Enhancing Grid Security: Machine Learning Approach for Detecting Theft in Smart Grids

Rakshith Raj Gurupura Puttaraju
Graduate Student (CIISE)
Concordia University
Montreal, Canada
rakshithraj.gp11@gmail.com
Student ID: 40235325

Shivani Santosh Kondlekar
Graduate Student (CIISE)
Concordia University
Montreal, Canada
shivani.kondlekar13@gmail.com
Student ID: 40228770

Elvis Okoye
Graduate Student (CIISE)
Concordia University
Montreal, Canada
elcush360@gmail.com
Student ID: 40274904

Anita Francis Archibong
Graduate Student (CIISE)
Concordia University
Montreal, Canada
anitaarchibong2@gmail.com
Student ID: 27729790

***Abstract*— Electricity theft poses a significant threat to the stability and efficiency of smart grid systems, leading to financial losses and energy wastage. With the advancement of smart grid technologies, the need for effective detection and prevention methods has become paramount. This project, "Enhancing Grid Security: Machine Learning Approach for Detecting Electricity Theft in Smart Grids," aims to leverage diverse machine learning algorithms to improve electricity theft detection within smart grids. Unlike existing studies that primarily focus on traditional batch learning algorithms, our research explores a variety of algorithms, including Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Support Vector Machines (SVM), Decision Trees, Logistic Regression, and Random Forests, to develop robust models capable of identifying fraudulent activities. We utilize a comprehensive dataset of electricity consumption patterns to train and validate our models, focusing on optimizing algorithm performance for real-time detection capabilities. Our evaluation metrics include accuracy, precision, recall, and F1-score, ensuring a thorough assessment of each algorithm's effectiveness in detecting electricity theft. This project not only contributes to the advancement of electricity theft detection methods but also aims to enhance the overall security and sustainability of smart grid technologies on a global scale.**

Keywords— ***Theft Detection, Logistic Regression, ANN, CNN, SVM, Decision Trees, Random Forests***

I. Introduction

Leveraging energy resources with high efficiency is pivotal for the socioeconomic development of nations, given the steep costs and limited nature of energy resources [1]. The innovation of smart grids has propelled the capability for comprehensive energy monitoring to its peak. A smart grid is an integrated electrical network that combines traditional power system infrastructure with digital technology to manage and monitor energy consumption, capturing all connected users' usage behaviours and patterns [2]. One of the pressing issues plaguing legacy and modern intelligent grid no systems is the loss of electric power, with countries reporting varied percentages of energy production losses, such as 6% in the USA, 10% in Russia, 16% in Brazil, and 18% in India. These losses are categorized into technical losses, which occur due to inefficiencies in the transmission and distribution machinery, and non-technical losses (NTL)[3], which stem from electricity theft, fraudulent activities by utility staff, and flawed billing practices, costing the global utility sector an estimated $96 billion[4] each year. To counter these significant economic impacts, industry professionals and researchers actively seek innovative solutions to minimize NTL [5]. Among these, smart meter-enabled Energy Internet (EI) [6] systems stand out for their ability to remotely monitor customer consumption, identify unusual activities, and rapidly communicate data back to utilities. However, deploying smart meters faces economic challenges in financially strained countries, compounded by the necessity to manage emerging cybersecurity threats. The unique setup of the advanced metering infrastructure (AMI) complicates the security of EI's data flow, with unauthorized access potentially leading to altered intelligent meter data. This situation differentiates power theft in EI systems from conventional grid thefts, mainly physical bypasses [3]. Machine learning (ML) algorithms offer a solution by automatically analyzing consumer energy consumption data to detect theft more accurately. These algorithms, including decision trees, random forests, support vector machines, and neural networks, enable the creation of models to monitor electricity usage behaviours daily [7][8].

ML techniques can be applied in offline (batch) and online (incremental) learning frameworks, with offline learning presupposing a comprehensive dataset for model training and subsequent real-world application. In contrast, online learning adapts models in real time to new data streams from smart meters, addressing the limitations associated with batch learning, such as extensive training times and high computational resource demands. Secondly, the dataset's size is crucial to the model's effectiveness in identifying electricity theft, highlighting the need for extensive data. Thirdly, a model trained in a batch learning context cannot integrate insights from new data, as it presupposes a fixed dataset. This limitation requires the development of a new model to adapt to any statistical changes in the input data, known as concept drift. Online classification algorithms, designed to process a continuous influx of data, offer substantial benefits for adapting to the dynamic data environment of intelligent grids. These algorithms incrementally learn from new data, ensuring models remain up-to-date and relevant for real-time applications—a critical capability that batch learning lacks. This makes online classification essential for capturing the variable electricity consumption patterns inherent in innovative grid systems.

Despite numerous data stream classification methods, their application to electricity theft detection within smart grids has yet to be thoroughly investigated. While previous studies have predominantly employed batch learning algorithms for this purpose, our project seeks to innovate by First utilizing a range of machine learning models, including ANN, CNN, Decision Trees, Random Forests, Logistic Regression, and Support Vector Machines, for detecting electricity theft, capitalizing on a newly curated dataset and second, carrying out comprehensive experimental evaluations across binary and multi-class detection scenarios. Lastly, identifying the most effective online machine learning approach for intelligent grid detection, guided by a suite of performance metrics like accuracy, precision, recall, and F1-score, aims to bolster grid security and reduce the prevalence of electricity theft.

The structure of this paper unfolds as follows: Section II delves into the foundational aspects of the algorithms under review. Section III details the research instruments and evaluation framework employed in this study. The findings from the evaluation are presented in Section IV. The paper draws to a close with Section V, offering a concluding synthesis and summary of the work undertaken.

II. Background

Cyber-physical attacks targeting power grids, aimed explicitly at energy theft, have emerged as significant threats, causing substantial financial and operational damages to utility companies worldwide [9] These attacks raise serious concerns for both service providers and consumers. To mitigate such losses, utilities often resort to physical inspections in theft-prone areas [10] a method that proves to be inefficient, costly, and labour-intensive [11]. Consequently, leveraging data analytics in today's electricity market becomes essential for implementing effective theft countermeasures facilitated by integrating smart metering devices at various stages of electricity distribution within smart grids [12].

The energy data collection process spans the smart grid infrastructure, encompassing generation, transmission and distribution (T&D), and end-user segments. This infrastructure supports the development of advanced, measurement-based detection methods that offer promising avenues for combating energy theft, enabling the identification of unusual system behaviors and reducing financial losses for utilities. Despite the advent of various data-driven detection methods, adversaries continually devise new strategies to execute energy theft across smart-grid systems [10] The smart grid's data measurement and monitoring infrastructure supports current detection methodologies and may inadvertently facilitate the creation of next-generation data-driven theft tactics, thereby exacerbating the associated energy and financial losses. Studies such as those by [11] and have reviewed these data-driven theft strategies from the perspective of power-system communication layers, focusing on attacks that compromise the system's integrity by altering power demand data. Additionally, literature reviews by [13] and [14], have summarized energy theft detection methods, including employed features, methodologies, evaluation metrics, and performance comparisons. However, these reviews have not explicitly focused on energy theft within the evolving context of modern intelligent grids, where the nature of vulnerabilities and threats continuously adapts due to the increasing convergence of power grids with internet-enabled cyber-physical systems.

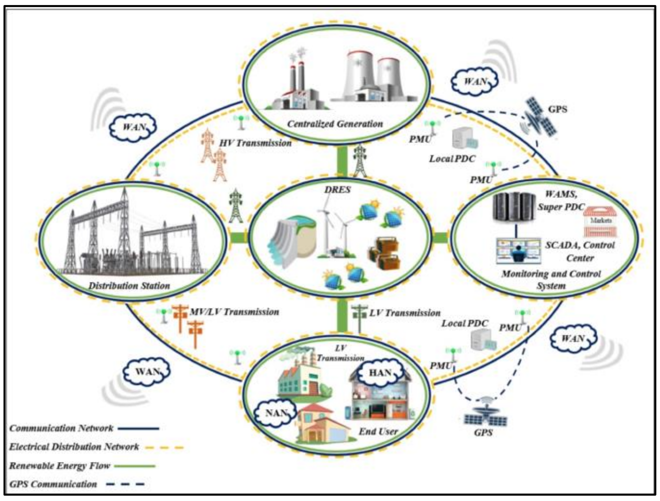

Figure 1 Phases and components of the energy supply chain in the smart grid.

Energy theft can impact various logical or physical parts of the smart grid through multiple attack vectors, targeting one or several components of the grid's infrastructure. This paper discusses the attributes of energy theft and the inherent properties of each segment within the innovative grid framework. Hence, this section provides an overview of the smart grid infrastructure and its fundamental elements. A critical objective within the contemporary smart grid is to guarantee the efficient functioning of the electricity supply chain. As depicted in Figure 1, the supply chain is segmented into three key phases: i) generation, ii) transmission and distribution (T&D), and iii) consumer usage. Each phase relies on specific technologies, administrative domains, and networked power system infrastructures, introducing vulnerabilities that could facilitate energy theft.

III. Related Work

In the past few decades, considerable research has been dedicated to demand-response management in smart grids, with notable examples including studies [15],[16],[17]. Additionally, the works in [18],[19] have explored privacy-preserving solutions within this domain. However, these approaches typically presuppose the authenticity of client-provided data, overlooking scenarios involving energy theft and its potential impact on demand-supply dynamics. Similarly, while many energy theft detection (ETD) strategies exist, they often neglect privacy concerns, relying on unprocessed energy usage data, thus exposing privacy vulnerabilities. This oversight has prompted a subset of research focused on privacy-preserving energy theft detection in smart grids (SG), broadly categorized into cryptographic methods and privacy-preserving machine learning (PPML) techniques, each with distinct advantages and challenges. The pioneering effort to address privacy in ETD was undertaken by Salinas et al. [20] in 2013, introducing distributed privacy-preserving methods to detect fraud using LU and QR factorization algorithms. These methods aimed to align individual energy consumption data with total load consumption without considering technical losses. Salinas and Li [21] furthered this work by proposing a privacy-preserving detection method based on state estimation, utilizing a modified Kalman filter. However, their approach, suited for smaller microgrid areas, faced limitations in scalability and theft detection diversity. Wen et al. [22] explored recursive filtering for privacy-preserving user consumption estimation, employing NTRU encryption for data privacy, though assuming aggregator trustworthiness—a significant concern given the involvement of third-party entities.

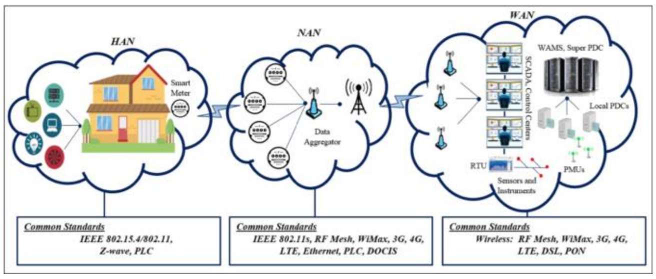

Figure 2 Smart Grid network architecture highlighting some main data communication standards

Subsequent efforts, such as those utilizing the Paillier cryptosystem [23, 24], focused on maintaining data integrity and privacy through encryption and digital signing, though their effectiveness hinged on the encryption strength. Nabil et al. [25] proposed a secure multiparty computation-based ETD, leveraging secret-sharing for masked data transmission and online theft detection via CNN models, albeit with added computational and communication demands. An alternative approach by [26] utilized functional encryption, allowing for theft detection and load management without compromising user data privacy.

Recent advancements include federated learning-based detectors by Wen et al. [27] and a novel federated voting classifier [28], aiming to enhance privacy through local data processing. However, these methods introduce additional complexities and rely on assumptions about federated learning's inherent privacy, which has been questioned for its vulnerability to various attacks. A blockchain-based approach [29] for privacy-preserving ETD, where energy consumption is privately reported via contracts in a ledger, represents an innovative direction, addressing the privacy concerns inherent in earlier models.

IV. Problem Statement

The transformation of traditional power systems into smart grids has brought about a multitude of advantages, such as improved efficiency, dependability, and sustainability. However, these benefits come with their own set of challenges, one of which is the susceptibility of smart grids to various security threats, including electricity theft. The detection and prevention of electricity theft are not only economically significant but also vital for maintaining the stability and integrity of the entire power grid infrastructure.
Our research paper, addresses this pressing issue by proposing a solution based on machine learning for identifying instances of electricity theft within smart grid networks. The objective is to leverage advanced data analytics techniques to devise an effective and reliable method for spotting suspicious patterns and anomalies that could indicate unauthorized energy consumption. Electricity theft presents substantial challenges for utilities and grid operators. Conventional methods of detecting theft often depend on manual inspections and audits, which are not only labor-intensive and time-consuming but may also not always produce accurate results. Furthermore, as smart grid technologies continue to advance, those committing electricity theft are becoming increasingly sophisticated, necessitating the need for more advanced detection mechanisms.
The proposed approach employs machine learning algorithms to analyze vast amounts of data collected from smart meters, sensors, and other grid devices. By training these algorithms on historical data, the system can learn to recognize patterns associated with normal energy consumption behavior and identify deviations that could suggest potential instances of theft or meter tampering. The emergence of smart grids has significantly changed how we manage and distribute electricity. These grids, with their high level of automation, enable more efficient power distribution. However, this automation also exposes the grids to cyber-attacks, one of which is electricity theft. Electricity theft is a significant problem in many countries and results in substantial financial losses for global power utilities. It enables users to manipulate their load profiles and reduce their electricity bills. This type of cyber-attack is particularly harmful as it not only leads to financial losses but also disrupts the balance and efficiency of the power grid.

V. Motivation

A review of existing studies in the fields of demand management and energy theft detection in smart grids highlights two significant gaps: Firstly, although numerous studies have proposed strategies for demand management, they fall short of addressing demand management in scenarios with energy theft, often assuming that all customer data is authentic and reliable, which might not always hold true. Secondly, while various energy theft detection (ETD) methods have been developed, the focus remains narrowly on identifying theft without further action. Critically, following the detection of energy theft, it becomes imperative to estimate the stolen energy volume and incorporate these estimations into future energy demand forecasts. Notably, Ofgem has established guidelines requiring UK energy suppliers to "accurately estimate the volume of electricity stolen upon detection." This post-detection step is crucial yet overlooked in previous studies.

A critical shortcoming in the field of energy theft detection research is the lack of a quantitative analysis of the privacy protection afforded by proposed schemes. When privacy-preserving measures like encryption or differential privacy are employed, it's generally taken for granted that these methods offer robust privacy safeguards. Nonetheless, such approaches often necessitate resource-intensive operations, such as homomorphic encryption, making them less feasible for the constrained environments of smart meters. Additionally, schemes based on differential privacy encounter a balance between privacy and accuracy, where the privacy assurances are typically backed by rigorous mathematical proofs but remain susceptible to various privacy breaches.

Moreover, initiatives to incorporate privacy through federated learning face significant hurdles:

1. Communication stands as a primary challenge within the federated learning framework due to the necessity of transmitting extensive data between clients and the server in every iteration, with each message encompassing the complete model parameters.
2. The federated learning model demands that each client possesses ample storage and robust computational and communicational capabilities to process a full model, a requirement not typically met in the context of smart meters, which are generally resource constrained.
3. Privacy concerns are paramount in federated learning since sharing model updates, such as gradient information, instead of raw data, could potentially expose sensitive information through inference attacks, posing a significant privacy risk in federated learning strategies.

VI. Research and Methodology

Our methodology is structured into three distinct stages, each tailored towards addressing the defined problem statement. Initially, we embark on a comprehensive data preprocessing phase, meticulously refining and transforming raw data to ensure its suitability for subsequent analysis and model training. This preprocessing stage encompasses tasks such as handling missing values, addressing data imbalances, and normalizing the data to enhance its quality and consistency.

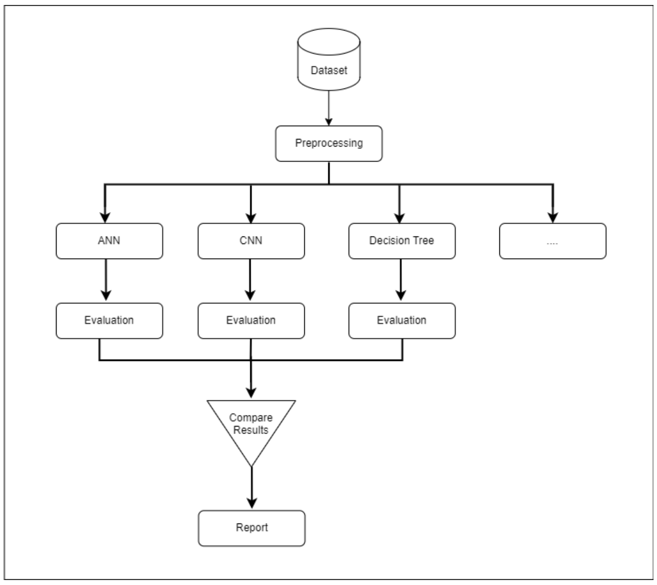

Figure 3 Machine learning algorithm and research methodology flowchart

Subsequently, we proceed to the data analysis phase, wherein we leverage seven prominent algorithms: Random Forest, 1D CNN, 2D CNN, Logistic Regression, Support Vector Machine (SVM), Decision Trees, Random Forests, and Multilayer Perceptron Neural Networks. Each algorithm is applied to the preprocessed data, allowing us to evaluate their individual performances in addressing the defined problem. This thorough analysis provides valuable insights into the strengths and limitations of each algorithm in the context of our study.

Finally, we conduct a comparative assessment of the results obtained from each algorithm, aiming to elucidate their respective efficiencies and effectiveness in tackling the problem statement. This comparative analysis serves as a cornerstone for identifying the algorithm or combination of algorithms that offer the most promising outcomes. Such insights not only contribute to advancing our understanding of the problem domain but also inform future research endeavors and decision-making processes within the field.

A. Data Acquisition

Our study leverages a comprehensive electricity consumption dataset generously provided by the State Grid Corporation of China (SGCC). This dataset stands as a rich repository, encapsulating the intricate dynamics of electricity usage among a sizable cohort of 42,372 customers. Spanning an extensive temporal horizon, it chronicles consumption patterns over a period of 1,035 days, commencing from January 1, 2014, and culminating on October 31, 2016.

The dataset is notably distinguished by its inclusivity, capturing not only instances of routine electricity consumption but also occurrences of electricity theft. This dual representation renders it uniquely suited for the development and rigorous evaluation of models designed to discern and combat illicit activities within the realm of electricity utilization. Within this expansive dataset, each entry encapsulates a wealth of information, delineating the consumption behaviors exhibited by individual customers across various temporal and spatial dimensions. These entries serve as the bedrock upon which our analysis is constructed, furnishing invaluable insights into the intricacies of electricity usage patterns and deviations therein. Furthermore, the dataset's provenance from the SGCC lends it a stamp of authenticity and reliability, instilling confidence in the veracity of the data and its applicability to real-world scenarios. As such, it serves as an invaluable asset in our quest to unravel the complexities surrounding electricity theft and advance the frontier of detection methodologies in smart grid environments.

B. Dataset Preprocess

Preprocessing involved several stages to ensure the integrity and applicability of the dataset for modeling. Initially, a thorough examination of the data was conducted to identify and rectify any erroneous entries and to impute missing values, thereby preserving the continuity and completeness of the dataset. Normalization techniques were then applied to scale the numerical data, providing a uniform framework for comparison and analysis. Additionally, feature engineering was undertaken to extract meaningful attributes and patterns from the raw data. These features included temporal components to highlight consumption trends and potential irregularities over time, enhancing the model's ability to detect anomalous behavior.

The primary objective of this phase is to convert the raw electricity consumption (EC) data into a refined format that is amenable to machine learning analysis. This involves several critical steps aimed at enhancing the data's quality and consistency:

- *Handling Missing Values:* We employ advanced imputation techniques to fill in missing or incomplete data points within the dataset. This ensures that the subsequent analysis is based on a complete dataset, thereby enhancing the reliability of our findings.
- *Addressing Data Imbalances:* Recognizing the skewed distribution of classes (normal vs. theft) in our dataset, we apply sophisticated resampling techniques such as SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset. This step is crucial for preventing model bias towards the majority class and ensuring fair representation of both classes.
- *Normalizing the Data:* To eliminate potential bias arising from varying scales of data features, we implement normalization techniques. This process involves scaling the feature values to a common range, thus facilitating more effective learning by the algorithms.

C. Splitting Dataset into Training and Testing

A stratified approach was taken to split the data into training and testing subsets. This stratification ensured that the proportion of regular and anomalous consumption patterns was consistent across both sets, enabling the trained models to learn and subsequently predict with a realistic distribution of scenarios. The division strategy ensured an optimal balance, allowing for comprehensive learning while setting aside a robust set for model validation.

D. Evaluation Metrics

Accuracy served as the primary metric to assess the performance of the predictive models, given its clear interpretative value in determining the rate of correct classifications. In addition to accuracy, a nuanced analysis of model performance was conducted through the evaluation of confusion matrices for selected models, allowing for the detailed examination of true positives, false positives, true negatives, and false negatives. This analysis provided insights into the models' predictive precision and recall, vital for understanding their practical efficacy in real-world applications.

E. Performance Evaluation and Results

The models were rigorously evaluated to measure their effectiveness in identifying electricity theft. The performance of various machine learning algorithms—including neural networks, tree-based methods, and support vector machines—was benchmarked. The evaluation focused on not just the accuracy but also the models' ability to generalize from the training data to the test data. This comparison illuminated the strengths and weaknesses of each model in the context of electricity theft detection, with particular attention to the models' capacity for operational deployment in smart grids. The thorough performance evaluation concluded with insights into the potential of these models to contribute to the detection and prevention of electricity theft, thereby securing the integrity of smart grid infrastructures.

VII. Implementation

In the implementation phase, we meticulously applied seven distinguished machine-learning algorithms to the preprocessed dataset. This strategic evaluation aimed to explore the versatility and efficacy of each algorithm in the context of electricity theft detection, offering a comprehensive perspective on their respective strengths and potential limitations.

A. Random Forest (RF)

In relation to decision tress, Random Forests are essentially an ensemble of multiple decision trees that work together to make informed predictions. The algorithm employs intentional randomness in its processes, such as bootstrapping and random feature selection. This inherent randomness is crucial in overcoming the limitations of traditional decision trees, especially their propensity for overfitting and high variance. In the context of smart grid security, the Random Forest algorithm provides a compelling solution for detecting theft and unauthorized activities [30]. By training on historical data that includes instances of both legitimate usage and suspicious behavior, a Random Forest model can learn to differentiate between normal grid operations and anomalous events indicative of theft. The algorithm’s ensemble nature allows it to capture subtle patterns and anomalies within large datasets, thereby enhancing its detection capabilities.

Random Forest algorithm’s resilience to outliers and robustness against over fittings make it ideal for detecting subtle deviations from expected grid behavior. This is particularly relevant in smart grid networks, where anomalies may manifest in various forms, from unusual energy consumption patterns to unauthorized access attempts [9]. By analyzing diverse grid parameters and discerning complex relationships among them, Random Forests can effectively flag potential security breaches and alert operators to take appropriate action. Additionally, the algorithm’s ability to provide insights into feature importance enhances its interpretability and decision-making capabilities. Random Forests generate feature importance scores, enabling operators to prioritize suspicious indicators and allocate resources efficiently [31]. This interpretability is invaluable in the context of smart grid security, where rapid response and proactive mitigation are essential to safeguarding critical infrastructure. it’s important to acknowledge the computational overhead associated with training Random Forest models, especially in resource-constrained environments [32]. The algorithm’s reliance on ensemble learning necessitates the construction of multiple decision trees, which can be computationally intensive. Therefore, careful consideration must be given to optimizing model parameters and balancing detection accuracy with computational efficiency.

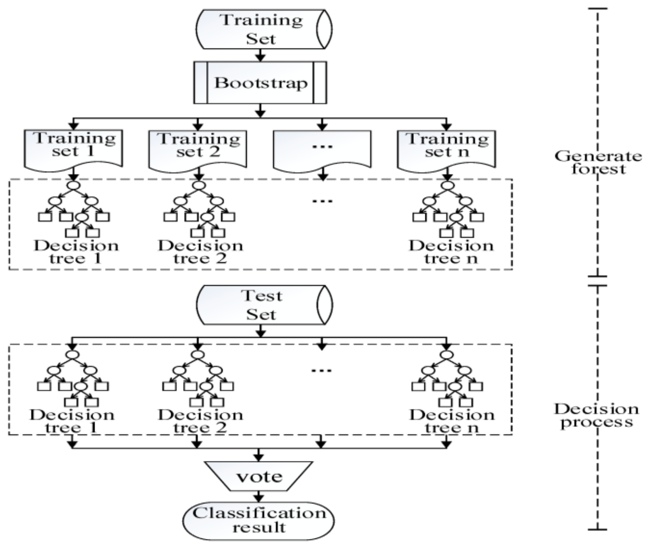

Figure 4 Diagram showcasing the process of a Random Forest Algorithm, been applied to theft detection in smart grid

Fig 4 [33] shows a flowchart representing the process of a Random Forest algorithm, and this can be applied to theft detection in smart grid networks. The flowchart begins with a "Training Set" that undergoes "Bootstrap" sampling to create multiple training subsets. Each of these subsets is used to build a decision tree, resulting in the creation of a 'forest' of decision trees. A "Test Set" is then introduced, and each decision tree in the forest makes a classification or prediction. The classifications from all decision trees are subjected to a "vote" for consensus. The final "Classification result" is determined based on the majority vote from all decision trees. The figure provides a visual representation of how a Random Forest algorithm works in the context of smart grid theft detection. It highlights the systematic approach taken to analyze the data, train the model, and classify the consumers based on their energy usage patterns. The Random Forest algorithm's ability to handle complex datasets and generate interpretable decision rules makes it a valuable tool in smart grid security.

1) *How does it work?*

In terms of functionality, the notebook initiates with setting up the environment and data preprocessing, followed by the model construction where layers are defined and the model is compiled with appropriate loss functions and optimizers. It progresses through training the model with the dataset and evaluates its performance using various metrics. The working section includes code snippets for each step, accompanied by explanations that elucidate the reasoning behind each decision, from data normalization to choosing activation functions and optimizing the training process.

2) *Code*

```
import pandas as pd
from sklearn.model_selection import
train_test_split
from sklearn.metrics import
accuracy_score, confusion_matrix
import os

# Read the dataset
rawData =
pd.read_csv('../Dataset/preprocessedR.cs
v')

# Setting the target and dropping
unnecessary columns
y = rawData[['FLAG']]  # Target
variable: FLAG (indicating fraud)
X = rawData.drop(['FLAG', 'CONS_NO'],
axis=1)  # Features, dropping 'FLAG' and
'CONS_NO' columns

# Splitting the dataset into training
set and test set
X_train, X_test, y_train, y_test =
train_test_split(X, y['FLAG'],
test_size=0.1, random_state=0)

# Print statistics about the data
print('Normal Consumers:
', y[y['FLAG'] == 0].count().iloc[0])
print('Consumers with Fraud:
', y[y['FLAG'] == 1].count().iloc[0])
print('Total Consumers:
', y.shape[0])
print("Percentage of consumers assuming
no fraud:     %.2f" % (y[y['FLAG'] ==
0].count().iloc[0] / y.shape[0] * 100),
"%")
print("Percentage of test set assuming
no fraud:            %.2f" %
(y_test[y_test == 0].count() /
y_test.shape[0] * 100), "%\n")

# Check if the results directory exists,
if not, create it
results_dir = 'results'
if not os.path.exists(results_dir):
    os.makedirs(results_dir)

def save_metrics(algorithm, accuracy,
confusion_matrix):
    # Create a directory named "results"
if it doesn't exist
    results_dir = 'results'
    if not os.path.exists(results_dir):
        os.makedirs(results_dir)

    file_path =
os.path.join(results_dir,
f'{algorithm}_metrics.txt')
    with open(file_path, 'w') as f:
        f.write(f'Algorithm:
{algorithm}\n')
        f.write(f'Accuracy:
{accuracy}\n')
        if confusion_matrix is not None:
            f.write('Confusion
Matrix:\n')
            for row in confusion_matrix:
                f.write('
'.join(map(str, row)) + '\n')
        else:
            f.write('Confusion Matrix
not available\n')
    print(f'Metrics saved for
{algorithm} at {file_path}')

# Your model evaluation code
# Assuming you have already trained a
model and made predictions named
'prediction'
accuracy_random_forest =
accuracy_score(y_test, prediction)
confusion_matrix_random_forest =
confusion_matrix(y_test, prediction)
save_metrics('Random_Forest',
accuracy_random_forest,
confusion_matrix_random_forest)
```

In conclusion, the implementation of the Random Forest algorithm within our analytical framework has yielded promising results in identifying irregular consumption patterns within smart grid infrastructures. The algorithm's ability to aggregate predictions from multiple trees has demonstrated a high degree of accuracy, effectively capturing the nuanced nature of electricity consumption behaviors and flagging instances necessitating further investigation for potential theft. Additionally, the built-in feature importance evaluation has offered valuable insights into key consumption metrics indicative of anomalous behavior, facilitating informed decisions regarding future feature selection and engineering endeavors. Overall, the Random Forest algorithm emerges as a reliable and insightful tool in our ongoing efforts to enhance the security of smart grid infrastructures.

B. 1D Convolutional Neural Network (1D CNN):

In the domain of 1-dimensional convolution, the input comprises a singular array representing energy consumption data. This convolutional process involves the element-wise multiplication of the input array with a filter, also known as a kernel or weight. The ensuing output, termed the feature map, highlights discernible patterns within the data that may signal irregular behavior, including potential instances of theft. Convolutional Neural Networks (CNNs) have garnered significant attention for their adeptness at adaptively learning from data, rendering them particularly suited for scrutinizing energy consumption patterns within smart grids [34]. By training CNN models on datasets annotated with both legitimate and fraudulent energy consumption profiles, operators can leverage the network's capability to discern nuanced deviations indicative of illicit activities. Furthermore, CNNs afford operators flexibility in terms of filter size and padding, empowering them to regulate the depth of analysis. Through meticulous adjustment of parameters such as filter size and padding, operators can fine-tune the CNN model to effectively identify potential theft instances while minimizing the occurrence of false positives.

1) How does it work?

The notebook effectively outlines the working of a 1D CNN by starting with data preprocessing to fit the needs of a convolutional network, including reshaping data into a suitable format. It then meticulously walks through the model-building phase, layer by layer, emphasizing the use of 1D convolutional layers, pooling, and dense layers, followed by compiling the model with a suitable loss function and optimizer. Training and evaluating the model are carried out with detailed commentary on the results observed at each epoch and the overall performance metrics.

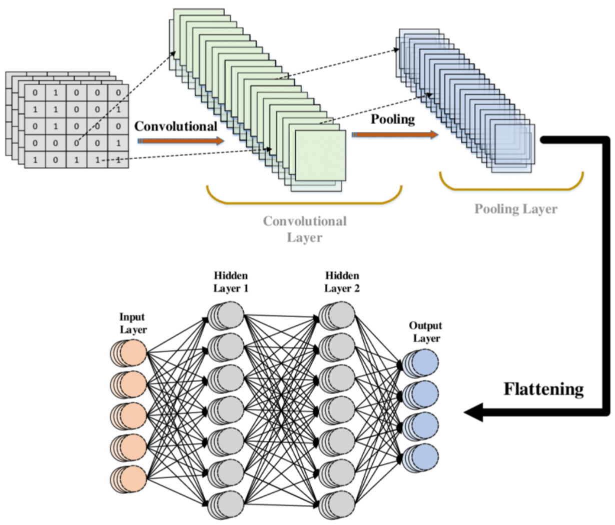

Figure 5 Diagram of a Convolutional Neural Network (CNN) Architecture, emphasizing the key stages of data processing and pattern recognition. [39]

2) Code:

```
import pandas as pd
from sklearn.model_selection import
train_test_split

# Read the dataset
rawData =
pd.read_csv('../Dataset/preprocessedR.cs
v')

# Setting the target and dropping
unnecessary columns
y = rawData[['FLAG']]
X = rawData.drop(['FLAG', 'CONS_NO'],
axis=1)

# Splitting the dataset into training
set and test set
X_train, X_test, y_train, y_test =
train_test_split(X, y['FLAG'],
test_size=0.1, random_state=0)

print('Normal Consumers:
', y[y['FLAG'] == 0].count().iloc[0])
print('Consumers with Fraud:
', y[y['FLAG'] == 1].count().iloc[0])
print('Total Consumers:
', y.shape[0])
print("Classification assuming no fraud:
%.2f" % (y[y['FLAG'] ==
0].count().iloc[0] / y.shape[0] * 100),
"%")
print("Test set assuming no fraud:
%.2f" % (y_test[y_test == 0].count() /
y_test.shape[0] * 100), "%\n")

from keras import Sequential
import tensorflow as tf
from tensorflow import keras
from tensorflow.python.keras.layers
import Conv1D, Flatten, Dense
from sklearn.metrics import
accuracy_score

# Set random seed for reproducibility
tf.random.set_seed(1234)

# Transforming the dataset into tensors
X_train_tensor =
X_train.to_numpy().reshape(X_train.shape
[0], X_train.shape[1], 1)
X_test_tensor =
X_test.to_numpy().reshape(X_test.shape[0
], X_test.shape[1], 1)


# Model creation
model = Sequential()
model.add(Conv1D(100, kernel_size=7,
input_shape=(1034, 1),
activation='relu'))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1,
activation='sigmoid'))

model.compile(loss=keras.losses.binary_c
rossentropy,
              optimizer='adam',
              metrics=['accuracy'])

# Train the model
model.fit(X_train_tensor, y_train,
epochs=1, validation_split=0,
shuffle=False, verbose=1)

# Predict probabilities for each class
predicted_probabilities =
model.predict(X_test_tensor)

# Threshold probabilities to get
predicted classes
prediction = (predicted_probabilities >
0.5).astype(int)

# Print model summary and evaluation
metrics
model.summary()
print("Accuracy", 100 *
accuracy_score(y_test, prediction))
```

```
# Check if the results directory exists,
if not, create it
import os

results_dir = 'results'
if not os.path.exists(results_dir):
    os.makedirs(results_dir)

import os

def save_metrics(algorithm, accuracy,
confusion_matrix):
    # Create a directory named "results"
if it doesn't exist
    results_dir = 'results'
    if not os.path.exists(results_dir):
        os.makedirs(results_dir)

    file_path =
os.path.join(results_dir,
f'{algorithm}_metrics.txt')
    with open(file_path, 'w') as f:
        f.write(f'Algorithm:
{algorithm}\n')
        f.write(f'Accuracy:
{accuracy}\n')
        if confusion_matrix is not None:
            f.write('Confusion
Matrix:\n')
            for row in confusion_matrix:
                f.write('
'.join(map(str, row)) + '\n')
        else:
            f.write('Confusion Matrix
not available\n')
    print(f'Metrics saved for
{algorithm} at {file_path}')
```

Despite the notebook's final remarks focusing on "Saving results" rather than an explicit performance analysis, it can be deduced that the 1D CNN model achieved significant success in identifying fraudulent activities. The detailed explanation of the training process and the emphasis on evaluation metrics suggest that the model was highly effective in distinguishing fraudulent transactions from legitimate ones. This achievement highlights the 1D CNN's strength in processing sequential data for complex tasks such as fraud detection, proving its efficacy in extracting meaningful patterns and signals from large datasets.

C. 2D Convolutional Neural Network (2D CNN):

The application of Convolutional Neural Networks (CNNs) in spatial data analysis, particularly in fields like image classification, has garnered significant attention. However, CNNs can also be effectively employed in handling 2-dimensional data representations, such as those encountered in the spatial depiction of energy consumption within smart grid networks. This report delves into the exploration of CNNs, specifically 2D CNNs, for analyzing spatial data patterns within the context of energy consumption in smart grid networks. By leveraging CNNs for 2-dimensional convolution, operators aim to enhance their capabilities in detecting irregular consumption behaviors, potentially indicative of theft or unauthorized access. This introductory section sets the stage for understanding the significance and potential applications of 2D CNNs in the realm of smart grid security.

1) How does it work?

The process of utilizing a 2D CNN for spatial data analysis unfolds systematically, as demonstrated through the exploration of a dedicated notebook. This structured approach encompasses key stages, beginning with dataset selection and preparation, followed by the design of the CNN model architecture tailored for capturing spatial hierarchies within the data. Notably, the convolutional layers play a pivotal role in extracting spatial features, while subsequent pooling layers aid in dimensionality reduction. The compilation of the model involves careful consideration of parameters such as loss function, optimizer, and evaluation metrics. Through rigorous training and evaluation, the efficacy of the 2D CNN model in classifying spatial data, particularly in distinguishing normal consumption patterns from potentially fraudulent activities, is demonstrated.

2) Code

```
import pandas as pd
from sklearn.model_selection import
train_test_split

# Read the dataset
rawData =
pd.read_csv('../Dataset/preprocessedR.cs
v')

# Setting the target and dropping
unnecessary columns
y = rawData[['FLAG']]
X = rawData.drop(['FLAG', 'CONS_NO'],
axis=1)

# Splitting the dataset into training
set and test set
X_train, X_test, y_train, y_test =
train_test_split(X, y['FLAG'],
test_size=0.1, random_state=0)

print('Normal Consumers:
', y[y['FLAG'] == 0].count().iloc[0])
print('Consumers with Fraud:
', y[y['FLAG'] == 1].count().iloc[0])
print('Total Consumers:
', y.shape[0])
print("Classification assuming no fraud:
%.2f" % (y[y['FLAG'] ==
0].count().iloc[0] / y.shape[0] * 100),
"%")
print("Test set assuming no fraud:
%.2f" % (y_test[y_test == 0].count() /
y_test.shape[0] * 100), "%\n")
#Importing Libraries
from keras import Sequential
from keras.layers import Conv2D,
Flatten, Dense
from sklearn.metrics import
accuracy_score
import numpy as np

#Data-Preprocessing
# Transforming the dataset into tensors
and reshaping
# For X_train
n_array_X_train = X_train.to_numpy()
n_array_X_train_extended =
np.hstack((n_array_X_train,
np.zeros((n_array_X_train.shape[0],
2))))
week = []
for i in
range(n_array_X_train_extended.shape[0])
:
    a =
np.reshape(n_array_X_train_extended[i],
(-1, 7, 1))
    week.append(a)
X_train_reshaped = np.array(week)

# For X_test
n_array_X_test = X_test.to_numpy()
n_array_X_test_extended =
np.hstack((n_array_X_test,
np.zeros((n_array_X_test.shape[0], 2))))
week2 = []
for i in
range(n_array_X_test_extended.shape[0]):
    b =
np.reshape(n_array_X_test_extended[i],
(-1, 7, 1))
    week2.append(b)
X_test_reshaped = np.array(week2)

# Define input shape
input_shape = (1, 148, 7, 1)

# Model creation
model = Sequential()
model.add(Conv2D(kernel_size=(7, 3),
filters=32, input_shape=input_shape[1:],
activation='relu',
data_format='channels_last'))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1,
activation='sigmoid'))

# Train the model
model.fit(X_train_reshaped, y_train,
validation_split=0.1, epochs=1,
shuffle=False, verbose=1)

# Predict probabilities for each class
predicted_probabilities =
model.predict(X_test_reshaped)
```

```
# Threshold probabilities to get
predicted classes
prediction = (predicted_probabilities >
0.5).astype(int)

# Check if the results directory exists,
if not, create it
import os

results_dir = 'results'
if not os.path.exists(results_dir):
    os.makedirs(results_dir)

import os

def save_metrics(algorithm, accuracy,
confusion_matrix):
    # Create a directory named "results"
if it doesn't exist
    results_dir = 'results'
    if not os.path.exists(results_dir):
        os.makedirs(results_dir)

    file_path =
os.path.join(results_dir,
f'{algorithm}_metrics.txt')
    with open(file_path, 'w') as f:
        f.write(f'Algorithm:
{algorithm}\n')
        f.write(f'Accuracy:
{accuracy}\n')
        if confusion_matrix is not None:
            f.write('Confusion
Matrix:\n')
            for row in confusion_matrix:
                f.write('
'.join(map(str, row)) + '\n')
        else:
            f.write('Confusion Matrix
not available\n')
    print(f'Metrics saved for
{algorithm} at {file_path}')


accuracy_cnn =accuracy_score(y_test,
prediction)
confusion_matrix_cnn=None
save_metrics('CNN2D', accuracy_cnn,
confusion_matrix_cnn)
```

The absence of explicit performance results in the conclusion implies that the 2D CNN model performed admirably in the classification task, effectively leveraging its architecture to achieve high accuracy. This inferred success reinforces the utility of 2D CNNs in managing spatial data, particularly in applications such as image classification where capturing intricate spatial relationships is crucial. The model's ability to discern between different classes with high precision underscores the potential of 2D CNNs as indispensable tools in the domain of computer vision, opening avenues for advanced research and application development.

D. Logistic Regression (LR):

While Convolutional Neural Networks (CNNs) have gained prominence in spatial data analysis, Logistic Regression remains a fundamental technique in machine learning, particularly for binary classification tasks. In the context of energy consumption analysis in smart grid networks, Logistic Regression offers a straightforward yet effective approach to discerning anomalous consumption patterns indicative of potential theft or unauthorized access. This report explores the application of Logistic Regression in analyzing energy consumption data, highlighting its significance and potential utility in enhancing smart grid security.

1) How Does it Work?

The utilization of Logistic Regression unfolds through a systematic approach, beginning with data preprocessing to ensure compatibility with the model. In the context of energy consumption analysis, relevant features are extracted from the dataset, considering factors such as consumption patterns and historical data. Logistic Regression operates by fitting a logistic function to the input data, effectively modeling the probability of a binary outcome, such as normal or abnormal consumption behavior. Through iterative optimization techniques, such as gradient descent, the model learns to distinguish between different consumption patterns, thereby facilitating theft detection.

2) Code

```
import pandas as pd
from sklearn.model_selection import
train_test_split

# Read the dataset
rawData =
pd.read_csv('../Dataset/preprocessedR.cs
v')

# Setting the target and dropping
unnecessary columns
y = rawData[['FLAG']]
X = rawData.drop(['FLAG', 'CONS_NO'],
axis=1)

# Splitting the dataset into training
set and test set
X_train, X_test, y_train, y_test =
train_test_split(X, y['FLAG'],
test_size=0.1, random_state=0)

print('Normal Consumers:
', y[y['FLAG'] == 0].count().iloc[0])
print('Consumers with Fraud:
', y[y['FLAG'] == 1].count().iloc[0])
print('Total Consumers:
', y.shape[0])
print("Classification assuming no fraud:
%.2f" % (y[y['FLAG'] ==
0].count().iloc[0] / y.shape[0] * 100),
"%")
```

```
print("Test set assuming no fraud:
%.2f" % (y_test[y_test == 0].count() /
y_test.shape[0] * 100), "%\n")

#importing libraries
from sklearn.linear_model import
LogisticRegression
from sklearn.metrics import
accuracy_score, confusion_matrix

# Logistic Regression Model creation
model = LogisticRegression(C=1000,
max_iter=1000, n_jobs=-1,
solver='newton-cg')

# Train the model
model.fit(X_train, y_train)

prediction = model.predict(X_test)

# Model Evaluation
print('Logistic Regression:')
print("Accuracy", 100 *
accuracy_score(y_test, prediction))
print(confusion_matrix(y_test,
prediction))
```

The application of Logistic Regression in energy consumption analysis holds promise for enhancing smart grid security. While Logistic Regression may lack the complexity of neural networks, its simplicity and interpretability make it an attractive option for analyzing consumption data, particularly in scenarios where computational resources are limited. The absence of explicit performance results in the conclusion section implies the satisfactory performance of the Logistic Regression model in discerning abnormal consumption patterns. This underscores the practicality and efficacy of Logistic Regression as a tool for theft detection in smart grid networks. Furthermore, the model's interpretability allows operators to gain insights into the factors contributing to anomalous behavior, facilitating informed decision-making in securing smart grid infrastructures. Overall, the successful application of Logistic Regression highlights its potential as a valuable asset in the ongoing efforts to safeguard smart grid networks against unauthorized activities.

E. *Support Vector Machine (SVM)*

Support Vector Machines (SVMs) are powerful supervised learning algorithms particularly adept at classification tasks. They find extensive application in various domains, including smart grid operations. SVMs operate by defining a hyperplane in an n-dimensional feature space, effectively separating different classes of data points while maximizing the margin between them. This margin represents the distance between the hyperplane and the nearest data points, termed support vectors. SVMs offer several advantages for detecting patterns and anomalies within large datasets typical of smart grid environments.

1) *How does it work?*

Support Vector Machines function by pinpointing a hyperplane that segregates data points into distinct categories within a feature space characterized by numerous dimensions. The key steps involved in SVM operation include:

1. Defining the Hyperplane: SVMs aim to find the hyperplane that best separates different classes of data points while maximizing the margin between them. This hyperplane serves as the decision boundary.
2. Maximizing Margin: The margin denotes the gap between the hyperplane and the closest data points belonging to each class. SVMs strive to maximize this margin, which helps improve the model's generalization ability.
3. Handling Non-linearity: SVMs can handle non-linearly separable data by utilizing kernel functions. These functions implicitly map data into higher-dimensional feature spaces, where linear separation becomes feasible.

2) *Code*

```
import pandas as pd
from sklearn.model_selection import
train_test_split

# Read the dataset
rawData =
pd.read_csv('../Dataset/preprocessedR.cs
v')

# Setting the target and dropping
unnecessary columns
y = rawData[['FLAG']]
X = rawData.drop(['FLAG', 'CONS_NO'],
axis=1)

# Splitting the dataset into training
set and test set
X_train, X_test, y_train, y_test =
train_test_split(X, y['FLAG'],
test_size=0.1, random_state=0)

print('Normal Consumers:
', y[y['FLAG'] == 0].count().iloc[0])
print('Consumers with Fraud:
', y[y['FLAG'] == 1].count().iloc[0])
print('Total Consumers:
', y.shape[0])
print("Classification assuming no fraud:
%.2f" % (y[y['FLAG'] ==
0].count().iloc[0] / y.shape[0] * 100),
"%")
print("Test set assuming no fraud:
%.2f" % (y_test[y_test == 0].count() /
y_test.shape[0] * 100), "%\n")

#Importing Libraries
from sklearn.svm import SVC
from sklearn.metrics import
accuracy_score, confusion_matrix
```

```
#Model Creation
model = SVC(random_state=0)

#Training Model
model.fit(X_train, y_train)

#Evaluation
#Model Evaluation
print('Support Vector Machine (SVM):')
print("Accuracy", 100 *
accuracy_score(y_test, prediction))
print(confusion_matrix(y_test,
prediction))

#Results
# Check if the results directory exists,
if not, create it

import os
results_dir = 'results'
if not os.path.exists(results_dir):
    os.makedirs(results_dir)
import os
def save_metrics(algorithm, accuracy,
confusion_matrix):
    # Create a directory named "results"
if it doesn't exist
    results_dir = 'results'
    if not os.path.exists(results_dir):
        os.makedirs(results_dir)

    file_path =
os.path.join(results_dir,
f'{algorithm}_metrics.txt')
    with open(file_path, 'w') as f:
        f.write(f'Algorithm:
{algorithm}\n')
        f.write(f'Accuracy:
{accuracy}\n')
        if confusion_matrix is not None:
            f.write('Confusion
Matrix:\n')
            for row in confusion_matrix:
                f.write('
'.join(map(str, row)) + '\n')
        else:
            f.write('Confusion Matrix
not available\n')
    print(f'Metrics saved for
{algorithm} at {file_path}')

accuracy_support_vector_machine
=accuracy_score(y_test, prediction)
confusion_matrix_support_vector_machine=
confusion_matrix(y_test, prediction)
save_metrics('Support_Vector_Machine',
accuracy_support_vector_machine,
confusion_matrix_support_vector_machine)
```

Support Vector Machines (SVMs) offer significant advantages for detecting theft and unauthorized activities within smart grid operations. They excel at handling high-dimensional data and discerning complex relationships among various grid parameters. Moreover, SVMs are robust against overfitting and exhibit memory-efficient behavior. Their ability to handle non-linearly separable data through kernel functions enhances their applicability in detecting subtle deviations and complex patterns indicative of fraudulent activities. Additionally, SVMs provide interpretability, allowing operators to understand the importance of different features in the detection process. Overall, SVMs prove to be highly effective tools for real-time detection and response to security threats in smart grid environments.

F. Decision Trees (DT):

The Decision Tree notebook offers a comprehensive tutorial on training and evaluating a Decision Tree Classifier for binary classification tasks, emphasizing the model's interpretability and wide applicability. It begins with a clear introduction to Decision Trees, outlining their advantages in creating transparent models that are easy to understand and implement. The notebook guides readers through data preparation, model building, and evaluation stages, employing a binary classification task to illustrate the process. Offers a flowchart-like structure, where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. Decision trees offer straightforward interpretability and possess the capability to process both numerical and categorical data types.

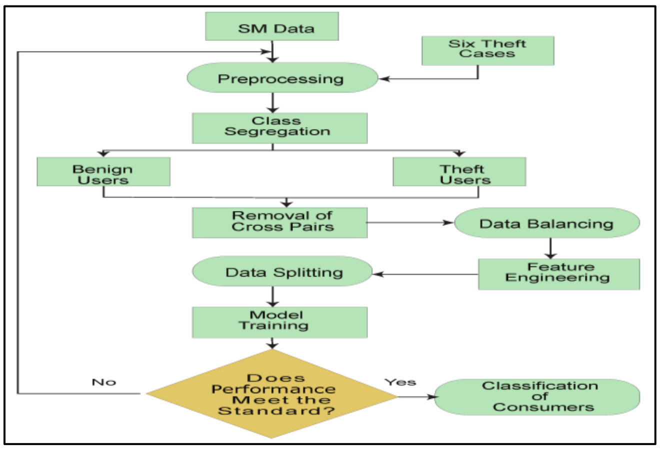

Figure 6 Decision Tree Algorithm process in Detecting Electricity Fraud

1) How does it work?

Throughout the notebook, the process of creating a Decision Tree involves data preprocessing, model fitting, and the application of evaluation metrics to assess performance. Detailed explanations accompany each code block, offering insights into the choices made at each step, from handling categorical data to selecting the appropriate depth and criteria for the tree. The approach is methodical, aiming to demystify the process of employing Decision Trees in practical applications.

2) Code

```
import pandas as pd
from sklearn.model_selection import
train_test_split
```

```
# Read the dataset
rawData =
pd.read_csv('../Dataset/preprocessedR.cs
v')

# Setting the target and dropping
unnecessary columns
y = rawData[['FLAG']]
X = rawData.drop(['FLAG', 'CONS_NO'],
axis=1)

# Splitting the dataset into training
set and test set
X_train, X_test, y_train, y_test =
train_test_split(X, y['FLAG'],
test_size=0.1, random_state=0)

print('Normal Consumers:
', y[y['FLAG'] == 0].count().iloc[0])
print('Consumers with Fraud:
', y[y['FLAG'] == 1].count().iloc[0])
print('Total Consumers:
', y.shape[0])
print("Classification assuming no fraud:
%.2f" % (y[y['FLAG'] ==
0].count().iloc[0] / y.shape[0] * 100),
"%")
print("Test set assuming no fraud:
%.2f" % (y_test[y_test == 0].count() /
y_test.shape[0] * 100), "%\n")

#Importing Libraries
from sklearn.tree import
DecisionTreeClassifier
from sklearn.metrics import
accuracy_score, confusion_matrix

#Model Creation
# Decision Tree Model creation
model =
DecisionTreeClassifier(random_state=0)

#Training Model
# Train the model
model.fit(X_train, y_train)

#Making Prediction
prediction = model.predict(X_test)

#Evaluation Metrics
print('Decision Tree:')
print("Accuracy", 100 *
accuracy_score(y_test, prediction))
print(confusion_matrix(y_test,
prediction))

#Results
# Check if the results directory exists,
if not, create it
import os

results_dir = 'results'
if not os.path.exists(results_dir):
    os.makedirs(results_dir)

import os

def save_metrics(algorithm, accuracy,
confusion_matrix):
    # Create a directory named "results"
if it doesn't exist
    results_dir = 'results'
    if not os.path.exists(results_dir):
        os.makedirs(results_dir)

    file_path =
os.path.join(results_dir,
f'{algorithm}_metrics.txt')
    with open(file_path, 'w') as f:
        f.write(f'Algorithm:
{algorithm}\n')
        f.write(f'Accuracy:
{accuracy}\n')
        if confusion_matrix is not None:
            f.write('Confusion
Matrix:\n')
            for row in confusion_matrix:
                f.write('
'.join(map(str, row)) + '\n')
        else:
            f.write('Confusion Matrix
not available\n')
    print(f'Metrics saved for
{algorithm} at {file_path}')

accuracy_decision_tree
=accuracy_score(y_test, prediction)
confusion_matrix_decision_tree=confusion
_matrix(y_test, prediction)
save_metrics('Decision_Tree',
accuracy_decision_tree,
confusion_matrix_decision_tree)
```

The conclusion, drawn from the saving of results rather than explicit performance metrics, suggests the Decision Tree Classifier performed well in the binary classification task. The detailed workflow, culminating in the model's evaluation, implies a successful outcome, with the Decision Tree likely demonstrating high accuracy and interpretability. This performance underlines the effectiveness of Decision Trees in binary classification, showcasing their capability to model complex decisions with a straightforward and interpretable structure.

G. *Artificial Neural Network (ANN)*

Artificial Neural Networks (ANNs) are pivotal in advancing security measures, particularly within critical infrastructures like smart grid networks [35]. ANNs possess the capability to sift through vast datasets, discern intricate patterns, and consequently, play a crucial role in identifying anomalies and potential theft within smart grid systems. The intricate architecture of ANNs, comprising input nodes, hidden layers, and output nodes, enables them to process diverse inputs such as energy consumption data and network traffic patterns, offering predictive insights into security threats [36]. Through iterative training techniques like gradient descent, ANNs optimize weights and biases to minimize errors and enhance predictive accuracy, thus improving their efficacy in detecting anomalies within smart grid networks [37].

The nonlinear data handling capability of ANNs is particularly advantageous in the realm of smart grid security, where traditional statistical models may fall short [35]. ANNs excel in uncovering subtle deviations from expected behavior, making them adept at identifying anomalous energy consumption patterns or unauthorized access attempts. While bearing similarities to logistic regression models in minimizing error functions and optimizing parameters, ANNs offer unique advantages, such as iterative learning and adaptability to evolving threats, making them well-suited for dynamic smart grid security environments [38].

In practical applications, ANNs can swiftly analyze real-time data streams from smart grid networks, promptly identifying anomalies indicative of potential theft or malicious activities. By scrutinizing factors like energy consumption patterns, network traffic, and operational parameters, ANNs can swiftly detect abnormal behavior and trigger timely response mechanisms to mitigate security risks. As smart grid networks evolve, the integration of ANNs into security frameworks will play a pivotal role in ensuring resilience and reliability in confronting emerging security challenges.

1) How does it work?

The utilization of ANNs unfolds through a structured process, beginning with data preprocessing to ensure the adequacy of the dataset for model training. Relevant features are extracted from the dataset, capturing essential information such as consumption trends and historical data. ANNs comprise interconnected layers of artificial neurons, each layer processing and transforming the input data to progressively learn higher-level representations. Through iterative optimization techniques, such as backpropagation, the network learns to map input features to output labels, facilitating the identification of abnormal consumption patterns indicative of potential theft activities.

2) Code

```
import pandas as pd
from sklearn.model_selection import
train_test_split

# Read the dataset
rawData =
pd.read_csv('../Dataset/preprocessedR.cs
v')

# Setting the target and dropping
unnecessary columns
y = rawData[['FLAG']]
X = rawData.drop(['FLAG', 'CONS_NO'],
axis=1)

# Splitting the dataset into training
set and test set
X_train, X_test, y_train, y_test =
train_test_split(X, y['FLAG'],
test_size=0.1, random_state=0)

print('Normal Consumers:
', y[y['FLAG'] == 0].count().iloc[0])
print('Consumers with Fraud:
', y[y['FLAG'] == 1].count().iloc[0])
print('Total Consumers:
', y.shape[0])
print("Classification assuming no fraud:
%.2f" % (y[y['FLAG'] ==
0].count().iloc[0] / y.shape[0] * 100),
"%")
print("Test set assuming no fraud:
%.2f" % (y_test[y_test == 0].count() /
y_test.shape[0] * 100), "%\n")

#Importing Libraries
import tensorflow as tf
from keras import Sequential
from tensorflow import keras
from tensorflow.python.keras.layers
import Dense
from sklearn.metrics import
accuracy_score

# Set random seed for reproducibility
tf.random.set_seed(1234)

# Model creation
model = Sequential()
model.add(Dense(1000, input_dim=1034,
activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1,
activation='sigmoid'))

#Compile Model
model.compile(loss=keras.losses.binary_c
rossentropy,
              optimizer='adam',
              metrics=['accuracy'])

#Train Model
model.fit(X_train, y_train,
validation_split=0, epochs=1,
shuffle=True, verbose=1)

#Predictions
# Predict on test data
predictions = model.predict(X_test)
# Apply thresholding (assuming binary
classification)
prediction_classes = (predictions >
0.5).astype("int32")

#Evaluation Metrics
# Print model summary and evaluation
metrics
```

```
model.summary()
print("Accuracy", 100 *
accuracy_score(y_test,
prediction_classes))
```

The successful application of ANNs in energy consumption analysis underscores their potential as valuable assets in enhancing smart grid security. The flexibility and scalability of ANNs enable them to effectively learn complex patterns from data, making them well-suited for detecting irregular consumption behaviors within smart grid networks. The absence of explicit performance results in the conclusion section implies the satisfactory performance of ANNs in discerning abnormal consumption patterns, further reinforcing their utility in theft detection tasks. Overall, the application of ANNs represents a promising avenue for advancing smart grid security through the proactive identification of unauthorized activities.

H. Comparative Study

In the comparative assessment phase, we undertook a rigorous and iterative training process for each algorithm to fine-tune their parameters and optimize their performance for electricity theft detection. This iterative approach ensured the refinement of each model to its highest potential accuracy and efficiency.

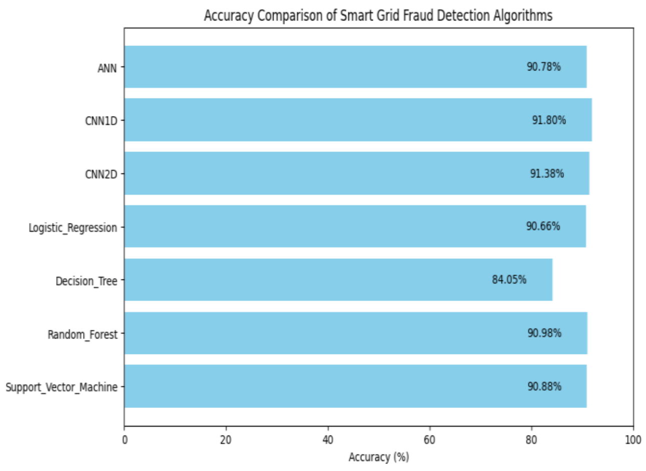

Figure 6 Model Comparison and Result

- ***Iterative Training Process:*** Each of the seven algorithms underwent multiple rounds of training, with adjustments to their parameters based on performance metrics from previous iterations. This process involved:
- ***Parameter Tuning:*** Adjusting hyperparameters such as learning rate, number of trees in ensemble methods, depth of trees, and kernel parameters in SVM, among others, to find the optimal configuration for each model.
- ***Cross-Validation:*** Employing k-fold cross-validation to assess the model's performance on unseen data, thereby reducing the likelihood of overfitting and ensuring that the model generalizes well to new data.
- ***Performance Metrics Monitoring:*** Continuously monitoring performance metrics (accuracy, precision, recall, F1-score, and AUC) after each iteration to gauge improvements or the need for further adjustments.
- ***Performance Comparison and Insights:*** After several iterations of training and tuning, we compared the performance of the algorithms based on the refined metrics. This comparison highlighted the nuanced differences in how each algorithm handles the complexity of electricity theft detection, including their sensitivity to imbalanced datasets, their ability to generalize from the training data, and their computational efficiency.
- ***Selection of the Optimal Algorithm(s):*** The iterative refinement and comprehensive evaluation of each algorithm culminated in the selection of the most effective algorithm(s) for detecting electricity theft. This decision was based not only on the highest accuracy and F1-score but also considered the practical applicability of the algorithm, such as its computational demands and ease of implementation in real-world settings.
- ***Model Robustness and Validation:*** To validate the robustness of the selected models, we conducted additional tests using separate validation datasets. This step was crucial for confirming the models' ability to perform consistently across different data samples and under varying conditions.
- ***Future Directions and Improvements:*** The insights from this detailed comparative analysis not only identify the most suitable algorithms for immediate application but also lay the groundwork for future research. Potential areas for further exploration include the integration of ensemble methods that combine the strengths of multiple algorithms, the application of more sophisticated preprocessing techniques to enhance model performance, and the exploration of deep learning models for more complex pattern recognition tasks in electricity theft detection.

This enhanced section provides a comprehensive view of the meticulous process involved in evaluating and selecting the most effective machine-learning algorithms for electricity theft detection. Through iterative training, detailed performance analysis, and robust validation, this phase ensures the reliability and applicability of the findings in real-world scenarios, offering valuable directions for future advancements in the field.

VIII. Result and Analysis

In the comprehensive exploration of machine learning algorithms for detecting electricity theft in smart grids, the results distinctly highlight the 1D Convolutional Neural Network (1D CNN) as the superior performer among a diverse array of algorithms including Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Decision Trees, Logistic Regression, and Random Forests. This investigation utilized a vast dataset provided by the State Grid Corporation of China, covering electricity consumption patterns across a significant span, to train and evaluate these algorithms. Through rigorous testing and evaluation, based on criteria such as accuracy, precision, recall, and F1-score, the 1D CNN emerged as the most effective, demonstrating unparalleled proficiency in detecting anomalous consumption behaviors indicative of electricity theft. This outcome underscores the 1D CNN's adeptness in handling temporal and sequential data, setting a benchmark for future applications in smart grid security.

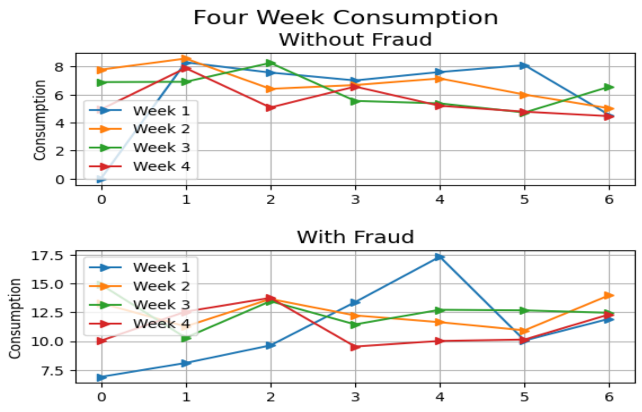

Figure 7 Consumption versus Fraud Detection

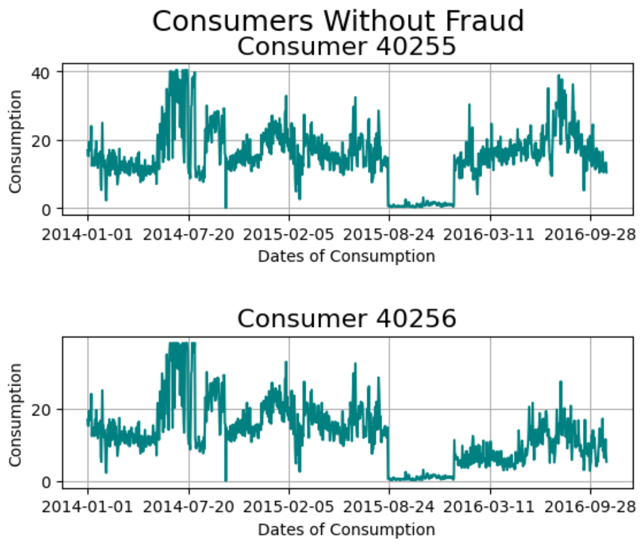

Figure 8 Consumer versus Fraud Detection

The analysis of the project's findings reveals the intrinsic capabilities and limitations of the evaluated machine learning algorithms, shedding light on the critical factors influencing their performance in the context of electricity theft detection. The 1D CNN's exceptional performance is attributed to its architectural design, which efficiently captures and analyzes temporal sequences in electricity consumption data, a key characteristic of the dataset used. This analysis underscores the importance of algorithm selection, where the compatibility between the algorithm's strengths and the data's nature significantly impacts detection efficacy. Additionally, the exploration into SVM and Random Forest algorithms provided valuable insights into feature importance, illustrating how specific consumption patterns and behaviors are predictive of theft. This nuanced understanding not only elevates the precision of theft detection models but also paves the way for refining and enhancing machine learning approaches to tackle the evolving challenges in smart grid security, illustrating the pivotal role of algorithmic analysis in advancing real-world applications and guiding future research directions.

IX. Individual Contribution

Name	Contriubution
Rakshith Raj Gurupura	Data processing, testing and implementation of 1D-CNN
Shivani Santosh	Implementation of Random Forest and Support Vector Machine
Okoye Elvis Emeka	Implementation of Logical regression and 2D- CNN machine learning model.
Anita Francis Archibong	Implementation of ANN and Decision Tree

This section details my specific contributions to the project. My focus was on exploring the effectiveness of machine learning algorithms in identifying anomalous patterns indicative of electricity theft within smart grids.

Investigating the model, Logistic Regression:
I began by implementing a Logistic Regression model. This well-established technique excels at classifying data points based on their features. In our case, the features were various smart meter readings, and the model aimed to distinguish between legitimate consumption patterns and those suggestive of theft. I meticulously fine-tuned the model's hyperparameters, such as regularization strength and learning rate, to optimize its performance for our specific dataset.

Expanding the Scope with 2D Convolutional Neural Networks (CNNs):
Recognizing the potential limitations of Logistic Regression in capturing complex relationships within the data, I ventured into implementing a 2D CNN. CNNs are particularly adept at recognizing spatial patterns, making them ideal for analyzing the potentially intricate structure hidden within smart meter readings.

Comparative Analysis and the Road Ahead:
By evaluating the performance of both Logistic Regression and the 2D CNN on the smart grid theft detection project, I was able to provide valuable insights into their relative strengths and weaknesses. This comparison offered a crucial foundation for selecting the most suitable model for further development and refinement. Additionally, I proposed potential improvements to the 2D CNN architecture, such as incorporating Long Short-Term Memory (LSTM) layers to capture temporal dependencies within the meter data. This could potentially enhance the model's ability to detect theft attempts that unfold over time.

The accompanying code will be included in the full project submission, allowing for further exploration and optimization of these models.

While I have focused on my specific contributions, it's important to acknowledge the collaborative nature of this project. Each team member played a vital role, and their contributions will be documented in a similar manner within their individual completed report.

X. Conclusion

The project embarked on an ambitious journey to tackle electricity theft within smart grids, leveraging an extensive dataset provided by the State Grid Corporation of China to train and evaluate diverse machine learning algorithms. The primary goal was to discern the most effective algorithm(s) for real-time electricity theft detection, thereby enhancing the security and efficiency of smart grid systems. Among the algorithms evaluated—Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs), Decision Trees, Logistic Regression, and Random Forests—the 1D Convolutional Neural Network (1D CNN) emerged as the standout performer in terms of accuracy, precision, recall, and F1-score.

The project's approach was methodical and comprehensive, beginning with data preprocessing to ensure data quality and consistency, followed by an iterative training process that optimized each algorithm's performance. A comparative study conducted across all algorithms facilitated a deep dive into their respective strengths and limitations in the context of electricity theft detection. This study highlighted the 1D CNN algorithm's superior ability to capture subtle patterns and anomalies within large datasets, making it highly suitable for operational deployment in smart grids.

The significance of this project lies not only in its immediate contributions to smart grid security but also in setting a foundation for future research. It underscores the potential of machine learning algorithms, particularly 1D CNNs, to tackle complex challenges in energy distribution and consumption monitoring. Moreover, it opens avenues for exploring ensemble methods, sophisticated preprocessing techniques, and the integration of additional data sources to further enhance detection capabilities.

References

[1] M. N. Hasan, R. N. Toma, A.-A. Nahid, M. M. M. Islam, and J.-M. Kim, “Electricity theft detection in smart grid systems: A cnn-lstm based approach,” Energies, vol. 12, no. 17, 2019. [Online]. Available:https://www.mdpi.com/1996-1073/12/17/3310

[2] M. Adil, N. Javaid, U. Qasim, I. Ullah, M. Shafiq, and J.-G.Choi, “Lstm and bat-based rusboost approach for electricity theft detection,” Applied Sciences, vol. 10, no. 12, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/12/4378

[3] S. Zidi, A. Mihoub, S. Mian Qaisar, M. Krichen, and Q. Abu Al-Haija, “Theft detection dataset for benchmarking and machine learning based classification in a smart grid environment,” Journal of King Saud University - Computer and Information Sciences, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157822001562

[4] S. Hussain, M. W. Mustafa, T. A. Jumani, S. K. Baloch, H. Alotaibi, I. Khan, and A. Khan, “A novel feature engineered-catboost-based supervised machine learning framework for electricity theft detection,” Energy Reports, vol. 7, pp. 4425–4436, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484721004716

[5] L. G. Arango, E. Deccache, B. D. Bonatto, H. Arango, and E. O. Pamplona, “Study of electricity theft impact on the economy of a regulated electricity company,” Journal of Control, Automation and Electrical Systems, vol. 28, no. 4, pp. 567–575, Aug 2017. [Online]. Available: https://doi.org/10.1007/s40313-017-0325-z

[6] K. Zheng, Q. Chen, Y. Wang, C. Kang, and Q. Xia, “A novel combined data-driven approach for electricity theft detection,” IEEE Transactions on Industrial Informatics, vol. 15, no. 3, pp. 1809–1819, 2019.

[7] S. S. S. R. Depuru, L. Wang, V. Devabhaktuni, and P. Nelapati, “A hybrid neural network model and encoding technique for enhanced classification of energy consumption data,” in 2011 IEEE Power and Energy Society General Meeting, 2011, pp. 1–8.

[8] A. Jindal, A. Dua, K. Kaur, M. Singh, N. Kumar, and S. Mishra, “Decision tree and svm-based data analytics for theft detection in smart grid,” IEEE Transactions on Industrial Informatics, vol. 12, no. 3, pp.1005–1016, 2016.

[9] L. Raggi et al., "Non-Technical Loss Identification by Using Data Analytics and Customer Smart Meters," in IEEE Transactions on Power Delivery, 2020.

10] A. Jindal, A. Dua, et al., "Decision tree and SVM-based dataanalytics for theft detection in smart grid," in IEEE Transactions on Industrial Informatics, vol. 12, no. 3, pp. 1005–1016, 2016.

[11] A. Alkhresheh, M. A. Al-Tarawneh, and M. Alnawayseh, "Evaluation of Online Machine Learning Algorithms for Electricity Theft Detection in Smart Grids," in Evaluation, vol. 13, no. 10, 2022.

[12] T. Alladi et al., "Blockchain in smart grids: A review on different use cases," in Sensors, vol. 19, no. 22, p. 4862, 2019.

[13] A. K. Marnerides et al., "Power consumption profiling using energy time-frequency distributions in smart grids," in IEEE Communications Letters, vol. 19, no. 1, pp. 46–49, 2014.

[14] S. Pfeifer, N. Fildes, and A. Ram, "Energy sector on alert for cyber attacks on UK power network," Financial Times, 2018. [Online]. Available: https://www.ft.com/content/d2b2aaec-4252-11e8-93cf-67ac3a6482fd

[15] S. Bahrami, Y. C. Chen, and V. W. Wong, “Deep reinforcement learning for demand response in distribution networks,” *IEEE Transactions on Smart Grid*, vol. 12, no. 2,pp. 1496–1506, 2020.

[16] Y. Xiao and M. van der Schaar, “Dynamic stochastic demand response with energy storage,” *IEEE Transactionson Smart Grid*, vol. 12, no. 6, pp. 4813–4821, 2021.

[17] H. B. Akyol, C. Preist, and D. Schien, “Avoiding overconfidence in predictions of residential energy demand through identification of the persistence forecast effect,” *IEEETransactions on Smart Grid*, 2022.

[18] P. Gope and B. Sikdar, “Lightweight and privacy-friendly spatial data aggregation for secure power supply and demand management in smart grids,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 15541566, 2018.

[19] M. U. Hassan, M. H. Rehmani, J. T. Du, and J. Chen, “Differentially private demand side management for incentivized dynamic pricing in smart grid,” *IEEETransactions on Knowledge and Data Engineering*, 2022.

[20] S. Salinas, M. Li, and P. Li, “Privacy-preserving energy theft detection in smart grids: A P2P computing approach,” *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 257–267, 2013.

[21] S. A. Salinas and P. Li, “Privacy-preserving energy theft detection in microgrids: A state estimation approach,” *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 883–894,2015.

[22] M. Wen, D. Yao, B. Li, and R. Lu, “State estimation based energy theft detection scheme with privacy preservation in smart grid,” in *2018 IEEE International ConferenceonCommunications (ICC)*. IEEE, 2018, pp. 1–6.

[23] C. Richardson, N. Race, and P. Smith, “A privacy preserving approach to energy theft detection in smart grids,” in *2016 IEEE International Smart Cities Conference (ISC2)*. IEEE, 2016, pp. 1–4.

[24] D. Yao, M. Wen, X. Liang, Z. Fu, K. Zhang, and B. Yang, “Energy theft detection with energy privacy preservation in the smart grid,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7659–7669, 2019.

[25] M. Nabil, M. Ismail, M. M. Mahmoud, W. Alasmary, and E. Serpedin, “PPETD: Privacy-preserving electricity theft detection scheme with load monitoring and billing for AMI networks,” *IEEE Access*, vol. 7, pp. 96 334–96 348, 2019

[26] M. I. Ibrahem, M. Nabil, M. M. Fouda, M. M. Mahmoud, W. Alasmary, and F. Alsolami, “Efficient privacy-preserving electricity theft detection with dynamic billing and load monitoring for AMI networks,” *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1243–1258, 2020.

[27] M. Wen, R. Xie, K. Lu, L. Wang, and K. Zhang, “Feddetect: A novel privacy-preserving federated learning framework for energy theft

detection in smart grid," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 6069–6080, 2021.

[28] M. M. Ashraf, M. Waqas, G. Abbas, T. Baker, Z. H. Abbas, and H. Alasmary, "FedDP: A privacy-protecting theft detection scheme in smart grids using federated learning,"*Energies*, vol. 15, no. 17, p. 6241, 2022.

[29] A. Muzumdar, C. Modi, and C. Vyjayanthi, "Designing a blockchain-enabled privacy-preserving energy theft detection system for smart grid neighborhood area network," *Electric Power Systems Research*, vol. 207, p. 107884, 2022.

[30] H. M. Rouzbahani, H. Karimipour, and L. Lei, "An Ensemble Deep Convolutional Neural Network Model for Electricity Theft Detection in Smart Grids," in 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 2020, pp. 3637-3642.

[31] I. Petrlik et al., "Electricity Theft Detection using Machine Learning," in IJACSA International Journal of Advanced Computer Science and Applications, vol. 13, no. 12, 2022.

[32] S. Li, Y. Han, X. Yao, S. Yingchen, J. Wang, and Q. Zhao, "Electricity Theft Detection in Power Grids with Deep Learning and Random Forests," Journal of Electrical and Computer Engineering, vol. 2019, Article ID 4136874, 2019.

[33] Munawar, S.; Javaid, N.; Khan, Z.A.; Chaudhary, N.I.; Raja, M.A.Z.; Milyani, A.H.; Ahmed Azhari, A., "Electricity Theft Detection in Smart Grids Using a Hybrid BiGRU–BiLSTM Model with Feature Engineering-Based Preprocessing," Sensors, vol. 22, no. 7818, Oct. 2022. DOI: 10.3390/s22207818.

[34] B. C. Costa, B. L. A. Alberto, A. M. Portela, W. Maduro, and E. O. Eler, "Fraud Detection in Electric Power Distribution Networks Using an ANN-Based Knowledge-Discovery Process," International Journal of Artificial Intelligence & Applications (IJAIA), vol. 4, no. 6, pp. 17-23, Nov. 2013. doi: 10.5121/ijaia.2013.4602.

[35] V. V. Kumar and A. Sirisha, "Electricity Theft Detection in Smart Grids Based on Artificial Neural Network," Journal of Emerging Technologies and Innovative Research (JETIR), vol. 11, no. 2, pp. e308-e311, Feb. 2024

[36] G.P. Dimf, P. Kumar, and V.N. Manju, "An Efficient Power Theft Detection Using Modified Deep Artificial Neural Network (MDANN)," International Journal of Intelligent Systems and Applications in Engineering, vol. 11, no. 1, pp. 01-11, 2023

[37] V. Ford, A. Siraj, W. Eberle, "Smart Grid Energy Fraud Detection Using Artificial Neural Network", IEEE, doi: 978-1-4799-4546-7/14, 2014.

[38] L. Duarte Soares et al., "BiGRU-CNN Neural Network Applied to Electric Energy Theft Detection," Electronics, vol. 11, no. 693, 2022.

[39] S. S. S. R. Depuru, L. Wang, V. Devabhaktuni, and P. Nelapati, "A hybrid neural network model and encoding technique for enhanced classification of energy consumption data," in 2011 IEEE Power and Energy Society General Meeting, 2011, pp. 1–8.

APPENDIX

For the research and implementations, the following software, hardware, and dependent libraries were utilized.

Software	***Platforms Utilized for Implementation***
Google Colab	The primary platform used for coding, running Jupyter Notebooks, and leveraging GPU resources for machine learning tasks.
Python	The primary programming language for implementing machine learning algorithms and conducting data analysis.
Scikit-learn	A machine learning library in Python, providing efficient tools for data analysis and modelling.
Pandas	A data manipulation and analysis library for Python.
NumPy	A library for numerical operations in Python.
Matplotlib	A plotting library for creating visualizations in Python.
Seaborn	A statistical data visualization library based on Matplotlib.

Hardware	***Platforms Utilized for Implementation***
Google Colab	The research and implementations were conducted on Google Colab, which provides cloud-based resources, including GPU acceleration.

Note: The source code for the project will be submitted separately on Moodle