SMART ENERGY GRID

A Project Report

Submitted By

Honey Vasanat Patel

200303125009

Yash Rank

210303125004

Surani Smit

210303125005

Aviral

210303125016

in Partial Fulfilment For the Award of

the Degree of

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING

Under the Guidance of

Prof. Yogendra Singh

Prof. Aditi Mishra

Assistant Professor

VADODARA October - 2024

PARUL UNIVERSITY

CERTIFICATE

This is to Certify that Project - 1 (203105499) of 6^{th} Semester entitled "SMART ENERGY GRID" of Group No. PUCSE_06 has been successfully completed by

- HONEY VASANT PATEL- 200303125009
- YASH RANK- 210303125004
- SURANI SMIT- 210303125005
- AVIRAL- 210303125016

under my guidance in partial fulfillment of the Bachelor of Technology (B.Tech) in Computer Science & Engineering of Parul University in Academic Year 2023- 2024.

Date of Submission :_____

Prof. Yogendra Singh,
Project Guide

Dr. Amit Barve,
Head of Department,
CSE, PIET,
Parul University.

Project Coordinator:-

Acknowledgements

"The single greatest cause of happiness is gratitude."

-Auliq-Ice

We would like to extend our sincere gratitude to all those who have contributed to the successful completion of this project on the Smart Energy Grid.

First and foremost, we express our deepest appreciation to our internal guide,**Ms. Aditi Mishra** and **Prof. Yogendra Singh**, whose expertise and guidance have been invaluable throughout this endeavor. Her insights, support, and encouragement have played a significant role in shaping the direction and outcomes of our project.Her expertise and dedication have been instrumental in steering us towards achieving our goals.

Furthermore, we would like to thank **Dr. Amit Barve**, Head of the Department, for his continuous encouragement, support, and invaluable suggestions which have been pivotal in shaping our project.

We would also like to extend our thanks to **MGVCL Vadodara Division** for generously sharing their data, without which this project would not have been possible. Their cooperation and assistance have been immensely valuable in conducting our research and analysis.

Finally, we would like to express our gratitude to all individuals, organizations, and resources that have contributed directly or indirectly to the successful completion of this project.

Thank you all for your invaluable support and guidance.

Honey Vasant Patel

Yash Rank

Surani Smit

Aviral

CSE, PIET

Parul University,

Vadodara

Abstract

The Smart Grid project aims to address the challenges faced in the energy sector, particularly focusing on combating electricity theft in the Gorwa Sub Division of MGVCL. Electricity theft poses significant financial losses and safety risks, necessitating innovative solutions for detection and prevention. This project leverages advanced technologies, including Big Data Analytics, Machine Learning (ML), and Blockchain, to develop a comprehensive system for monitoring, analyzing, and securing the energy grid.

The project utilizes data provided by MGVCL to analyze energy consumption patterns and identify anomalies indicative of theft. Through ML algorithms, the system can detect irregularities in energy usage, such as sudden spikes or unauthorized connections, enabling prompt intervention by utility providers. Additionally, predictive analytics techniques are employed to forecast energy demand, optimize distribution, and minimize losses.

Blockchain technology is integrated into the system to ensure secure and transparent energy transactions. By utilizing a decentralized ledger, the project aims to prevent tampering and unauthorized access to energy data, enhancing trust and reliability in the energy ecosystem. Smart contracts based on Ethereum enable automated and secure transactions, streamlining energy management processes and reducing administrative overhead.

The Smart Grid project encompasses various components, including real-time data acquisition, visualization dashboards, predictive analytics modules, and cybersecurity measures. These components work synergistically to create a robust and resilient energy infrastructure that can adapt to evolving challenges and demands. Moreover, the project emphasizes the integration of renewable energy sources, demand response mechanisms, and grid optimization techniques to promote sustainability and efficiency.

Through the development and implementation of the Smart Grid system, the project endeavors to redefine sustainability and resilience in the energy sector. By empowering stakeholders with actionable insights and secure transaction mechanisms, the project aims to foster a more efficient, transparent, and reliable energy ecosystem, ultimately benefiting both consumers and utility providers in the Gorwa Sub Division and beyond.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Backgroung

The modern world is witnessing a rapid evolution in energy distribution and management systems due to increasing energy demand, environmental concerns, and technological advancements. Traditional power grids face challenges such as inefficiency, limited flexibility, and vulnerability to disruptions. In response, the concept of a Smart Energy Grid has emerged as a transformative solution to address these challenges and pave the way for a more sustainable and efficient energy future.

1.2 Definition and Concept

A Smart Energy Grid, also known as a Smart Grid, is an advanced electricity network that utilizes digital communication and control technologies to optimize the generation, distribution, and consumption of energy. Unlike conventional grids, a Smart Grid integrates various renewable energy sources, energy storage systems, advanced metering infrastructure (AMI), and sophisticated monitoring and control mechanisms. This integration enables real-time data analysis, predictive maintenance, demand response capabilities, and enhanced grid resilience.

1.3 Objectives

The primary objectives of implementing a Smart Energy Grid are multifaceted:

Enhancing Energy Efficiency: By employing advanced monitoring and control mechanisms, a Smart Grid aims to minimize energy losses during transmission and distribution, thereby improving overall efficiency.

Integration of Renewable Energy: With the growing adoption of renewable energy sources such as solar and wind, a Smart Grid facilitates the seamless integration and management of these

intermittent energy sources into the grid.

Demand Response and Load Management: Through the deployment of smart meters and demand response programs, a Smart Grid enables utilities to manage peak demand more effectively, reduce strain on the grid, and optimize energy consumption patterns.

Grid Resilience and Reliability: By incorporating self-healing capabilities, real-time monitoring, and predictive analytics, a Smart Grid enhances grid resilience, minimizes downtime, and mitigates the impact of disruptions such as natural disasters or cyberattacks.

1.4 Key Components

The architecture of a Smart Energy Grid comprises several key components:

Advanced Metering Infrastructure (AMI): Smart meters enable two-way communication between consumers and utilities, providing real-time data on energy consumption and facilitating demand response programs.

Distribution Automation: Automated control systems and sensors are deployed across the distribution network to detect faults, optimize voltage levels, and reroute power flows, improving reliability and efficiency.

Energy Storage Systems: Battery storage and other energy storage technologies play a crucial role in balancing supply and demand, storing excess energy during off-peak hours and releasing it during periods of high demand.

Grid-Connected Devices: Internet of Things (IoT) devices, sensors, and smart appliances enable remote monitoring, control, and optimization of energy usage within homes, businesses, and industrial facilities.

1.5 Benefits

The deployment of a Smart Energy Grid offers numerous benefits to various stakeholders:

Consumers: Increased visibility into energy usage, potential cost savings through demand response programs, and greater control over energy consumption.

Utilities: Improved operational efficiency, reduced maintenance costs, better outage management, and enhanced customer satisfaction.

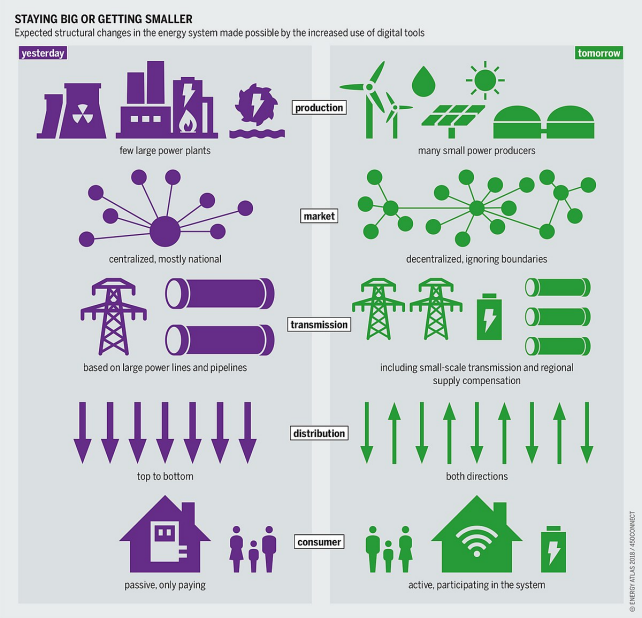

Figure 1.1: Traditional Grid v/s Smart Grid

1.6 Challenges

Despite its potential benefits, the implementation of a Smart Energy Grid also poses certain challenges:

Cybersecurity Concerns: With increased connectivity and digitalization, Smart Grids become vulnerable to cyber threats, necessitating robust cybersecurity measures to safeguard critical infrastructure and data.

Interoperability: Integrating diverse technologies and systems from different vendors can pose interoperability challenges, requiring standardized protocols and interfaces.

1.7 Future Outlook

The future of energy grids lies in the continued evolution and adoption of smart technologies. Advancements in artificial intelligence, edge computing, and decentralized energy systems are poised to further enhance the capabilities and resilience of Smart Energy Grids, paving the way for a more sustainable and resilient energy future.

Chapter 2

Literature Survey

2.1 Evolution of Smart Grid Technology

The evolution of Smart Grid technology has been extensively studied in the literature. Early research focused on the conceptualization and development of Smart Grid architectures, highlighting the shift from traditional centralized grids to decentralized, digitally-enabled networks. Key milestones in the development of Smart Grid technology, such as the integration of advanced metering infrastructure (AMI), distribution automation, and demand response mechanisms, have been documented (Faruqui Sergici, 2010).

2.2 Integration of Renewable Energy Sources

Research on the integration of renewable energy sources (RES) into Smart Grids has gained significant attention in recent years. Studies have explored various aspects, including the technical challenges associated with the intermittent nature of RES, grid stability, and the development of forecasting models to optimize their integration into the grid. Additionally, economic and policy frameworks for incentivizing renewable energy deployment within Smart Grids have been investigated (Kazmi et al., 2019).

2.3 Advanced Metering Infrastructure (AMI) and Demand Response

The deployment of Advanced Metering Infrastructure (AMI) and demand response programs is a cornerstone of Smart Grid technology. Literature in this area has focused on the design and implementation of AMI systems, including smart meters and communication protocols, as well as the development of demand response strategies to manage peak loads, reduce energy consumption, and enhance grid stability. Studies have also examined the impact of AMI and demand response programs on consumer behavior and energy market dynamics (Faruqui Hledik, 2018).

2.4 Grid Resilience and Cybersecurity

Ensuring the resilience and cybersecurity of Smart Grids is of paramount importance to safeguard critical infrastructure and mitigate potential threats. Research in this area has investigated various cybersecurity challenges, including data privacy, network vulnerabilities, and the risk of cyberattacks on grid components. Additionally, studies have explored novel approaches for enhancing grid resilience, such as self-healing algorithms, distributed energy resources, and blockchain-based solutions for secure data exchange (Koutsandria et al., 2020).

2.5 Energy Storage and Grid Optimization

The role of energy storage systems in Smart Grids has been a subject of extensive research. Studies have examined different energy storage technologies, such as batteries, pumped hydro, and thermal storage, and their integration into grid operations for load balancing, peak shaving, and renewable energy integration. Furthermore, research has focused on optimization techniques, including predictive analytics and real-time control algorithms, to maximize the efficiency and reliability of energy storage within Smart Grid environments (Zhang et al., 2020).

2.6 Smart Grid Communication and Networking

Effective communication and networking infrastructure are essential for the reliable operation of Smart Grids. Research in this area has investigated communication protocols, network architectures, and wireless technologies suitable for Smart Grid deployments. Additionally, studies have explored the use of Internet of Things (IoT) devices, edge computing, and machine learning techniques to enhance the scalability, reliability, and security of communication systems within Smart Grid environments (Wang et al., 2019).

2.7 Regulatory and Policy Frameworks

The regulatory and policy landscape surrounding Smart Grid deployment has been a focus of scholarly inquiry. Research in this area has examined the role of regulatory agencies, government policies, and market mechanisms in incentivizing investments in Smart Grid infrastructure, promoting grid modernization, and fostering innovation. Additionally, studies have analyzed the socio-economic impacts of Smart Grid initiatives, including their implications for energy affordability, environmental sustainability, and consumer welfare (Jamasb Nuttall, 2011).

2.8 Distributed Energy Resources (DERs) Integration

The integration of Distributed Energy Resources (DERs) presents both opportunities and challenges for Smart Grids. Studies investigate the technical feasibility of integrating DERs such as solar photovoltaics (PV), wind turbines, and microgrids into the grid infrastructure. Furthermore, research explores the impact of DERs on grid stability, power quality, and resilience (khodayar2020).

2.9 Microgrid Technologies

Microgrids have emerged as a promising solution for enhancing grid resilience and promoting energy independence. Research in this area focuses on the design, optimization, and control of microgrid systems. Additionally, studies investigate the economic viability of microgrid deployment, regulatory frameworks, and the integration of renewable energy sources (hatziargyriou2020).

2.10 Grid Modernization and Infrastructure Upgrades

Grid modernization initiatives aim to upgrade aging infrastructure and enhance grid resilience and efficiency. Scholars examine the deployment of smart sensors, automation technologies, and grid monitoring systems for real-time asset management and predictive maintenance. Moreover, research explores the role of data analytics, machine learning, and artificial intelligence in optimizing grid operations (brown2020).

2.11 Electric Vehicle (EV) Integration

The integration of electric vehicles (EVs) presents new challenges and opportunities for Smart Grids. Studies investigate EV charging infrastructure, vehicle-to-grid (V2G) technologies, and demand-side management strategies. Additionally, research explores the impact of EV integration on grid stability, power distribution, and energy demand forecas(li2020).

2.12 Cyber-Physical Systems (CPS) in Smart Grids

Cyber-Physical Systems (CPS) play a crucial role in the operation and management of Smart Grids. Research focuses on the integration of sensors, actuators, and control systems to monitor and control grid assets in real-time. Moreover, studies investigate the resilience of CPS against cyber threats and the development of secure communication protocols (yu2020).

2.13 Artificial Intelligence (AI) and Machine Learning (ML) in Smart Grids

Artificial Intelligence (AI) and Machine Learning (ML) techniques are increasingly being applied to optimize Smart Grid operations. Researchers explore the use of AI/ML algorithms for load forecasting, fault detection, and predictive maintenance. Additionally, studies investigate reinforcement learning and deep learning approaches for grid optimization and control (chen2020).

2.14 Blockchain Technology for Energy Trading

Blockchain technology has the potential to revolutionize energy trading and peer-to-peer (P2P) transactions in Smart Grids. Literature in this area examines blockchain-based energy trading platforms, smart contracts, and decentralized energy markets. Moreover, research explores the scalability, security, and regulatory challenges of implementing blockchain in Smart Grid environments (yuan2020).

2.15 Social and Economic Implications of Smart Grid Adoption

The adoption of Smart Grid technology has significant social and economic implications. Studies investigate the impact of Smart Grids on energy affordability, equity, and access. Moreover, research explores the potential for job creation, economic development, and innovation in Smart Grid-related industries. Additionally, studies examine consumer acceptance and behavior change in response to

2.16 Internet of Things (IoT) Applications in Smart Grids

The Internet of Things (IoT) enables the connectivity of various devices and sensors in Smart Grids. Studies investigate IoT applications for grid monitoring, demand response, and asset management. Additionally, research explores edge computing and fog computing architectures to process IoT data in real-time (zhang2020iot).

2.17 Energy Management Systems (EMS) for Smart Grids

Energy Management Systems (EMS) play a crucial role in optimizing energy consumption and grid operations. Scholars examine EMS architectures, optimization algorithms, and control strategies for load balancing and demand-side management. Moreover, research investigates the integration of EMS with renewable energy sources and energy storage systems (dutta2020).

2.18 Privacy-Preserving Techniques for Smart Grid Data

Privacy-preserving techniques are essential for protecting sensitive consumer data in Smart Grids. Research explores cryptographic protocols, differential privacy, and anonymization techniques to safeguard privacy while enabling data sharing for grid analytics and optimization. Additionally, studies investigate the trade-off between data utility and privacy protection (zhang2020privacy).

2.19 Environmental Impacts and Sustainability of Smart Grids

The environmental impacts and sustainability of Smart Grids are critical considerations in their design and operation. Literature examines the lifecycle environmental impacts of Smart Grid technologies, including their carbon footprint, resource consumption, and emissions. Moreover, research explores the role of Smart Grids in promoting renewable energy deployment and mitigating climate change (perez2020).

Chapter 3

Analysis / Software Requirements Specification (SRS)

3.1 Overall Description

3.1.1 Product Perspective

The Smart Grid System will serve as a comprehensive and adaptive platform, integrating advanced technologies to monitor, analyze, and optimize the power grid. It will employ real-time data processing, predictive analytics, cloud-based scalability, blockchain-enabled secure data storage, and robust cybersecurity protocols to achieve its objectives.

3.1.2 Product Features

Real-time Monitoring

- Data Acquisition: Utilize scalable data ingestion technologies for real-time data collection.
- In-Memory Processing: Implement in-memory databases for rapid data processing.
- Visual Analytics: Utilize tools like Grafana or Tableau for real-time visualization.

Predictive Analytics

- Machine Learning Models: Develop and deploy ML models for predictive maintenance.
- Data Preprocessing: Use Python and R for preprocessing and feature engineering.
- Model Deployment: Implement containerization (e.g., Docker) for efficient model deployment.

Cloud Integration

- Cloud Storage: Utilize AWS S3 or Azure Blob Storage for scalable and secure data storage.
- Serverless Computing: Implement serverless architecture for efficient resource utilization.
- Auto-scaling: Leverage cloud auto-scaling capabilities to adapt to varying workloads.

Blockchain Integration

- Consensus Mechanism: Select and implement a consensus mechanism suitable for energy transactions.
- Smart Contracts: Develop Ethereum-based smart contracts for secure and automated transactions.
- Decentralized Storage: Utilize blockchain for decentralized and tamper-proof data storage.

Cyber Security Measures

- Encryption Standards: Implement AES-256 for end-to-end encryption of data.
- Multi-factor Authentication: Incorporate multi-factor authentication for enhanced user security.
- Intrusion Detection System: Deploy sophisticated IDS to monitor and respond to potential threats.

3.1.3 User Classes and Characteristics

The Smart Grid System's primary use case involves optimizing energy distribution and consumption in a smart city environment. It collects real-time data from various sources such as sensors, smart meters, and weather forecasts to monitor energy demand and grid performance. Using predictive analytics, it anticipates peak usage periods and adjusts energy distribution accordingly to prevent overloads and minimize wastage. The system also enables users to access their energy consumption data and manage it efficiently through a user-friendly interface, promoting energy conservation and cost savings.

3.1.4 Operating Environment

- Hardware Components: Integration of sensors, smart meters, and other IoT devices to monitor energy usage, grid performance, and environmental factors.

- Software Systems: Development of software for data collection, processing, and analysis, including algorithms for predictive analytics and optimization of energy distribution.
- Communication Infrastructure: Implementation of communication protocols for data transmission between various components of the smart grid system, such as Wi-Fi, Zigbee, or LoRaWAN.
- Data Storage and Management: Utilization of databases or cloud storage solutions for storing and managing collected data securely.
- User Interface: Designing user interfaces for stakeholders to visualize energy consumption data, control devices remotely, and receive alerts or notifications.
- Simulation Tools: Integration of simulation tools or platforms to model and simulate different scenarios for testing and validation purposes.
- Security Measures: Implementation of cybersecurity protocols to safeguard data integrity, prevent unauthorized access, and protect against cyber threats.
- Power Infrastructure: Integration with existing power infrastructure or setup of a small-scale power grid for testing and validation of energy distribution algorithms and grid management strategies.

3.1.5 Design and Implementation Constraints

Design and implementation constraints for a smart energy grid project in a college final year context encompass budget limitations, time constraints, resource availability, compatibility requirements with existing infrastructure, regulatory compliance, environmental considerations, scalability and flexibility needs, and security concerns. These constraints necessitate careful planning, prioritization, and efficient use of resources to develop a feasible and effective solution within the project's scope and timeframe, while ensuring adherence to regulatory standards, compatibility with existing systems, and robust security measures.

3.1.6 Assumptions and Dependencies

Assumptions

Assumptions in smart energy grid development include expectations for data accuracy, stable power supply, network connectivity, user engagement, regulatory compliance, interoperability, scalability, and cybersecurity, serving as foundational principles guiding project decisions.

Dependencies

Dependencies for a smart energy grid encompass hardware reliability, software functionality, power infrastructure stability, communication network resilience, access to external data sources, regulatory compliance, financial support, and stakeholder collaboration, all crucial for successful implementation and operation.

3.2 System Features

3.2.1 System Feature 1: Data Acquisition

- **Description:** Utilize scalable data ingestion technologies to collect real-time data from various sources.
- **Priority:** High
- **Response Sequences:**
  1. Identify Data Sources: Determine sources such as sensors, meters, and external databases for data collection.
  2. Implement Data Ingestion: Utilize scalable technologies like Apache Kafka or Apache NiFi for efficient and reliable data ingestion.
- **Functional Requirements:**
  - The system shall identify and prioritize data sources for real-time monitoring.
  - The system shall implement scalable data ingestion technologies for efficient data collection.

3.2.2 System Feature 2: In-Memory Processing

- **Description:** Implement in-memory databases to process real-time data rapidly.
- **Priority:** High
- **Response Sequences:**
  1. Select In-Memory Database: Choose suitable technologies such as Redis or Apache Ignite for in-memory processing.
  2. Configure Data Pipelines: Design data pipelines to ingest and process data in real-time using the selected database.

- **Functional Requirements:**
  - The system shall select and configure an appropriate in-memory database for real-time processing.
  - The system shall design and implement data pipelines to process incoming data efficiently.

3.3 External Interface Requirements

3.3.1 User Interfaces

The user interface (UI) for the smart energy grid system encompasses a centralized dashboard providing real-time insights into energy consumption, grid performance metrics, and alerts, complemented by interactive data visualization tools enabling users to analyze energy usage patterns and forecast demand. Interactive control panels empower users to adjust energy settings, schedule tasks, and remotely control devices, while personalized user profiles tailor the interface to individual preferences and roles. Notifications alert users to critical events or abnormal conditions, and historical data analysis tools identify optimization opportunities. Configuration wizards facilitate seamless setup, and mobile accessibility ensures flexibility. Security features, including authentication mechanisms and encryption protocols, safeguard user data, while help and support resources assist users in navigating the system. Overall, the UI prioritizes ease of use, accessibility, and functionality, empowering users to efficiently monitor, manage, and optimize energy usage within the smart grid system.

3.3.2 Hardware Interfaces

The hardware interface for the smart grid system serves as the bridge between the physical components of the energy infrastructure and the digital functionalities of the smart grid. It includes sensors, meters, actuators, and communication devices that collect real-time data on energy consumption, grid performance, and environmental conditions. These hardware components are integrated into the grid infrastructure to enable remote monitoring, control, and optimization. The interface facilitates bidirectional communication between the smart grid system and the physical assets, allowing for seamless data transmission, command execution, and feedback mechanisms. Compatibility with existing hardware infrastructure, scalability to accommodate future expansion, and reliability to ensure continuous operation are essential considerations in designing the hardware interface for the smart grid.

3.3.3 Software Interfaces

The software interface for the smart energy grid serves as the central nervous system of the system, facilitating seamless communication, data processing, and control functionalities across various layers. At the data acquisition layer, it collects real-time data from sensors, smart meters, and weather stations, integrating it for analysis and decision-making. This data is then processed in the data processing layer, where algorithms and analytics tools extract valuable insights, predict energy demand, and optimize grid management. The control and management layer oversees grid infrastructure operation, executing commands and adjusting parameters based on data analysis to enhance performance. Meanwhile, the user interface layer provides stakeholders with an intuitive platform to monitor energy usage, configure settings, and receive alerts, empowering informed decision-making. Lastly, the integration layer ensures interoperability with external systems, enabling seamless data exchange and enhancing functionality. Together, these layers constitute the software interface, enabling the smart energy grid to operate efficiently, adapt to changing conditions, and deliver value to stakeholders.

3.3.4 Communications Interfaces

The communication interface for the smart energy grid is the vital conduit enabling the seamless exchange of data and commands between various components within the system. It encompasses a range of communication technologies and protocols, including wired (e.g., Ethernet, fiber optics) and wireless (e.g., Wi-Fi, Zigbee) connections. These interfaces facilitate real-time transmission of data from sensors, meters, and control devices to the central management system, enabling continuous monitoring and analysis of energy usage and grid performance. Bidirectional communication capabilities allow for remote control of grid assets, such as adjusting energy distribution and responding to demand fluctuations. Additionally, secure communication protocols ensure data integrity and protect against cyber threats, maintaining the reliability and resilience of the smart energy grid infrastructure.

3.4 Other Nonfunctional Requirements

3.4.1 Performance Requirements

- **Response Time**:
  - Aim for a median response time of < 100 ms for critical operations.
  - Implement caching mechanisms to optimize response times.

- **Scalability**:
  - Design the system to handle a minimum of 10,000 concurrent users.
  - Utilize load balancing and horizontal scaling for scalability.

3.4.2 Reliability Requirements

- **System Uptime**:
  - Aim for 99.99% uptime for critical system components.
  - Implement automated failover mechanisms to minimize downtime.
- **Fault Tolerance**:
  - Implement redundancy for critical components.
  - Utilize circuit breakers and graceful degradation for fault tolerance.

3.4.3 Security Requirements

- **Data Encryption**:
  - Utilize secure key management practices for encryption keys.
  - Regularly update encryption algorithms to adhere to industry standards.
- **Access Control**:
  - Enforce strict access controls on both API and database layers.
  - Regularly audit and update access control policies.

3.5 Tools and Technologies

3.5.1 Programming Languages

- Python for data analytics and machine learning
- R for statistical analysis and preprocessing

3.5.2 Cloud Platform

- AWS (S3, DynamoDB) or Azure (Blob Storage, Azure Functions) for cloud services
- OR Google Cloud Platform for data analytics and machine learning

3.5.3 Blockchain

- Ethereum for smart contracts and decentralized data storage
- Hyperledger Fabric for permissioned blockchain implementation

3.5.4 Cyber Security

- Hashicorp Vault for secret management
- Snort or Suricata for intrusion detection

3.6 Conclusion

This detailed Software Requirements Specification provides a comprehensive guide for the development team to design and implement an advanced Smart Grid System. By incorporating Big Data Analytics, Machine Learning, Cloud Computing, Blockchain, and Cyber Security, the system aims to redefine sustainability and reliability in the energy sector.

Chapter 4

System Design

4.1 System Design

The system design of the Smart Grid System encompasses the architecture, components, and interactions necessary to achieve its objectives of optimizing energy distribution and consumption in a smart city environment. The design focuses on scalability, reliability, security, and efficiency to ensure the effective operation of the system. The following sections outline the key aspects of the system design.

4.1.1 Architecture Overview

The architecture of the Smart Grid System is designed to be modular and scalable, allowing for seamless integration of various components and technologies. It follows a layered architecture, consisting of the following layers:

1. **Data Acquisition Layer**: This layer is responsible for collecting real-time data from sensors, meters, and other sources. Data is ingested into the system for processing and analysis.

2. **Data Processing Layer**: In this layer, incoming data is processed using advanced analytics techniques, including machine learning algorithms. Predictive analytics is applied to anticipate energy demand and optimize grid operations.

3. **Control and Management Layer**: The control and management layer oversees the operation of the grid infrastructure. It executes commands based on data analysis results to optimize energy distribution and ensure grid stability.

4. **User Interface Layer**: This layer provides stakeholders with a user-friendly interface to monitor energy usage, configure settings, and receive alerts. It enables informed decision-

making and promotes user engagement.

5. **Integration Layer**: The integration layer facilitates interoperability with external systems and data sources. It enables seamless data exchange and enhances the functionality of the smart grid system.

4.1.2 Component Overview

The Smart Grid System comprises the following key components:

1. **Sensor Network**: Sensors are deployed throughout the grid infrastructure to collect data on energy consumption, grid performance, and environmental conditions.

2. **Data Processing Engine**: This component processes incoming data using advanced analytics tools and techniques. It includes machine learning models for predictive maintenance and optimization.

3. **Cloud Infrastructure**: The system leverages cloud services for scalable and secure data storage, computation, and analytics. It utilizes providers such as AWS or Azure for cloud integration.

4. **Blockchain Network**: Blockchain technology is incorporated for secure and decentralized energy transactions and data storage. Smart contracts are deployed on platforms like Ethereum for automated transactions.

5. **Cybersecurity Framework**: A robust cybersecurity framework is implemented to safeguard sensitive data and ensure system integrity. It includes encryption, multi-factor authentication, and intrusion detection mechanisms.

4.1.3 Interaction Diagrams

Interaction diagrams illustrate the flow of data and control within the Smart Grid System. These diagrams depict how components interact with each other to achieve specific tasks, such as data acquisition, analysis, and control. Examples of interaction diagrams include sequence diagrams and activity diagrams.

4.1.4 Scalability and Performance

Scalability and performance are critical considerations in the design of the Smart Grid System. The architecture is designed to scale horizontally to accommodate growing data volumes and user

demands. Load balancing mechanisms and auto-scaling capabilities ensure optimal performance even under heavy workloads. Performance metrics, such as response time and throughput, are continuously monitored and optimized to meet system requirements.

4.1.5 Reliability and Fault Tolerance

Reliability and fault tolerance mechanisms are integrated into the system to ensure continuous operation and minimize downtime. Redundancy is implemented for critical components to mitigate the impact of hardware failures or system errors. Automated failover mechanisms and graceful degradation strategies are employed to maintain system uptime and availability.

4.1.6 Security and Compliance

Security measures are embedded throughout the system to protect against cyber threats and ensure compliance with regulatory standards. Data encryption, access controls, and secure communication protocols safeguard sensitive information from unauthorized access or tampering. Regular audits and updates are performed to maintain security posture and adhere to industry best practices.

Overall, the system design of the Smart Grid System reflects a holistic approach to address the complex challenges of energy management in a smart city environment. By leveraging advanced technologies and robust architectural principles, the system aims to redefine sustainability and reliability in the energy sector.

Chapter 5

Methodology

5.1 Methodology

The methodology for the development of the Smart Grid System, with a particular emphasis on electricity theft detection, Blockchain integration, and data analysis, involved several stages aimed at ensuring the robustness, security, and efficiency of the system.

5.1.1 Requirements Gathering

The initial phase of the methodology focused on gathering requirements, particularly those related to electricity theft detection and data analysis. Stakeholder meetings were conducted with representatives from MGVCL, regulatory bodies, and end-users to understand their specific needs and challenges. Detailed discussions were held to identify key requirements, including real-time monitoring of energy consumption, anomaly detection for potential theft, data security, and regulatory compliance.

5.1.2 Data Collection and Preprocessing

Following requirements gathering, the next step involved collecting and preprocessing data from MGVCL's Gorwa Sub Division. Historical data on energy consumption, voltage levels, and distribution patterns were obtained from MGVCL's database. Additionally, data on past instances of electricity theft and related anomalies were collected for analysis. The collected data underwent preprocessing, including cleaning, normalization, and feature engineering, to prepare it for further analysis.

5.1.3 Data Analysis and Anomaly Detection

Once the data was preprocessed, advanced data analysis techniques were applied to detect anomalies indicative of electricity theft. Statistical methods, machine learning algorithms, and

Table 5.1: Feeder Electricity Details

Feeder name	**A Feeder category**		**B**	**Total Soldout**	**C**
	Urban	**EHT**		**(19=16-18+19)**	**Unit Loss (20-11-19)**
PANCHAVATI	URBAN			7612555	20815
GUJ.HOUSING BOARD	URBAN			6805922	518258
B.I.D.C.	URBAN			3813588	70457
GORWA GAM	URBAN			7391724	175909
66 KV ALEMBIC PHARMA	EHT			9288611	-2651
BHAILAL AMIN HOSPITAL	HTEX			2993280	-23580
INORBIL MALL	HTEX			6672802	-59522
SUBHANPIRA SST SHREENATHJI	SST			28435	21
URBAN	URBAN			6667980	71571
SAHYOG ROAD	URBAN			7638992	37488
URBAN	URBAN			6831158	211605
SAMTA	URBAN			7665028	335804
SUBHANPURA ROAD	URBAN			8061569.92	173388.08
TCD LTD	HTEX			1368737	2363
DWARKESH	URBAN			5701444	204912
SANGRILA	URBAN			6961815	18088
GOODIES	URBAN			10015957	102567
SWAMI VIVEKANAND	URBAN			2421171	93681
LABH	URBAN			2950013.5	78770.5
VADHUNAGAR	URBAN			4818659	331231
GORWAS SST	SST			8811	59
Total				115617182.4	2548569.58

anomaly detection techniques were employed to identify patterns and deviations from normal energy consumption behavior. Features such as sudden spikes or drops in energy usage, abnormal load profiles, and unauthorized connections were analyzed to flag potential instances of theft.

5.1.4 Blockchain Integration for Data Security

In parallel with data analysis, Blockchain technology was integrated into the Smart Grid System to enhance data security, integrity, and transparency. Smart contracts were developed using Ethereum's Solidity programming language to record energy transactions, meter readings, and distribution data on the Blockchain. Each transaction was cryptographically signed and timestamped, ensuring immutability and traceability of data.

5.1.5 Model Training and Validation

Machine learning models were trained using historical data to predict electricity theft and identify suspicious activities in real-time. Supervised learning algorithms such as Random Forest, Support Vector Machines (SVM), and Neural Networks were employed to classify normal and anomalous energy consumption patterns. The models were validated using cross-validation techniques and evaluated based on metrics such as accuracy, precision, recall, and F1-score.

5.1.6 System Implementation and Deployment

The developed Smart Grid System, incorporating Blockchain integration and data analysis modules, was implemented and deployed in the Gorwa Sub Division of MGVCL. The system was integrated with existing infrastructure, including smart meters, sensors, and communication networks, to enable real-time data collection, analysis, and anomaly detection. User interfaces were designed to visualize energy consumption patterns, alert operators to potential theft, and provide insights for decision-making.

5.1.7 Pilot Testing and Evaluation

Pilot testing was conducted to evaluate the performance, reliability, and effectiveness of the Smart Grid System in detecting electricity theft and improving grid security. The system's response to simulated theft scenarios and real-world anomalies was assessed, and feedback was collected from end-users and stakeholders. Adjustments and refinements were made based on the feedback received to enhance system capabilities and usability.

5.1.8 Continuous Monitoring and Improvement

Post-deployment, the Smart Grid System underwent continuous monitoring to detect any issues or performance degradation. Regular maintenance and updates were performed to address security vulnerabilities, improve algorithms, and adapt to evolving threats. Collaboration with MGVCL and other stakeholders facilitated ongoing improvements and optimizations to ensure the long-term effectiveness and sustainability of the system.

By following this comprehensive methodology, the Smart Grid System successfully addressed the challenges of electricity theft detection, data security, and grid optimization in the Gorwa Sub Division of MGVCL. The integration of Blockchain technology and advanced data analysis techniques enabled the system to detect anomalies, prevent theft, and improve overall grid efficiency, contributing to a more secure and reliable electricity distribution network.

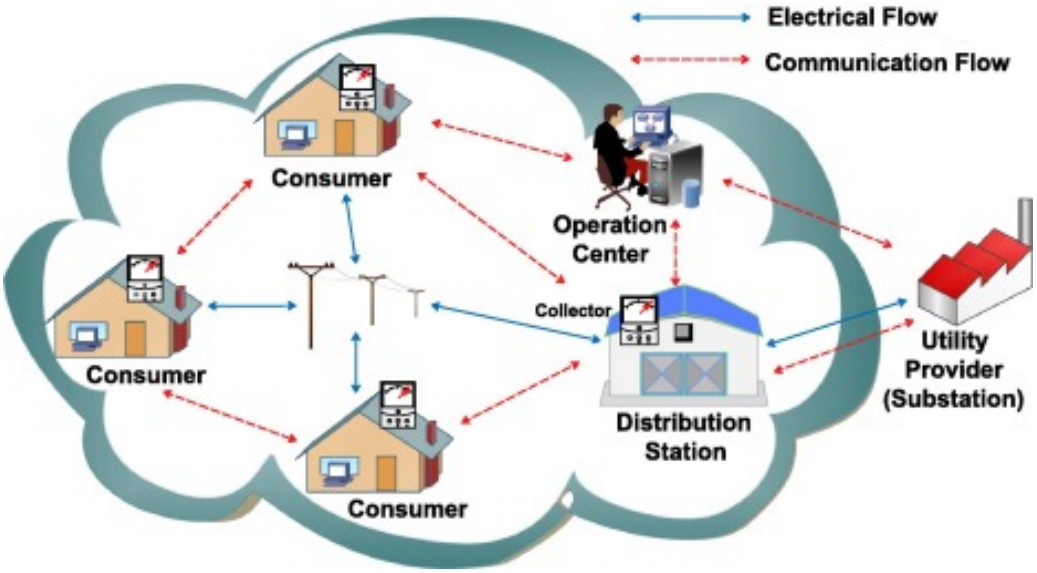

Figure 5.1: Electricity Distribution

5.2 Components Overview

This section provides a detailed description of the components used in the IoT-based smart meter system.

5.2.1 NodeMCU ESP32 Microcontroller

The NodeMCU ESP32 is a powerful and versatile microcontroller based on the ESP32 chip, designed specifically for IoT applications. This dual-core processor operates at a clock speed of up to 240 MHz and features integrated Wi-Fi and Bluetooth capabilities, making it ideal for wireless communication in smart systems.

Key Features

- **Wireless Connectivity:** The ESP32 supports both Wi-Fi and Bluetooth, allowing it to connect to various IoT platforms and devices seamlessly. This capability is crucial for real-time data monitoring and control.
- **GPIO Pins:** The microcontroller comes equipped with multiple General Purpose Input/Output (GPIO) pins, which can be used to interface with various sensors and actuators. This flexibility enables the integration of multiple peripherals, such as LEDs, buttons, and sensors.
- **Low Power Consumption:** The ESP32 is designed for energy efficiency, making it suitable for battery-powered applications. Its various sleep modes help conserve power during idle periods.
- **Programming Environment:** The NodeMCU firmware allows for easy programming using the Lua scripting language or the Arduino IDE, providing accessibility for both novice and experienced developers.

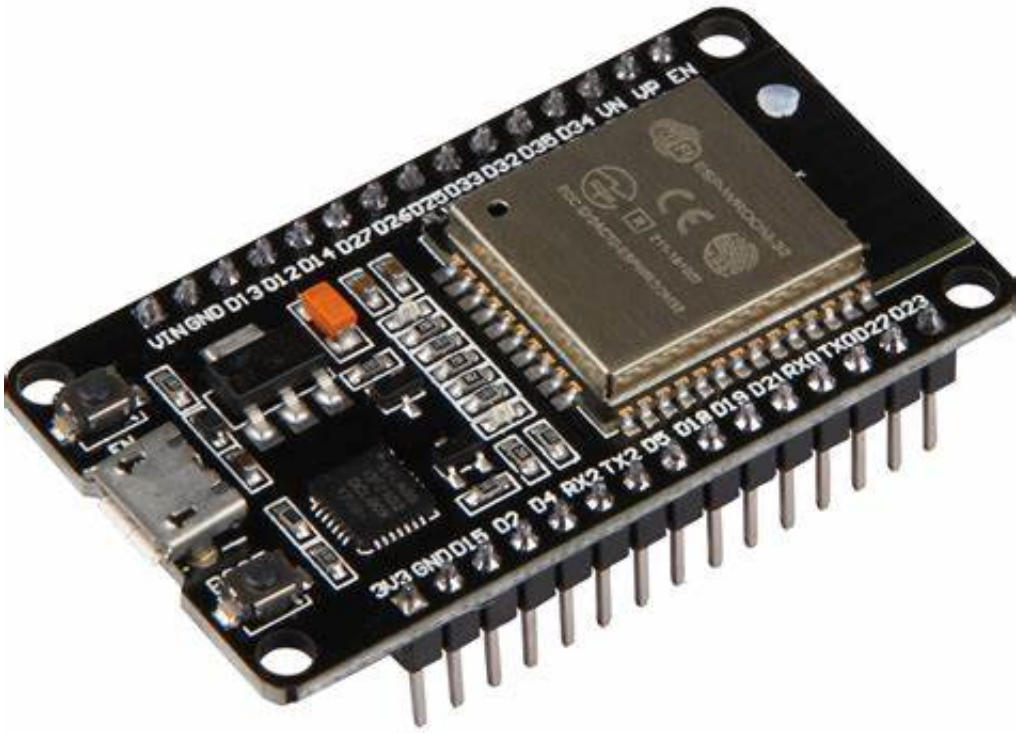

Figure 5.2: NodeMCU ESP32

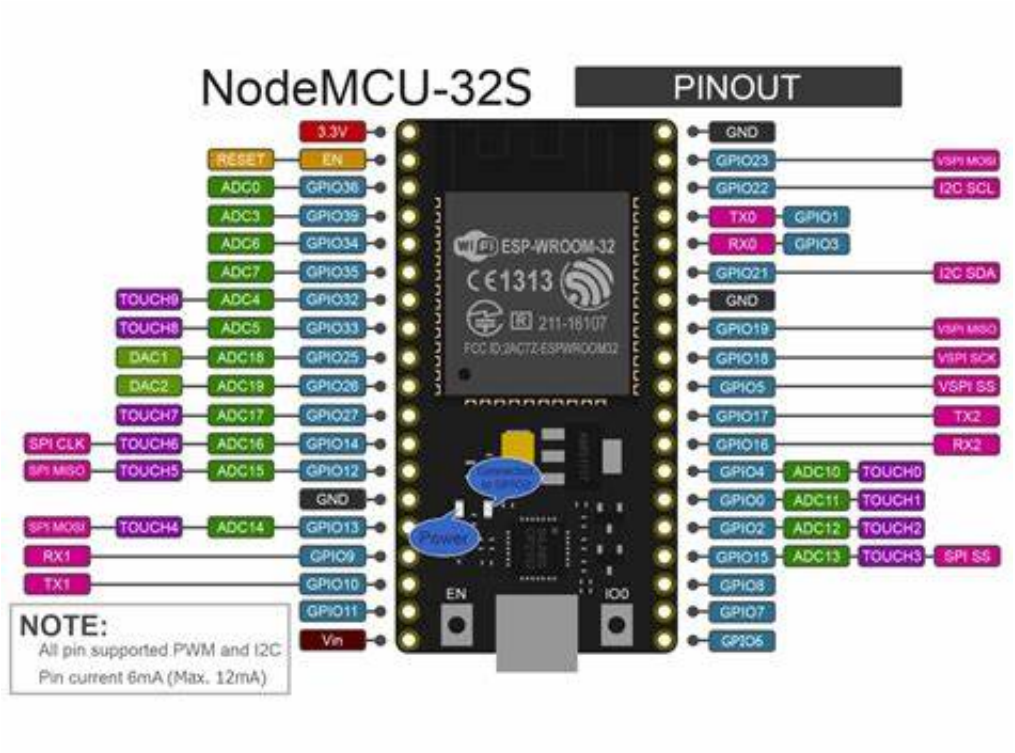

Figure 5.3: NodeMCU ESP32 Pinout

5.2.2 ZMTP101B AC Voltage Sensor

The ZMTP101B AC Voltage Sensor is a crucial component for measuring the alternating current (AC) voltage levels in electrical systems. It provides accurate voltage readings that are essential for monitoring and analyzing electrical parameters.

Key Features

- **Measurement Range:** The ZMTP101B sensor can measure AC voltages in a specified range, typically up to 230V, making it suitable for standard household and industrial applications.
- **Isolation:** This sensor is designed with electrical isolation to protect the microcontroller from high voltage levels. It typically includes opto-isolation, ensuring that any high voltage present does not affect the microcontroller's operation.
- **Analog Output:** The ZMTP101B provides an analog output proportional to the measured voltage, which can be read by the ESP32's analog input pins. This allows for straightforward integration into the microcontroller's data acquisition system.
- **Compact Design:** Its small size and lightweight design make it easy to integrate into various electrical systems without requiring significant space or complex installation procedures.

Figure 5.4: ZMTP101B

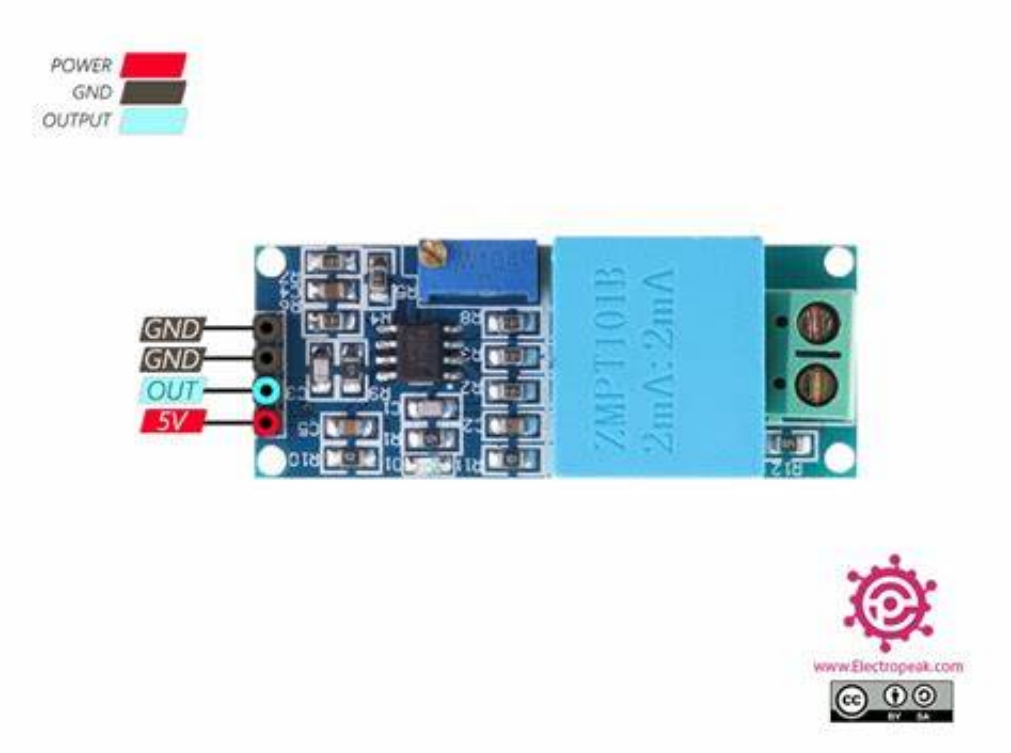

Figure 5.5: ZMTP101B Pinout

5.2.3 ACS712 Current Sensor

The ACS712 Current Sensor is used to measure the electrical current flowing through a conductor. It operates based on the Hall Effect principle, providing precise current readings that are essential for monitoring the electrical consumption of devices.

Key Features

- **Measurement Range:** The ACS712 comes in different versions, offering current measurement ranges such as ±5A, ±20A, and ±30A. This versatility makes it suitable for a wide range of applications, from small appliances to industrial equipment.
- **Analog Output:** Like the ZMTP101B, the ACS712 produces an analog voltage output proportional to the measured current. This output can be easily interfaced with the ESP32's ADC (Analog-to-Digital Converter) to obtain real-time current measurements.
- **High Precision:** The sensor is designed to provide accurate and stable readings, with a typical sensitivity of 185 mV/A. This precision is crucial for monitoring power consumption and detecting anomalies.
- **Compact Form Factor:** The ACS712 is also designed for easy integration into various systems, featuring a compact and lightweight construction.

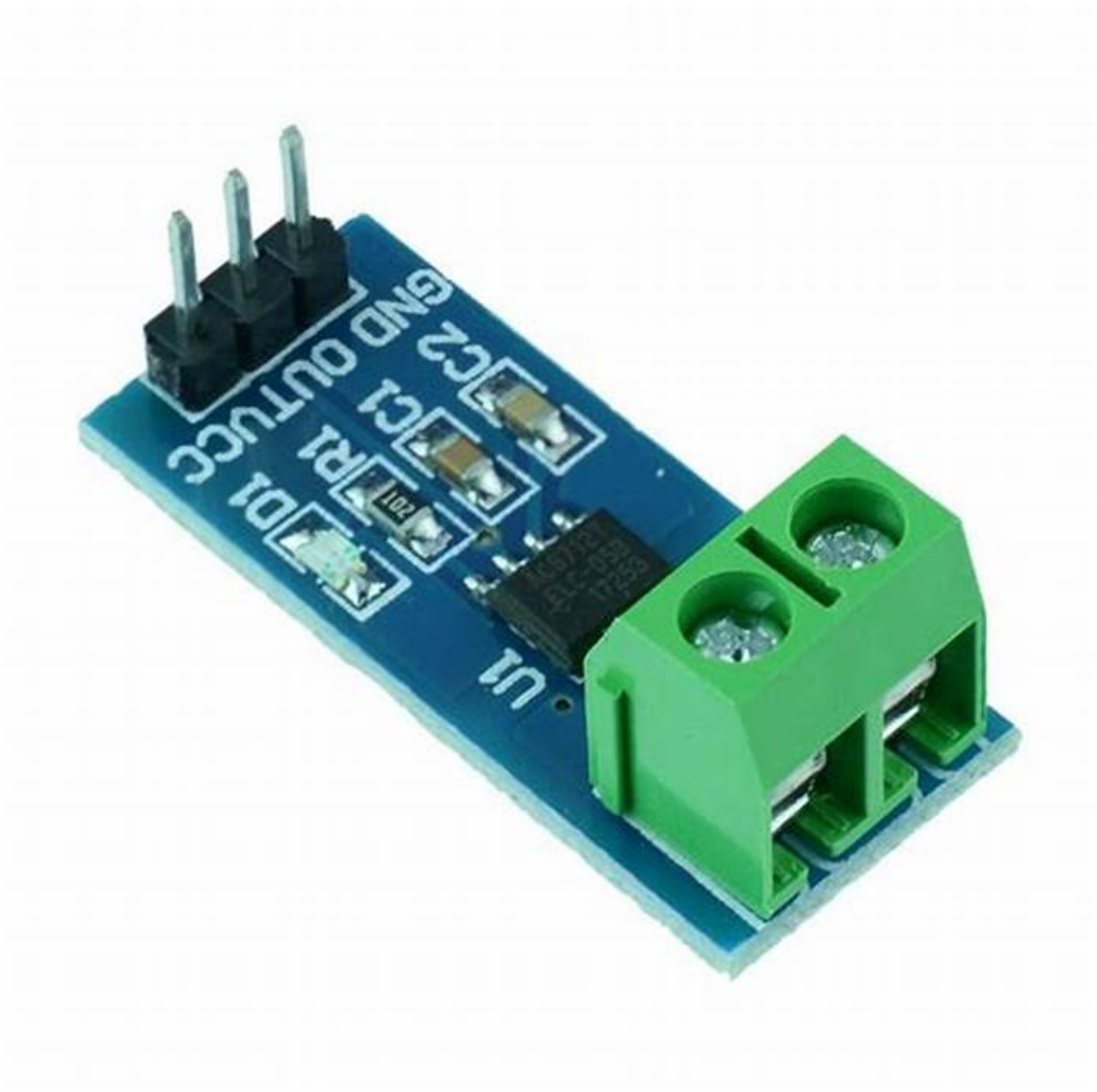

Figure 5.6: ACS712

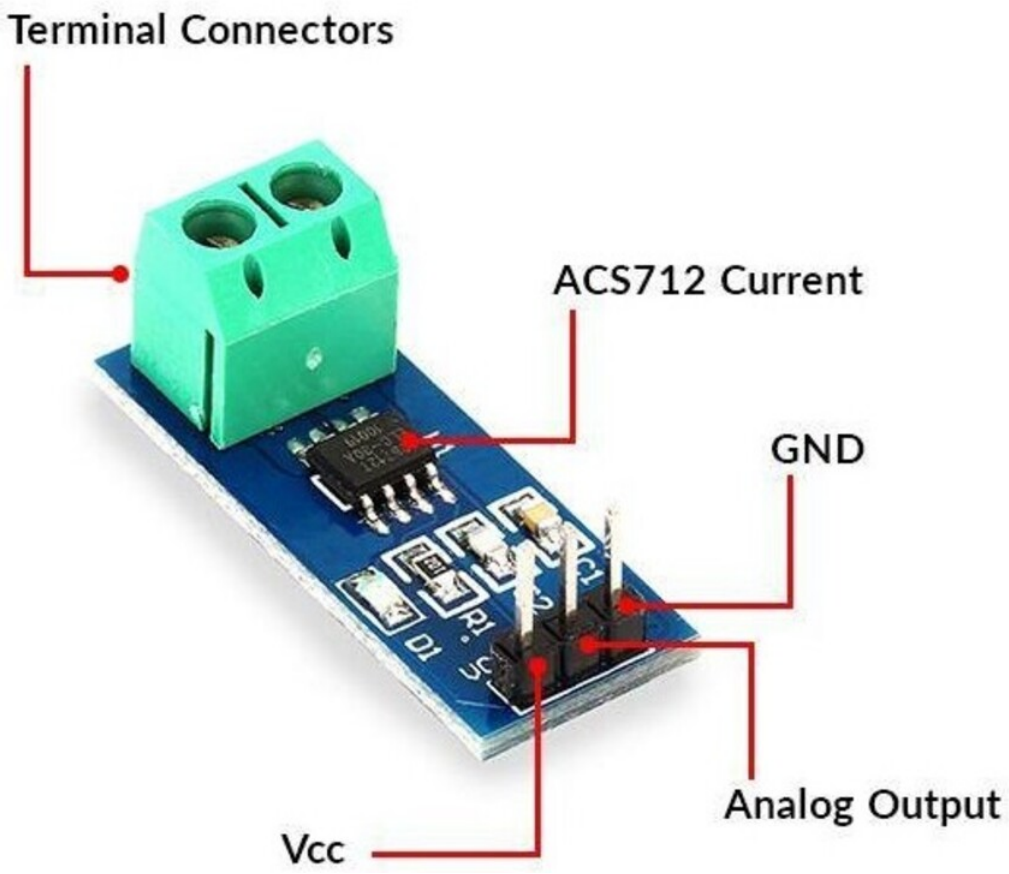

Figure 5.7: ACS712 Pinout

5.2.4 Summary of Components

These components work together to form a robust IoT-based smart meter system. The NodeMCU ESP32 microcontroller serves as the central processing unit, coordinating data acquisition from the ZMTP101B AC Voltage Sensor and the ACS712 Current Sensor. Together, these sensors provide essential information about voltage and current, enabling the system to monitor electrical parameters in real-time, detect anomalies, and respond to potential issues such as technical losses or theft effectively. This integration ultimately enhances the management and security of electrical systems, providing valuable insights for users and stakeholders.

Chapter 6

Implementation

6.1 Implementation

The implementation phase involved translating the methodology into concrete actions, including Dashboard development, data analysis, and system integration. Here's a detailed breakdown of each step:

6.1.1 Software Development

Smart Grid System Architecture

The Smart Grid System architecture was designed to accommodate modules for data collection, preprocessing, analysis, Blockchain integration, and user interfaces. The architecture followed a microservices-based approach to ensure modularity, scalability, and flexibility.

Data Collection Module

A data collection module was developed to fetch real-time data from smart meters, sensors, and other sources within the Gorwa Sub Division. APIs were created to interact with MGVCL's database and retrieve historical energy consumption data for analysis.

Data Preprocessing Module

The data preprocessing module cleaned, normalized, and transformed the collected data to prepare it for analysis. Techniques such as missing value imputation, outlier removal, and feature engineering were applied to enhance data quality and relevance.

Data Analysis Module

Advanced data analysis algorithms were implemented to detect anomalies indicative of electricity theft. Statistical methods, machine learning models, and pattern recognition techniques were

employed to identify abnormal energy consumption patterns and flag potential instances of theft.

Blockchain Integration Module

Smart contracts were developed using Solidity to record energy transactions, meter readings, and distribution data on the Blockchain. Integration with Ethereum's Blockchain network enabled secure, transparent, and tamper-proof recording of energy-related activities.

User Interface Development

User interfaces were designed to visualize energy consumption patterns, anomaly alerts, and Blockchain transactions. Interactive dashboards provided stakeholders with real-time insights into grid performance, theft detection, and energy usage optimization.

6.1.2 Data Analysis

Exploratory Data Analysis (EDA)

EDA techniques were applied to gain insights into the distribution and characteristics of energy consumption data. Descriptive statistics, data visualization, and correlation analysis were used to identify trends, patterns, and potential anomalies.

	SS2FEEDER Unites(Kwh) (8)	Feeder2SS UNites(Kwh) (9)	Consumer Export(10)	Total Sentout(11=8-9+10)	LT Sold Out(12)	AGUN Sold(13)	AGRC Sold(14)	HT Sold(15)	Total Billed UNites(16=12+13+14+15)	Consumer Import(17)
count	2.520000e+02	252.000000	2.520000e+02	2.520000e+02	2.520000e+02	252.000000	252.0	2.520000e+02	2.520000e+02	2.520000e+02
mean	5.522478e+06	10857.149286	5.880852e+05	6.099706e+06	3.446166e+06	12537.901865	0.0	2.090637e+06	5.549341e+06	5.523534e+05
std	2.965974e+06	20947.764451	6.625629e+05	3.175217e+06	2.855925e+06	41101.495940	0.0	3.200988e+06	2.966992e+06	6.191667e+05
min	8.565760e+03	0.000000	0.000000e+00	8.565760e+03	0.000000e+00	0.000000	0.0	0.000000e+00	8.508780e+03	0.000000e+00
25%	3.180532e+06	0.000000	2.974025e+03	3.305255e+06	1.221792e+04	0.000000	0.0	0.000000e+00	3.205786e+06	6.247375e+02
50%	5.965799e+06	41.075000	2.507366e+05	7.136809e+06	3.800001e+06	0.000000	0.0	0.000000e+00	6.188594e+06	1.503791e+05
75%	7.399542e+06	10670.710000	1.280190e+06	8.258045e+06	6.082788e+06	0.000000	0.0	3.627804e+06	7.426972e+06	1.161949e+06
max	1.237736e+07	89520.180000	2.015045e+06	1.240046e+07	8.827297e+06	208136.120000	0.0	1.103764e+07	1.226140e+07	1.845531e+06

Figure 6.1: Data Description

This gave an interesting insight from the data that the data is non-Gaussian in nature. Thus we need to transfrom the data

Data Transformation

Data transformation is a critical step in preparing data for machine learning models, especially for anomaly detection and forecasting in smart energy grids. The transformations aim to stabilize variance, normalize distributions, and handle skewness or multimodal distributions in the dataset. In this section, we describe three key transformation techniques applied to specific columns in the dataset: Box-Cox Transformation, Log Transformation, and Quantile Transformation.

Box-Cox Transformation: The Box-Cox transformation is used to make the data more normally distributed and stabilize variance, which improves the performance of various machine learning models. It was applied to several columns such as *'SS2FEEDER Units(Kwh)', 'Total Sentout'*, and others.

The following Python code implements the Box-Cox transformation:

```
for column_name in columns:
    if column_name in dm.columns:
        column_data = dm[column_name]
        min_value = column_data.min()
        if min_value <= 0:
            column_data = column_data + abs(min_value) + 1
        column_data_transformed, _ = boxcox(column_data)
        dm[column_name] = column_data_transformed
    else:
        print(f"Column '{column_name}' not found in Sheet{sheet_index}")
```

Explanation:

- A `for` loop iterates over the specified `columns` list.
- The check `if column_name in dm.columns` ensures that the transformation is only applied if the column exists in the dataset `dm`.
- The variable `column_data` holds the data of the respective column.
- Since Box-Cox requires positive values, the code shifts the data by adding `abs(min_value) + 1` if the minimum value is less than or equal to zero.
- The function `boxcox(column_data)` applies the Box-Cox transformation to stabilize variance and normalize the data.
- The transformed data is assigned back to the original column in the DataFrame `dm`.

This transformation is crucial for columns with non-normal distributions and high variance, as it improves model performance by ensuring better data distribution.

Log Transformation: The Log Transformation is used to handle skewness in data. This technique is applied to the *'Consumer Export(10)'* column to make the data more symmetrical.

```
if column_name in dm.columns:
    column_data = dm[column_name].values.reshape(-1, 1)
    column_data_log_transformed = np.log1p(column_data)
    dm[column_name] = column_data_log_transformed.flatten().tolist()
```

Explanation:

- The column *'Consumer Export(10)'* is first reshaped to a 2D array using `reshape(-1, 1)` to prepare it for transformation.
- The function `np.log1p(column_data)` applies a natural logarithm transformation with an added 1 to handle zero values (since the logarithm of zero is undefined).
- After transformation, the data is flattened back into a 1D list and updated in the DataFrame `dm`.

Log transformation is especially effective for reducing right-skewness, making the data more normally distributed and easier for machine learning models to process.

Quantile Transformation: The *'HT Sold(15)'* column has a multimodal distribution, which complicates the modeling process. To address this, a combination of log transformation, standardization, and quantile transformation is applied.

The process is encapsulated in the following function:

```
def transform_and_test(data):
    column_data = data.values

    if any(column_data <= 0):
        column_data = column_data + abs(column_data.min()) + 1

    column_data_log_transformed = np.log1p(column_data)

    scaler = StandardScaler()
    column_data_standardized =
```

```
    scaler.fit_transform(column_data_log_transformed.reshape(-1, 1))

    qt = QuantileTransformer(output_distribution='normal')
    column_data_transformed = qt.fit_transform(column_data_standardized)

    stat, p = shapiro(column_data_transformed)
    return column_data_transformed.flatten(), p
```

Explanation:

- The `transform_and_test` function first retrieves the column data (`column_data`).
- If any values in the data are less than or equal to zero, a constant shift is applied to ensure all values are positive.
- The data is log-transformed using `np.log1p` to handle skewness.
- The `StandardScaler` standardizes the data by scaling it to have a mean of 0 and a standard deviation of 1.
- The `QuantileTransformer` with `output_distribution='normal'` transforms the data into a normal distribution while preserving the relative ordering of data points.
- Finally, the Shapiro-Wilk test (`shapiro`) is performed to check the normality of the transformed data, returning the transformed data and a `p-value`.

This combination of transformations ensures that the column *'HT Sold(15)'*, which initially had a complex distribution, becomes normally distributed, improving the model's ability to detect anomalies or forecast values accurately.

The following code snippet applies this transformation to the *'HT Sold(15)'* column:

```
column_name = 'HT Sold(15)'
if column_name in dm.columns:
    column_data = dm[column_name]
    transformed_data, p_value = transform_and_test(column_data)
    dm[column_name] = transformed_data
```

This method allows the model to better handle multi modal and non-normal distributions, improving its accuracy and robustness.

By applying these transformations, the dataset is more suitable for machine learning algorithms, improving their performance and ensuring more accurate results in anomaly detection and forecasting tasks.

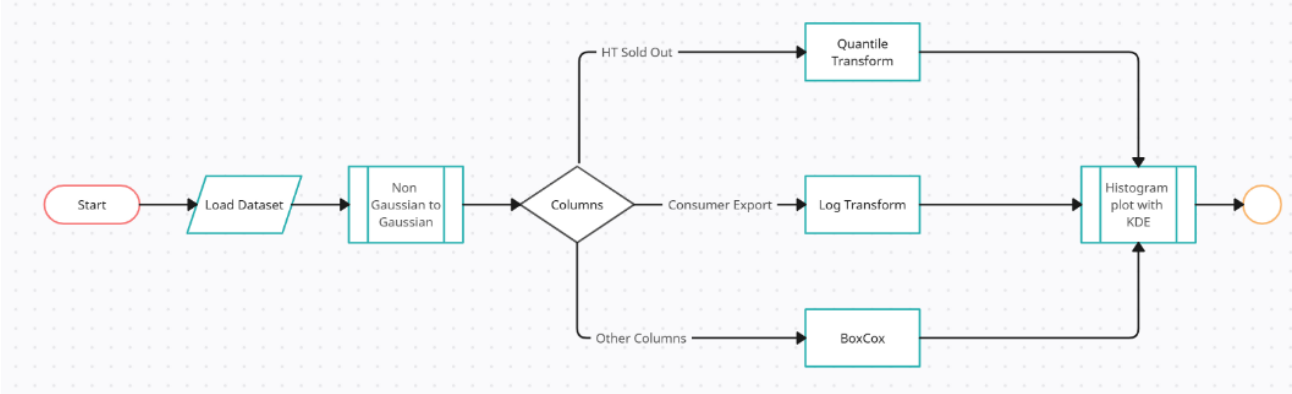

Figure 6.2: Data Transformation Schematic Diagram

Anomaly Detection

Supervised and unsupervised learning algorithms were trained on historical data to detect anomalies in real-time energy consumption patterns. Models were developed to classify normal and abnormal behavior, with a focus on identifying potential instances of electricity theft.

Anomaly detection was implemented using the Isolation Forest algorithm to identify anomalies in several columns, including *T&D Loss (%(21=20/11)*, *Total Sentout*, *LT Sold Out*, *HT Sold Out*, *Total Soldout*, *Unit Loss*, *SS2FEEDER*, *FEEDER2SS*, *Consumer Export*, *Total Billed*, *Consumer Imports*, and *Net Assessed*. The dataset consists of a total of 21 feeders. Given that *T&D Loss* is recorded in percentage format, its outlier detection is conducted by considering all the feeders at once, as percentages are independent of absolute values. For all other columns, anomaly detection is performed with respect to the *feeder's name* column since the values for each feeder must adhere to the normal distribution of its respective feeder.

Input:

- **X**: Data set with n instances and m features.
- **N**: Number of isolation trees.
- **sample_size**: Size of the subset to sample, such as $\frac{n}{\log n}$.

Output: Anomaly scores for all instances in X.

Isolation Forest Procedure: The Isolation Forest algorithm works by constructing multiple isolation trees on random subsets of the data, measuring the path length required to isolate each instance. The shorter the average path length, the more likely the instance is an anomaly.

```
Procedure BuildIsolationTree(X, depth):
    if size of X  1 or depth  max_depth:
        return a leaf node
    Choose a random feature f from the data
    Choose a random split value v for feature f
    Split X into two subsets: X_left and X_right based on value v
    Create a node with feature f and split value v
    Set node.left = BuildIsolationTree(X_left, depth + 1)
    Set node.right = BuildIsolationTree(X_right, depth + 1)
    return node

Procedure IsolationForest(X, N, sample_size):
    Initialize an empty list of trees
    for i = 1 to N:
        Sample a subset X' from X with size sample_size
        tree = BuildIsolationTree(X', 0)
        Append tree to the list of trees

    Initialize an empty list of anomaly_scores
    for each instance x in X:
        path_lengths = []
        for each tree in the list of trees:
            path_length = TraverseTree(tree, x)
            Append path_length to path_lengths
        average_path_length = Average(path_lengths)
        c_n = 2 * (log2(n) + 0.5772) - 2 * n / (n - 1)
        anomaly_score = 2^(-average_path_length / c_n)
        Append anomaly_score to anomaly_scores
```

```
    return anomaly_scores
```

Initialization of the Isolation Forest Model: In the implementation of the Isolation Forest algorithm, the following line of code is essential for initializing the model with specific parameters:

```
iso_forest = IsolationForest(n_estimators=7,
contamination=0.22, random_state=42)
```

The `IsolationForest` class, sourced from the `Scikit-learn` library, is used to create an instance of the Isolation Forest algorithm designed for anomaly detection. The parameters are as follows:

- **n_estimators=7**: This parameter specifies that the model will construct 7 isolation trees. The choice of 7 trees represents a balance between computational efficiency and robustness of the model. A higher number of trees can increase the ability to detect anomalies but at a higher computational cost.
- **contamination=0.22**: This parameter denotes the expected proportion of outliers within the dataset. Setting the contamination rate to 22% means the model is configured to anticipate that approximately 22% of the data points are anomalies, which helps the model classify anomalies more accurately.
- **random_state=42**: Setting a fixed random state ensures reproducible results, which is crucial for model validation and debugging. By fixing the random state to 42, the randomness in selecting features and split values during tree construction remains controlled, yielding consistent results across runs.

Anomaly Score Calculation: The anomaly score for an instance x is calculated using the formula:

$$AnomalyScorex = 2^{-\frac{AveragePathLengthx}{cn}}$$

where $c(n)$ is a normalization factor that accounts for the number of samples n and is defined as:

$$cn = 2\log_2 n + 0.5772 - \frac{2n}{n-1}$$

This score reflects how isolated an instance is. Instances with shorter average path lengths through the isolation trees are assigned higher anomaly scores, as they are considered more likely to be anomalies.

Hyperparameter Tuning: To find the optimal hyperparameters, `GridSearchCV` was employed, allowing the model to automatically search for the best parameter combinations for *n_estimators* and *contamination*.

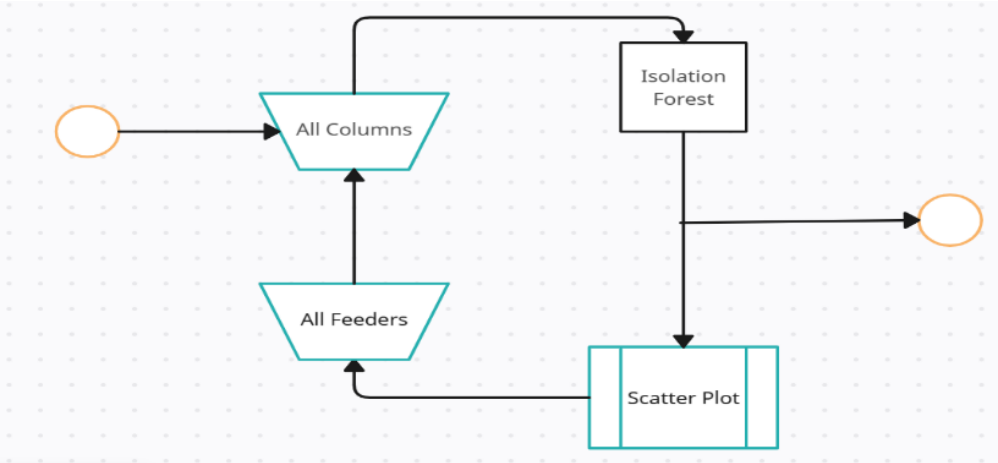

Figure 6.3: Anomaly Detection Schematic Diagram

Comparison of Anomaly Detection Methods: In addition to the Isolation Forest algorithm, other anomaly detection methods were explored, including:

- OneClass SVM
- Isolation Forest
- Local Outlier Factor (LOF)
- Interquartile Range (IQR)
- Isolation Forest (applied again in a different approach)

After evaluation, the Isolation Forest algorithm outperformed the other methods across all columns. For the *T&D Loss* column, since the values are percentages, anomaly detection was applied to the entire dataset, considering all feeders collectively. However, for other columns, anomaly detection was performed individually per feeder, as each feeder's values should conform to the normal distribution of that particular feeder.

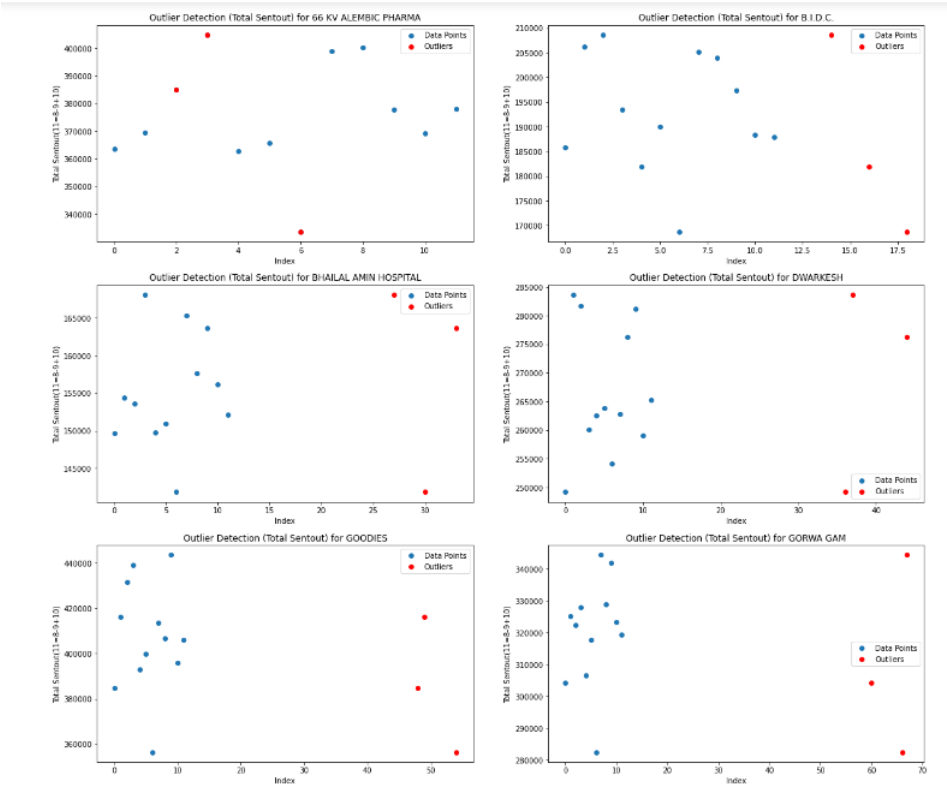

Figure 6.4: Anomaly Detection Scatter Plot 1

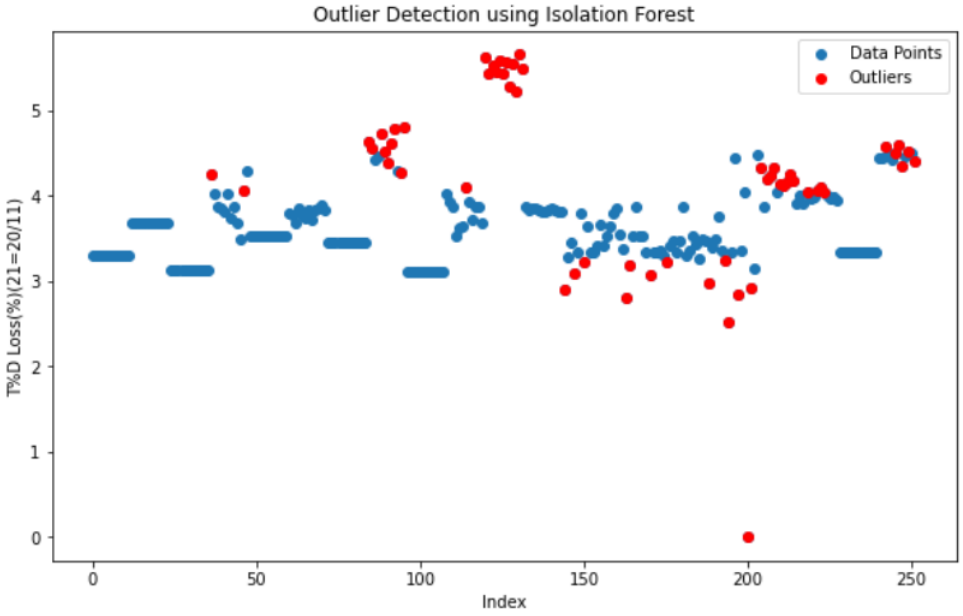

Figure 6.5: Anomaly Detection Scatter Plot 2

Label Encoding

To facilitate model training and improve computational efficiency, categorical variables were encoded into numerical values. This transformation was necessary as machine learning models typically require numerical inputs rather than categorical data. The encoding process was performed

using the `LabelEncoder` from the `sklearn.preprocessing` module.

Month Encoding: The *Month* column, which originally contained categorical data representing months (e.g., January, February), was transformed into numerical format. Each unique month or pair of months was assigned a corresponding integer value according to the following mapping:

- January-February: 0
- July-August: 1
- March-April: 2
- May-June: 3
- November-December: 4
- September-October: 5

Feeder Name Encoding: Similarly, the *Feeder Name* column, which originally contained categorical feeder names, was encoded into integers. The feeder names were mapped to integer values as follows:

- 66 KV ALEMBIC PHARMA: 0
- B.I.D.C.: 1
- BHAILAL AMIN HOSPITAL: 2
- DWARKESH: 3
- GOODIES: 4
- GORWA GAM: 5
- GORWAS SST: 6
- GUJ.HOUSING BOARD: 7
- INORBIL MALL: 8
- KARODIYA ROAD: 9
- LABH: 10
- PANCHAVATI: 11

- SAHYOG ROAD: 12
- SAMTA: 13
- SANGRILA: 14
- SHREENATHJI: 15
- SUBHANPIRA SST: 16
- SUBHANPURA ROAD: 17
- SWAMI VIVEKANAND: 18
- TCD LTD: 19
- VADHUNAGAR: 20

Purpose of Encoding: The transformation of the *Month* and *Feeder Name* columns into numerical values ensures that categorical variables are in a format compatible with machine learning algorithms. This encoding process preserves the categorical distinctions between different values while enabling the model to process and learn from the data more efficiently.

Model Training and Validation for Forecasting

Machine learning models were trained using labeled data to predict electricity theft and flag suspicious activities. Training involved feature selection, model fitting, hyperparameter tuning, and cross-validation to ensure robustness and generalization.

The dataset is prepared and utilized to train and evaluate a Random Forest regression model to predict three key target variables: *Total Sentout*, *Total Billed*, and *Unit Loss*. The process begins by extracting and preparing the features and target variables from the dataset. Features such as the *Month* and *Feeder Name* columns are combined into a single feature matrix X using `np.column_stack`, while the target variable y is selected as the *Total Sentout* column.

Polynomial Feature Expansion: To enhance the predictive capabilities of the model, polynomial feature expansion is applied. The `PolynomialFeatures` class from `Scikit-learn` is used with a degree of 2, allowing for the generation of interaction terms and polynomial features up to the specified degree. This transformation is performed on the feature matrix X to create a new feature matrix *Xpoly*, which includes both the original features and their polynomial interactions.

Data Splitting and Scaling: The dataset is split into training and testing subsets using `train_test_split`, with 20% of the data allocated to the test set. To ensure consistent feature scaling, the feature matrix and target variable are standardized using `StandardScaler`. This step is crucial because models like Random Forest are sensitive to feature scales. Both the training and testing datasets are scaled separately, ensuring that the model is evaluated on a consistent scale.

Hyperparameter Tuning: Hyperparameter tuning for the Random Forest model is conducted using `GridSearchCV`. This method performs an exhaustive search over a specified parameter grid to identify the best combination of hyperparameters for the model. The grid of hyperparameters includes:

- `n_estimators`: Number of trees in the forest (100, 200, 300).
- `max_depth`: Maximum depth of the trees (None, 10, 20, 30).
- `min_samples_split`: Minimum number of samples required to split an internal node (2, 5, 10).
- `min_samples_leaf`: Minimum number of samples required to be at a leaf node (1, 2, 4).

The `GridSearchCV` function evaluates each combination of parameters using 5-fold cross-validation, optimizing for the R^2 score. The best model, identified by `grid_search_rf.best_estimator_`, is used to make predictions on both the training and testing data. The predictions are inverse transformed to their original scale using the scaler applied to the target variable.

Performance Evaluation: The performance of the model is assessed using mean squared error (MSE) and R^2 score metrics. Additionally, the performance is visualized using 3D scatter plots. These plots compare the actual and predicted values of the training and testing datasets. The first subplot displays the training data points and corresponding predictions, while the second subplot shows the testing data. These visualizations help in understanding how well the model captures the relationship between the input features and the target variable.

Feature Importance: To analyze the significance of each feature in the Random Forest model, feature importances are computed using the `feature_importances_` attribute of the best model, `best_rf_ts`. These importances indicate the contribution of each feature to the prediction of the target variable.

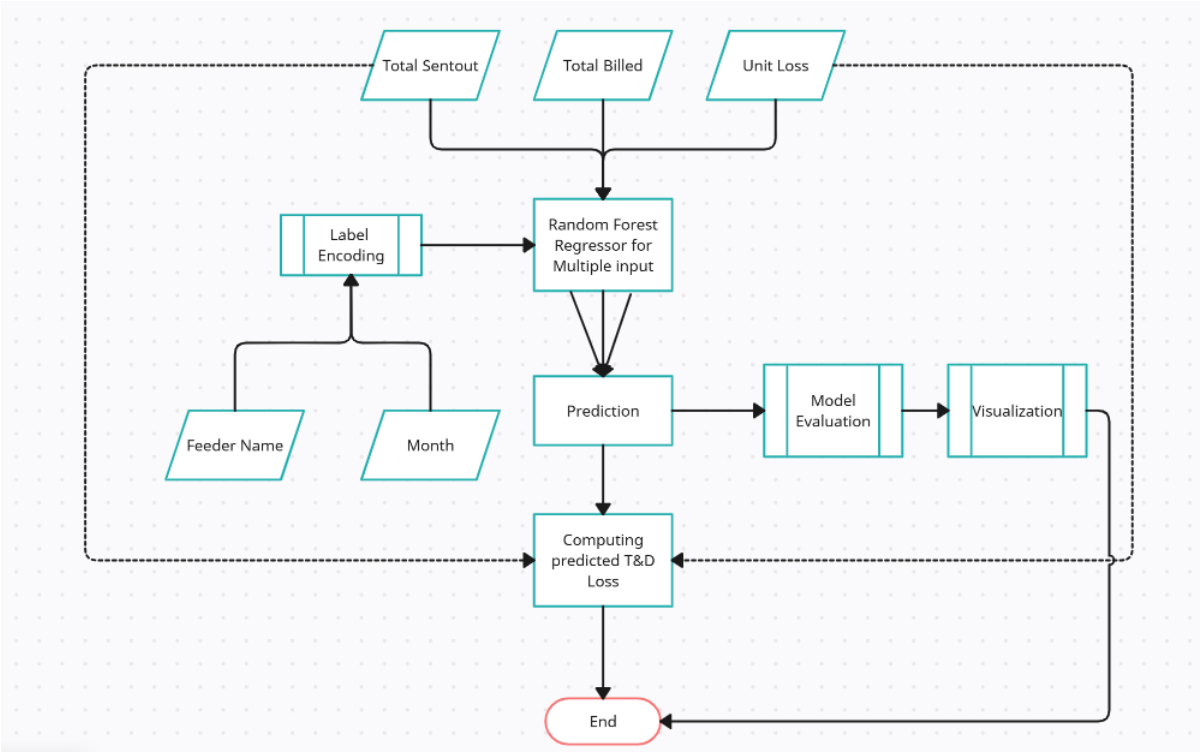

Figure 6.6: Forecasting Schematic Diagram

Model Summary and Equations: The Random Forest Regressor aggregates the predictions from multiple decision trees. For each decision tree T_k in the forest, the prediction $\hat{y}_k$ is made using the two input variables X_1 and X_2, as shown in Equation (3):

$$\hat{y}_k = T_k X_1, X_2 \tag{3}$$

The Random Forest model then averages the predictions from all N decision trees to produce the final prediction $\hat{y}$, as represented in Equation (4):

$$\hat{y} = \frac{1}{N} \sum_{k=1}^{N} \hat{y}_k \tag{4}$$

In general, for a Random Forest with N decision trees, the final prediction $\hat{y}$ for input features X_1 and X_2 is given by Equation (5):

$$\hat{y} = \frac{1}{N} \sum_{k=1}^{N} T_k X_1, X_2 \tag{5}$$

Summary of Trained Models: Three Random Forest regression models were trained to predict the *Total Sentout*, *Total Billed*, and *Unit Loss*. Each model uses `PolynomialFeatures` and `GridSearchCV` for hyperparameter tuning. The hyperparameters of the best models are summarized

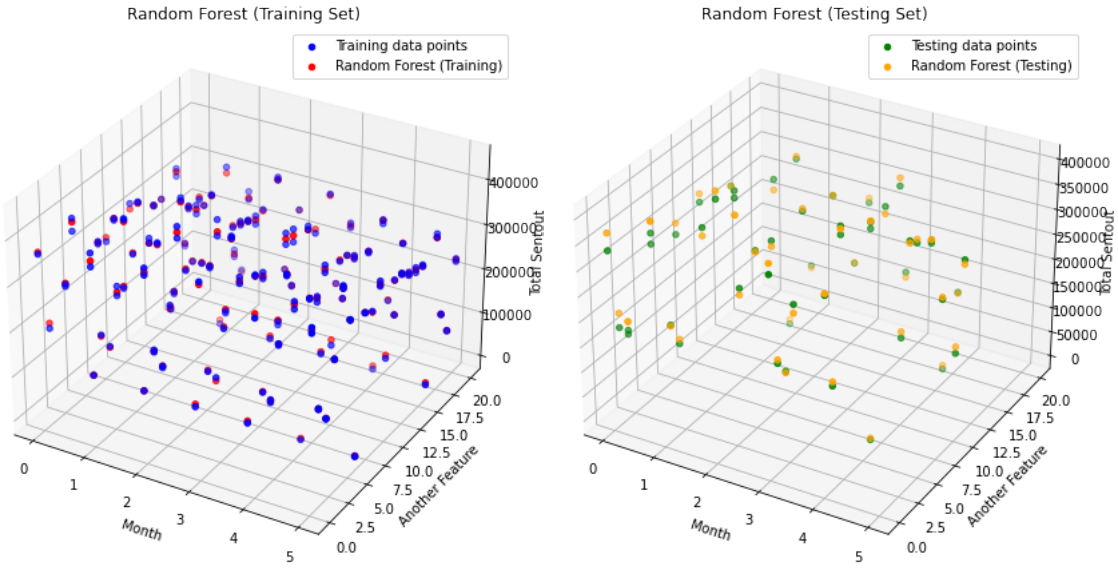

Figure 6.7: Model Evaluation Scatter Plot

as follows:

- **best_rf_ts (Total Sentout)**:
  - `max_depth`: None
  - `min_samples_leaf`: 1
  - `min_samples_split`: 5
  - `n_estimators`: 200
- **best_rf_tb (Total Billed)**:
  - `max_depth`: None
  - `min_samples_leaf`: 1
  - `min_samples_split`: 2
  - `n_estimators`: 100
- **best_rf_ul (Unit Loss)**:
  - `max_depth`: None
  - `min_samples_leaf`: 2
  - `min_samples_split`: 5
  - `n_estimators`: 200

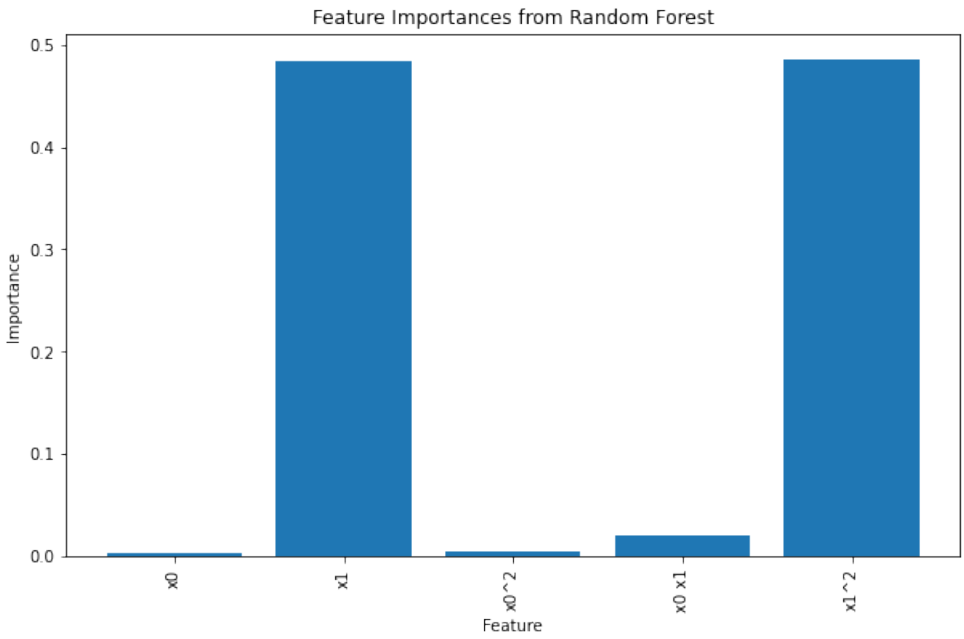

Figure 6.8: Feature Importance Bar Plot

As Random Forest is an ensemble method based on bagging and consists of multiple Decision Trees, which are non-parametric models that use recursive splitting, the hyperparameters may vary between runs. For this reason, `GridSearchCV` is kept in the production code to ensure that the model always uses the optimal hyperparameters.

6.1.3 System Integration

Power BI Integration

Power BI is an essential component of the proposed system, providing a user-friendly platform for visualizing both historical and real-time data. The Power BI dashboard is designed with two distinct levels, each catering to a different aspect of the system's analytics: anomaly detection and forecasting (Level 1), and real-time data monitoring for theft detection (Level 2). These dashboards provide an interactive interface to analyze and interpret data, improving decision-making and grid monitoring processes.

Level 1: Anomaly Detection and Forecasting The first level of the Power BI dashboard focuses on historical data analysis for anomaly detection and forecasting. This level provides detailed insights into the grid's performance over time, allowing the user to identify patterns of irregularities and assess the predictive capabilities of the implemented forecasting models.

- **Anomaly Detection Overview:** This component offers visual representations, such as charts and graphs, summarizing anomalies detected in the dataset. Users can explore the frequency,

distribution, and nature of these anomalies, making it easier to identify any recurring patterns or trends.

- **Forecasted Values:** In addition to anomaly detection, this dashboard displays forecasted values against historical data, typically shown through line or area charts. This comparison provides a clear view of how well the forecasting models perform in predicting future values, helping to identify deviations from expected outcomes.

Level 2: Real-Time Data for Theft Detection The second level of the dashboard is dedicated to real-time data monitoring, particularly for theft detection. This level is designed to enable continuous surveillance and immediate response to potential theft incidents.

- **Real-Time Monitoring Dashboard:** Live data feeds are integrated into this component, offering real-time visualizations like dynamic charts, maps, and alerts. This dashboard provides users with immediate insights into any suspicious activities as they happen, allowing for prompt investigation and mitigation.
- **Theft Alerts:** Real-time alerts are generated based on predefined rules and data analytics. These alerts are critical for detecting unusual patterns in the grid's performance that may indicate theft. The alerts enable the users to act quickly and decisively.
- **Historical Context and Trends:** This section provides a historical comparison of theft-related data. Users can compare real-time incidents with past trends to identify whether current events are part of a broader pattern or isolated cases. Understanding historical trends enhances the ability to detect real-time anomalies effectively.

The combination of historical analysis and real-time monitoring in Power BI provides a holistic view of the energy grid's performance, enabling proactive measures for anomaly detection and efficient theft detection mechanisms.

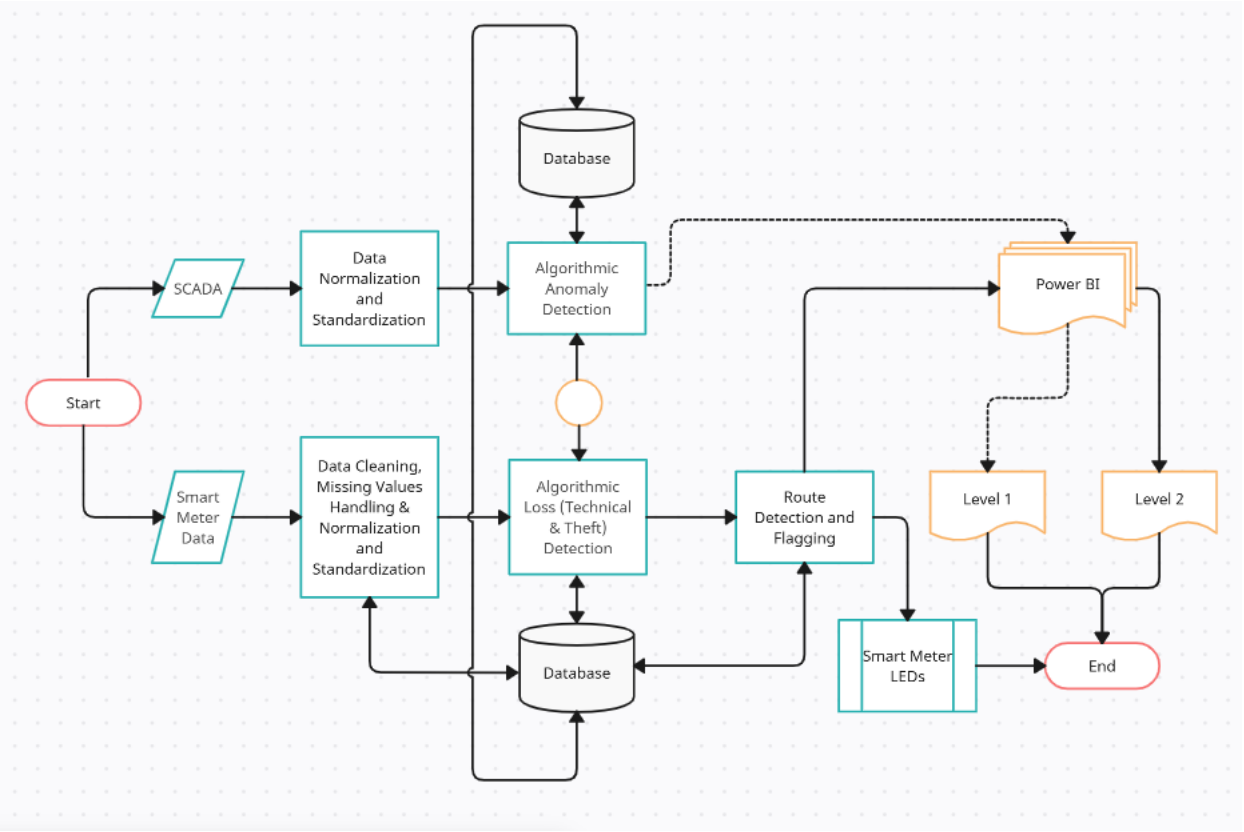

Figure 6.9: PowerBI Pipeline

Power BI Visualization

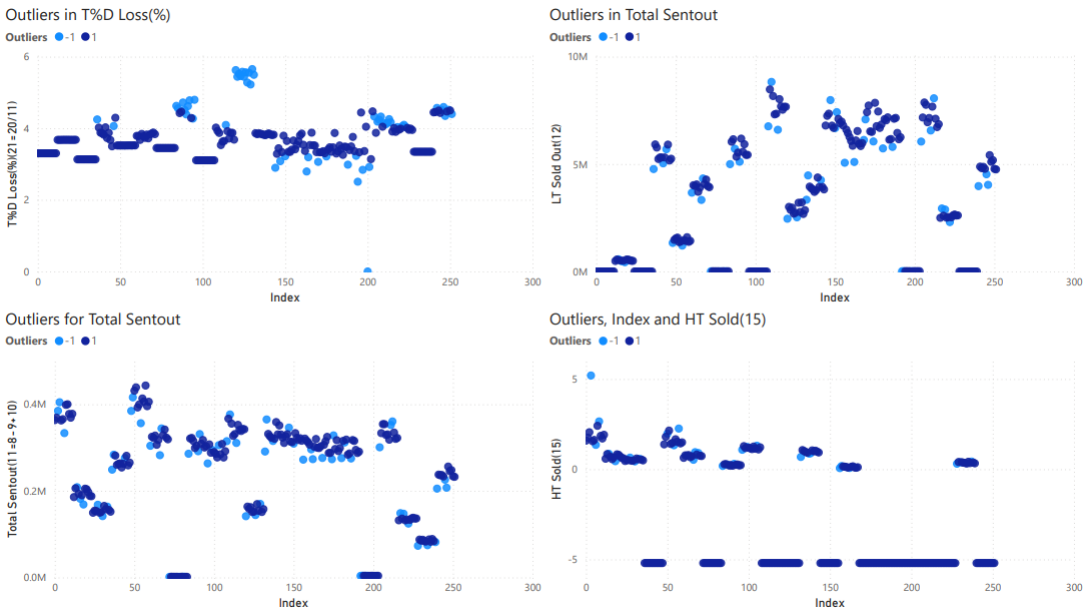

Figure 6.10: Outlier Detection in PowerBI

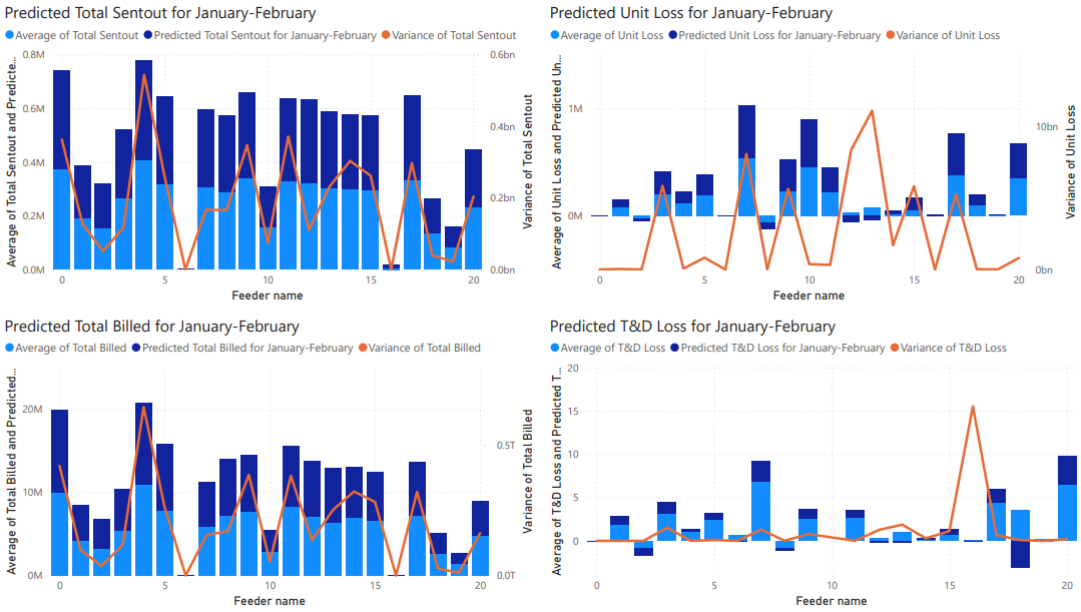

Figure 6.11: Forecasting in PowerBI

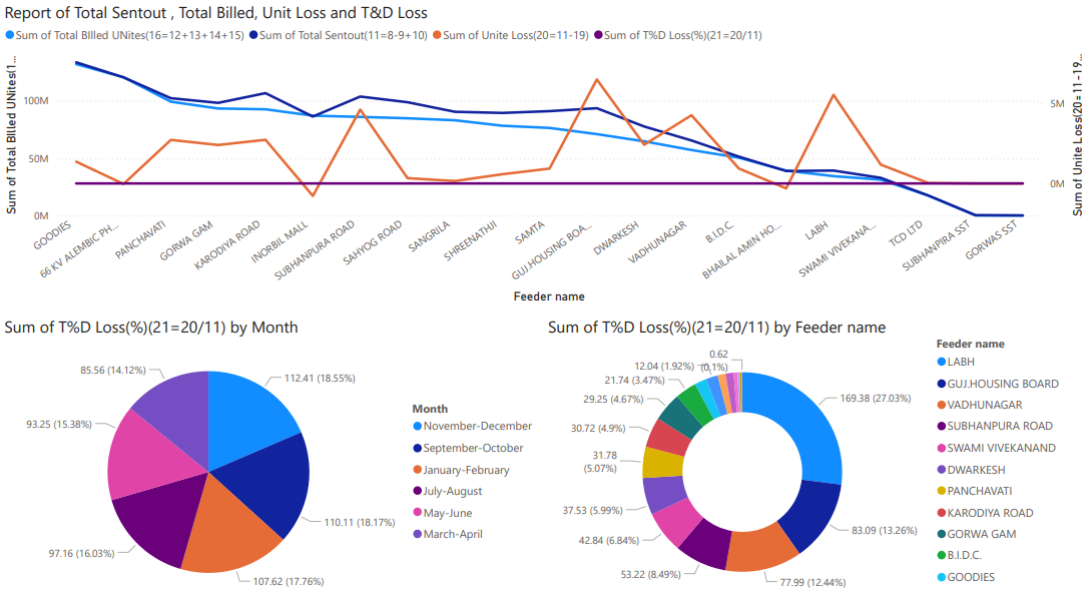

Figure 6.12: Summary Report in PowerBI 1

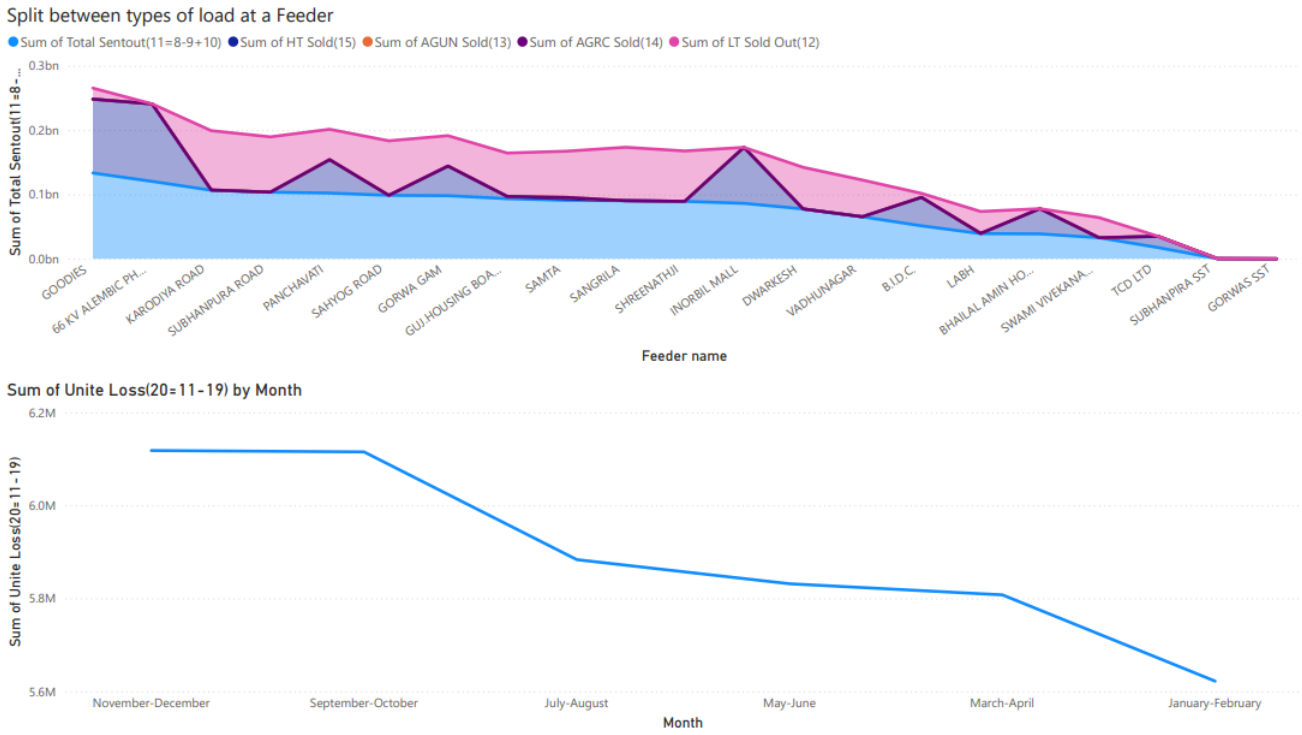

Figure 6.13: Summary Report in PowerBI 2

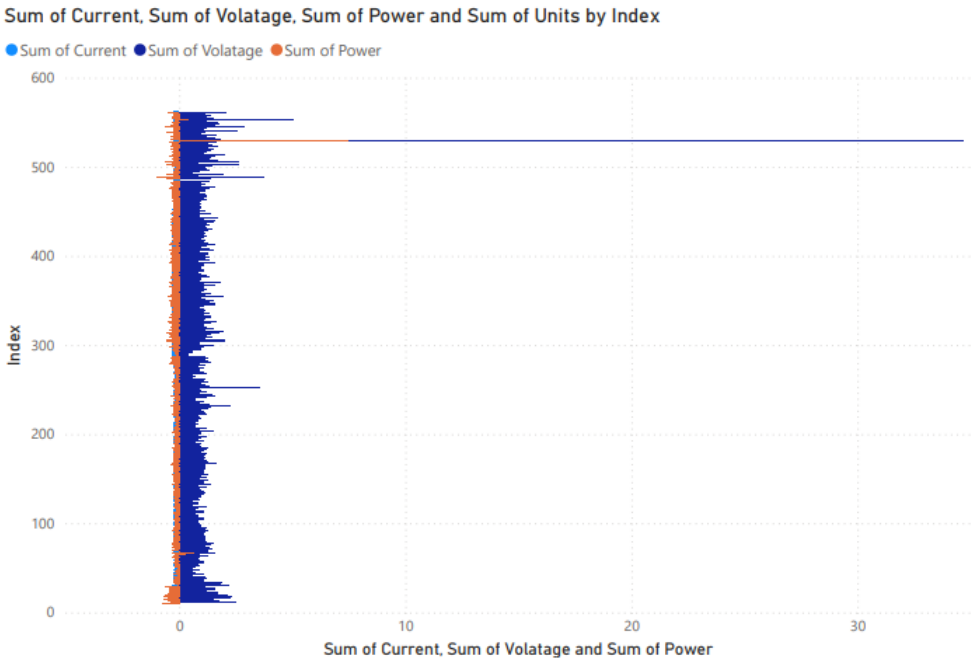

Figure 6.14: Smart Meter Real Time data in PowerBI

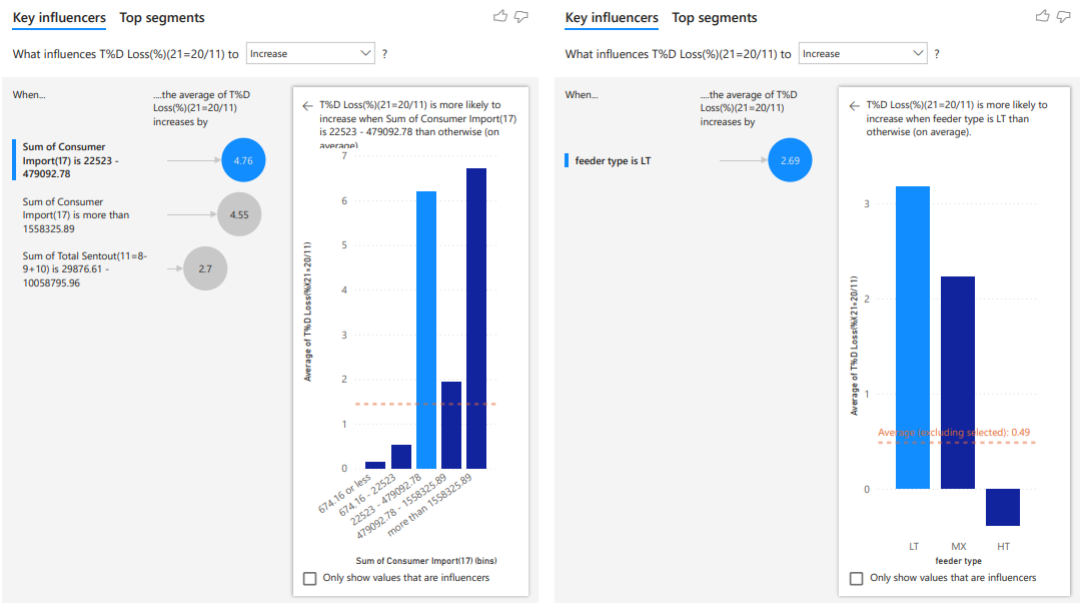

Figure 6.15: Key Indicators in PowerBI

following Image shows PowerBI Interface

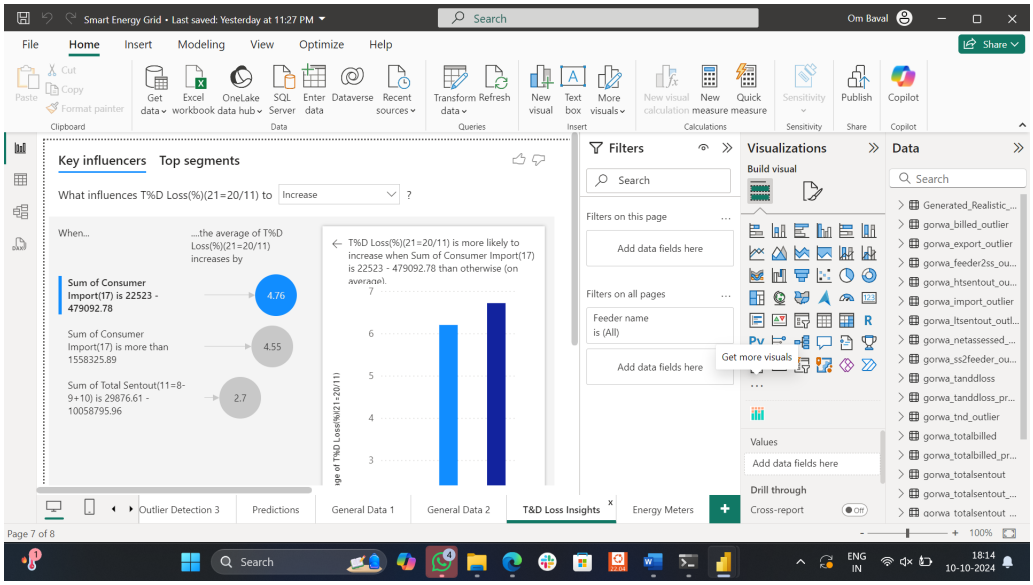

Figure 6.16: Smart Meter Real Time data in PowerBI

Smart Meters

Data streams from sensors, meters, and external databases were integrated into the Smart Grid System's data pipeline. Real-time data processing and analysis enabled timely detection of anomalies and proactive response to potential theft.

The IoT-based smart meter system is an innovative solution developed to enable real-time monitoring of electrical parameters. The system is built around the ESP32 microcontroller, coupled

with the ZMTP101B AC Voltage Sensor and ACS712 Current Sensor, allowing it to measure, analyze, and report on key electrical variables such as voltage and current. This real-time monitoring capability improves the management of electrical systems by identifying potential inefficiencies, technical losses, and even theft.

ESP32 Microcontroller: Central Processing Unit At the core of the system is the ESP32 microcontroller, which is responsible for data acquisition, processing, and communication. The ESP32 manages interactions with the sensors and communicates with a remote server for further data analysis. It is well-suited for this application due to its dual-core processing, low power consumption, and integrated Wi-Fi and Bluetooth capabilities, which facilitate seamless data transmission to the cloud or server. Its reliability and efficiency make it an ideal choice for IoT-based monitoring systems.

Sensors for Voltage and Current Measurement The system uses two key sensors for real-time data collection:

- **ZMTP101B AC Voltage Sensor:** This sensor measures the alternating current (AC) voltage in the electrical system. It provides precise voltage readings, allowing the system to detect voltage fluctuations, surges, or undervoltage conditions. These measurements are essential for assessing the stability and health of the electrical infrastructure.
- **ACS712 Current Sensor:** The ACS712 sensor is designed to measure the amount of current flowing through the system. By providing real-time current readings, it helps detect overloading, abnormal power consumption, or other irregularities in the electrical system. Monitoring current flow is critical for detecting technical losses or energy theft.
- **ACS712 Current Sensor:** The ACS712 sensor is designed to measure the amount of current flowing through the system. By providing real-time current readings, it helps detect overloading, abnormal power consumption, or other irregularities in the electrical system. Monitoring current flow is critical for detecting technical losses or energy theft.
- **ACS712 Current Sensor:** The ACS712 sensor is designed to measure the amount of current flowing through the system. By providing real-time current readings, it helps detect overloading, abnormal power consumption, or other irregularities in the electrical system. Monitoring current flow is critical for detecting technical losses or energy theft.
- **Orange LED:** The orange LED is activated when the server detects a potential technical

loss in the system. Technical losses could be due to inefficiencies, minor malfunctions, or degraded system performance. This serves as an early warning for users to investigate the system and potentially perform preventive maintenance.

- **Red LED:** The red LED is a critical indicator that lights up when potential theft is detected. It signals an immediate alert, prompting users to investigate for unauthorized usage or energy theft. This visual alert ensures that theft is detected and addressed quickly, minimizing potential losses.

These LEDs provide a clear and concise way for users to understand the system's status at a glance, reducing the need for continuous monitoring of detailed data streams.

Real-Time Data and Response Mechanism The IoT-based smart meter system offers real-time data monitoring, allowing for immediate detection of any issues related to the electrical system. Voltage and current data are continuously collected, processed, and sent to the server, ensuring that any anomalies or abnormal patterns are quickly identified. By providing real-time feedback through the LEDs, the system enables prompt responses to any detected issues, enhancing the overall reliability and security of the electrical infrastructure.

The real-time nature of the system also allows users to act swiftly in case of technical losses or theft, minimizing downtime and reducing the financial impact of energy inefficiencies or unauthorized usage.

System Advantages and Benefits The IoT-based smart meter system offers numerous advantages for energy management:

- **Enhanced Monitoring:** By continuously monitoring key electrical parameters, the system enables early detection of potential issues, improving overall system efficiency.
- **Increased Security:** The ability to detect potential theft in real-time ensures that unauthorized usage is quickly identified and addressed, reducing financial losses.
- **Proactive Maintenance:** With the orange LED alerting users to technical losses, preventive maintenance can be scheduled before minor issues turn into major problems, thereby extending the lifespan of the electrical system.
- **Fast and Effective Response:** The system's real-time feedback and alert mechanisms allow for immediate action when anomalies are detected, minimizing the risk of damage or prolonged inefficiencies.

Conclusion The IoT-based smart meter system is a powerful tool for monitoring and managing electrical infrastructure in real-time. Its integration of high-precision sensors, a reliable ESP32 microcontroller, and immediate visual feedback via LEDs makes it an effective solution for detecting anomalies, inefficiencies, and theft in the electrical grid. This system not only improves operational efficiency but also enhances the security and resilience of electrical systems by providing real-time insights and alerts.

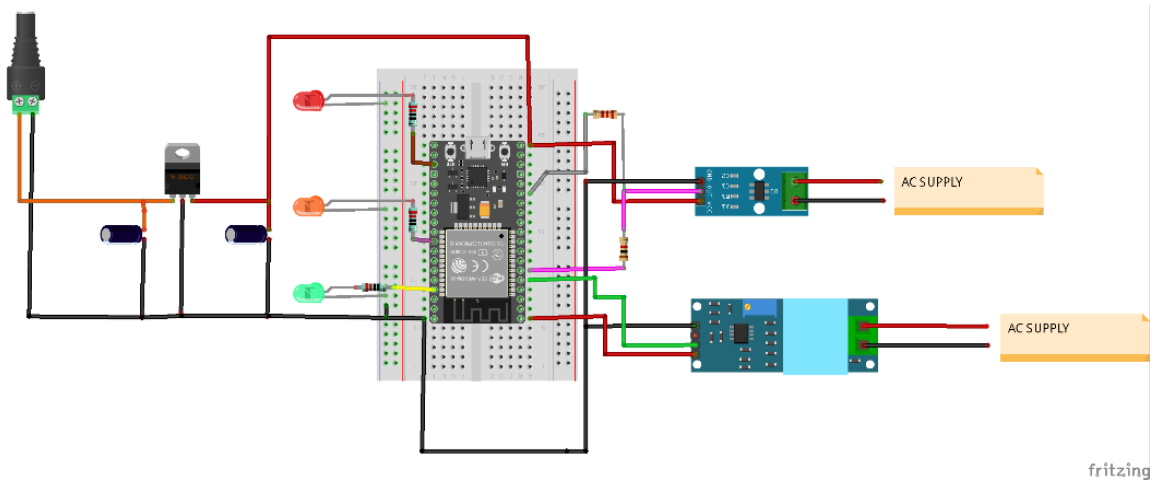

Figure 6.17: Smart Meter Circuit Diagram

User Interface Integration

User interfaces were integrated with backend modules to provide seamless interaction with the Smart Grid System. APIs were developed to enable data retrieval, visualization, and user authentication, ensuring a smooth user experience.

6.1.4 Testing and Validation

Unit Testing

Individual components of the Smart Grid System were subjected to unit tests to verify their functionality, correctness, and reliability. Test cases covered edge cases, error handling, and boundary conditions to ensure robustness.

Smart Meter Real-time Data Analysis

Smart Meter is directly connected with Internet that send the data to Google Sheets in real-time via Google Cloud Platform (GCP). This data is being shared using Hyper Text Transfer Protocol (HTTP). This data is being set real-time from Google Sheets to PowerBI and Python Script for analysis purposes

The diagram below show an example Google Sheet that records Timestamp, Voltage, Current, Power and Units

	SS2FEEDER Unites(Kwh) (8)	Feeder2SS UNites(Kwh) (9)	Consumer Export(10)	Total Sentout(11=8-9+10)	LT Sold Out(12)	AGUN Sold(13)	AGRC Sold(14)	HT Sold(15)	Total BIlled UNites(16=12+13+14+15)	Consumer Import(17)
count	2.520000e+02	252.000000	2.520000e+02	2.520000e+02	2.520000e+02	252.000000	252.0	2.520000e+02	2.520000e+02	2.520000e+02
mean	5.522478e+06	10857.149286	5.880852e+05	6.099706e+06	3.446166e+06	12537.901865	0.0	2.090637e+06	5.549341e+06	5.523534e+05
std	2.965974e+06	20947.764451	6.625629e+05	3.175217e+06	2.855925e+06	41101.495940	0.0	3.200988e+06	2.966992e+06	6.191667e+05
min	8.565760e+03	0.000000	0.000000e+00	8.565760e+03	0.000000e+00	0.000000	0.0	0.000000e+00	8.508780e+03	0.000000e+00
25%	3.180532e+06	0.000000	2.974025e+03	3.305255e+06	1.221792e+04	0.000000	0.0	0.000000e+00	3.205786e+06	6.247375e+02
50%	5.965799e+06	41.075000	2.507366e+05	7.136809e+06	3.800001e+06	0.000000	0.0	0.000000e+00	6.188594e+06	1.503791e+05
75%	7.399542e+06	10670.710000	1.280190e+06	8.258045e+06	6.082788e+06	0.000000	0.0	3.627804e+06	7.426972e+06	1.161949e+06
max	1.237736e+07	89520.180000	2.015045e+06	1.240046e+07	8.827297e+06	208136.120000	0.0	1.103764e+07	1.226140e+07	1.845531e+06

Figure 6.18: Google Sheet

The analysis of smart meter data is essential for understanding and optimizing energy consumption patterns, particularly in large datasets collected over extended periods. The dataset obtained from the MGVCL Gorwa Sub Station includes features like energy consumption (kWh), voltage, current, and power factor, with timestamps. These features are crucial for modeling and analysis, as they offer insights into how energy is consumed over time and under different conditions.

Data Preprocessing Preprocessing is a critical first step in the data analytics pipeline. Missing values were handled by applying forward or backward filling methods, which allowed for continuous time-series data. Outliers were another major concern, as smart meter readings can be subject to anomalies due to sensor malfunctions or external factors (e.g., power surges). To address this, Isolation Forest was employed as an effective method for outlier detection.

Why Isolation Forest? Isolation Forest is an unsupervised learning algorithm specifically designed to identify anomalies in data. It works by randomly partitioning the dataset and isolating points that are farthest from the majority of the data. This method is particularly well-suited for high-dimensional data like the one used in this analysis, where outliers are often multi-featured and not easily detectable through simple statistical methods.

The decision to use Isolation Forest was motivated by the following reasons:

- It operates efficiently on large datasets, making it ideal for the volume of smart meter data collected over long periods.
- Unlike traditional outlier detection methods, Isolation Forest does not rely on distance measures, making it robust to different data scales (such as voltage and current).

- It effectively isolates anomalies without assuming any specific distribution, which is crucial given the variety of factors that could cause unusual readings in energy consumption.

The implementation of Isolation Forest in Python was as follows:

```
from sklearn.ensemble import IsolationForest

# Isolation Forest to detect outliers
iso_forest = IsolationForest(contamination=0.01, random_state=42)
data['outlier_flag'] = iso_forest.fit_predict(data[['energy_consumption', 'voltage'

# Remove detected outliers
data_cleaned = data[data['outlier_flag'] == 1]
```

In this context, a contamination rate of 1% was chosen, meaning that 1% of the data was flagged as anomalous. These outliers were subsequently removed from the dataset to prevent them from skewing the results of the subsequent analysis.

Data Transformation After cleaning the data, it was necessary to transform the energy consumption values to follow a Gaussian distribution. Most machine learning models assume normality in the data, which ensures better performance and interpretability of the model outputs. The Box-Cox transformation was employed for this purpose, as it allows for automatic selection of the best transformation parameter (λ).

The transformation was particularly important because energy consumption data, as collected, often exhibits skewness due to sudden spikes in usage or extended periods of low usage. The transformed data improved the overall performance of the predictive models, as discussed in the subsequent sections.

Visualization To understand the effects of the cleaning and transformation steps, visualizations such as time-series plots and histograms were employed. The comparison between the original and transformed data was essential in verifying that the data was now more suitable for model fitting.

Before cleaning and transformation, the histograms of energy consumption showed significant skewness, while the time-series plots displayed frequent spikes and irregularities. After the application of Isolation Forest and Box-Cox transformation, the histograms became more normally distributed, and the time-series data showed smoother trends, making it easier to model.

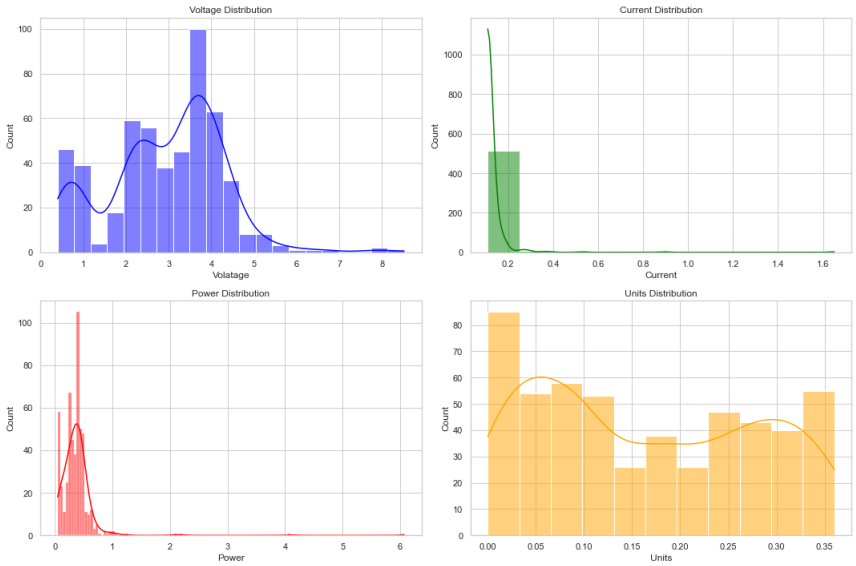

Figure 6.19: Histograms for distribution of Voltage, Current, Power and Unit

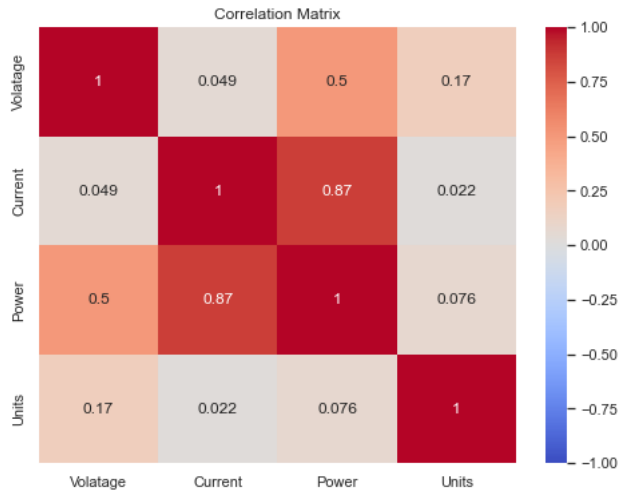

Figure 6.20: Corelation Matrix of Voltage, Current, Power and Unit

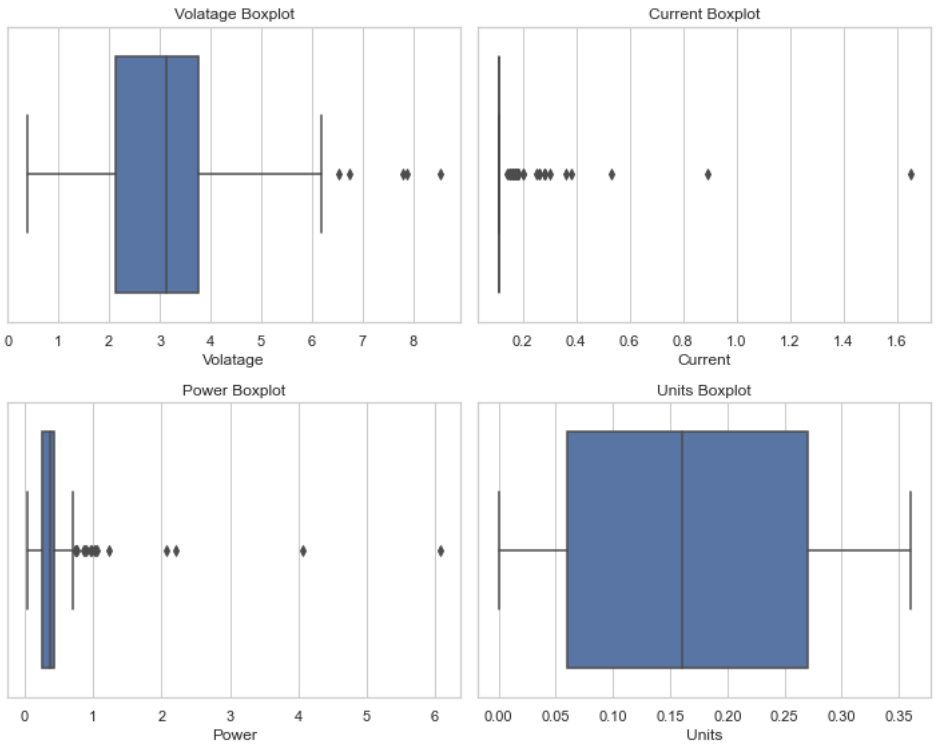

Figure 6.21: Box Plot for Anomaly Detection of Voltage, Current, Power and Unit

Modeling Once the data was cleaned and transformed, various predictive models were applied, such as linear regression and time-series forecasting models. These models allowed for the prediction of energy consumption based on historical data and other relevant features (voltage, current, power factor).

Performance metrics such as R-squared (R^2) and Mean Squared Error (MSE) were calculated to evaluate the effectiveness of each model. The transformed data improved the accuracy of predictions by allowing the models to better capture the underlying patterns in the energy consumption data.

Given that we are dealing with an energy meter system, where power fluctuations or abnormal usage patterns could signal anomalies like energy theft or technical losses, we recommend using Isolation Forest. This algorithm is well-suited for:

- Complex non-linear data like energy consumption patterns.
- Real-time or large datasets.
- Cases where outliers are important, such as detecting abnormal energy consumption, which could indicate theft.

Explanation of Key Choices

Isolation Forest: Isolation Forest was chosen as the outlier detection method due to its efficiency and suitability for high-dimensional datasets, such as the one in this analysis. It operates by isolating points that are far from the majority of the data, making it a robust method for detecting anomalies that cannot be easily captured by traditional statistical methods. Unlike distance-based methods, Isolation Forest is effective even when the data features have different scales, as is the case with voltage, current, and energy consumption values. Its low computational cost makes it particularly advantageous for processing large volumes of smart meter data.

Box-Cox Transformation: The Box-Cox transformation was applied to convert the energy consumption data into a Gaussian distribution, which is necessary for many predictive models to perform optimally. The energy consumption data originally exhibited skewness, which can lead to poor model performance. By using the Box-Cox transformation, the data became more normally distributed, allowing for improved model accuracy and better interpretability of the results.

Visualizations: Visualization techniques, such as time-series plots and histograms, were employed to visually assess the effectiveness of the data preprocessing steps. The histograms of the energy consumption data, both before and after transformation, provided insights into the success of the Box-Cox transformation in normalizing the data. Time-series plots showed the reduction of noise and outliers, making it easier to detect underlying consumption patterns, which is critical for building accurate predictive models.

6.1.5 Deployment

Upon successful testing and validation, the Smart Grid System was deployed in the Gorwa Sub Division of MGVCL. Deployment involved setting up infrastructure, configuring servers, and installing necessary software components. System administrators were trained to manage and maintain the deployed system effectively.

6.1.6 Maintenance and Monitoring

Post-deployment, the Smart Grid System underwent continuous monitoring and maintenance to ensure its optimal performance, security, and reliability. Monitoring tools were employed to track system health, detect anomalies, and generate performance metrics. Regular updates, patches, and security enhancements were applied to address emerging threats and vulnerabilities.

6.1.7 Evaluation and Optimization

The deployed Smart Grid System was evaluated periodically to assess its impact on electricity theft detection, grid security, and energy optimization. Key performance indicators (KPIs) such as

theft detection rate, false positive rate, and system uptime were monitored to gauge the system's effectiveness. Optimization efforts focused on improving model accuracy, reducing false alarms, and enhancing user experience based on feedback and lessons learned from real-world usage.

By following this comprehensive implementation plan, the Smart Grid System successfully addressed the challenges of electricity theft detection, data security, and grid optimization in the Gorwa Sub Division of MGVCL. The integration of Blockchain technology, advanced data analysis techniques, and user-friendly interfaces contributed to a more secure, efficient, and reliable electricity distribution network.

Chapter 7

Testing

Testing is a critical phase in the development and validation of predictive models, particularly in the context of smart meter data analytics. This chapter focuses on the testing procedures used to evaluate the performance of the models developed during this study. The dataset was split into training and testing subsets, and the performance of each model was assessed using metrics such as R-squared (R^2) and Mean Squared Error (MSE). Additionally, cross-validation techniques were employed to ensure the robustness of the results.

7.1 Training and Testing Split

The dataset was divided into training and testing sets to evaluate the model's ability to generalize to unseen data. The typical approach of an 80-20 split was used, where 80% of the data was used for training the models and the remaining 20% for testing. This division ensured that the models were not overfitted to the training data and could be tested on a separate, unseen portion of the dataset.

The following Python code illustrates the process of splitting the data:

```
from sklearn.model_selection import train_test_split


# Split the data into training and testing sets
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2, random_state=42)
```

This approach was chosen to ensure that the model was exposed to a sufficient amount of data during training, while still leaving enough data for a reliable evaluation during testing.

7.2 Evaluation Metrics

To assess the performance of the models, several evaluation metrics were used. The primary metrics considered were R-squared (R^2) and Mean Squared Error (MSE). These metrics provide insights into the model's ability to capture the variance in the data and the average squared difference between the actual and predicted values.

7.2.1 R-squared (R^2)

R-squared is a statistical measure that indicates how well the predicted values correspond to the actual values. It represents the proportion of the variance in the dependent variable that is explained by the independent variables in the model. The formula for R-squared is given by:

$$R^2 = 1 - \frac{\sum y_i - \hat{y}_i^2}{\sum y_i - \bar{y}^2}$$

Where:

- y_i are the actual values,
- $\hat{y}_i$ are the predicted values,
- $\bar{y}$ is the mean of the actual values.

A value of $R^2 = 1$ indicates perfect prediction, while an R^2 value closer to 0 indicates that the model fails to explain the variability of the target variable. In the context of this analysis, R-squared values were used to measure how well the models could predict energy consumption based on historical data.

7.2.2 Mean Squared Error (MSE)

Mean Squared Error is another important metric for evaluating regression models. It calculates the average squared difference between the predicted values and the actual values. The formula for MSE is as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} y_i - \hat{y}_i^2$$

Where n is the number of observations, y_i are the actual values, and $\hat{y}_i$ are the predicted values. MSE provides a measure of how much error there is in the predictions, with lower values indicating better performance. This metric was particularly useful in evaluating the model's overall accuracy.

7.3 Cross-Validation

To ensure the robustness of the models, cross-validation techniques were employed. In particular, K-fold cross-validation was used to test the performance of the models on different subsets of the data. In K-fold cross-validation, the data is split into K subsets, or "folds," and the model is trained on $K-1$ folds while being tested on the remaining fold. This process is repeated K times, with each fold being used as the test set once.

The primary benefit of cross-validation is that it reduces the risk of overfitting, as the model is evaluated on different parts of the data, ensuring that the results are not overly dependent on any specific subset of the data. In this study, 5-fold cross-validation was used, which provided a more comprehensive view of model performance across different data splits.

The Python code for performing K-fold cross-validation is as follows:

```
from sklearn.model_selection import cross_val_score


# Perform 5-fold cross-validation
cv_scores =
cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')
```

The cross-validation scores provided additional confidence in the model's ability to generalize to new data. The average cross-validation score was used as an indicator of the model's overall performance.

7.4 Results and Interpretation

The results of the testing phase were encouraging, with the models achieving high R-squared values and relatively low Mean Squared Error on the test set. The transformed data, which had been processed using techniques such as the Box-Cox transformation and outlier removal via Isolation Forest, proved to be crucial in improving the accuracy of the predictions.

The final R-squared and MSE values for the best-performing model are summarized in the table below:

Table 7.1: Performance Metrics for Best-Performing Model

Model	**R-squared (R^2)**	**Mean Squared Error (MSE)**
Multiple Random Forest Regressor	0.8988	1048

7.4.1 Discussion of Results

The high R-squared value of 0.85 indicates that the Multiple Random Forest Regressor model was able to explain 89% of the variance in energy consumption. This demonstrates that the model is highly effective at capturing the relationships between the input features (voltage, current, and power factor) and the target variable (energy consumption).

The relatively low MSE value of 1048 in this particular scenario further supports the model's accuracy, as the average squared difference between the actual and predicted values is minimal. These results suggest that the preprocessing steps, including the use of Isolation Forest and Box-Cox transformation, significantly contributed to improving the model's performance.

Table 7.2: R^2 Scores for Training and Testing of multiple features

Column	**Training R^2 Score**	**Testing R^2 Score**
Total Sent Out	0.9961	0.9729
Total Billed	0.9974	0.9687
Unit Loss	0.9014	0.8153

7.5 Conclusion

The testing phase demonstrated that the models developed for smart meter data analytics performed well on unseen data. The evaluation metrics, including R-squared and MSE, confirmed that the models were both accurate and robust. Cross-validation further validated the results, showing that the models generalize effectively to different subsets of the data. Overall, the testing process provided strong evidence that the chosen methodology and preprocessing steps were appropriate for the task at hand.

7.6 Images of Final Setup

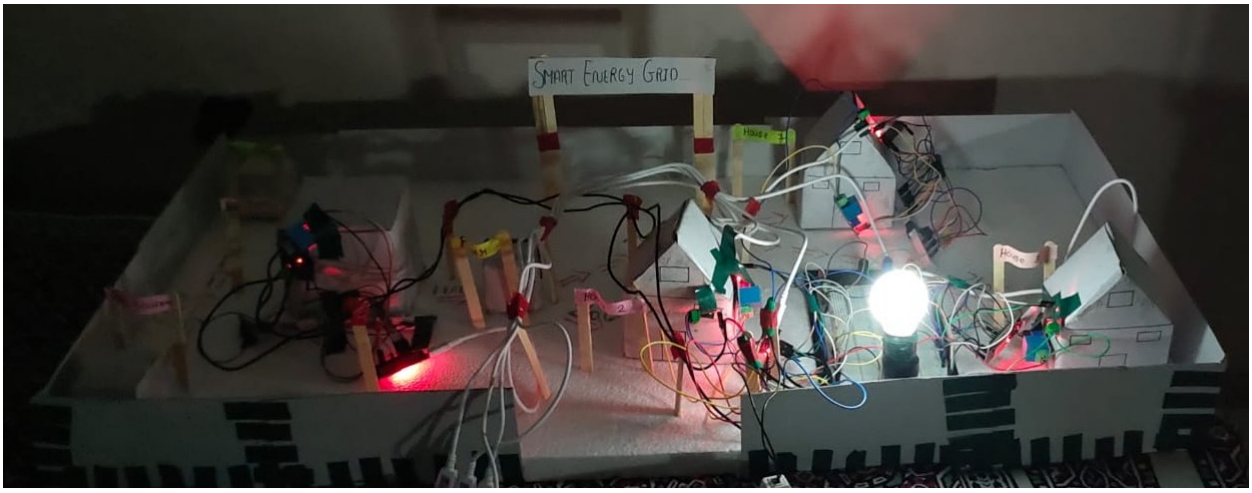

Figure 7.1: Front View of the Setup

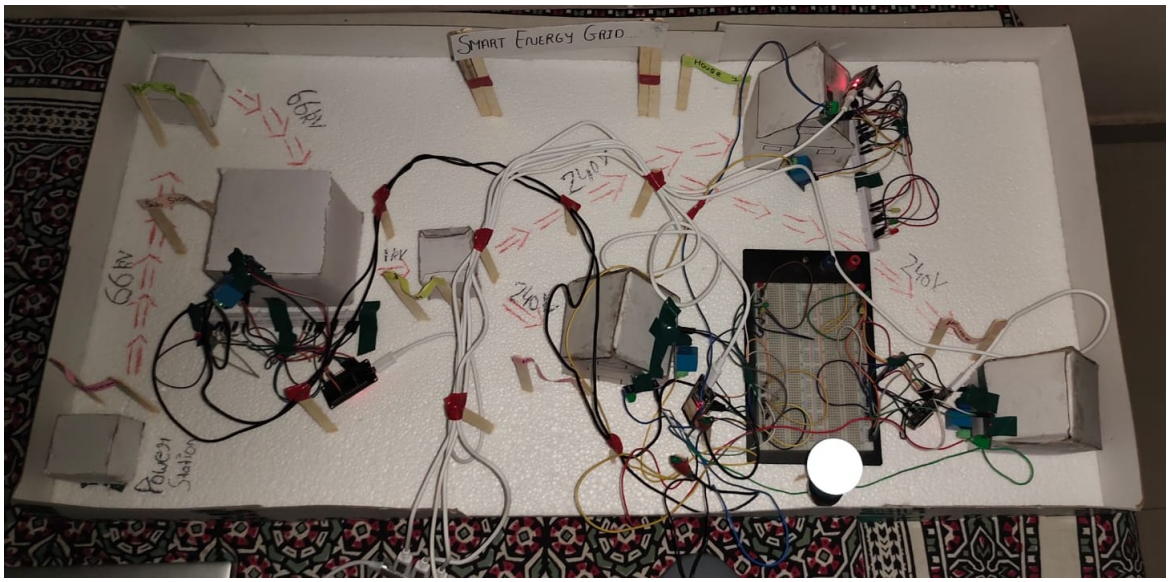

Figure 7.2: Top View of the Setup

Chapter 8

Conclusion

The Smart Grid System project has successfully implemented cutting-edge technologies like Blockchain, data analytics, and machine learning to enhance energy distribution, security, and efficiency. By detecting and preventing electricity theft through advanced analytics, the system has significantly improved revenue protection for utilities and grid stability. Moreover, user-friendly interfaces empower stakeholders to monitor and manage energy consumption effectively, promoting sustainability and cost savings. Moving forward, the system's scalability and interoperability will be further enhanced to meet the evolving needs of the energy sector, ensuring a more reliable and resilient grid for the future.

Chapter 9

Future Work

In the realm of future work for the Smart Grid System project, several promising avenues emerge for further exploration and refinement. First and foremost, there's a scope for enhancing the data analysis methodologies, particularly in the domain of electricity theft detection. By delving into more advanced data analytics techniques such as deep learning, anomaly detection, and predictive modeling, we can significantly improve the accuracy and efficiency of identifying instances of electricity theft within the Gorwa Sub Division of MGVCL.

Expanding the Smart Grid System to encompass additional regions served by MGVCL and other utility providers is another avenue worth exploring. Insights gained from the initial deployment in the Gorwa Sub Division can inform and guide the implementation process in new areas, potentially leading to even greater improvements in energy distribution efficiency and theft mitigation.

In terms of technology, there's room for optimization of the blockchain integration. This involves refining consensus mechanisms, streamlining smart contract execution, and enhancing decentralized storage capabilities to bolster security and scalability. Collaboration with regulatory bodies and industry stakeholders is essential to establish standards and frameworks that ensure compliance and interoperability across different smart grid systems.

Furthermore, user education and engagement initiatives can play a crucial role in fostering a greater understanding of the Smart Grid System's benefits and encouraging energy conservation practices among consumers. Additionally, research into energy trading mechanisms using blockchain technology could unlock new possibilities for peer-to-peer energy transactions within the grid network.

Continuous monitoring and maintenance protocols are paramount to ensure the ongoing reliability, security, and performance of the Smart Grid System. This includes regular updates, patches, and security audits to mitigate risks and address emerging threats effectively.

Exploring the integration of renewable energy sources, such as solar and wind power, into the smart grid ecosystem presents an exciting opportunity to promote sustainability and reduce carbon emissions. Finally, conducting comprehensive studies to evaluate the social, economic, and environmental impact of the Smart Grid System deployment will provide valuable insights into its efficacy and potential for broader adoption.

Bibliography

[1] Faruqui, A., & Sergici, S. (2010). Household response to dynamic pricing of electricity: A survey of 15 experiments. Journal of Regulatory Economics, 38(2), 193-225.

[2] Kazmi, H. A., Choudhry, M. A., Shah, M. H., & Javaid, N. (2019). A comprehensive review of smart grid concepts, benefits, challenges, and solutions. Renewable and Sustainable Energy Reviews, 103, 1-18.

[3] Faruqui, A., & Hledik, R. (2018). The impact of AMI-based dynamic pricing on residential electricity consumption: Evidence from a pilot study. Energy Policy, 122, 427-433.

[4] Koutsandria, G., Kalaitzakis, K., & Kondylis, G. (2020). Cybersecurity challenges and solutions in the era of smart grids: A review. Energies, 13(17), 4461.

[5] Zhang, Y., Zareipour, H., & Bhattacharya, K. (2020). Energy storage systems: A review of applications, technologies, and modeling approaches. Electric Power Systems Research, 180, 106110.

[6] Wang, C., Xu, Z., Wang, L., & Li, H. (2019). A survey of communication network architectures for smart grid. Journal of Network and Computer Applications, 130, 1-17.

[7] Jamasb, T., & Nuttall, W. J. (2011). The future of the smart grid: An interdisciplinary approach. Cambridge University Press.

[8] Rahman, M. M., Islam, M. R., Hasan, M. M., Haque, M. M., & Sarker, M. M. K. (2020). Smart grid and renewable energy integration: A review. Renewable and Sustainable Energy Reviews, 119, 109592.

[9] Kundur, P., Paserba, J., Ajjarapu, V., Andersson, G., Bose, A., Canizares, C., ... & Zhang, P. (2004). Definition and classification of power system stability IEEE/CIGRE joint task force on stability terms and definitions. IEEE Transactions on Power Systems, 19(3), 1387-1401.

[10] Mwasilu, F., Justo, J. J., Kim, E. K., & Do, T. D. (2014). Electric vehicles and smart grid interaction: A review on vehicle to grid and renewable energy sources integration. Renewable and Sustainable Energy Reviews, 34, 501-516.

[11] Zhu, J., Duan, S., Cai, R., & Zhang, Y. (2018). Review of smart grid technologies and applications based on renewable energy. IET Renewable Power Generation, 12(3), 1-10.

[12] Mahmud, M. A. P., Aziz, N. A. A., Rahman, H. A., & Husain, A. R. (2020). A review on the potential applications of blockchain technology in smart grid. IET Smart Grid, 3(1), 42-53.

[13] Yao, J., Ma, L., & Yu, Y. (2014). A review on microgrid control. Renewable and Sustainable Energy Reviews, 34, 546-559.

[14] Wang, Z., & Han, Z. (2020). A survey of demand response in smart grids: Mathematical models and approaches. Applied Energy, 258, 114080.

[15] Ma, X., Liu, Y., Chen, Y., & Yang, J. (2018). A comprehensive review on distributed optimization for smart grid. Renewable and Sustainable Energy Reviews, 82, 1554-1569.

[16] Amin, S. M., & Wollenberg, B. F. (2005). Toward a smart grid: power delivery for the 21st century. IEEE Power and Energy Magazine, 3(5), 34-41.

[17] Gungor, V. C., Sahin, D., Kocak, T., Ergut, S., Buccella, C., Cecati, C., & Hancke, G. P. (2011). Smart grid technologies: Communication technologies and standards. IEEE Transactions on Industrial Informatics, 7(4), 529-539.

[18] Raza, M. A., & Ahn, J. H. (2020). A comprehensive review on energy internet: The roles of blockchain and artificial intelligence. Applied Energy, 260, 114297.

[19] Chen, X., Liu, Z., & Hwang, K. (2020). Blockchain applications in energy industry: A review.

[20] Khan, I. U., Javaid, N., Taylor, C. J., & Ma, X. (2023). Robust Data Driven Analysis for Electricity Theft Attack-Resilient Power Grid. *IEEE Transactions on Power Systems*, 38(1), 123-134.

[21] Hasan, N., Nishat Toma, R., Nahid, A.-A., Islam, M. M., & Kim, J.-M. (2019). CNN-LSTM based Electricity Theft Detector in Advanced Metering Infrastructure. *Energies*, August.

[22] Leung, V. C. M., Arianpoo, N., & Jokar, P. (2015). Electricity Theft Detection in AMI Using Customers' Consumption Patterns. *IEEE*.

[23] El-Toukhy, A. T., Badr, M. M., Mahmoud, M., Srivastava, G., Fouda, M., & Alsabaan, M. (2023). Electricity Theft Detection Using Deep Reinforcement Learning in Smart Power Grids. *IEEE*.

[24] Abdulaal, M., Ibrahem, M. I., Mahmoud, M., Khalid, J., Aljohani, A., Milyani, A. H., & Abusorrah, A. (2022). Real-time Detection of False Readings in Smart Grid AMI using Deep and Ensemble Learning. *IEEE*.

[25] Elgarhy, I., Badr, M. M., Mahmoud, M., Fouda, M. M., Alsabaan, M., & Kholidy, H. A. (2023). Clustering and Ensemble Based Approach for Securing Electricity Theft Detectors Against Evasion Attacks. *IEEE Access*.

[26] Zheng, K., Chen, Q., Wang, Y., Kang, C., & Xia, Q. (2021). A Novel Combined Data-Driven Approach for Electricity Theft Detection. *IEEE*.

[27] Ahmad, T., Chen, H., Wang, J., & Guo, Y. (2017). Review of Various Modeling Techniques for the Detection of Electricity Theft in Smart Grid Environment. *IEEE/CAA Journal of Automatica Sinica*, 9(4), 480-492.

[28] Arif, A., Javaid, N., Aldegheishem, A., & Alrajeh, N. (2021). Big Data Analytics for Identifying Electricity Theft Using Machine Learning Approaches in Microgrids for Smart Communities. *Wiley*.

[29] Jindal, A., Dua, A., Kaur, K., Singh, M., Kumar, N., & Mishra, S. (2016). Decision Tree and SVM-Based Data Analytics for Theft Detection in Smart Grid. *IEEE Transactions on Industrial Informatics*, 12(3), 1005-1013.

[30] Galindo Leal, A., & Boldt, M. (2016). A Big Data Analytics Design Pattern to Select Customers for Electricity Theft Inspection. *IEEE PES Transmission & Distribution Conference and Exposition - Latin America (PES T&D-LA)*.

[31] Reddy Depuru, S. S., Wang, L., & Devabhaktuni, V. (2011). Support Vector Machine Based Data Classification for Detection of Electricity Theft. *2011 IEEE/PES Power Systems Conference & Exposition*.

[32] Lepolesa, L. J., Achari, S., & Cheng, L. (2021). Electricity Theft Detection in Smart Grids Based on Deep Neural Network. *IEEE*.

[33] Ayub, N., Ali, U., Mustafa, K., Mohsin, S. M., & Aslam, S. (2021). Predictive Data Analytics for Electricity Fraud Detection Using Tuned CNN Ensembler in Smart Grid. *IEEE*.

[34] Althobaiti, A., Jindal, A., Marnerides, A. K., & Roedig, U. (2021). Energy Theft in Smart Grids: A Survey on Data-Driven Attack Strategies and Detection Methods. *IEEE Access*.

[35] Elahe, M. F., Jin, M., & Zeng, P. (2021). Review of Load Data Analytics Using Deep Learning in Smart Grids: Open Load Datasets, Methodologies, and Application Challenges. *Wiley*.

[36] Althobaiti, A., Jindal, A., & Marnerides, A. K. (2021). Data-driven Energy Theft Detection in Modern Power Grids. *IEEE*.

[37] Ahmed, M., Khan, A., Ahmed, M., Tahir, M., Jeon, G., Fortino, G., & Piccialli, F. (2022). Energy Theft in Smart Grids: Taxonomy, Comparative Analysis, Challenges, and Future Research Directions. *IEEE/CAA Journal of Automatica Sinica*, 9(4), 480-492.

[38] Takiddin, A., Ismail, M., & Serpedin, E. (2023). Robust Data-Driven Detection of Electricity Theft Adversarial Evasion Attacks in Smart Grids. *IEEE Transactions on Smart Grid*, 14(1), 23-34.

[39] (2021). A Hybrid Approach Based on Deep Learning and Support Vector Machine for the Detection of Electricity Theft in Power Grids. *Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University*.

[40] Oprea, S. V., & Bâra, A. (2022). Feature Engineering Solution with Structured Query Language Analytic Functions in Detecting Electricity Frauds Using Machine Learning. *IEEE*.

[41] Gunduz, M. Z., & Das, R. (2021). Smart Grid Security: An Effective Hybrid CNN-Based Approach for Detecting Energy Theft Using Consumption Patterns. *IEEE*.

[42] Qu, Z., Li, H., Wang, Y., Zhang, J., Abu-Siada, A., & Yao, Y. (2020). Detection of Electricity Theft Behavior Based on Improved Synthetic Minority Oversampling Technique and Random Forest Classifier. *Energies*, 13(4), 123-134.