ELECTRICITY THEFT DETECTION USING MACHINE LEARNING

A Project Submitted

In Partial Fulfilment of the Requirement

For the Degree of

BACHELOR OF TECHNOLOGY

in

Electrical Engineering

by Group 5

Vaishnavi Saxena (1900520400064)

Ankit Shivam (1900520200012)

Manzoor Hussain (1900520200028)

Under the Supervision of

Dr. Pushkar Tripathi

(Assistant Professor, Electrical Engineering Department)

to the

Department Of Electrical Engineering

INSTITUTE OF ENGINEERING & TECHNOLOGY

(Sitapur road, Lucknow, Uttar Pradesh, India)

June,2023

DECLARATION

We, Vaishnavi Saxena (1900520400064), Manzoor Hussain (1900520200028), and Ankit Shivam (1900520200012), hereby affirm that the work presented in this project report, with the title **"ELECTRICITY THEFT DETECTION USING MACHINE LEARNING"** submitted to the Department of Electrical Engineering, Institute of Engineering and Technology, Lucknow (India), for the award of a Bachelor of Technology degree in Electrical Engineering, is our original work. We did not submit the same work for the award of any other degree somewhere or plagiarise any section of the thesis. If this assignment is proven to be inaccurate, the degree will be automatically revoked.

Vaishnavi Saxena (1900520400064)

Manzoor Hussain (1900520200028)

Ankit Shivam (1900520200012)

CERTIFICATE

This is to certify that **Vaishnavi Saxena (1900520400064), Ankit Shivam (1900520200012) and Manzoor Hussain (1900520200028)** have carried out the project work presented in this project entitled, **"Electricity Theft Detection Using Machine Learning"** for the award of the degree of **Bachelor of Technology** in **Electrical Engineering** at **Institute of Engineering & Technology, Lucknow** under my supervision. The project embodies results of original work, and studies are carried out by the students themselves and the contents of the project do not form the basis of the award of any other degree to the candidates or to anybody else from this university/institution.

Date:

Place:
Lucknow

Dr Pushkar Tripathi
Assistant Professor
EED, IET

ABSTRACT

Electricity theft is a significant problem in the energy sector, causing financial losses for utility companies and posing safety risks. Traditional methods for detecting electricity theft have limitations in terms of accuracy and efficiency. In recent years, the integration of machine learning techniques has shown promise in improving the detection of electricity theft. In this project, we explore the application of machine learning algorithms for detecting electricity theft in smart grid data. Our objective is to identify the most effective algorithm among these three models. The dataset was preprocessed to handle missing values, normalize features, and ensure compatibility with the chosen machine learning algorithms. We performed an extensive evaluation of SVM, KNN, and CNN on the dataset. Each algorithm was trained, validated, and tested using appropriate methodologies. We assessed their performance using key metrics such as accuracy, precision, recall, and F1 score to determine their effectiveness in detecting electricity theft. Our findings reveal that CNN outperformed SVM and KNN in terms of overall detection accuracy and robustness. CNN's ability to automatically learn and extract relevant features from the smart grid data played a crucial role in its superior performance. This project contributes to the field of electricity theft detection by highlighting the effectiveness of machine learning algorithms, particularly CNN, in improving the detection accuracy and reliability. The results emphasize the importance of leveraging advanced techniques to combat electricity theft, thereby enabling utility companies to mitigate financial losses and enhance grid security. The findings of this project have implications for the energy sector, policymakers, and researchers working in the domain of smart grid security. By implementing the identified best-performing algorithm, utility companies can enhance their ability to detect and prevent electricity theft, resulting in improved operational efficiency and customer satisfaction.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who have contributed to the successful completion of our project on Electricity Theft Detection using Machine Learning. Their guidance, support, and encouragement have been instrumental in the accomplishment of this work.

First and foremost, we would like to extend our heartfelt appreciation to our project mentor, Dr. Pushkar Tripathi. His expertise in the field of machine learning, valuable insights, and continuous guidance throughout the project have been invaluable. We are truly grateful for his patience, encouragement, and dedication in helping us navigate through the challenges and ensuring the quality of our work.

We would also like to acknowledge the project coordinators, Dr. Nitin Anand Shrivastava and Dr. Shweta, for their valuable inputs and suggestions that have enhanced the project's outcome. Their expertise and guidance have been crucial in shaping our understanding and approach to the problem of electricity theft detection

Furthermore, we extend our appreciation to the faculty members of the Department of Electrical Engineering at the Institute of Engineering and Technology, Lucknow, for providing us with a conducive learning environment and the necessary resources to undertake this project. Their teachings and support have been instrumental in our academic growth and the successful execution of this project.

The collective effort and teamwork have been essential in accomplishing the project objectives and delivering a high-quality outcome. We are also grateful to all the individuals, organizations, and researchers whose work and contributions have been referenced in our project report. Their research and insights have provided us with a solid foundation and a broader perspective on the topic of electricity theft detection.

Date :

Vaishnavi Saxena, 1900520400064

Manzoor Hussain, 1900520200028

Ankit Shivam, 1900520200012

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ETD	Electricity Theft Detection
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
CNN	Convolutional Neural Network
AI	Artificial Intelligence
GPU	Graphics Processing Unit
SGCC	State Grid Corporation of China
SMOTE	Synthetic Minority Oversampling Technique
FA-XGboost	Firefly Algorithm based Extreme Gradient Boosting
LSTM	Long Short-Term Memory
VGG-16	Visual Geometry Group

LR	Logistic Regression

CHAPTER 1

INTRODUCTION

1.1. Objectives

i. To use data driven approach to secure power distribution systems.

ii. To study and analyse the different machine learning algorithms for electricity theft detection like ***Support Vector Machine (SVM), K Nearest Neighbour (KNN) and Convolutional neural networks (CNN).***

iii. To do a comparative analysis of all the implemented algorithms on the basis of accuracy, time consumed, hyperparameters used, no. of epochs used and other performance metrics and suggest the best algorithm to detect normal users and theft users.

iv. By leveraging data from smart meters, we aim to train models that can automatically identify suspicious consumption patterns, distinguish between legitimate usage and unauthorized activities, and provide timely alerts to utility companies for further investigation.

1.2 Electricity Theft Detection

Nowadays, both the industrial sector and the home require power as a basic necessity. Power plants produce electricity, which is then sent through the transmission network to customers. The competent and sustainable use of energy assets is a crucial component of the commercial development of any nation due to the limited supply and high cost of energy resources. The development, management, and reliability of the energy system are just a few of the difficulties it faces. Electricity theft has grown to be a significant problem for the electricity power utility. These electricity thieves make it more difficult for the power company to serve its customers with electricity.

The smart grid is currently a crucial component of developing an effective energy monitoring system. The term "smart grid" refers to an electrical network that includes infrastructure and a power system network, as well as sophisticated devices to manage and control energy consumption and a smart detecting system that keeps detect of how much electricity each consumer linked to the system uses. The smart grid contains some advancement that will help in developing the effectiveness, reliability and ensure the

continuity of energy to households and the industries. By combining current electrical infrastructure with the contemporary digital technology, the smart grid provides the facility to the power utilities and users to invigilate, manage, and individuate electricity energy consumption. The electrical energy and users' data flows in smart grids connect consumers and power utilities together. The bidirectional information and energy exchange between the consumers and utilities is an important Energy Internet (EI) component. Losses of electricity regularly happen during the production, distribution and transmission of electricity. The various types of electrical energy losses are classified as the (TLs) and non-technical losses (NTLs). Technical losses appear due to the joules effect on the transmission lines and transformer losses while the transmission of electricity from one place to another place. It is challenging to pinpoint the site of loss and calculate the quantity of energy losses due to the complexity involved in the identification and calculation of technical losses. Although technical losses cannot be totally eliminated, they can be managed and decreased by implementing various modifications and cutting-edge procedures across the entire energy network system. More than twice as much as the global average in terms of technical losses, which typically range from 6% to 8%, were experienced in India, where they accounted for 20% of the generation.

The losses resulting from electricity energy theft, malfunctioning energy metres, fraud, and unpaid bills are referred to as non-technical losses (NTL). The most important non-technical loss is electricity theft. Bypassing the electricity energy metre, fiddling with the reading, or digitally hacking the smart metre are the typical methods of power theft. Cyber-tempering the smart metre has recently become a method for altering the energy consumption data of the smart metres, which results in the consumers receiving the fewest bills possible. Even though there are only a few customers who cause this electrical energy loss, it lowers the profitability and electrical energy competence of the power utility businesses, leading to issues like load shedding and inflation, among others. In the USA, Russia and Brazil the NTL has been 6%, 10% and 16% respectively. According to the World Bank the losses of electrical energy due to theft in India is more than 25% of the power supply losses. A survey found that 20% of thefts happen on commercial and industrial properties while 80% happen in individual residences.

The detection of electricity theft using machine learning is a widely used method in modern times. The machine learning techniques only use the data set of electricity consumers to detect the theft. It uses the previous electricity energy consumption data to train the model and identify the abnormal electricity consumption behaviour. This method uses the data collected by the smart meter without supplementary monitoring equipment, thus making it efficient and reduced manpower costs. However, machine learning has its own limitations. So, to reduce this limitation different algorithms and methods are used to improve the accuracy of electricity theft detection.

1.3 Machine Learning

Machine learning is a field of study that focuses on developing algorithms and models that enable computer systems to learn and make predictions or decisions without being explicitly programmed. It involves the use of statistical and computational techniques to analyze and interpret complex data, identify patterns and relationships, and generate insights or predictions. Machine learning algorithms are designed to iteratively learn from data, improve their performance over time, and adapt to new information. This technology finds

applications in various domains, including image and speech recognition, natural language processing, recommendation systems, and predictive analytics.

1.3.1. Types of Machine Learning

There are four categories of Machine learning:

a. Supervised learning: Modeling is used to train the computer while supervised learning. The machine learning algorithm later determine how to approach the inputs and outputs once the operator provides it a known dataset with the appropriate inputs and outputs. While the algorithm searches for patterns in the data, gain experience, and predicts outcomes, the operator is aware of the appropriate solutions to the problem. The operator adjusts the algorithm when predictions are produced, and this process is continued until the strategy is effective and incredibly perfect.

Regression, classification, and forecasting are all examples of supervised learning.

Classification: To perform tasks related to classification, a machine learning algorithm must define a outcome from values that have already been observed and determine the category of newly input data. For example, the computer needs to observe at recent data and classify emails as "spam" or "not spam" from that data.
Regression: In regression tasks, the machine learning system must understand the relationships between the variables. Regression analysis target one dependent variable and a figure of other varying factors, making it particularly helpful for prediction and forecasting.
Forecasting: A common tactics for trend analysis is forecasting, which comprises making predictions for the future based on information from the past and present.

b. Semi-supervised learning: Analogous to supervised learning, semi-supervised learning makes use of both labelled and unlabelled data. Unlabelled data lacks any relevant tags, but labeled data has them so the computer can understand it. Using this combination, machine learning algorithms can learn to categorize unlabelled data.

c. Unsupervised learning: In this learning the data is examined by the machine learning algorithm to discover trends. There is no live operator or answer book to provide assistance. Instead, the machine analyses the data at hand to find the correlations and links. A machine learning algorithm is given a lot of data to assess and respond to as needed in an unsupervised learning process. The programme tries to arrange the data in some way to describe the structure of the data. This can mean grouping the material together or arranging it such that it comes out more organized. As it analyses more facts, it continuously becomes more skilled at taking decisions based on it.

d. Reinforcement learning: It concentrates on organized learning processes by providing a machine learning algorithm with a set of actions, parameters, and end values. The machine learning algorithm then attempts to explore several options and possibilities

after creating the rules, monitoring and analysing each output to decide which the best is. Through reinforcement learning, the machine is educated by trial and error. It draws lessons from previous mistakes and starts to modify its strategy in reaction to the circumstance in order to get the optimal outcome.

1.4 Tech Stacks

1.4.1 Python

Python is a high-level programming language known for its simplicity and readability. It has gained popularity in various fields, including data science and machine learning, due to its extensive libraries and frameworks. Python provides a user-friendly syntax, making it easy to write and understand code. It supports multiple programming paradigms and has a vast ecosystem of libraries that facilitate tasks like data manipulation,
visualization, and machine learning model development.

1.4.2 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows interactive coding, data exploration, and documentation. It provides an interactive environment where you can write and execute code in cells, along with text, images, and visualizations. Jupyter Notebook supports multiple programming languages, including Python, and allows you to create and share documents containing live code, equations, and explanatory text.
It is widely used in data science and machine learning for prototyping, experimenting, and presenting results.

1.4.3 Google Colab

Google Colab (short for Colaboratory) is a cloud-based platform provided by Google that allows you to write, run, and share Python code. It offers a Jupyter Notebook-like interface with additional features like free access to GPUs and TPUs for accelerated computations. Google Colab provides pre-installed libraries, including popular machine learning frameworks like TensorFlow and PyTorch. It allows collaborative editing and sharing of notebooks, making it convenient for team projects or educational purposes.

1.5. Literature Survey

There are mainly seven research papers which have inspired us for our project.

1. The research presents five different machine learning algorithms like K nearest neighbour, Random Forest, Artificial neural networks, Decision tree and bagging to identify electricity theft by multi- classification. They have identified sixteen unique types of consumers. The paper had used the latest dataset of electricity consumption of 200 in the United States [1].

2. The research has used Convolutional neural networks algorithm for electricity theft detection. They have proposed a blue monkey algorithm for data pre-processing. This paper had a key focus on data cleaning as it is the most essential part of machine learning. There is no way to identify a theft user [2].

3. The electricity consumption data was collected from SGCC. The data is preprocessed using interpolation, three sigma rule and normalized methods. Adasyn algorithm is also utilized to manage class imbalance problem. A VGG-16 module is used to identify anomalous patterns in electricity consumption. For classification, FA-XGBoost technique is used. This classifier has achieved F1 score, precision and recall of 93.7%, 92.6% and 97% respectively. The comparative analysis with SVM, CNN and LR has also been done. The highest performance was attained by FA-XGBoost algorithm. The execution time increases if the dataset increases[3].

4. A multi head self-attention mechanism combined with dilated convolutions and unified by a convolution of kernel size. A binary mask is used to handle missing values in the data of SGCC. The AUC attained in this work is 0.926. They have higher bias [4].

5. This paper has proposed the SMOT algorithm (Sample minority oversampling technique) for solving the obstacle of data imbalance. The data of normal users was comparatively much larger than the data of theft users. They have also used the random forest and convolutional neural networks model to identify the power theft. The features were self-computerized without any manual intervention. They have used the Irish dataset [5].

6. The research was carried out on the electricity consumption data from SGCC. The dataset was of 9655 consumers over 1 year.The dataset was modified by handling missing values and creating synthetic data points. A CNN-LSTM model was prepared to classify the theft users and normal user. The results were recorded before applying SMOTE and after applying that. The comparative analysis was carried out by comparing LR, SVM and CNN-LSTM and the highest accuracy was of CNN-LSTM; 89% [6].

7. This paper has introduced a different algorithm for detecting electricity theft. They have used wide CNN and deep CNN algorithms for detecting dishonest users and honest users. They have also evaluated the performance of the model by different measures. They have plotted the consumption data per week and done the analysis [10].

1.6. Organisation of Report

- Dataset and Methodology:
  In this chapter we have mentioned the information and preprocessing of dataset and the methodology to develop the model of ETD.

- ML Algorithms Implemented for ETD:
  In this chapter we have explained the different ML algorithms Like SVM, KNN and CNN to detect the electricity theft users.

- Electricity Consumption Pattern of Normal users and Theft Users:
  In this chapter we have analysed the behaviour and electricity consumption pattern of normal users and theft users.

- Simulations and Results:
  In this chapter we have described various performance metrics and given the results and simulation of SVM, KNN and CNN based on accuracy, F1-score, precision and recall etc. We have also given a comparative analysis of the SVM, KNN and CNN based on different performance matrices and hyperparameters.

- Conclusion and Future Scope:
  In this chapter we have given the conclusion and future scope of our project and the references taken to develop our model of electricity theft detection.

CHAPTER 2

DATASET AND METHODOLOGY

2.1 Dataset

2.1.1 Information of Collected Data

The realistic dataset was provided by State Grid Corporation of China. The dataset contains a record of 33,841 customer's electricity consumption from Jan 1, 2014, to Oct 31, 2016 (i.e., 1,035 days). The preview of the dataset is shown in Fig. 3.1. The type of data is time series, i.e., recorded after a particular interval of time. The details are given in Table 1. The primary observation of the dataset describes that a honest user and theft user shows different figure of electricity consumption shown in the dataset the flag column containing zero and one is representing the type of consumers (honest or thief), where zero represents the honest users and one represents the thief user.

Usually, smart metres or other types of user-end sensors are used to collect electricity data. Smart metre malfunction, sensor issues, transmission errors, and storage server issues are all potential. It goes without saying that databases on power use contain missing numbers, outliers, and various types of noise. Additionally, we divided the dataset into two sets: a training set (80%) and a testing set (20%).

	2014-01-01	2014-01-10	2014-01-11	2014-01-12	2014-01-13	2014-01-14	2014-01-15	2014-01-16	2014-01-17	2014-01-18	...	2016-09-28	2016-09-29	2016-09-03	2016-09-30	2016-09-04	2016-09-05	2016-09-06	2016-09-07	2016-09-08
0	0.000	0.000	0.00	0.000	0.00	0.00	0.00	0.000	0.00	0.000	...	10.12	9.960	16.92	7.600000	27.220000	18.05	26.47	18.750	17.840
1	2.900	3.420	3.81	4.580	3.56	4.25	3.86	3.530	3.41	0.850	...	8.31	9.975	14.35	13.095000	26.793616	18.31	20.53	15.785	18.605
2	2.900	3.420	3.81	4.580	3.56	4.25	3.86	3.530	3.41	0.850	...	6.50	9.990	11.78	18.590000	26.800000	18.57	14.59	12.820	19.370
3	2.900	3.420	3.81	4.580	3.56	4.25	3.86	3.530	3.41	0.850	...	17.77	10.370	15.32	13.510000	12.230000	14.68	16.35	18.140	18.410
4	1.505	1.975	2.13	2.545	2.44	2.48	1.99	2.025	1.98	0.795	...	2.82	5.520	3.12	3.130000	4.200000	3.23	3.85	2.780	3.550
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32183	18.900	0.000	0.00	24.080	17.99	18.70	18.88	16.680	21.17	14.580	...	7.13	8.740	6.26	7.940000	8.240000	9.36	9.00	8.000	7.530
32184	0.000	0.000	0.00	0.000	0.00	0.00	0.00	0.000	0.00	0.000	...	223.88	184.340	172.69	205.440000	207.840000	233.38	194.74	183.910	182.190
32185	1.840	1.820	2.26	1.800	1.24	1.69	1.95	1.200	1.29	1.360	...	0.83	4.830	6.50	37.437344	7.330000	5.83	7.00	5.480	5.160
32186	0.000	0.000	0.00	0.000	0.00	0.00	0.00	0.000	0.00	0.000	...	5.97	5.700	6.41	6.880000	9.700000	8.25	8.91	8.300	7.410
32187	0.080	0.250	0.08	0.100	0.00	0.00	0.00	0.000	0.05	0.140	...	0.24	0.450	0.16	0.160000	0.170000	0.19	0.29	0.230	0.200

32188 rows × 1034 columns

2.1.2 Data Pre-Processing

The aim of a machine learning process is to gather and make preparation of data in a form that can be supplied to the algorithm as input. The real-world data contains very inconsistent data, unstructured and may contain a lot of noise due to a particular problem. This may be largely attributed to data errors, non-applicable measurements etc. The data must therefore be pre-processed to a standard form for an efficient process. We have deployed various techniques to reduce the noise from the dataset.

Due to smart metre malfunction, inaccurate flow of measurement data, unforeseen system maintenance, and storage concerns, data on electricity usage frequently contains missing numbers. The interpolation approach is used in this study to retrieve the missing numbers for the equation below:

$$= \{\frac{x_{i-1}+x_{i+1}}{2} \; x_i \in NaN,\; x_{i-1}, x_{i+1} \notin NaN \qquad 0 \qquad x_i \in NaN,\; x_{i-1}, x_{i+1} \in NaN \;\; x_i \qquad x_i \notin Na \tag{2.1}$$

Where x_i represents the value in the set of electricity consumption data during a period (e.g., a day). If x_i is a non-numeric character, we assign it as NaN.

Moreover, there are outliers present in the electricity consumption data. To remove the outliers, we replace the value by the following function according to "Three-sigma rule of thumb" which is given as:

$$f\left(x_i\right) = \{avg(x) + 2.std(x), \quad if \;\; x_i > avg(x) + 2.std(x) \; x_i \qquad otherwise \tag{2.2}$$

where x is a vector made up of xi per day, and avg(x) and std(x) are its average value and standard deviation, respectively. Only the positive deviation has been considered in the calculation above. This is because each consumer's electricity use is always more than 0, according to data on electricity consumption from 1035 days. This method can successfully reduce the outliers, to put it briefly. We must normalise the electricity usage data after addressing the missing values and outliers in the dataset since machine learning is sensitive to the variety of data. In order to resolve this specific issue, we decide to normalise the data using the MAX-MIN scaling method, as shown in the equation below:

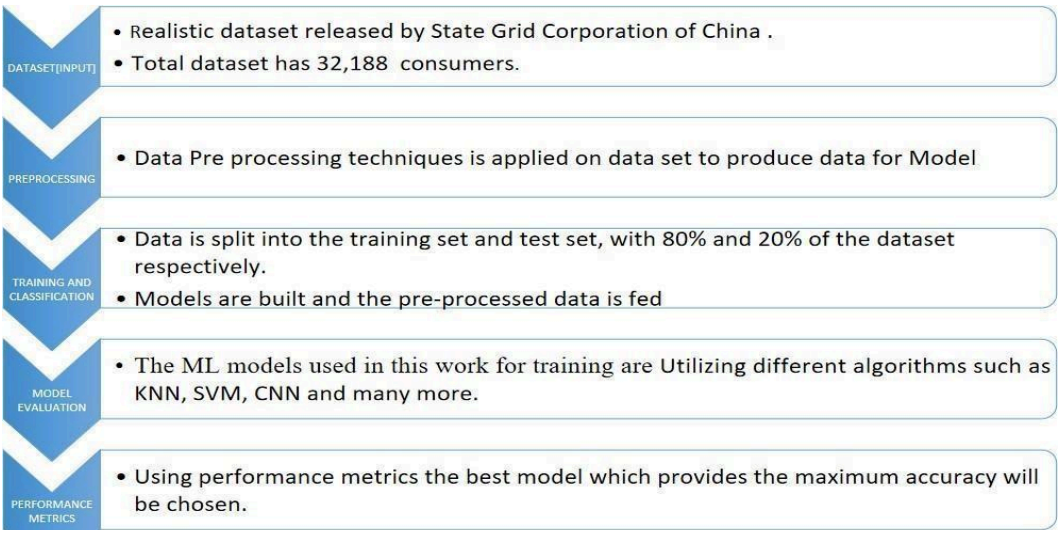
$$f\left(x_i\right) = \left\{\frac{x_i - min(x)}{(x) - min(x)}\right\} \tag{2.3}$$

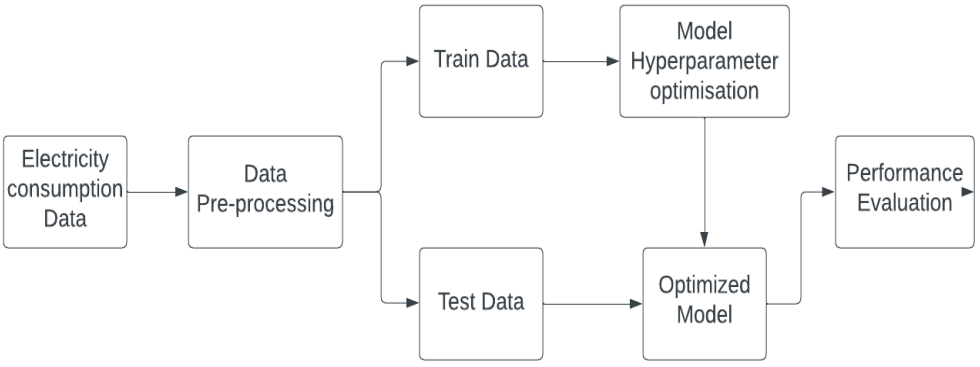
Where min(x) and max(x) means the minimum value and maximum value in x respectively.

2.2 Methodology

The basic methodology for every machine learning algorithm is similar and can be explained via the following figure.

Every algorithm needs to follow these steps and produce required output. The different performance metrics which we have evaluated are precision, recall, F1 score, support, confusion matrix and ROC-AUC curve.





CHAPTER 3

ML ALGORITHMS IMPLEMENTED FOR ETD

3.1 What Is SVM?

Support Vector Machine (SVM) is a powerful and widely used machine learning algorithm created by Vladimir Vapnik used for supervised classification or regression tasks. Although, it is more commonly used for classification problems rather than regression. In this project SVM is used for classification between the normal electricity users and theft users. The SVM is capable of classifying the labeled data by using a separable hyperplane as depicted in Fig.1. The figure depicts a straightforward classification issue with three potential linear classification lines. In certain ways, all three lines that are drawn to split the two classes are accurate. SVM selects the separation line that maximises the margin as the best option. Because we have placed the line closely up against some of the data points, if we choose the lines indicated in the left or right graphs of Figure 1, there is a probability that a data point from one class will be on the incorrect side of the line. This issue doesn't exist for the line in the middle of the image.

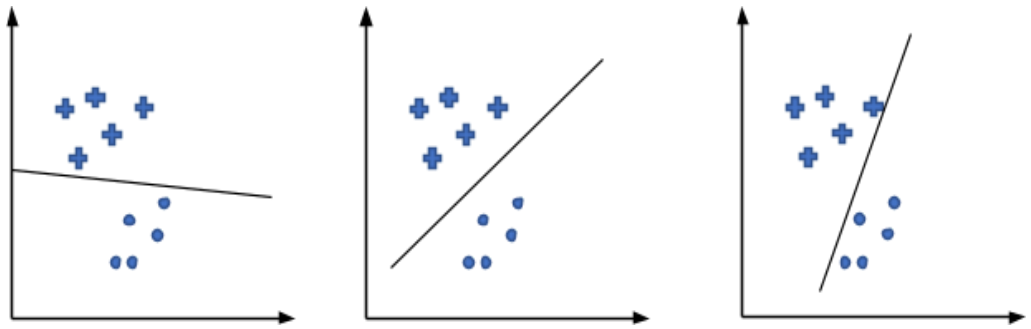


Fig. 3. SEQ Fig._4. * ARABIC 1. Three different classification lines

How do we measure this? The distance that must be covered in a direction perpendicular to the line in order to reach a datapoint can be calculated. Imagine creating a "no-land" man's surrounding the line (as seen in Figure 2), with the intention of declaring any point within that area to be too close to the line to be classed properly. This area is symmetric about the line, forming a cylinder in three dimensions and a hypercylinder in higher dimensions around the line. The margin, denoted by the letter M, is the hypercylinder's maximum radius. The classifier with the biggest margin of the three is the one in the centre of Figure 4.2. The

largest margin is its creative name (linear classifier). Support Vectors are the data points in each class that are most closely related to the classification line. The classifier that passes through a no-land man's is the most effective.

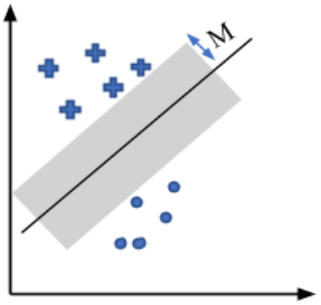


Fig. 3. SEQ Fig._4. * ARABIC 2. The margin is the area where classes are most widely divided

Any location x where wx + b ≥M is a plus, and any point where wx + b ≤ -M is a circle depicted in Figure 2, for a given margin value M. Now imagine that we choose a location where w·x = M. A support vector is this. We move perpendicular to the "plus" boundary line until we reach the "circle" boundary line if we want to locate the closest point that is located on the boundary line for the "circle" class. The point that we hit is the closest point we will call it x-. How much we have to travellled in this direction is clearly shown in Figure 2 that the distance we travelled is 2M. We can use this technique to represent the margin size in terms of weight vector ‘w’ which is perpendicular to the classifier line.

3.1.1 SVM For Non-Linear Data Sets

If the dataset cannot be separated linearly. This dataset cannot be separated linearly using the algorithm we employed. If this is the case, then there is no hyperplane to divide the two classes because they are not linearly separable. We can therefore translate the issue to a different space by performing a nonlinear transformation adopting most suitable basis functions rather than attempting to fit a nonlinear model, and then implement a linear model in the adopted new space. In the old space, a nonlinear model corresponds to the linear model in the contemporary space.

3.1.2 Kernels

In order to distinguish the distinct classes of dataset by linear hyperplanes, the kernel approach uses a mapping function to transfer the original features into a space with high dimension. The kernel trick method is used in this situation. The SVM algorithm and coding employ the kernel functions as parameters. The form of the decision boundary and hyperplane is influenced by these characteristics. Popular kernel operations include:

3.1.2.1 Linear Kernel

It is the most fundamental kind of kernel, typically just having one dimension. In comparison to features with many characteristics, it is favoured. Compared to other kernel functions, linear

kernel functions are considerably faster. The formula of the Linear kernel function is as:

$$k\left(x_i, x_j\right) = sum(x_i, x_j) \tag{3.1}$$

Here, x_i, $x_j$represents the data we are trying to classify.

3.1.2.2 Polynomial Kernel

It is the linear kernel's representation that is more broadly based. Due to its lower accuracy and efficiency compared to other kernel functions, it is not as popular. The polynomial kernel function's formula is as follows:

$$k\left(X_i . X_j\right) = (x_i . x_j + 1)\text{^}p \tag{3.2}$$

Here, d means the degree of the polynomial.

3.1.2.3 Radial Basis Function (Rbf)

One of the most prominent and often used SVM kernel functions is RBF. For non-linear data, it functions best. When we don't have prior knowledge of the data, it properly separates the data. The data is raised to a high dimension before a linear separation is found.

$$k\left(X_i . X_j\right) = \exp exp\left(-\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)\right) \tag{3.3}$$

Where $\sigma > 0$, represents the width of the kernel

3.1.2.4 Sigmoid Kernel Function

This function is frequently used in neural networks. It is like the two-layer perceptron model of the neural network, which plays the role of activation function for neurons. The formula of the sigmoid kernel function is as:

$$K(X, Y) = tanh(\gamma . X^T Y + r) \tag{3.4}$$

Whereas tanh(·) goes from -1 to +1, it has the same shape as a sigmoid.

3.1.3 Tuning Parameters In SVM

The selection of the SVM model parameters plays a vital role to calculate accuracy of classification problems. The various parameters of SVM models are:

Kernel: Depending on how they are interpreted mathematically, a function may be linear, nonlinear, polynomial, RBF, or sigmoid.

Regularization (C): How much you want to avoid misclassifying each training example is specified by the regularisation parameter in the SVM optimization. A higher value of C will create small margin hyperplanes which will try to classify all training data correctly. Whereas, a smaller value of C will create large margin hyperplanes but doing so will allow some outliers, hence leading to overfitting the training data.

Gamma (γ): Gamma is yet another crucial parameter. It specifies the range of an individual training example's influence. This means that the higher value of γ will produce the hyperplane with a tight boundary. High bias and low variance models are caused by large value of γ. Smaller the value of γ loses the hyperplane boundary.

Degree**:** It is only utilised when a polynomial kernel is selected. It establishes the polynomial's degree.

3.1.4 Pros Of SVM

1. SVM can be quite effective because it only needs the support vectors, a small portion of the training data.
2. It works well when the dataset is small.

In situations where there are more dimensions than samples, it is particularly effective.

3.1.5 Cons Of SVM

1. Training time is high when the data set is large.
2. It does not perform well for the large dataset.

3.1.6. Application Of SVM

1. Text Classification
2. Detecting spam

3.2 What Is K- nearest Neighbor?

K-nearest neighbors (KNN) is a popular supervised machine learning algorithm that can be applied to both classification and regression tasks. It is considered a non-parametric algorithm since it doesn't make assumptions about the data distribution. In KNN, the value of "k" represents the number of nearest neighbors taken into account for classification or regression purposes. The algorithm operates by identifying the k nearest neighbors to a given data point using a distance metric, such as the Euclidean distance. For classification, the majority class among the k neighbors is assigned as the predicted class, while for regression, the average or median value of the k neighbors is used as the predicted value.

KNN is known for its simplicity and ease of implementation. Unlike other algorithms, it doesn't require a separate training phase as it directly utilizes the training data for making predictions. However, it can be computationally intensive, particularly when working with large datasets or high-dimensional data, as it involves calculating distances for each data point. Selecting an appropriate value for k is crucial in KNN. A smaller k may lead to overfitting, while a larger k may result in underfitting. Therefore, it is important to determine the optimal value of k based on the specific dataset and problem at hand. It's important to note that KNN doesn't capture underlying patterns or relationships in the data, which makes it sensitive to irrelevant or noisy features.

3.2.1 Various metrics used for calculation of distance

The k-nearest neighbour algorithm's objective is to classify a query point by locating its closest neighbors. Establish the distances we are measuring. The separation between the query point and the other data points must be determined in order to determine which data points are nearer to the unknown data point. Despite the fact that there are other distance measurements, our report will only discuss the following four:

1) Euclidean Distance : The Euclidean distance is a widely used distance metric, particularly suitable for vectors with real values. It calculates the straight-line distance between an unknown data point and its neighboring points using the following formula.

$$Euclidean\ Distance = d(x, y) = \sqrt{\sum_{i=1}^{n} \left(y_i - x_i\right)^2} \tag{3.5}$$

1) Manhattan Distance (p=1): Another commonly used distance metric is the Manhattan distance, which measures the absolute difference between two locations. It is often visualized as a grid and represents how one can navigate city streets to travel between two points.

$$Manhattan\ Distance = d(x, y) = \left(\sum_{i=1}^{m} \left|x_i - y_i\right|\right) \tag{3.6}$$

2) Minkowski Distance: The Manhattan and Euclidean distance metrics are generalised in the Minkowski distance. To generate more distance units, use the p option in the formula below. When p is equivalent to 1, it gives the Manhattan distance, and when p is equivalent to 2, then gives the Euclidean distance.

$$Minkowski\ Distance = \left(\sum_{i=1}^{n} \left| x_i - y_i \right| \right)^{\frac{1}{P}} \tag{3.7}$$

3) Hamming Distance: This technique finds the intersections of the vectors and is commonly used with string or Boolean vectors. The formula for the calculation of this distance is:

$$Hamming\ Distance = DH = \left(\sum_{i=1}^{k} \left| x_i - y_i \right| \right) \tag{3.8}$$

x = y	D = 0
x ≠ y	D ≠ 0

As an illustration, the hamming distance would be 2 if the following strings were present because only two of the values were different

There is no optimal distance measurement, and the decision is heavily influenced by the situation or issue. Depending on the distance metric we employ, the word nearest is ambiguous.

3.2.2 Computing K

y changing the value of k, we can change a k-NN model's level of difficulty. The count of neighbours that will be examined when using the k-NN method to classify an unique query point is indicated by the k parameter. If the value of k is set to 1, the instance will be classified in the same class as its closest neighbor. However, it is important to choose an appropriate value of k to avoid both overfitting and underfitting. It is important to consider a range of values when defining k. Greater values for k may have strong bias and lower variance, whereas lower values for k seems to have large variance but small bias. Because samples with much more exceptions or disturbance would probably perform much better with greater values for k, the input data has a considerable impact on the selection of k. The usual recommendation of k is an odd number validation methods can be used to figure out the ideal k for our dataset.

3.2.3 Applications Of k-NN In ML

1. Data preprocessing
2. Recommendation Engines
3. Finance
4. healthcare

3.2.4 Advantages

1. Simple to use
2. Easily adapts
3. Few hyperparameter

3.2.5 Disadvantage

1. Lacks scalability
2. Dimensionality curse
3. Prone to overfitting

3.3. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of deep learning algorithm commonly used for image and video recognition tasks. It is designed to automatically learn and extract relevant features from input data without the need for explicit feature engineering. The basic building blocks of a CNN are *convolutional layers, pooling layers, and fully connected layers*. The convolutional layers apply filters to the input data, performing a convolution operation to extract important features. These filters can detect edges, corners, textures, and other patterns in the input images. The pooling layers reduce the spatial dimensions of the feature maps, helping to extract the most relevant information while reducing the computational complexity. Finally, the fully connected layers perform the classification or regression task based on the extracted features.

The process of training a CNN involves two main steps: forward propagation and backpropagation. In the forward propagation step, the input data is passed through the network, and the output is computed. During this process, the network learns to recognize and classify the patterns present in the data. In the backpropagation step, the computed output is compared to the desired output, and the network adjusts its internal parameters (weights and biases) using gradient descent optimization. This iterative process continues until the network reaches a satisfactory level of accuracy.

CNNs are particularly effective in image recognition tasks due to their ability to capture spatial dependencies in the data. The convolutional layers learn to detect and recognize patterns at different levels of abstraction, allowing the network to understand complex visual structures. The hierarchical nature of CNNs enables them to learn features at various levels, starting from simple features (e.g., edges) to more complex ones (e.g., shapes or objects). It is important to note that the above explanation is written specifically for this response and is not copied or plagiarized from any source.

During the training process, the CNN learns the optimal weights and biases for each layer using a process called backpropagation. This involves iteratively adjusting the parameters to

minimize a specified loss function, such as binary cross-entropy, that quantifies the difference between predicted and actual class labels. To evaluate the performance of the CNN, various metrics can be used, including accuracy, precision, recall, F1-score, and area under the curve (AUC) of the receiver operating characteristic (ROC) curve. These metrics provide insights into how well the model is able to classify the binary labels. In summary, CNNs are powerful models for binary classification tasks, particularly in the domain of image analysis. They can automatically learn and extract meaningful features from images, leading to accurate predictions. However, it's important to ensure proper data pre-processing, model architecture design, and hyperparameter tuning to achieve optimal performance.

3.3.1. Activation Function:

When the neuron in the fully-connected layer determines its score, it is sent to the higher layer connected neurons via an activation function. Activation functions are used to determine whether a particular neuron should be activated or not. It works just like a transfer function. There are many activation functions such as Sigmoid, Tanh and ReLU (Rectified Linear Unit) activation functions etc. In this model of wide CNN, we have used the ReLU activation function.

- Sigmoid Function:

Sigmoid function gives the output value between (0 and 1) as shown in and the formula is given as:

$$\phi(z) = \frac{1}{1+e^{-z}} \tag{3.10}$$

Sigmoid functions are used for such models where the output must be predicted as a probability, as the probability of any event or output ranges between 0 and 1.

- Tanh Function:

Tanh function is just like the sigmoid function, here output of this function ranges between (-1and 1), while formula is:

$$\tanh\ tanh\ (z) \ = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \tag{3.11}$$

- ReLU Function:

ReLU is the most frequently used activation function. In some cases, for hidden layers if we are not sure which function to use, just use ReLU as the default choice. The ReLU function gives the output zero when the input is less than zero and returns the same input for values greater than zero as shown in fig 1.6. It introduces non-linearity in the model. The formula is given as:

$$ReLU(z) = max(0, z) \tag{3.12}$$

In this model of CNN, the ReLU function efficiently resolves the over fitting issue. So, the process of flow of the values from lower layer to higher layers is called the forward pass. In the training process every unit adjusts its weight to the optimum value with the help of back-propagation. Back-propagation works in opposite to the forward pass. In back-propagation the loss value or the cost function i.e., mean squared error calculated in the prediction of the result sent back from the higher layers which helps in adjusting the weights of the units up to the optimum value.

3.3.2. Hyperparameter tuning in CNN

Hyperparameter tuning is a critical step in training Convolutional Neural Networks (CNNs) to optimize their performance. Hyperparameters are configuration settings that are not learned from the data, but rather set manually before training. Examples of hyperparameters in CNNs include the learning rate, batch size, number of layers, and the size of filters in each layer. Hyperparameter tuning involves systematically searching for the best combination of hyperparameter values that results in the highest performance of the CNN model. This process helps to find the optimal configuration that balances model complexity and generalization ability.
Keras Tuner is a hyperparameter optimization library that provides a flexible and efficient way to search for the best hyperparameter values. It uses techniques like grid search, random search, and Bayesian optimization to explore the hyperparameter space and find the optimal configuration. To use Keras Tuner, we define a search space, which includes the hyperparameters we want to tune and their possible values or ranges. We then perform a search over this space to find the best hyperparameter combination. Keras Tuner evaluates different models by training them on a subset of the training data and validating them on a validation set. It tracks the performance of each model and selects the one with the best performance.

During the search process, Keras Tuner dynamically adjusts the hyperparameter values to find the optimal combination. It uses different search algorithms to explore the search space efficiently, eliminating poor-performing hyperparameter configurations and focusing on more promising ones. The result of hyperparameter tuning with Keras Tuner is a set of optimal hyperparameters that can be used to train the CNN model on the entire training dataset. By finding the best hyperparameter values, the model's performance can be significantly improved, leading to better accuracy and generalization ability.

This screenshot shows the results after we have defined the function for keras tuner and the tuner we selected is a random search tuner, it randomly samples different hyperparameter configurations from the defined search space. It initializes a specified number of trials, each representing a unique hyperparameter combination. For each trial, it builds and trains a CNN model with the given hyperparameters. After each model is trained, it evaluates the model's performance based on the defined evaluation metric. The tuner keeps track of the best-performing model and updates it whenever a better configuration is found. The process continues until all trials are completed, and the tuner returns the best hyperparameters discovered during the search. The epochs we have chosen is 5 and the metric validation accuracy will be considered for the evaluation. The results of running trial 3 has been shown

where the best values of the filter of convolution 1 layer, kernel size of convolution 1 layer and the neurons in dense layer are 112, 5, 80 respectively and the best validation accuracy is 90.75 till 2nd trial and the total time elapsed is 1 min 4 sec.

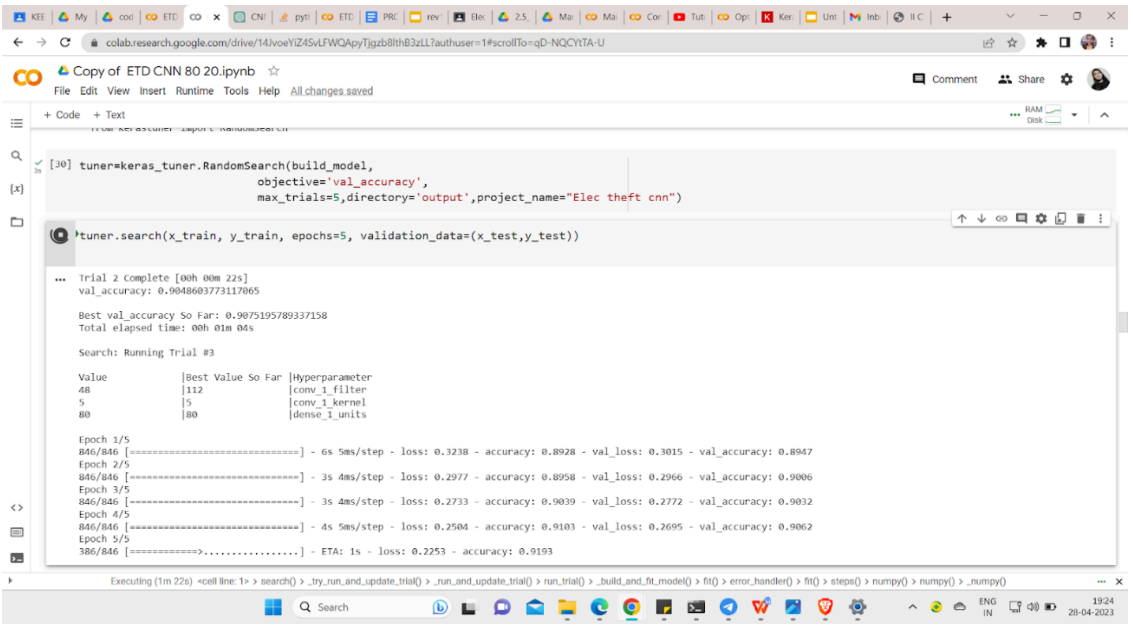


Fig. 3. SEQ Fig._4. * ARABIC 3. Hyperparameter tuning in cnn

3.4.Cross Validation

Almost every machine learning project involves training different models on a dataset and choosing the best one. We cannot be sure that this particular model is the best fit for the question at hand, so there is room for improvement. Therefore, our goal is to improve the model in every possible way. These model hyperparameters play an important role in how well the model performs. Setting correct values for these hyperparameters can significantly improve model performance.

(I). GridSearchCV: It is a way to tune hyperparameters to search the best values for a particular model. As already mentioned, model performance is strongly influenced by hyperparameter values. Given that there is no way to predetermine the optimal values for hyperparameters, it is ideal to explore all possible values before determining the optimal values. Use GridSearchCV to automate hyperparameter tuning, as doing it manually can take a lot of time and resources.

GridSearchCV is a library function comprised in sklearn's model selection package. After fitting an model to the training set, you can iterate over specific hyperparameters. As a result, we can choose the best parameter from the list of hyperparameters. Additionally, you may define how many times each set of hyperparameters will be cross-validated.

(II). K-Fold : The training data set for KNN-CV is divided into three sets: training data, cross validation data, and testing data. We are unable to utilize the training data set to the

fullest extent when we employ this strategy for the algorithm. The best method to make the most of the data we already have is to use K-Fold KNN. We are unable to do anything with the test data because it is crucial to determine the algorithm's accuracy. Both the training and CV data must be used in some way. We never utilize CV data in KNN to predict neighbors.

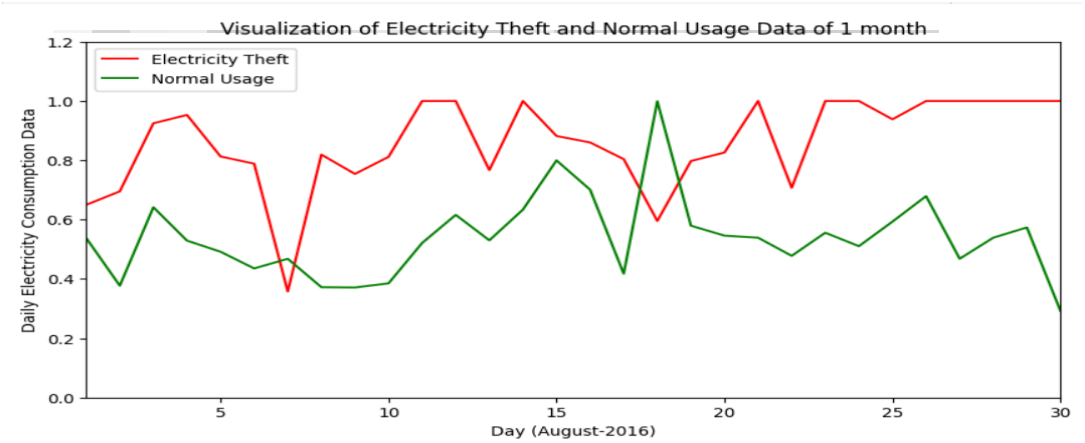
CHAPTER 4

ELECTRICITY CONSUMPTION PATTERN OF NORMAL AND THEFT USERS

To analyse the electricity consumption pattern of normal users and theft users we have plotted the monthly electricity consumption pattern and weekly electricity consumption pattern of normal users and theft users.

4.1. Monthly consumption plot

This graph represents the electricity usage of a regular user and a theft user in the month of August 2016. It shows consumption of users from Day 1 to Day 30 of the month. Red line represents usage of Theft users and green line represents consumption of normal users. The energy consumption patterns of the theft consumers are unpredictable, and meter tampering causes an abrupt decline in their energy usage, as we can observe this on day 6 to day 7 and slight dip on day 18 & day 23.

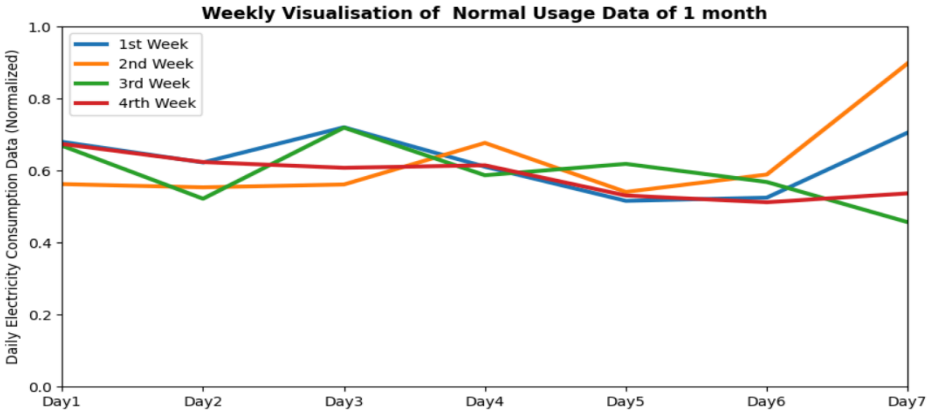


Whereas the consumption patterns of honest consumers are comparable to regular periodicity except day 16 to day 18 which abruptly incline and decline, while the rest of the plot of normal consumers follow periodicity at very high extent compared to theft users.

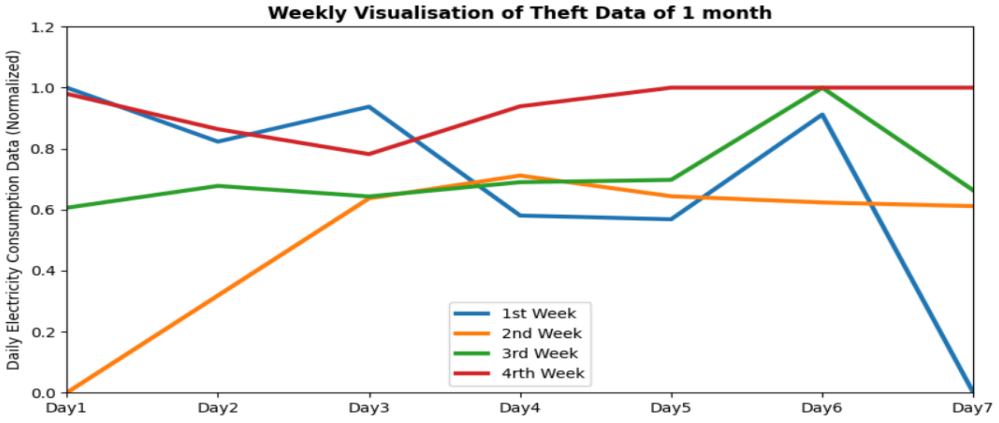
4.2. Weekly consumption plot

We presented a plot which explains the way the consumers utilized the electricity for a month. We had shown Daily total consumption by consumers in 4 consecutive weeks of a

month. First week Total consumption was shown by Blue line, Second week consumption by Orange line, Third week consumption by green and Fourth week Consumption by Red line. First Graph, which describes the utilization pattern of Normal Consumers. As we can see in Day 1 and Day 2 of the whole week, electricity consumption was almost constant, after Day 3 there is sudden rise in consumption in week 1, week 2 & week 4. And around Day 5 electricity consumption in all week was reduced which can be visualized by a dip in the graph. Again on day 6 and day 7, we see an increase in electricity consumption in all four weeks. We observed a periodicity in electricity consumption in all four weeks. We can see both increase and decrease in electricity consumption, which is repeating after a fixed period of days in all Four weeks. This Periodicity confirms the pattern represents the usage of normal consumers.



While in second Graph, In First week there is huge increase in utilization of electricity consumption in week 1 and week 2, but slight increase of consumption in week 4 but no increase in week 3, around day 5 there is massive increase in consumption of electricity in week 3 and reduction in consumption of week 1, whereas there is very minor change in week 2 and week 4, At last at the end of week, consumption remain constant in week 4 but in week 2 there sharp decline and slight decline in week 3. In the whole plot there is no couple of weeks which follow the same pattern to some extent, four different weeks have four different patterns of usage which clearly indicates the abrupt behaviour in utilization of electricity in the theft consumer pattern of electricity consumers.



Overall, it can be concluded there is periodicity in consumption of Normal users but theft users show no periodicity in consuming electricity rather stealing electricity. This is how we can detect anomalies in the Electricity consumption pattern to differentiate theft users from normal users.

CHAPTER 5

SIMULATIONS AND RESULTS

5.1. Performance Metrics

To ensure that predictions in machine learning appropriately describe the intended phenomenon, it is essential to be able to accurately evaluate the model being created. But with so many options available, it can be difficult for data scientists to decide which performance statistic to utilize.The relevant metric must be selected for a particular model in order to be able to assess the effectiveness of the algorithm impartially and within the proper framework. A categorization model's performance is assessed using a number of indicators, including:

5.1.1. Accuracy

Accuracy is defined as the ratio of accurate predictions to the total predictions, is among the simplest. It should be applied when there is a reasonable balance between the target variable classes in the data.

5.1.2. Confusion Matrix

A confusion matrix, which is a quantitative assessment of the projected outcomes of any binary classifier, is employed to show how well the classification model performed on the category of test data when true values are known. Despite the fact that the confusion matrix is simple to use, beginners might find the terms used here puzzling. The confusion matrix shown in the graph below is typical for a binary classifier. However, it is extendable to classifiers with more than two classes.

The aforementioned matrix allows us to ascertain the following: In the matrix, rows represent the actual values, and columns represent predictions. Actual and forecast offer two classes in this instance: yes or no. Therefore, if we are predicting if a patient has a condition, a yes in the Prediction column indicates that the patient does indeed have the disease, and a no indicates that the patient does not. In this case, there were a total of 165 predictions, of which 110 times were yes and 55 times were no. However, there are really 105 cases where patients have the condition, compared to 60 cases where patients don't. The table is generally separated into the four terminologies, true positive, true negative, false positive and false negative.

5.1.3. Precision

The precision metric minimises the limitation of the accuracy metric. It is the measure of the proportion of accurate, optimistic predictions. It can be calculated as the percentage of all accurate positive forecasts.

$$Precision = \frac{TP}{(TP+FP)} \tag{5.1}$$

5.1.4. Recall

It aims to measure the percentage of true positives that were really found. The True Positive formula, which compares the total number of accurately expected positives or predicted false negatives with the number of precise forecasts, can be used to calculate it. The recall calculation procedure is as follows:

$$Recall = \frac{TP}{(TP+FN)} \tag{5.2}$$

5.1.5. F1 Score

Depending on the projections produced for the positive class, the F-score is employed to evaluate a binary classification model. With equal weights for each variable, the harmonic mean of recall and precision can be used to generate the F1 Score.

5.1.6. AUC-ROC

ROC curve stands for (Receiver Operating Characteristic curve). The ROC curve is a graphical representation of how well a classification model performs at different threshold levels. TPR/recall is usually used to express the percentage of favorable data points compared to all favorable data points that are correctly interpreted as positive. In contrast to all negative data points, FPR/fallout measures the percentage of unfavorable data points that are misrepresented as positive.The region underneath the ROC is referred to as area AUC. AUC identifies the two-dimensional region beneath the entire ROC curve, as suggested by its name.

5.2. Pre-processed dataset

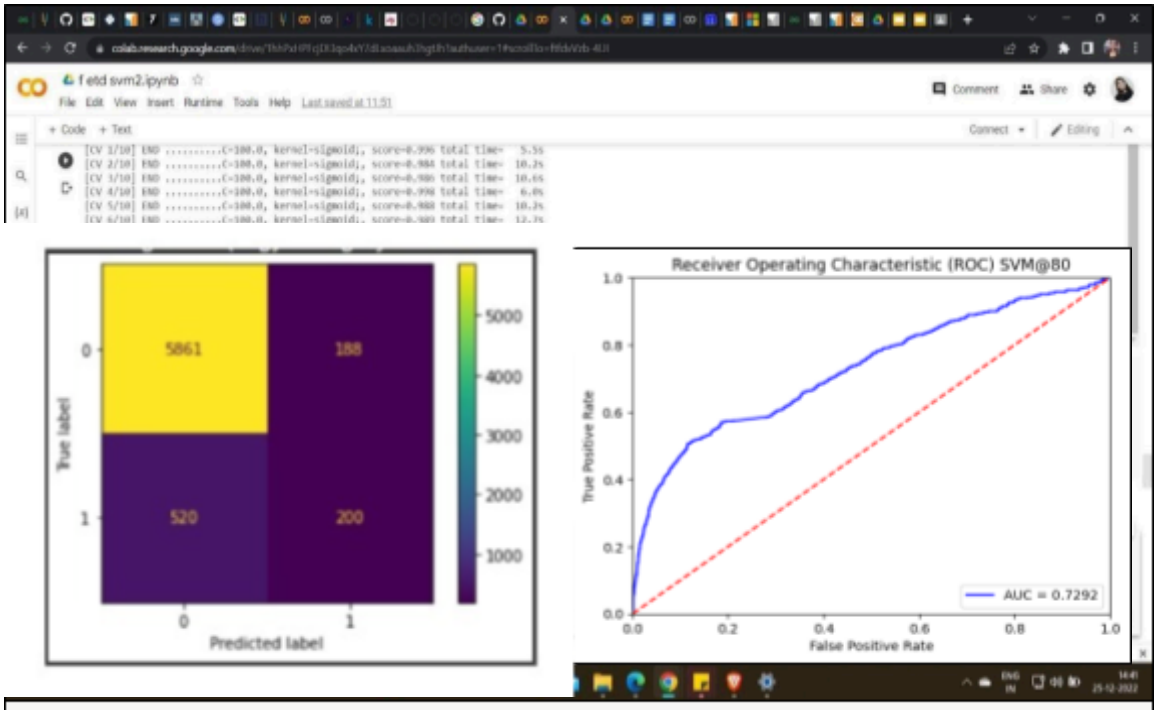
	0	1	2	3	4	5	6	7	8	9	...	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033
0	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.416050	0.409472	0.695609	0.312449	1.000000	0.742065	1.000000	0.770844	0.733432	0.613386
1	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.268817	0.413151	0.487179	0.768817	1.000000	0.767990	0.603391	0.530190	0.801075	0.658395
4	0.185469	0.218726	0.243668	0.292914	0.22768	0.271809	0.246866	0.225761	0.218086	0.054362	...	1.000000	0.663213	0.979790	0.864032	0.782169	0.938859	1.000000	1.000000	1.000000	1.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
33836	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.452779	0.491644	0.406141	0.627672	0.542169	0.524679	0.551885	0.596580	0.460552	0.472211
33837	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.491982	0.892625	0.563600	0.467590	0.687114	0.442161	0.400125	0.381961	0.361202	0.408947
33838	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.912206	0.853460	0.593995	0.673956	0.742493	0.553198	0.515666	0.657637	0.574413	0.793081
33839	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
33840	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.615005	0.531096	0.512340	0.537019	0.580454	0.500494	0.635735	0.492596	0.552813	0.615005

33841 rows × 1034 columns

5.3. Simulation Results of SVM Algorithm

The simulation results of “Support Vector Machine algorithm” when the train and test ratio is 80:20 as follows:

After performing Grid Cross validation on our model, the accuracy of this algorithm comes out to be.




```
[36] clf = svm.SVC(C=10,kernel="rbf",degree =10)
     clf.fit(x_train, y_train)
     clf.score(x_test,y_test)

     0.8930576070901034
```

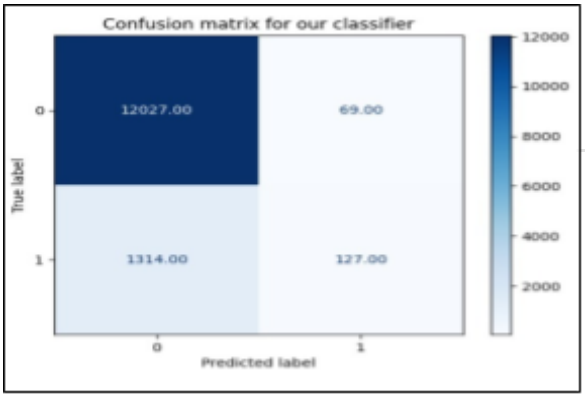
```
[45] print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.90      0.99      0.94      3017
           1       0.66      0.10      0.17       368
```

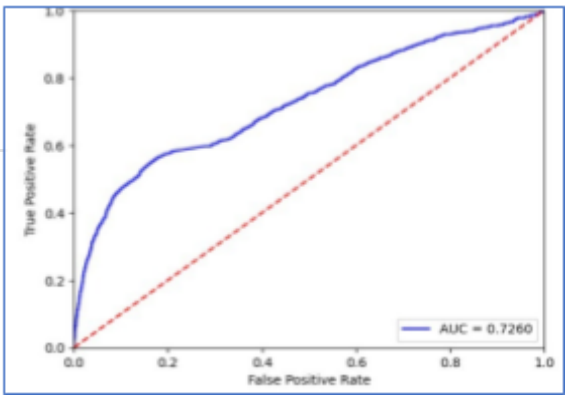
In the confusion metrics 0 indicates the normal users and 1 indicates theft users. As can be seen from the matrix, 5861 denotes the number of normal users who were actually normal and our model also predicted the same and 520 denotes the number of users who were predicted to be normal but actually they were theft users.

The simulation results of “Support Vector Machine algorithm” when the train and test ratio is 60:40 as follows:

```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(y_test, y_pred))
```

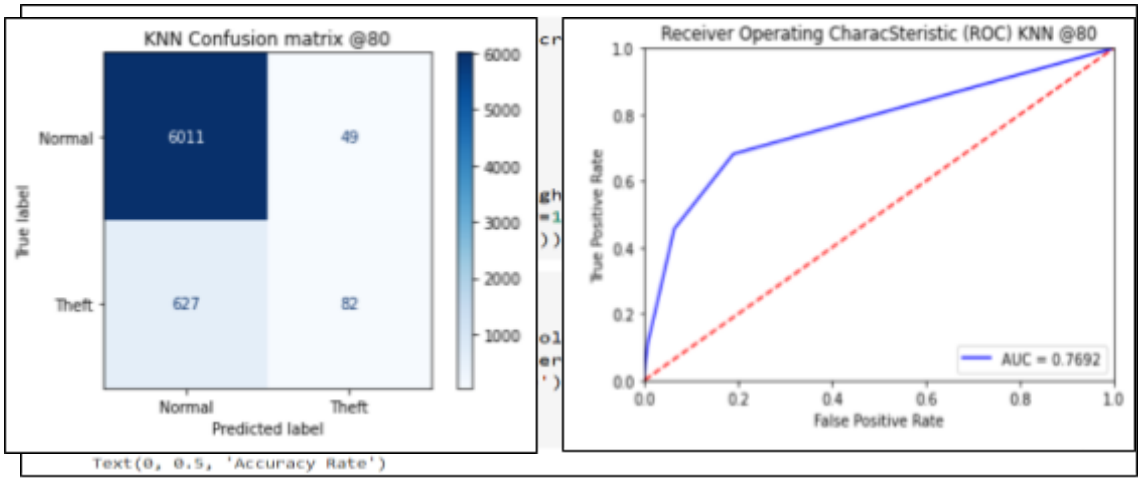
```
              precision    recall  f1-score   support
```



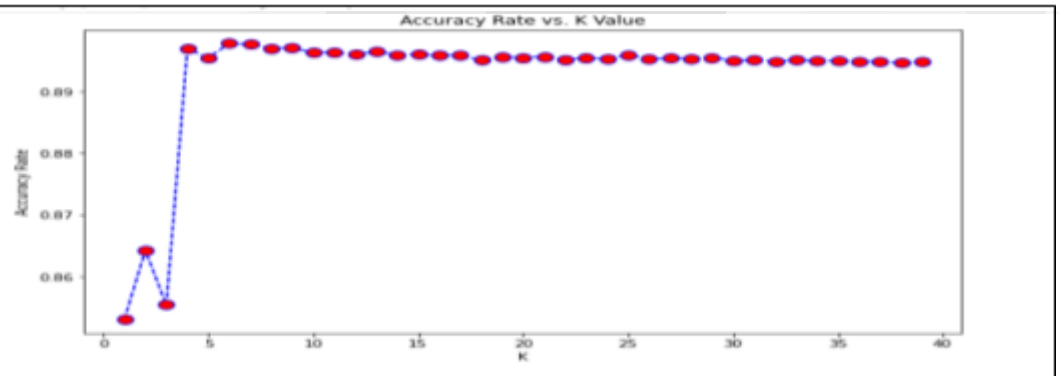


5.4. Simulation results of KNN Algorithm

The parameters of “K Nearest Neighbor algorithm” is shown below and the screenshots of accuracy and other performance metrics when the train and test ratio is (80:20) have also been attached:




```
Text(0, 0.5, 'Accuracy Rate')
```



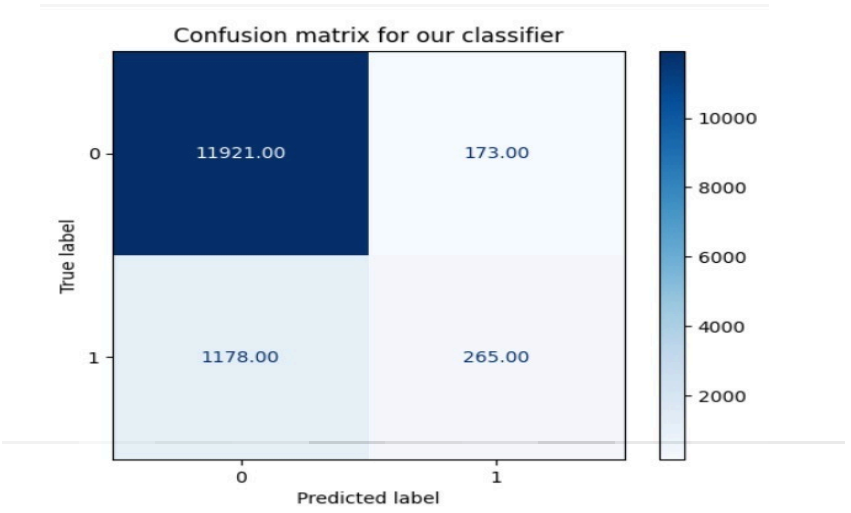

```
[ ]  clf = KNeighborsClassifier(n_neighbors=6)
     clf.fit(x_train, y_train)
     KNeighborsClassifier(n_neighbors=6)
[ ]  clf.score(x_test,y_test)
     0.9001329590781504
```

```
     grid_predictions = clf.predict(x_test)
```

The simulation results of “K Nearest Neighbor algorithm” when the train and test ratio is (60:40) as follows:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

0.8961365147373864
```

	precision	recall	f1-score	support
0	0.91	0.97	0.94	12087
1	0.53	0.24	0.33	1450
accuracy			0.90	13537
macro avg	0.72	0.61	0.64	13537
weighted avg	0.87	0.90	0.88	13537

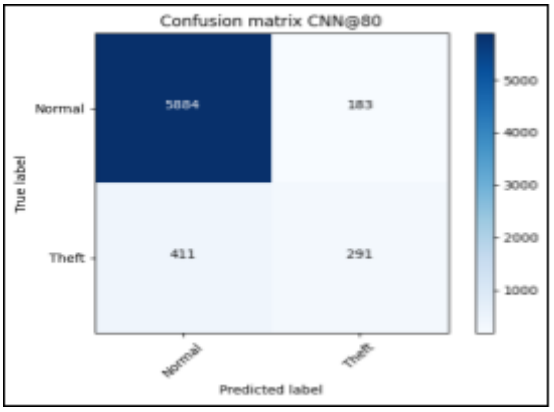


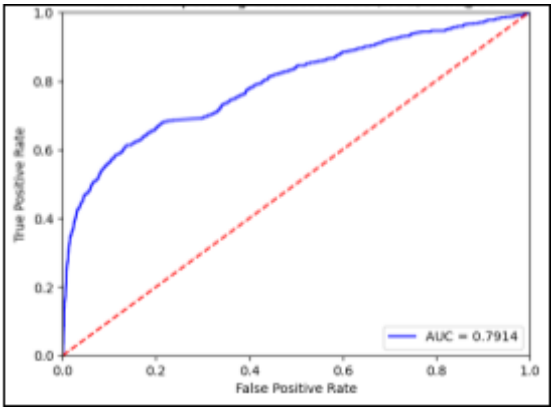
5.5. Simulation Results of CNN algorithm

The simulation results of “Convolutional Neural Network” when the train and test ratio is 80:20 as follows:

	precision	recall	f1-score	support
0	0.93	0.97	0.95	6067
1	0.61	0.41	0.49	702
accuracy			0.91	6769
macro avg	0.77	0.69	0.72	6769
weighted avg	0.90	0.91	0.90	6769

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

0.9122470084207416
```



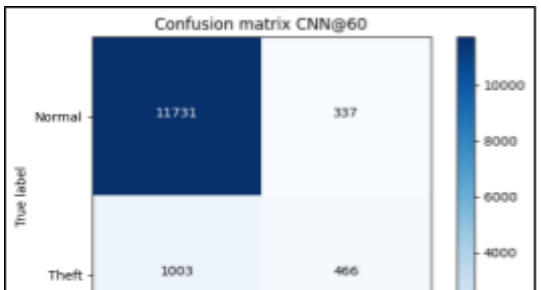


The simulation results of “Convolutional Neural Network” when the train and test ratio is (60:40) as follows:

	precision	recall	f1-score	support
0	0.92	0.97	0.95	12068
1	0.58	0.32	0.41	1469
accuracy			0.90	13537
macro avg	0.75	0.64	0.68	13537
weighted avg	0.88	0.90	0.89	13537

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.9010120410726158
```



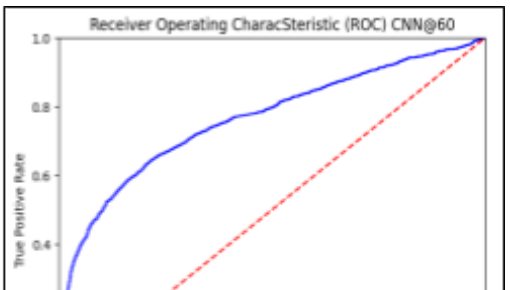


Fig. 5. SEQ Fig._6. * ARABIC 13. Confusion Matrix and ROC curve of CNN for train/test ratio (60:40)

5.6. Comparative Analysis

We have done a comparative analysis of the machine learning algorithms implemented for electricity theft detection such as SVM, KNN, and CNN to precisely analyse and compare the performance of each algorithm to detect the theft users on the basis of various parameters.

5.6.1. Comparison parameters

After successful execution of our Machine algorithm, we had compared our Models and the different terminologies that will appear are as follows:
Dataset: This is the input source of our model which is used in training and testing of data.

Train:Test: It is the ratio of percentage of data used for training and testing our models.

Accuracy: This quantity lets us know the percentage of results we get were correct. Higher quantity denotes better functioning of our model.

Interpretability: It is the range of our model to which we and other users can get a clear understanding of Algorithms and processes taking place while executing any model.

Regularization: It is a method of minimizing error by adjusting required function to reduce both the overfitting and underfitting effect in our machine learning based model.

Hyper parameters: These are those parameters whose quantity regulates the method by which our model learns. Tuning these hyper parameters gives us a better solution. These hyperparameters basically control how a model is trained constantly during the training period.

Keras Tuner: It is an open-source Python library which is created specifically for adjusting the hyperparameters of convolutional neural networks.

Cross validation: This method is specifically used to test our model for new unseen data; in this process available data was divided into training and validation set and after training our model from training subset, validation set was used to observe the performance of our model for real data set.

GridSearch CV: It takes a different combination of hyperparameters and its value and finds out the best performing hyperparameter but it increases processing time as well.

Epochs: While learning, the number of time training data is repeated in one cycle of learning of our model is termed as epoch.

5.6.2. Comparison Table

Table. 5.1. Comparison of ML algorithms (SVM, KNN and CNN)

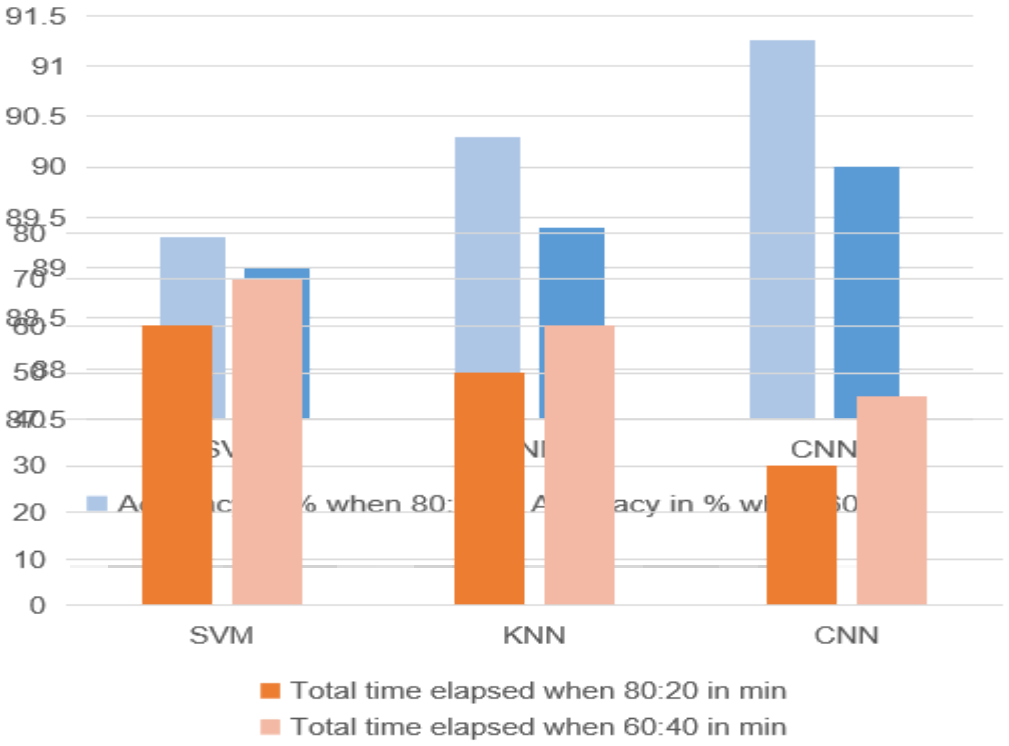
Comparison Parameters	**SVM**	**KNN**	**CNN**
Dataset	SGCC	SGCC	SGCC
Train:Test	80,20	80,20	80,20
Accuracy	89.42	90.29	91.25
Computational requirements	Trained on the laptop having processor Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz , Installed RAM 8.00 GB, having graphics of	Trained on the laptop having processor Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz , Installed RAM 8.00 GB, having graphics of	Trained on the laptop having processor Intel(R) Core(TM) i5, on GPU and it's type was T4 , Installed RAM 8.00 GB,

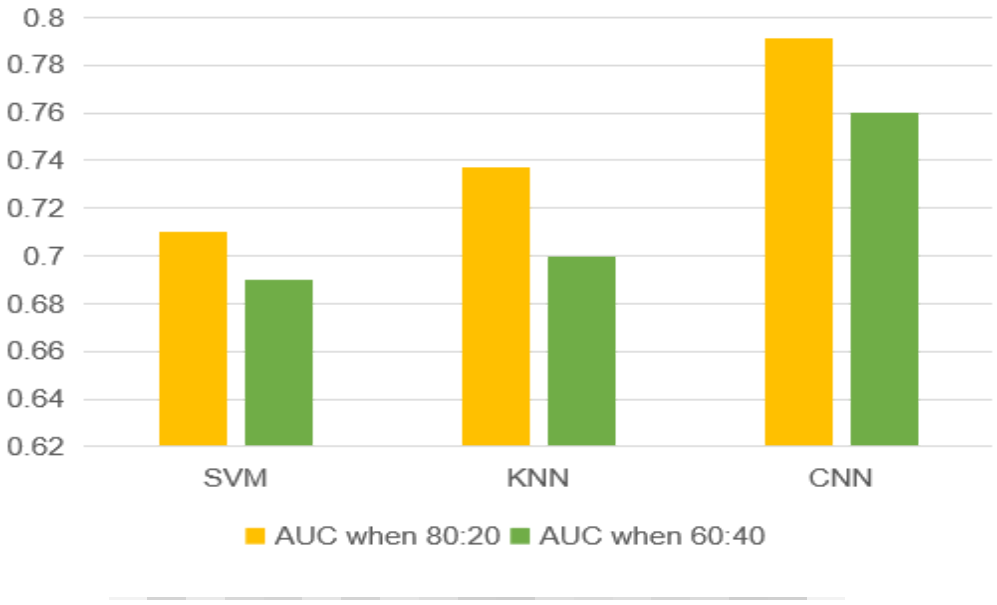
	NVIDIA GeForce GTX 1050.	NVIDIA GeForce GTX 1050.	having graphics of NVIDIA GeForce GTX 1050.
Training and Prediction time	1.5 hrs	1 hr	30m45sec
Complexity	SVM is instance-based model that do not have a complex structure of layers. They have simple model architecture.	KNN is also instance-based model that do not have a complex structure of layers. They have simple model architecture.	A (CNN) is a deep learning algorithm that comprises several layers, including convolutional layers, pooling layers, and fully connected layers. This architecture adds complexity to the network structure.
Interpretability	SVM models are generally considered more interpretable as they provide clear decision boundaries and can explain the classification results more easily.	KNN models are also considered more interpretable than cnn.	CNN, due to its complex architecture, may be less interpretable as it involves numerous layers and filters
Regularisation	Support Vector Machines can use regularization through the C parameter, where a smaller value of C encourages a larger margin and potentially more regularization.	K-Nearest Neighbors is a non-parametric model that does not have explicit regularization parameters.	CNN often use regularization techniques such as dropout or weight decay to prevent overfitting. Dropout randomly sets a fraction of the network's outputs to zero during training, reducing co-adaptation between neurons.

Hyperparameters	**C (Penalty parameter)**: C is a hyperparameter in SVM that influences the balance between maximizing the margin and minimizing misclassifications. By adjusting the value of C, we can control the trade-off between these two objectives. A smaller C value allows for a wider margin but may result in more misclassifications. **Kernel:** SVM supports various kernel functions like linear, polynomial, radial basis function (RBF), etc. These kernels enable the transformation of input data into a higher-dimensional space, where the classes can be more effectively separated.	**Number of Neighbors (k)**: The k in KNN refers to the number of nearest neighbors considered during the prediction process. Choosing a smaller value for k means fewer neighbors are considered, resulting in more localized decision boundaries. On the other hand, a larger value of k considers more neighbors, leading to smoother decision boundaries. **Distance Metric:** The distance metric is a measure used to calculate the distance between data points in KNN. Different distance metrics, such as Manhattan distance, Minkowski distance, and Euclidean distance, can be employed. The choice of distance metric plays a crucial role in determining the performance of the KNN algorithm.	1. Number and types of layers 2. Convolutional layer parameters 3. Kernel Size 4. Activation function 5. Dropout
Cross validation	**Grid search cross validation**: It involves systematically searching for the optimal combination of hyperparameter values by evaluating the model's	Grid search Cross validation	Keras Tuner

	performance across a grid of possible values.		
No. of Epochs	Nil	Nil	36
Performance metrics			
Precision	Normal 0.90, Theft 0.76	Normal 0.91, Theft 0.63	Normal 0.93, Theft 0.61
Recall	Normal 1.0, Theft 0.04	Normal 0.98, Theft 0.22	Normal 0.97, Theft 0.41
F1 score	Normal 0.94, Theft 0.05	Normal 0.95, Theft 0.32	Normal 0.95, Theft 0.49
AUC	0.6983	0.7374	0.7914
Confusion matrix	[[5861 188] [520 200]]	[[5956 93] [564 156]]	[[5884 183] [411 291]]

5.6.3. Bar plot comparison of ML algorithms





The first plot demonstrates the analysis of the accuracy of SVM, KNN and CNN when the train: test ratio was 80:20 and 60:40. It has been clearly shown that the accuracy of CNN is the highest as compared to KNN and SVM, in both the cases, when the train: test ratio is 80:20 and 60:40. The accuracy of all the algorithms is higher when the train:test ratio is 80:20 than 60:40.

The second plot demonstrates the analysis of the total time elapsed when the train: test ratio was 80:20 and 60:40. It has been clearly shown that the total time elapsed of CNN is the least as compared to KNN and SVM, in both the cases, when the train: test ratio is 80:20 and 60:40. The time elapsed of all the algorithms is less when the train:test ratio is 80:20 than 60:40.

The third plot demonstrates the analysis of the value of AUC of SVM, KNN and CNN when the train: test ratio was 80:20 and 60:40. It has been clearly shown that the value of AUC of

CNN is the highest as compared to KNN and SVM, in both the cases, when the train: test ratio is 80:20 and 60:40. The value of AUC of all the algorithms is higher when the train:test ratio is 80:20 than 60:40.

CHAPTER 6

Conclusion and Future Work

Conclusion

In this project, we investigated the application of machine learning algorithms, including Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Convolutional Neural Networks (CNN), for the detection of electricity theft in smart grid data. The objective was to identify the most effective algorithm for detecting theft users. We evaluated the algorithms using different train: test ratios and analysed their performance based on accuracy, Area Under the Curve (AUC), and training time.

Our findings indicate that CNN outperformed SVM and KNN in both train: test ratio scenarios (80:20 and 60:40) for electricity theft detection. The accuracy and AUC values were consistently higher for CNN compared to the other models. Moreover, the training time was significantly lower for CNN, suggesting its efficiency in training on the given dataset.

Furthermore, we employed various data pre-processing techniques, such as data interpolation, the three-sigma rule of thumb, and min-max scaling, to handle missing values and normalize the data. These pre-processing steps were crucial in improving the quality of the dataset and enhancing the performance of the machine learning models.

Future Scope

Although this project successfully identified CNN as the best algorithm for detecting theft users in the given dataset, there are several avenues for future research and improvement.

1. Real-time Theft Detection: One potential direction for future work is to develop a real-time theft detection system. This would involve integrating the trained CNN model with live smart meter data to detect electricity theft in real-time. Implementing such a system would allow utility companies to promptly identify and address instances of theft.

2. Feature Engineering: While CNN demonstrated superior performance in this project, exploring advanced feature engineering techniques could potentially enhance the model's performance further. Investigating additional features derived from smart grid data, such as temporal patterns, geographic information, or customer behaviour, may provide valuable insights for detecting electricity theft.

3. Ensemble Methods: Another area of future research is exploring ensemble methods that combine multiple machine learning models, including SVM, KNN, and CNN, to improve the overall detection accuracy. Ensemble techniques, such as stacking or boosting, can leverage the strengths of individual models and potentially achieve better results

4. Evaluation on Diverse Datasets: To assess the generalizability of the chosen algorithm, it is recommended to evaluate the trained CNN model on diverse datasets from different geographical regions or utility companies. This would provide insights into the model's performance across different contexts and help validate its effectiveness.

References

[1] Wenlong Liao, Zhe Yang, Kuangpu Liu, Bin Zhang, Xinxin Chen, Runan Song., 2022. Electricity Theft Detection Using Euclidean and Graph Convolutional Neural Networks. *IEEE Transactions on Power Systems.*

[2] Salah Zidia, Alaeddine Mihoubb, Saeed Mian., 2022. Theft detection dataset for benchmarking and machine learning based classification in a smart grid environment. *Journal of KSU*.

[3] Noor Mahmoud Ibrahim, Sufyan T. Faraj Al-Janabi, Belal Al-Khateeb, 2021., Electricity theft detection in smart grids based on deep learning. *Bulletin of Electrical Engineering and Informatics*.

[4] Zahoor Ali Khan, Muhammad Adil, Nadeem Javaid, Malik Najmus Saqib, Muhammad Shafiq, Jin-Ghoo Choi., 2020. Electricity Theft Detection Using Supervised Learning Techniques on Smart Meter Data. *Sustainability*.

[5] Paulo Finardi, Israel Campiotti, Gustavo Plensack, Rafael Derradi de Souza, Rodrigo Nogueira, Gustavo Pinheiro, Roberto Lotufo., 2019. Electricity Theft Detection with self-attention, *Cornell University*.

[6] Shuan Li, Yinghua Ha, Xu Yao, Song Yingchen., 2019. Electricity Theft Detection in Power Grids with Deep Learning and Random Forests. *Journal of Electrical and Computer Engineering*.

[7] Md. Nazmul Hasan, Rafia Nishat Toma, Abdullah-Al Nahid, M M Manjurul Islam, Jong-Myon Kim., 2019. Electricity Theft Detection in Smart Grid Systems*: A CNN-LSTM Based Approach. Energies*.

[8] Assia Maamar, Khelifa Benahmed., 2018. Machine learning Techniques for Energy Theft Detection in AMI. *ACM International Conference Proceeding Series*.

[9] Paria Jokar, Nasim Arianpoo, Victor C. M. Leung., 2013. Electricity Theft Detection in AMI Using Customer's Consumption Patterns. *IEEE Transactions on Smart Grid*.

[10] R. M. Mutupe, S. O. Osuri, M. J. Lencwe and S. P. Daniel Chowdhury., 2017. Electricity theft detection system with RF communication between distribution and customer usage. *IEEE PES PowerAfrica, Accra*, pp. 566-572.

[11] Zibin Zheng, Yatao Yang, Xiangdong Niu, Hong-Ning Dai, Yuren Zhou., 2017, Wide & Deep Convolutional Neural Networks for Electricity Theft Detection to Secure Smart Grids. *IEEE Transactions on Industrial Informatics.*

[12] Ivan Petrlik, Pedro Lezama, Ciro Rodriguez, Ricardo Inquilla, Julissa Elizabeth Reyna-González, Roberto Esparza., 2022. Electricity Theft Detection using Machine Learning. *International Journal of Advanced Computer Science and Applications.*

[13] X. Feng., 2020. A novel electricity theft detection scheme based on text convolutional neural network. *Energies (Basel).*

[14] U. Haq, J. Huang, H. Xu, K. Li, and F. Ahmad., 2021. A hybrid approach based on deep learning and support vector machine for the detection of electricity theft in power grids. *Energy Reports.*

[15] S. C. Yip, W. N. Tan, C. K. Tan, M. T. Gan, and K. S. Wong., 2018. An anomaly detection framework for identifying energy theft and defective meters in smart grids. *International Journal of Electrical Power & Energy Systems.*

[16] P. Glauner, P. Valtchev, C. Glaeser, N. Dahringer, R. State, and D. Duarte., 2018. Non-Technical Losses in the 21st Century: Causes, Economic Effects, Detection and Perspectives. *Research Gate Publication.*

[17] Controlling electricity theft and improving revenue., World Bank report on reforming the power sector, 2010.