A Mini Project Report

On

DETECTING ELECTRICITY THEFT IN POWER GRID USING CNN-RF HYBRID MODEL

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of

Mr.L.Manikandan

Assistant Professor

By

M. Rachana (18D21A05L3)

K. Supraja (18D21A05J6)

P. Siri (18D21A05M1)

ESTD: 2001

SRIDEVI WOMEN'S ENGINEERING COLLEGE

Department Of Computer Science and Engineering

(Approved by AICTE, affiliated to JNTU, HYD and Accredited by NBA and NAAC
An ISO 9001:2015 Certified Institution)

V.N.PALLY, Gandipet, Hyderabad-107

2021-2022

Department of Computer Science and Engineering

SRIDEVI WOMEN'S ENGINEERING COLLEGE

(Approved by AICTE, affiliated to JNTUH and Accredited by NBA and NAAC An ISO 9001:2015 Certified Institution)

V.N.PALLY, Gandipet, Hyderabad-107

2021-2022

CERTIFICATE

This is to certify that the MINOR PROJECT report entitled "**DETECTING ELECTRICTY THEFT IN POWER GRID USING CNN-RF HYBRID MODEL"** is being submitted **K.Supraja (18D21A05J6), M.Rachana (18D21A05L3), P.Siri (18D21A05M1)** in partial fulfillment for the award of degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by them.

UNDER THE GUIDANCE OF	**COORDINATOR**	**HEAD OF THE DEPARTMENT**
Mr. L. Manikandan	Dr.M.RamaSubramanian	Dr.A.Gautami Latha
Assistant Professor	Professor	Professor & HOD

EXTERNAL EXAMINER

T | TruProjects
Find Your Dream Projects

CERTIFICATE OF COMPLETION

DATE: 16/11/2021

This is to certify that **MS M. RACHANA, K. SUPRAJA, P. SIRI** bearing with **REGNOS**: **18D21A05L3, 18D21A05J6, 18D21A05M1** from **SRIDEVI WOMEN'S ENGINEERING COLLEGE** successfully completed **MINI** project work at **TRU PROJECTS EDUCATIONAL SERVICES PRIVATE LIMITED from 15/09/2021 to 15/11/2021** during the project. They worked on the project entitled **"DETECTING ELECTRICITY THEFT IN POWER GRID USING CNN-RF HYBRID MODEL"**

They was found punctual, hardworking and interested to learn the technologies. During the project work They demonstrated good skills with self— motivate attitude towards learning.

Their association with the team was fruitful. We wish them all the best for future!

For **TRU PROJECTS**

J. Manoj Kumar

Tru Projects Educational Services Pvt. Ltd. HYDERABAD

J MANOJ KUMAR

MANAGING DIRECTOR

DECLARATION

We hereby declare that the mini project entitled **“DETECTING ELECTRICTY THEFT IN POWER GRID USING CNN-RF HYBRID MODEL”** is the work done during the period from 15.09.2021 to 15.11.2021 and submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad

M.Rachana	**18D21A05L3**
K.Supraja	**18D21A05J6**
P.Siri	**18D21A05M1**

ACKNOWLEDGMENT

There are many people who helped, directly or indirectly to complete our mini project successfully. We would like to take this opportunity to thank one and all. First of all, we would like to express our deep gratitude towards our internal guide **Mr. L. Manikandan,** Assistant Professor, Department of CSE for his support in completion of our mini project if any dissertation.

We express our sincere gratitude to Coordinator **Dr. M. Ramasubramanian**, professor, Department of CSE, for their support and motivation, and cooperation for the successful completion of this project.

We also wish to express our sincere thanks to **Dr. A. Gautami Latha**, Professor and HOD, Department of CSE for her valuable advices and suggestions.

We are also extremely thankful to **Dr. B. L. Malleswari**, Principal, Sridevi Women's Engineering College, for providing the facilities to complete the project.

Finally, we would like to thank all our family and friends for their help and constant cooperation during our project period. Finally, we are very much intended to our parents for their moral support and encouragement.

M.Rachana	**18D21A05L3**
K.Supraja	**18D21A05J6**
P.Siri	**18D21A05M1**

INDEX

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

As one of the major factors of the nontechnical losses (NTLs) in distribution networks, the electricity theft causes significant harm to power grids, which influences power supply quality and reduces operating profits. In order to help utility companies solve the problems of inefficient electricity inspection and irregular power consumption, a novel hybrid convolutional neural network-random forest (CNN-RF) model for automatic electricity theft detection is presented in this paper. In this model, a convolutional neural network (CNN) firstly is designed to learn the features between different hours of the day and different days from massive and varying smart meter data by the operations of convolution and down sampling. In addition, a dropout layer is added to retard the risk of overfitting, and the back propagation algorithm is applied to update network parameters in the training phase. And then, the random forest (RF) is trained based on the obtained features to detect whether the consumer steals electricity. To build the RF in the hybrid model, the grid search algorithm is adopted to determine optimal parameters. Finally, experiments are conducted based on real energy consumption data, and the results show that the proposed detection model outperforms other methods in terms of accuracy and efficiency.

CHAPTER 1
INTRODUCTION

1.1 PURPOSE

The loss of energy in electricity transmission and distribution is an important problem faced by power companies all over the world. The energy losses are usually classified into technical losses (TLs) and nontechnical losses (NTLs). These electricity fraud behaviors may bring about the revenue loss of power companies.

These applications motivate the CNN applied for feature extraction from high-resolution smart meter data in electricity theft detection. In, a wide and deep convolutional neural network (CNN) model was developed and applied to analyse the electricity theft in smart grids. In a plain CNN, the softmax classifier layer is the same as a general single hidden layer feed forward neural network (SLFN) and trained through the backpropagation algorithm. On the one hand, the SLFN is likely to be over trained leading to degradation of its generalization performance when it performs the backpropagation algorithm.

1.2 SCOPE

The Aim of project is Electricity theft is the criminal act of taking power through unlawful strategies without paying for the devoured power tax which brings about income misfortune for the utility supplier. The CNN is proposed to automatically capture various features of customers' consumption behaviors from smart meter data, which is one of the key factors in the success of the electricity theft detection model. To improve detection performance, the RF is used to replace the softmax classifier detecting the patterns of consumers based on extracted features. This model has been trained and tested with real data from all the customers of electricity utility in Ireland and London.

1.3 MODEL DIAGRAM/OVERVIEW

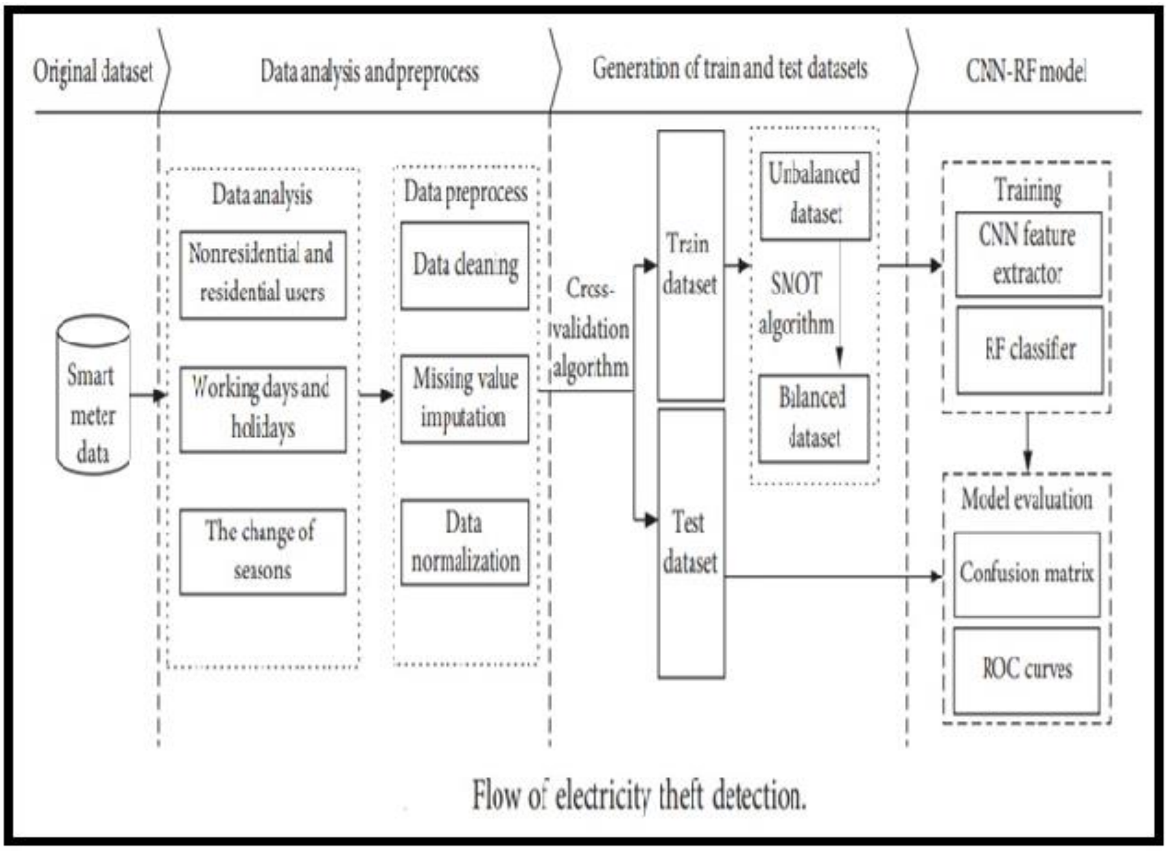


Fig. 1.1 Model Overview

CHAPTER 2
LITERATURE SURVEY

[1] TECHNICAL AND NONTECHNICAL LOSSES IN POWER SYSTEM AND ITS ECONOMIC CONSEQUENCE IN INDIAN ECONOMY

India faces endemic electrical energy and peaking shortages. These shortages have had a very detrimental effect on the overall economic growth of the country. As total distribution system losses equals technical losses plus non-technical losses. The reasons cited for such high losses are; lack of adequate T & D capacity, too many transformation stages, improper load distribution and extensive rural electrification etc. Simply Losses could be defined as the difference between the metered units of electricity entering the distribution network and those leaving the network paid for through electricity accounts, whether estimated or metered, in a well defined period of time. Technical losses are regarded as the electrical system losses which are caused by network impedance, current flows and auxiliary supplies. The sources of technical losses may be directly driven by network investment or by network operation. Non-technical losses arise from several areas including theft, un-billed accounts, and estimated customer accounts, errors due to the approximation of consumption by un-metered supplies and metering errors. The purpose of this paper is to analyse Technical and Non Technical Losses with the help of a case study and MATLAB Simulation in power systems.

[2] A MULTI-SENSOR ENERGY THEFT DETECTION FRAMEWORK FOR ADVANCED METERING INFRASTRUCTURES

The advanced metering infrastructure (AMI) is a crucial component of the smart grid, replacing traditional analog devices with computerized smart meters. Smart meters have not only allowed for efficient management of many end-users, but also have made AMI an attractive target for remote exploits and local physical tampering with the end goal of stealing energy. While smart meters posses multiple sensors and data sources that can indicate energy theft, in practice, the individual methods exhibit many false positives. In this paper, we present AMIDS, an AMI intrusion detection system that uses information fusion to combine the sensors and consumption data from a smart meter to more accurately detect energy theft. AMIDS combines meter audit logs of physical and cyber events with consumption data to more accurately model and detect theft-related behavior. Our experimental results on normal and anomalous load profiles show that AMIDS can identify energy theft efforts with high accuracy. Furthermore, AMIDS correctly identified legitimate load profile changes that more elementary analyses classified as malicious.

[3] SECURITY AND PRIVACY CHALLENGES IN THE SMART GRID

The electrical grid is undergoing one of the largest transitions in its long history, the move to smart grid technology. This new grid lets customers and providers more efficiently manage and generate power. As with many new technologies, the smart grid also introduces new security concerns. This article considers the state of global smart grid deployments and the operational, ecological, and financial motivations behind them as well as potential sources and costs of security failures. Future initiatives might address the security challenges future deployments are almost certain to face.

[4] THE CHALLENGE OF NON-TECHNICAL LOSS DETECTION USING ARTIFICIAL INTELLIGENCE: A SURVEY

Detection of non-technical losses (NTL) which include electricity theft, faulty meters or billing errors has attracted increasing attention from researchers in electrical engineering and computer science. NTLs cause significant harm to the economy, as in some countries they may range up to 40% of the total electricity distributed. The predominant research direction is employing artificial intelligence to predict whether a customer causes NTL. This paper first provides an overview of how NTLs are defined and their impact on economies, which include loss of revenue and profit of electricity providers and decrease of the stability and reliability of electrical power grids. It then surveys the state-of-the-art research efforts in a up-to-date and comprehensive review of algorithms, features and data sets used. It finally identifies the key scientific and engineering challenges in NTL detection and suggests how they could be addressed in the future.

[5] NON-TECHNICAL LOSS DETECTION USING STATE ESTIMATION AND ANALYSIS OF VARIANCE

Illegal use of electric energy is a widespread practice in many parts of the world. Smart metering enables the improvement of customer load model, theft and stressed asset detections. In this paper, a state estimation based approach for distribution transformer load estimation is exploited to detect meter malfunction/tampering and provide quantitative evidences of non-technical loss (NTL). A measure of overall fit of the estimated values to the pseudo feeder bus injection measurements based on customer metering data aggregated at the distribution transformers is used to localize the electricity usage irregularity. Following the state estimation results, an analysis of variance (ANOVA) is used to create a suspect list of customers with metering problems and estimate the actual usage. Typical Taiwan Power Company (TPC) distribution feeder data are used in the tests. Results of NTL detection under meter defect and energy theft scenarios are presented. Experiences indicate that the proposed scheme can give a good trace of the actual usage at feeder buses and supplement the current meter data validation estimation and editing (VEE) process to improve meter data quality.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Existing machine learning solutions can be further categorized into classification and clustering models, as is presented. Although aforementioned machine learning detection methods are innovative and remarkable, their performances are still not satisfactory enough for practice. For example, most of these approaches require manual feature extraction, which partly results from their limited ability to handle high-dimensional data. Indeed, traditional hand designed features include the mean, standard deviation, maximum, and minimum of consumption data. The process of manual feature extraction is a tedious and time-consuming task and cannot capture the 2D features from smart meter data.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

- ❖ Performances are still not satisfactory enough for practice.
- ❖ Time-consuming task and cannot capture the 2D features from smart meter data.

3.2 PROBLEM STATEMENT

As one of the major factors of the nontechnical losses (NTLs) in distribution networks, the electricity theft causes significant harm to power grids, which influences power supply quality and reduces operating profits. In order to help utility companies solve the problems of inefficient electricity inspection and irregular power consumption, a novel hybrid convolutional neural network-random forest (CNN-RF) model for automatic electricity theft detection is to be presented.

3.3 PROPOSED SYSTEM

The main aim of the methodology described in this paper is to provide the utilities with a ranked list of their customers, according to their probability of having an anomaly in their electricity meter. As shown in Figure 1, the electricity theft detection system is divided into three main stages as follows:

- Data analysis and preprocess: to explain the reason of applying a CNN for feature extraction, we firstly analyse the factors that affect the behaviors of electricity consumers. For the data preprocess, we consider several tasks such as data cleaning (resolving outliers), missing value imputation, and data transformation (normalization).
- Generation of train and test datasets: to evaluate the performance of the methodology described in this paper, the preprocessed dataset is split into the train dataset and test dataset by the cross-validation algorithm. -e train dataset is used to train the parameters of our model, whilst the test dataset is used to assess how well the model generalizes to new, unseen customer samples. Given that electricity theft consumers remarkably outnumber non-fraudulent ones, the imbalanced nature of the dataset can have a major negative impact on the performance of supervised machine learning methods. To reduce this bias, the synthetic minority oversampling technique (SMOT) algorithm is used to make the number of electricity thefts and non-fraudulent consumers equal in the train dataset.
- Classification using the CNN-RF model: in the proposed CNN-RF model, the CNN firstly is designed to learn the features between different hours of the day and different days from massive and varying smart meter data by the operations of convolution and down sampling. And then, RF classification is trained based on the obtained features to detect whether the consumer steals electricity. Finally, the confusion matrix and receiver operating characteristic (ROC) curves are used to evaluate the accuracy of the CNN-RF model on the test dataset.

3.3.1 ADVANTAGES OF PROPOSED SYSTEM

1.To help utility companies solve the problems of inefficient electricity inspection and irregular power consumption

2. To detect whether the consumer steals electricity or not

CHAPTER 4
SYSTEM REQUIREMENT SPECIFICATION

4.1 FUNCTIONAL REQUIREMENTS

1. Data Collection
2. Data uploading
3. Data Preprocessing
4. Training And Testing
5. Modeling
6. Predicting

4.2 NON FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *"how fast does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Data Integrity requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement

4.3 HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

• **Operating system**	**: windows, linux**
• **Processor**	**: minimum intel i3**
• **Ram**	**: minimum 4 gb**
• **Hard disk**	**: minimum 250gb**

4.4 SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.
The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

- **Python ide 3.7 version (or)**
- **Anaconda 3.7 (or)**
- **Jupiter (or)**
- **Google colab**

CHAPTER 5
SYSTEM DESIGN

5.1 SYSTEM SPECIFICATIONS (ARCHITECTURE)

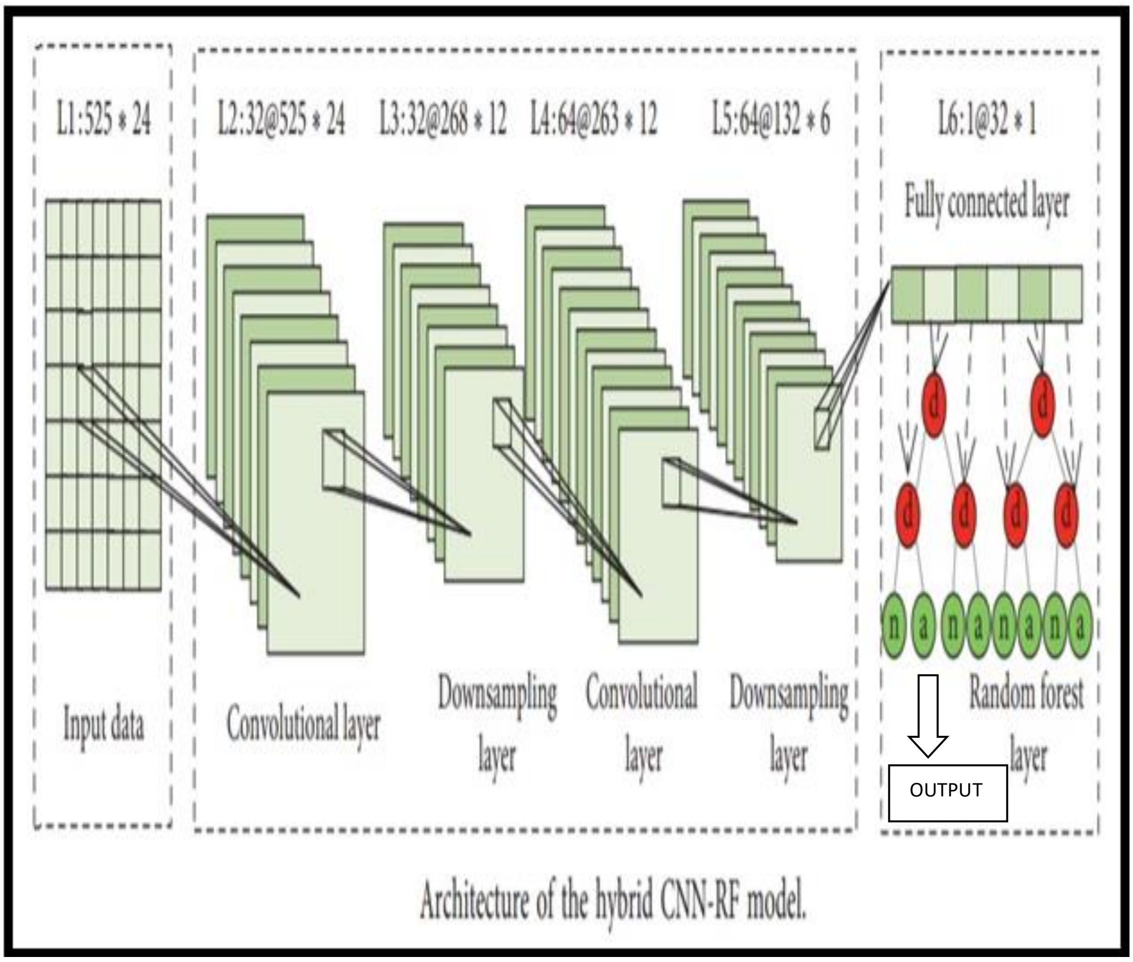


Fig 5.1 SYSTEM ARCHITECTURE

In this section, the design of the CNN-RF structure is presented in detail and some techniques are proposed to reduce overfitting. Moreover, correlative parameters including the number of filters and the size of each feature map and kernel are given. Finally, the training process of the proposed CNN-RF algorithm is introduced.

5.1.1 CNN-RF Architecture: The proposed CNN-RF model is designed by integrating CNN and RF classifiers. As shown in Figure 6, the architecture of the CNN-RF mainly includes an automatic feature extractor and a trainable RF classifier.

The feature extractor CNN consists of convolutional layers, downsampling layers, and a fully connection layer. The CNN is a deep supervised learning architecture, which usually includes multiple layers and can be trained with the backpropagation algorithm. It also can explore the intricate distribution in smart meter data by performing the stochastic gradient method. The RF classifier consists of a combination of tree classifiers, in which each tree contributes with a single vote to the assignment of the most frequent class in input data. In the following, the design of each part of the CNN-RF structure is presented in detail

5.1.2. Convolutional Layer: The main purpose of convolutional layers is to learn feature representation of input data and reduce the influence of noise. -e convolutional layer is composed of several feature filters which are used to compute different feature maps. Specially, to reduce the parameters of networks and lower the complication of relevant layers during the process of generating feature maps, the weights of each kernel in each feature map are shared. Moreover, each neuron in the convolutional layer connects the local region of front layers, and then a ReLU activation function is applied on the convolved results. Mathematically, the outputs of the convolutional layers can be expressed as $y_{\text{conv}}(X_l^{f_l}) = \delta\left(\sum_{f_l=1}^{F_l} W_l^{f_l} * X_l^{f_l} + b_l^{f_l}\right),$

where δ is the activation function, ∗ is the convolution operation, and both W_1^{f1} and b_1^{f1} are the learnable parameters in the f-th feature filter.

5.1.3. Downsampling Layer: Downsampling is an important concept of the CNN, which is usually placed between two convolutional layers. It can reduce the number of parameters and achieve dimensionality reduction. In particular, each feature map of the pooling layer is connected to its corresponding feature map of the previous convolutional layer. And the size of output feature maps is reduced without reducing the number of output feature maps in the downsampling layer. In the literature, downsampling operations mainly include max-pooling and average-pooling. The max-pooling operation transforms small windows into single values by maximum which is expressed as $y_{\text{pool}}(X_l^{f_l}) = \max_{m \in M}\{X_{l,m}\}$, but average-pooling returns average values of activations in a small window. In, experiments show that the maxpooling operation shows better performance than averagepooling. Therefore, the max-pooling operation is usually implemented in the downsampling layer:

$y_{\text{pool}}(X_l^{f_l}) = \max_{m \in M}\{X_{l,m}\}$ where M is a set of activations in the pooling window and m is the index of activations in the pooling window.

5.1.4. Fully Connected Layer: With the features extracted by sequential convolution and pooling, a fully connected layer is applied for flattening feature maps into one vector as follows:

$y_{f_l}(X_l) = \delta(W_l \cdot X_l + b_l),$ where W_l is the weight of the l-th layer and b is the bias of the l-th layer.

5.1.5. RF Classifier Layer: Traditionally, the softmax classifier is used for the last output layer in the CNN, but the proposed model uses the RF classifier to predict the class based on the obtained features. Therefore, the RF classifier layer can be defined as $y_{\text{out}}(X_L) = \text{sigm}(W_{\text{rf}} \cdot X_l + b_l),$ where sigm(·) is the sigmoid function, which maps abnormal values to 0 and normal values to 1. The parameter set W_{rf} in the RF layer includes the number of decision trees and the maximum depth of the tree, which are obtained by the grid search algorithm.

5.2 SYSTEM COMPONENTS (MODULES)

In propose paper author is performing following steps

1) **Reading dataset:** using this module reading power consumption dataset
2) **Preprocess dataset:** using this module we will normalize and clean dataset by removing missing dataset
3) **Train CNN Model:** using this module we will train CNN with dataset and then will extract trained features from CNN and then input this trained features to random forest algorithm to build theft prediction model. To remove irrelevant features we have added DROPOUT layer.
4) **Train CNN with Random Forest:** using this module will train random forest with CNN features and then calculate precision, recall, FSCORE and accuracy
5) **Train CNN with SVM:** using this module will train SVM with CNN features and then calculate precision, recall, FSCORE and accuracy

6) **Train Random Forest without CNN:** Here we trained random forest on normal dataset without using CNN features and then calculate precision, recall, FSCORE and accuracy
7) **Train SVM without CNN:** Here we trained SVM on normal dataset without using CNN features and then calculate precision, recall, FSCORE and accuracy
8) **Comparison Graph:** using this we will display comparison graph between all algorithms
9) **Predict Electricity Theft:** Using this module we will upload test data and then CNN-RF will predict whether test records contains ENERGY THEFT or not

5.3 UML DIAGRAMS

The underlying premise of UML is that no one diagram can capture the different elements of a system in its entirety. Hence, UML is made up of nine diagrams that can be used to model a system at different points of time in the software life cycle of a system.

5.3.1 Use case diagram:

The use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The use case diagram shows which actors interact with each use case.

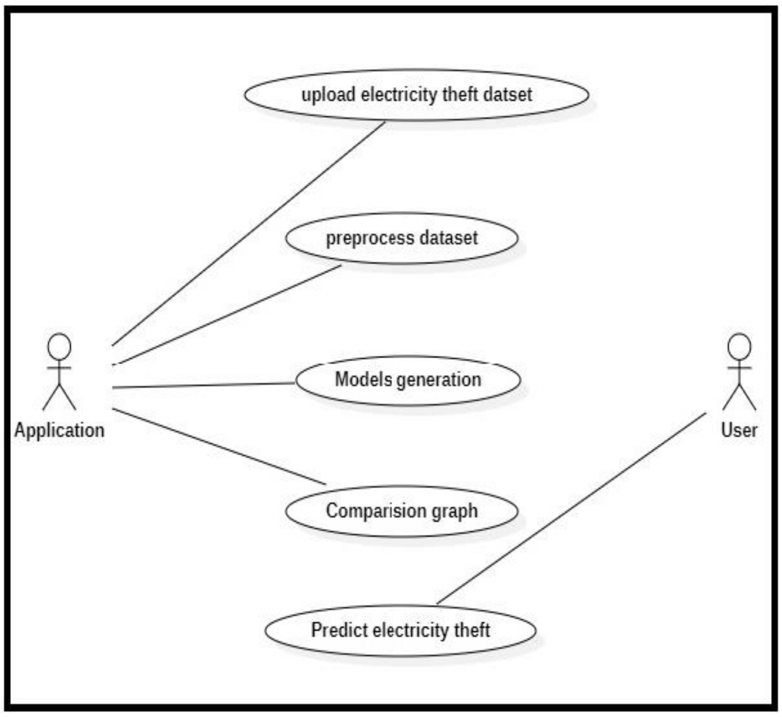


Fig 5.2 Use Case Diagram

5.3.2 Class diagram:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class.

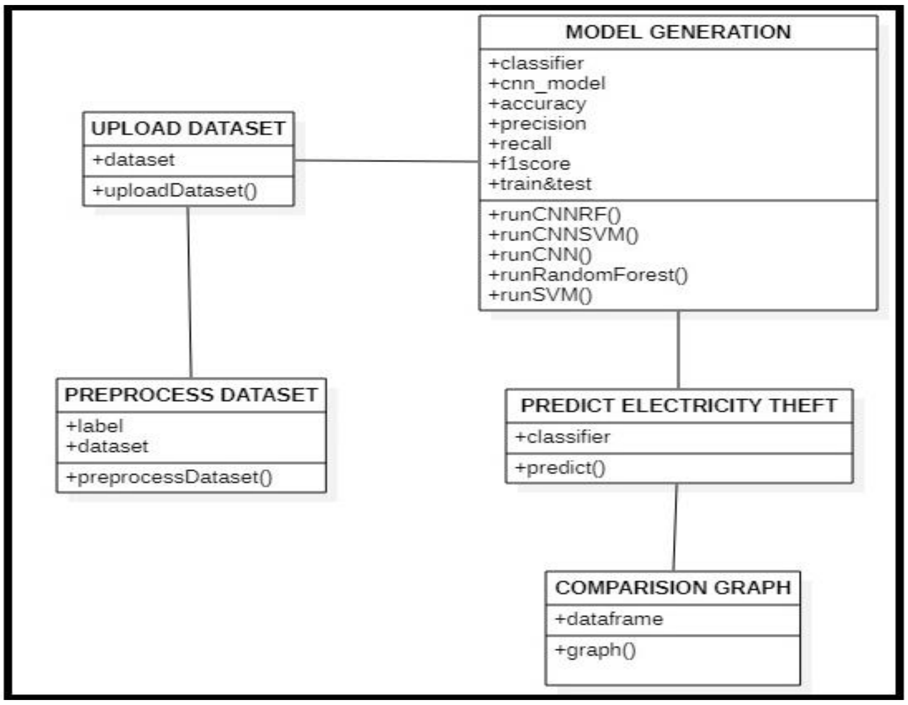


Fig 5.3 Class Diagram

5.3.3 State Chart diagram:

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.

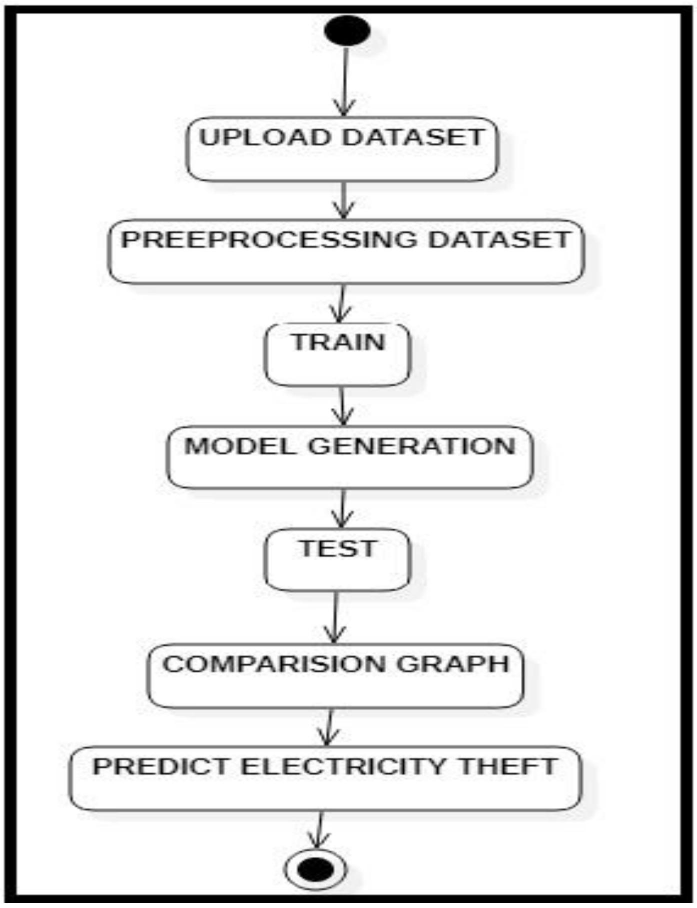


Fig 5.4 State Diagram

5.3.4 Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

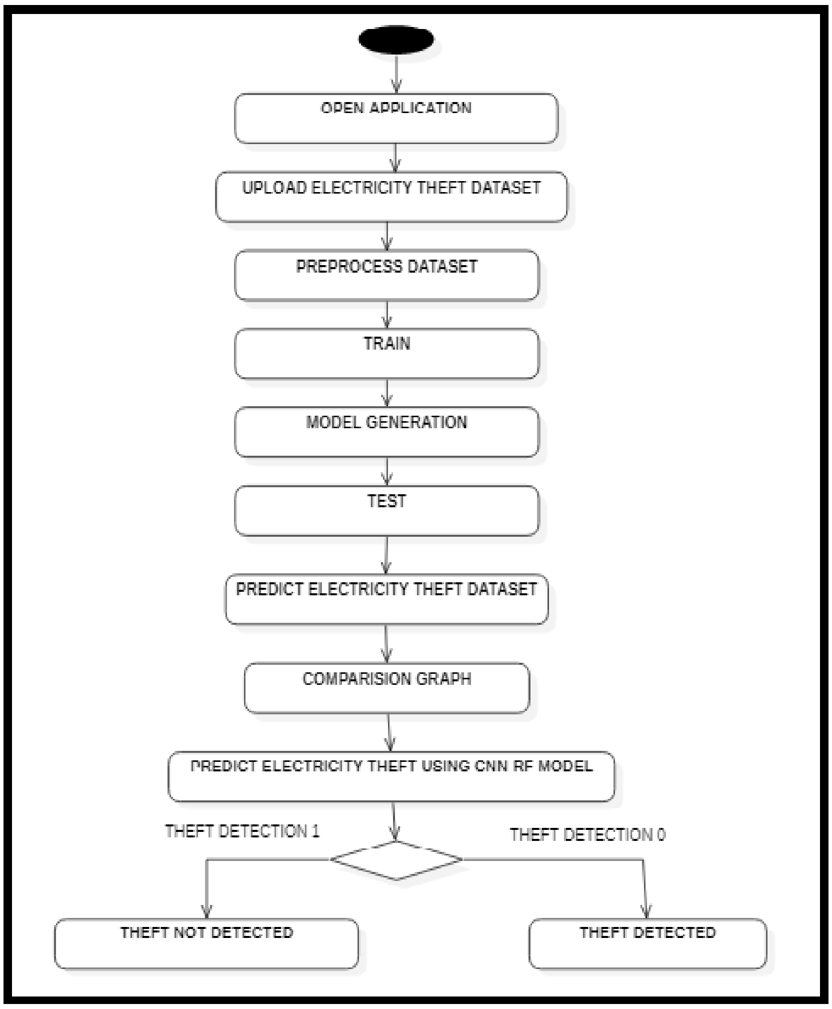


Fig 5.5 Activity Diagram

5.3.5 Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

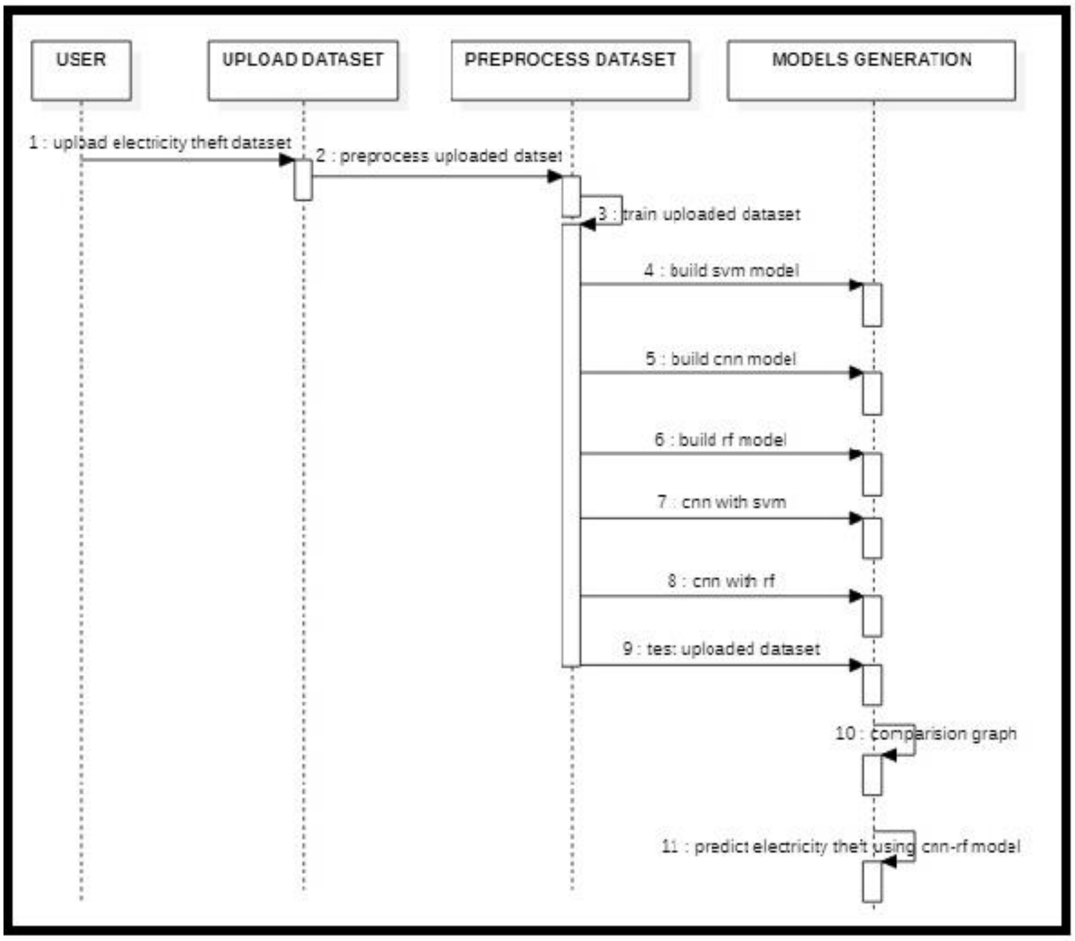

USER
UPLOAD DATASET
PREPROCESS DATASET
MODELS GENERATION
1 : upload electricity theft dataset
2 : preprocess uploaded datset
3 : train uploaded dataset
4 : build svm model
5 : build cnn model
6 : build rf model
7 : cnn with svm
8 : cnn with rf
9 : test uploaded dataset
10 : comparision graph
11 : predict electricity theft using cnn-rf model

Fig 5.6 Sequence Diagram

5.3.6 Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

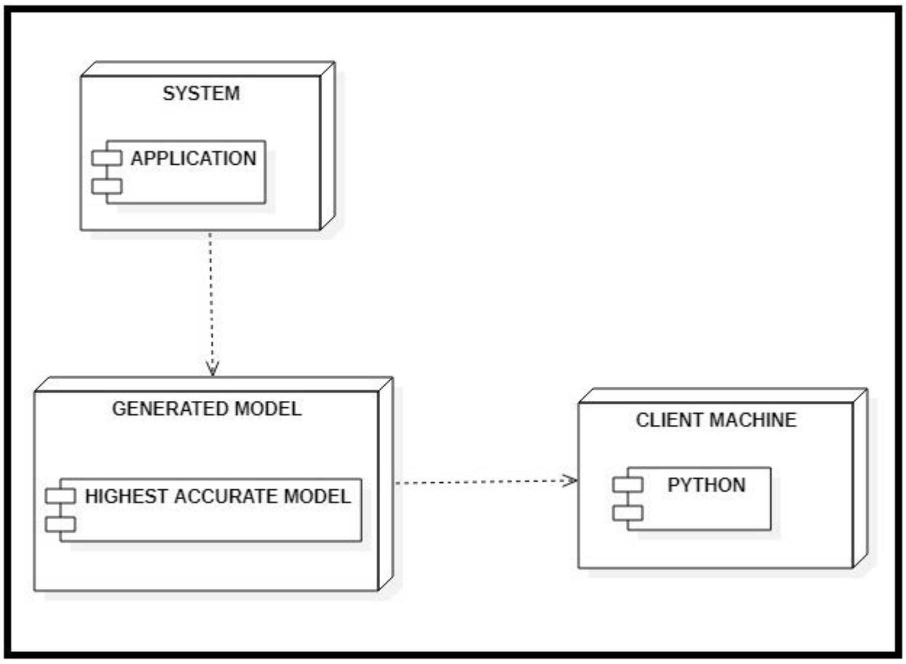


Fig 5.7 Component Diagram

5.3.7 Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

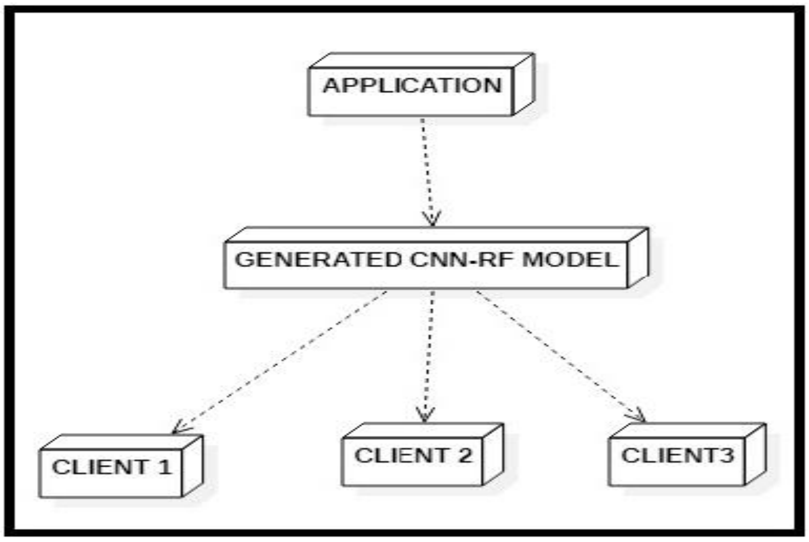


Fig 5.8 Deployment Diagram

CHAPTER 6
IMPLEMENTATION

6.1 SAMPLE CODE

Step 1: Import Libraries:

```
from tkinter import *

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askopenfilename

import pandas as pd

from tkinter import simpledialog

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import normalize

from keras.models import Sequential, Model

from keras.layers import Dense, Dropout, Activation

from keras.utils.np_utils import to_categorical

from keras.models import model_from_json

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

from sklearn import svm

import os

import matplotlib.pyplot as plt
```

(where tkinter used for GUI(front end) ,pandas - data preprocessing, numpy - mathematical , matplotlib - data visualization , sklearn - machine learning , tensorflow and keras - machine learning and AI back end purpose)

Step 2: Defining the main function and setting the title & size of tkinter

```
main = tkinter.Tk()

main.title("Electricity Theft Detection in Power Grids with Deep Learning and Random Forests")

main.geometry("1000x650")
```

Step 3: Defining the global function

```
global filename

global cnn_model

global X, Y

global le

global dataset

accuracy = []

precision = []

recall = []

fscore = []

global classifier
```

Step 4: Defining the upload function

```
def uploadDataset():
    global filename
    global dataset
    filename = filedialog.askopenfilename(initialdir = "Dataset")
    text.delete('1.0', END)
    text.insert(END,filename+' Loaded\n')
    dataset = pd.read_csv(filename)
    text.insert(END,str(dataset.head())+"\n\n")
```

Step 5: Defining the preprocess Data set function

```
def preprocessDataset():
    global X, Y
    global le
    global dataset
    le = LabelEncoder()
    text.delete('1.0', END)
    dataset.fillna(0, inplace = True)
    dataset['client_id'] = pd.Series(le.fit_transform(dataset['client_id'].astype(str)))
    dataset['label'] = dataset['label'].astype('uint8')
    print(dataset.info())
    dataset.drop(['creation_date'], axis = 1,inplace=True)
    text.insert(END,str(dataset.head())+"\n\n")
    dataset = dataset.values
    X = dataset[:,0:dataset.shape[1]-1]
    Y = dataset[:,dataset.shape[1]-1]
    Y = Y.astype('uint8')
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    Y = Y.astype('uint8')
    text.insert(END,"Total records found in dataset to train Deep Learning : "+str(X.shape[0])+"\n\n")
```

Model building

Step 6: CNN

```
def runCNN():
    global X, Y
    text.delete('1.0', END)
    global cnn_model
    accuracy.clear()
    precision.clear()
    recall.clear()
    fscore.clear()
    Y1 = to_categorical(Y)
    Y1 = Y1.astype('uint8')
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:

            loaded_model_json = json_file.read()

            cnn_model = model_from_json(loaded_model_json)

        json_file.close()

        cnn_model.load_weights("model/model_weights.h5")

        cnn_model._make_predict_function()

        print(cnn_model.summary())


    else:

        counts = np.bincount(Y1[:, 0])

        weight_for_0 = 1.0 / counts[0]

        weight_for_1 = 1.0 / counts[1]

        class_weight = {0: weight_for_0, 1: weight_for_1}

        cnn_model = Sequential() #creating RNN model object

        cnn_model.add(Dense(256, input_dim=X.shape[1], activation='relu', kernel_initializer =
"uniform")) #defining one layer with 256 filters to filter dataset

        cnn_model.add(Dense(128, activation='relu', kernel_initializer = "uniform"))#defining
another layer to filter dataset with 128 layers

        cnn_model.add(Dense(2, activation='softmax',kernel_initializer = "uniform")) #after
building model need to predict two classes such as normal or Dyslipidemia disease

        cnn_model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy']) #while filtering and training dataset need to display accuracy

        print(cnn_model.summary()) #display rnn details

        hist = cnn_model.fit(X, Y1, epochs=20, batch_size=64,class_weight=class_weight)

        cnn_model.save_weights('model/model_weights.h5')

        model_json = cnn_model.to_json()
        with open("model/model.json", "w") as json_file:
            json_file.write(model_json)
        json_file.close()
    X_train, X_test, y_train, y_test = train_test_split(X, Y1, test_size=0.2, random_state=0)
    y_test = np.argmax(y_test, axis=1)
    predict = cnn_model.predict(X_test)
    predict = np.argmax(predict, axis=1)
    p = precision_score(y_test, predict,average='macro') * 100
    r = recall_score(y_test, predict,average='macro') * 100
    f = f1_score(y_test, predict,average='macro') * 100
    a = accuracy_score(y_test,predict)*100
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
    text.insert(END,"CNN Precision : "+str(p)+"\n")
    text.insert(END,"CNN Recall    : "+str(r)+"\n")
    text.insert(END,"CNN FMeasure  : "+str(f)+"\n")
    text.insert(END,"CNN Accuracy  : "+str(f)+"\n\n")
```

Step 7: CNNRF

```
def runCNNRF():
    global classifier
    global X, Y
    global cnn_model
```

```
predict = cnn_model.predict(X)

YY = []

for i in range(len(predict)):

    val = np.argmax(predict[i])

    YY.append(val)

YY = np.asarray(YY)

extract = Model(cnn_model.inputs, cnn_model.layers[-2].output)

XX = extract.predict(X)

rfc = RandomForestClassifier(n_estimators=200, random_state=0)

rfc.fit(XX, YY)

classifier = rfc

X_train, X_test, y_train, y_test = train_test_split(XX, YY, test_size=0.2, random_state=0)

predict = rfc.predict(X_test)

p = precision_score(y_test, predict,average='macro') * 100

r = recall_score(y_test, predict,average='macro') * 100

f = f1_score(y_test, predict,average='macro') * 100

a = accuracy_score(y_test,predict)*100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

text.insert(END,"CNN with Random Forest Precision : "+str(p)+"\n")

text.insert(END,"CNN with Random Forest Recall    : "+str(r)+"\n")

text.insert(END,"CNN with Random Forest FMeasure  : "+str(f)+"\n")
```

```
    text.insert(END,"CNN with Random Forest Accuracy  : "+str(f)+"\n\n")
```

Step 8: CNNSVM

```
def runCNNSVM():
    global X, Y
    global cnn_model
    predict = cnn_model.predict(X)
    YY = []
    for i in range(len(predict)):
        val = np.argmax(predict[i])
        YY.append(val)
    YY = np.asarray(YY)
    extract = Model(cnn_model.inputs, cnn_model.layers[-2].output)
    XX = extract.predict(X)
    rfc = svm.SVC()
    rfc.fit(XX, YY)
    X_train, X_test, y_train, y_test = train_test_split(XX, YY, test_size=0.2, 
random_state=0)
    predict = rfc.predict(X_test)
    p = precision_score(y_test, predict,average='macro') * 100
    r = recall_score(y_test, predict,average='macro') * 100
    f = f1_score(y_test, predict,average='macro') * 100
    a = accuracy_score(y_test,predict)*100
    accuracy.append(a)
    precision.append(p)

    recall.append(r)

    fscore.append(f)

    text.insert(END,"CNN with SVM Precision : "+str(p)+"\n")

    text.insert(END,"CNN with SVM Recall    : "+str(r)+"\n")

    text.insert(END,"CNN with SVM FMeasure  : "+str(f)+"\n")

    text.insert(END,"CNN with SVM Accuracy  : "+str(f)+"\n\n")
```

Step 9: Random Forest

```
def runRandomForest():

    global X, Y

    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

    rfc = RandomForestClassifier(n_estimators=200, random_state=0)

    rfc.fit(X_train, y_train)

    predict = rfc.predict(X_test)

    p = precision_score(y_test, predict,average='macro') * 100

    r = recall_score(y_test, predict,average='macro') * 100

    f = f1_score(y_test, predict,average='macro') * 100

    a = accuracy_score(y_test,predict)*100

    accuracy.append(a)

    precision.append(p)

    recall.append(r)

    fscore.append(f)

    text.insert(END,"Random Forest Precision : "+str(p)+"\n")

    text.insert(END,"Random Forest Recall    : "+str(r)+"\n")
    text.insert(END,"Random Forest FMeasure  : "+str(f)+"\n")
    text.insert(END,"Random Forest Accuracy  : "+str(f)+"\n\n")
```

Step 10: SVM

```
def runSVM():
    global X, Y
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
    rfc = svm.SVC()
    rfc.fit(X_train, y_train)
    predict = rfc.predict(X_test)
    p = precision_score(y_test, predict,average='macro') * 100
    r = recall_score(y_test, predict,average='macro') * 100
    f = f1_score(y_test, predict,average='macro') * 100
    a = accuracy_score(y_test,predict)*100
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
    text.insert(END,"SVM Precision : "+str(p)+"\n")
    text.insert(END,"SVM Recall    : "+str(r)+"\n")
    text.insert(END,"SVM FMeasure  : "+str(f)+"\n")
    text.insert(END,"SVM Accuracy  : "+str(f)+"\n\n")
```

Step 11: Defining prediction function

```
def predict():
    global classifier
    global cnn_model
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir = "Dataset")
    test = pd.read_csv(filename)
    test.fillna(0, inplace = True)
    test = test.values
    data = test
    extract = Model(cnn_model.inputs, cnn_model.layers[-2].output)
    test = extract.predict(test)
    predict = classifier.predict(test)
    for i in range(len(predict)):
        if predict[i] == 1:
            text.insert(END,str(data[i])+" ===> record detected as ENERGY THEFT\n\n")
        if predict[i] == 0:
            text.insert(END,str(data[i])+" ===> record NOT detected as ENERGY THEFT\n\n")
```

Step 12: Accuracy comparison graph

```
def graph():
    df = pd.DataFrame([['CNN','Precision',precision[0]],['CNN','Recall',recall[0]],['CNN','F1 Score',fscore[0]],['CNN','Accuracy',accuracy[0]],
                       ['CNN-RF','Precision',precision[1]],['CNN-RF','Recall',recall[1]],['CNN-RF','F1 Score',fscore[1]],['CNN-RF','Accuracy',accuracy[1]],
                       ['CNN-SVM','Precision',precision[2]],['CNN-SVM','Recall',recall[2]],['CNN-SVM','F1 Score',fscore[2]],['CNN-SVM','Accuracy',accuracy[2]],
                       ['RF','Precision',precision[3]],['RF','Recall',recall[3]],['RF','F1 Score',fscore[3]],['RF','Accuracy',accuracy[3]],
                ['SVM','Precision',precision[3]],['SVM','Recall',recall[3]],['SVM','F1 Score',fscore[3]],['SVM','Accuracy',accuracy[3]],

                ],columns=['Parameters','Algorithms','Value'])

    df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')

    plt.show()
```

Step 13: Defining the button size and configuration

```
font = ('times', 16, 'bold')

title = Label(main, text='Electricity Theft Detection in Power Grids with Deep Learning and Random Forests', justify=LEFT)

title.config(bg='White', fg='blue')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=100,y=5)

title.pack()


font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Electricity Theft Dataset", command=uploadDataset)

uploadButton.place(x=200,y=100)

uploadButton.config(font=font1)


preprocessButton = Button(main, text="Preprocess Dataset", command=preprocessDataset)

preprocessButton.place(x=500,y=100)

preprocessButton.config(font=font1)
```

```
cnnButton = Button(main, text="Generate CNN Model", command=runCNN)

cnnButton.place(x=200,y=150)

cnnButton.config(font=font1)


cnnrfButton = Button(main, text="CNN with Random Forest", command=runCNNRF)

cnnrfButton.place(x=500,y=150)

cnnrfButton.config(font=font1)


cnnsvmButton = Button(main, text="CNN with SVM", command=runCNNSVM)

cnnsvmButton.place(x=200,y=200)

cnnsvmButton.config(font=font1)


rfButton = Button(main, text="Run Random Forest", command=runRandomForest)

rfButton.place(x=500,y=200)

rfButton.config(font=font1)


svmButton = Button(main, text="Run SVM Algorithm", command=runSVM)

svmButton.place(x=200,y=250)

svmButton.config(font=font1)


predictButton = Button(main, text="Predict Electricity Theft", command=predict)

predictButton.place(x=500,y=250)

predictButton.config(font=font1)
```

```
graphButton = Button(main, text="Comparison Graph", command=graph)

graphButton.place(x=800,y=250)

graphButton.config(font=font1)



font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=120)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=300)

text.config(font=font1)

main.config(bg='#87CEEB')

main.mainloop()
```

CHAPTER 7
SYSTEM TESTING

7.1. TESTING STRATERGIES

7.1.1 Unit Testing

Unit testing, a testing technique using which individual modules are tested to determine if there are issues by the developer himself, it is concerned with functional correctness of the standalone modules. The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Unit Testing Techniques:

Black Box Testing - Using which the user interface, input and output are tested.

White Box Testing –Used to test each one of those functions behavior is tested.

7.1.2 Functional Testing

Functional testing systematic demonstrations that functions tested are available are specified by the business and technical requirements, system documentation, and user manuals.

7.1.3 Integration Testing

Integration Testing done upon completion of unit testing, the units or modules are to be integrated which gives raise too integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

7.1.4 Data Flow Testing

Data flow testing is a family of testing strategies based on selecting paths through the program’s control flow in order to explore sequence of events related to the status of Variables or data object. Dataflow Testing focuses on the points at which variables receive and the points at which these values are used.

7.1.5 User Interface Testing

User interface testing, a testing technique used to identify the presence of defects is a product/software under test by Graphical User interface [GUI].

7.2 TEST CASES:

TEST CASE No.	INPUT	OUTPUT	DESCRIPTION	RESULT
Test case 1 (unit testing - uploading dataset module)	Dataset	Dataset -5 records	Dataset that project deals with is uploaded.	SUCCESS
Test case 2 (Unit testing of Preprocessing uploaded dataset)	Dataset	Clean Dataset - 5 records	Uploaded dataset is Preprocessed using predefined python libraries.	SUCCESS
Test Case 3 (Unit testing of Generating various models)	Preprocessed Dataset	Precision,recall,f1_score,accuracy of algorithm adopted	Precision,recall,f1_score, accuracy are calculated to the corresponding algorithm adopted using predefined python libraries.	SUCCESS
Test case 4 (Functional Testing of predict theft)	Preprocessed Dataset	Status of Theft Detection	Using predefined python modules, various model are built for detecting electricity theft and it's status is printed on output window.	SUCCESS
Test case 5 (Functional testing of comparison graph)	Accuracy,recall,f1_score,precision of various models	Comparison graph	Comparing the input parameters, a graph is plotted and displayed on output window.	SUCCESS

Test case 6 (Integration Testing)	Dataset	Accurate status of electricity theft	Different modules of the final model are integrated appropriately providing accurate result.	SUCCESS
Test case 7 (Data Flow Testing)	Dataset	Accurate status of electricity theft	Dataset is flown through all the modules and final model that provides accurate output is built.	SUCCESS
Test case 8 (User Interface Testing of Upload Electricity Theft)	Dataset	Preprocessed dataset	User interface (GUI) are functioning according to the label of the button.	SUCCESS
Test case 8 (User Interface Testing of Generate model)	Preprocessed Dataset	Precision,recall,f1_score,accuracy of algorithm adopted	User interface (GUI) are functioning according to the label of the button.	SUCCESS
Test case 9 (User Interface Testing of Predict Electricity Theft)	Preprocessed Dataset	Status of electricity theft	User interface (GUI) are functioning according to the label of the button.	SUCCESS
Test case 10 (User Interface Testing of Comparison graph)	Accuracy,recall,f1_score,precision of various models.	Comparison graph	User interface (GUI) are functioning according to the label of the button.	SUCCESS

Table 7.1 Test Cases

7.3 RESULTS AND DISCUSSIONS (SCREEN SHOTS)

To run project double click on ‘run.bat’ file to get below screen

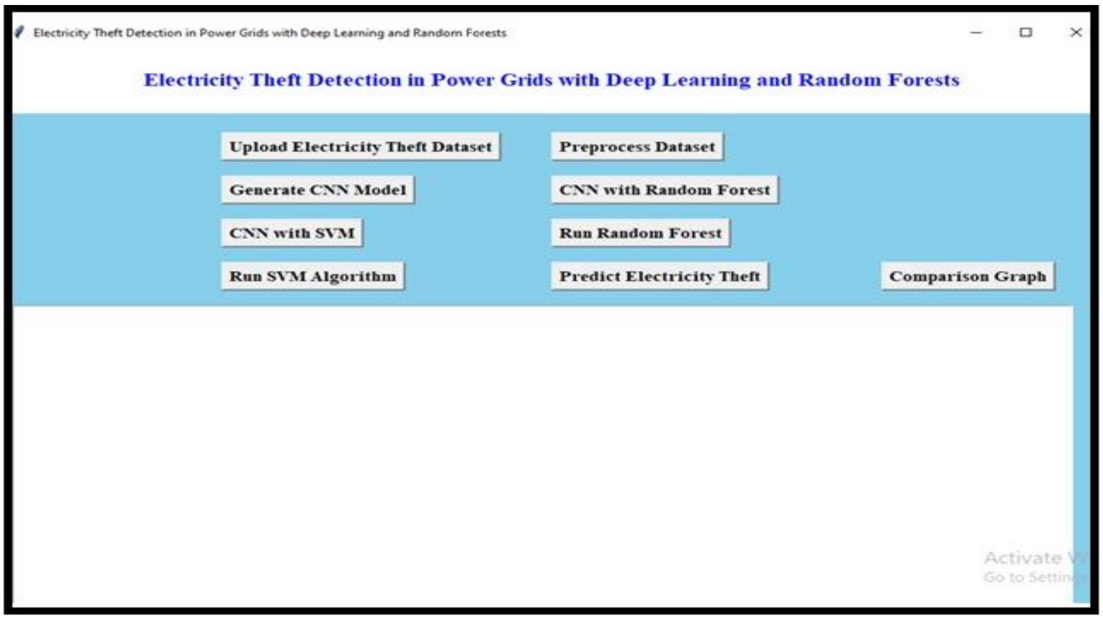
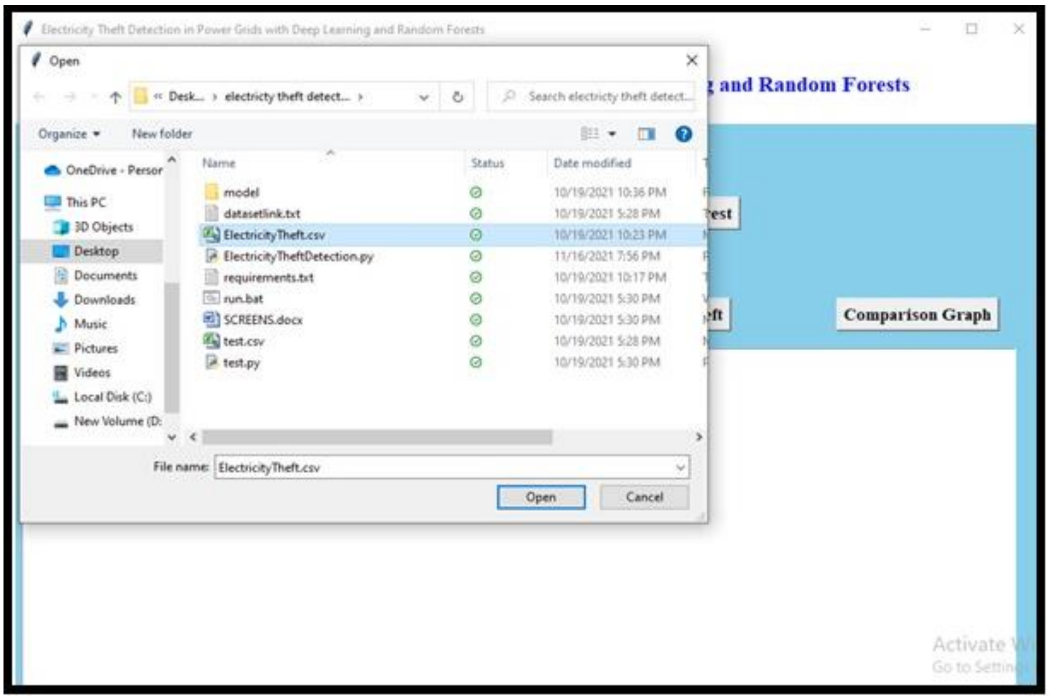

Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Upload Electricity Theft Dataset
Preprocess Dataset
Generate CNN Model
CNN with Random Forest
CNN with SVM
Run Random Forest
Run SVM Algorithm
Predict Electricity Theft
Comparison Graph

Fig 7.1 MAIN OUTPUT WINDOW

In above screen click on ‘Upload Electricity Theft Dataset’ button to upload dataset

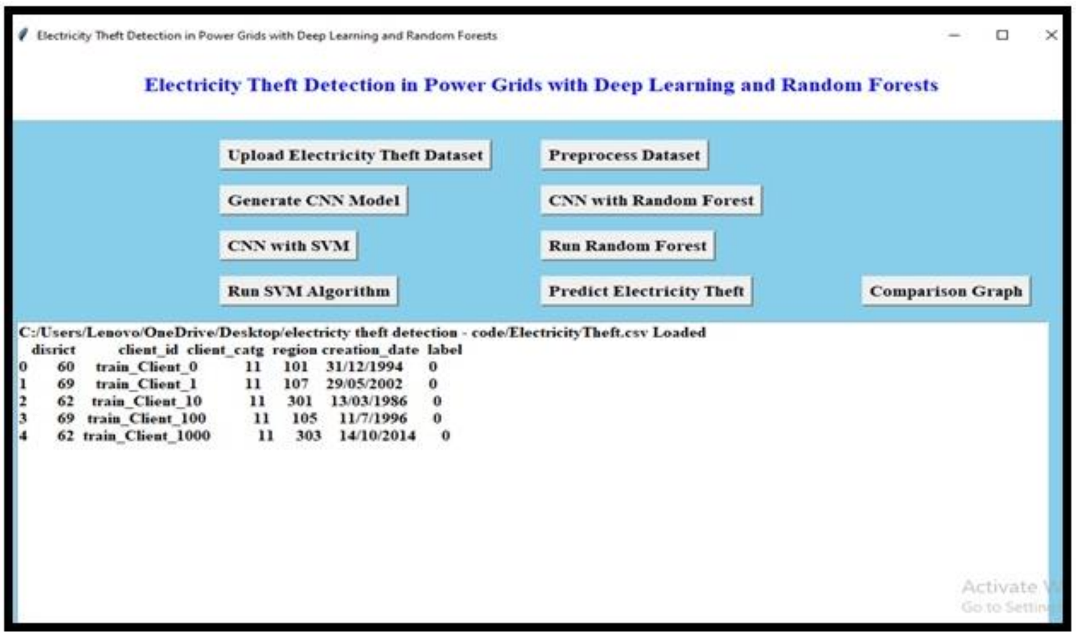


Name	Status	Date modified
model	✓	10/19/2021 10:36 PM
datasetlink.txt	✓	10/19/2021 5:28 PM
ElectricityTheft.csv	✓	10/19/2021 10:23 PM
ElectricityTheftDetection.py	✓	11/16/2021 7:56 PM
requirements.txt	✓	10/19/2021 10:17 PM
run.bat	✓	10/19/2021 5:30 PM
SCREENS.docx	✓	10/19/2021 5:30 PM
test.csv	✓	10/19/2021 5:28 PM
test.py	✓	10/19/2021 5:30 PM

File name: ElectricityTheft.csv

Fig 7.2 UPLOADING THEFT DATASET

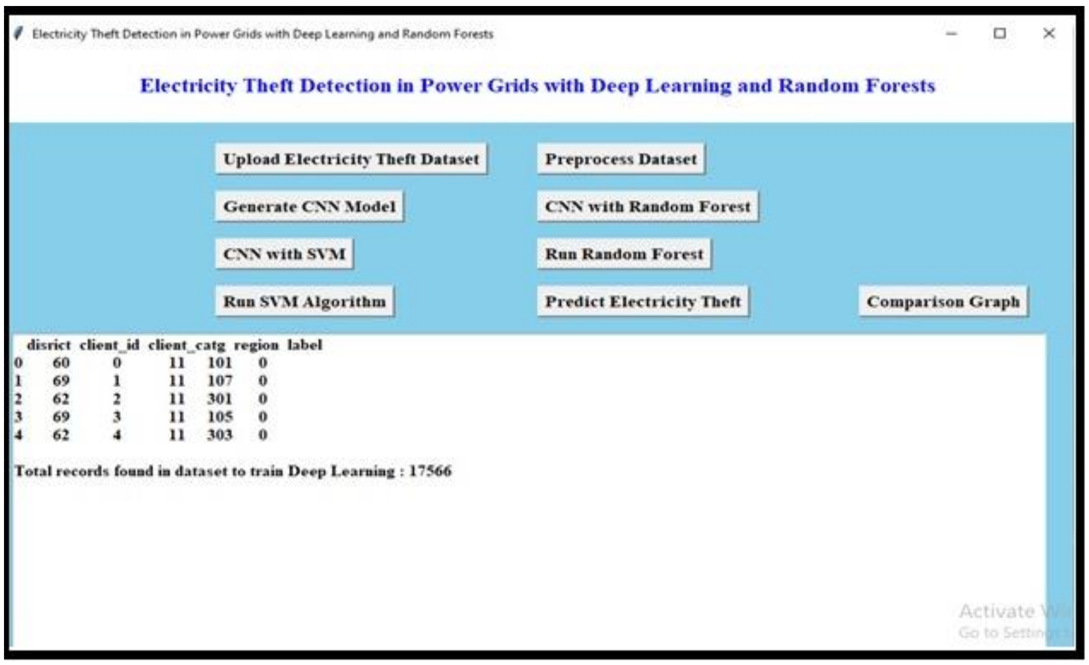
In above screen selecting and uploading ‘ElectricityTheft.csv’ file and then click on ‘Open’ button to load dataset and to get below screen


Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Upload Electricity Theft Dataset
Preprocess Dataset
Generate CNN Model
CNN with Random Forest
CNN with SVM
Run Random Forest
Run SVM Algorithm
Predict Electricity Theft
Comparison Graph
Activate W
Go to Settin

```
C:/Users/Lenovo/OneDrive/Desktop/electricty theft detection - code/ElectricityTheft.csv Loaded
  disrict        client_id client_catg  region creation_date  label
0     60    train_Client_0          11     101    31/12/1994      0
1     69    train_Client_1          11     107    29/05/2002      0
2     62   train_Client_10          11     301    13/03/1986      0
3     69  train_Client_100          11     105     11/7/1996      0
4     62 train_Client_1000          11     303    14/10/2014      0
```

Fig 7.3 DATASET

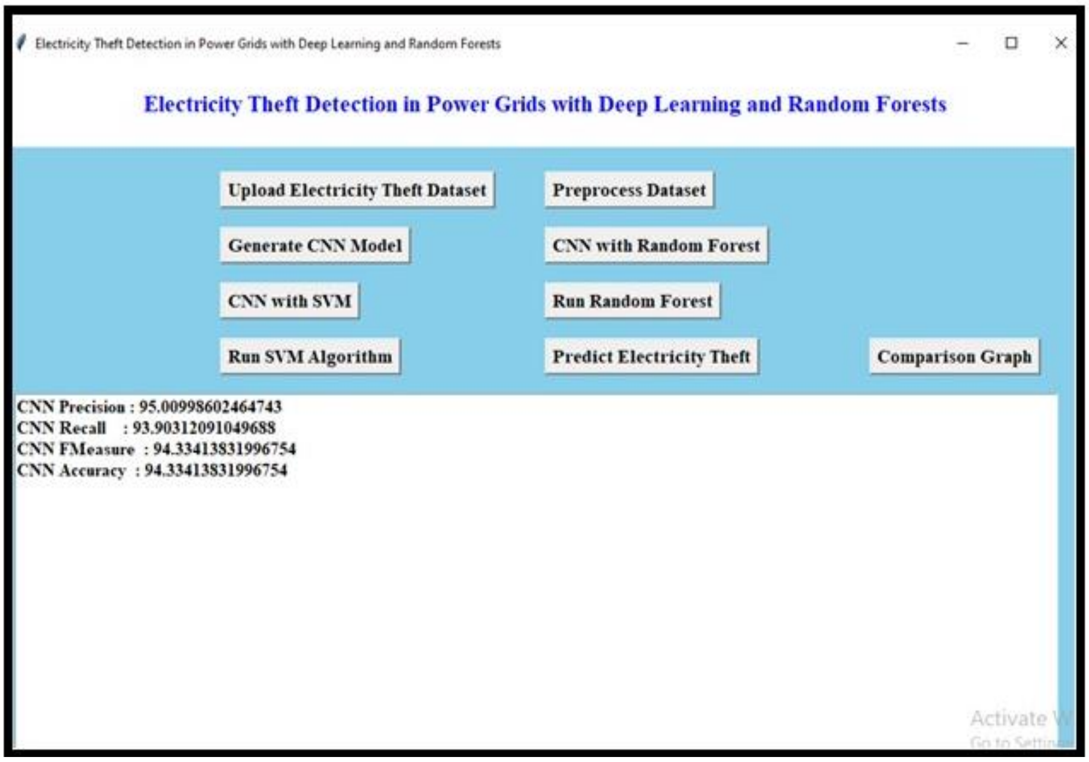
In above screen dataset loaded and displaying few records from dataset and this records contains non-numeric values and this values will not accept by machine learning so we need to convert to numeric by assigning integer ID so click on 'Preprocess Dataset' button to clean data.


Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Upload Electricity Theft Dataset
Preprocess Dataset
Generate CNN Model
CNN with Random Forest
CNN with SVM
Run Random Forest
Run SVM Algorithm
Predict Electricity Theft
Comparison Graph
Activate W
Go to Settings

```
  disrict  client_id  client_catg  region  label
0     60          0           11     101      0
1     69          1           11     107      0
2     62          2           11     301      0
3     69          3           11     105      0
4     62          4           11     303      0

Total records found in dataset to train Deep Learning : 17566
```

Fig 7.4 PREPROCESSED DATASET

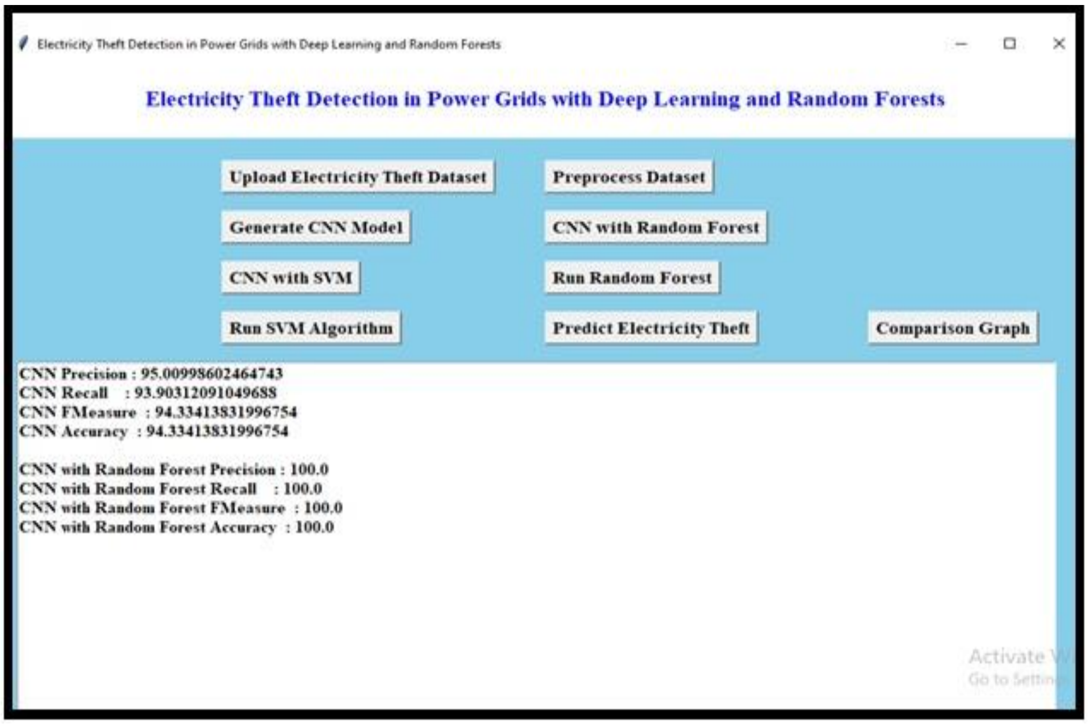
In above screen dataset converted to numeric format and now click on 'Generate CNN Model' button to train CNN with above dataset


Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Upload Electricity Theft Dataset
Preprocess Dataset
Generate CNN Model
CNN with Random Forest
CNN with SVM
Run Random Forest
Run SVM Algorithm
Predict Electricity Theft
Comparison Graph

```
CNN Precision : 95.00998602464743
CNN Recall    : 93.90312091049688
CNN FMeasure  : 94.33413831996754
CNN Accuracy  : 94.33413831996754
```

Fig 7.5 CNN METRICS

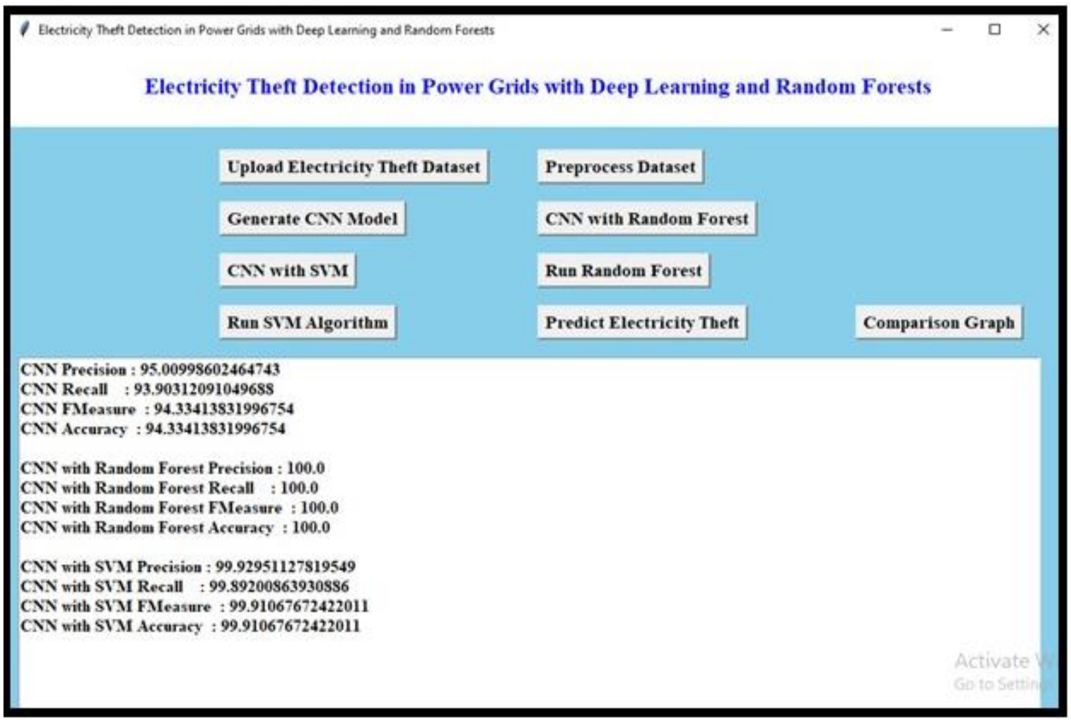
In above screen with normal CNN we got 94% accuracy and now click on ‘CNN with Random Forest’ button to train CNN with RF


Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Upload Electricity Theft Dataset
Preprocess Dataset
Generate CNN Model
CNN with Random Forest
CNN with SVM
Run Random Forest
Run SVM Algorithm
Predict Electricity Theft
Comparison Graph

```
CNN Precision : 95.00998602464743
CNN Recall    : 93.90312091049688
CNN FMeasure  : 94.33413831996754
CNN Accuracy  : 94.33413831996754

CNN with Random Forest Precision : 100.0
CNN with Random Forest Recall    : 100.0
CNN with Random Forest FMeasure  : 100.0
CNN with Random Forest Accuracy  : 100.0
```

Fig 7.6 CNN AND RF METRICS

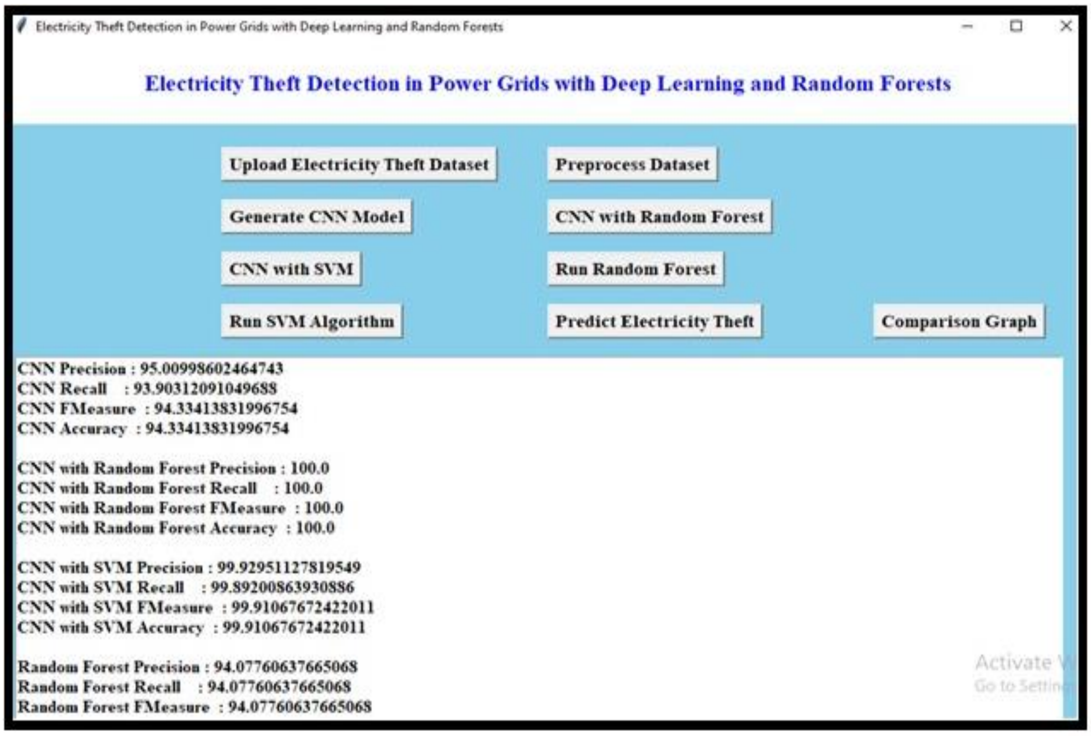
In above screen with CNN-RF we got 100% accuracy and now click on ‘CNN with SVM’ button to train dataset with CNN and SVM


Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Upload Electricity Theft Dataset
Preprocess Dataset
Generate CNN Model
CNN with Random Forest
CNN with SVM
Run Random Forest
Run SVM Algorithm
Predict Electricity Theft
Comparison Graph

```
CNN Precision : 95.00998602464743
CNN Recall    : 93.90312091049688
CNN FMeasure  : 94.33413831996754
CNN Accuracy  : 94.33413831996754

CNN with Random Forest Precision : 100.0
CNN with Random Forest Recall    : 100.0
CNN with Random Forest FMeasure  : 100.0
CNN with Random Forest Accuracy  : 100.0

CNN with SVM Precision : 99.92951127819549
CNN with SVM Recall    : 99.89200863930886
CNN with SVM FMeasure  : 99.91067672422011
CNN with SVM Accuracy  : 99.91067672422011
```

Fig 7.7 CNN WITH SVM METRICS

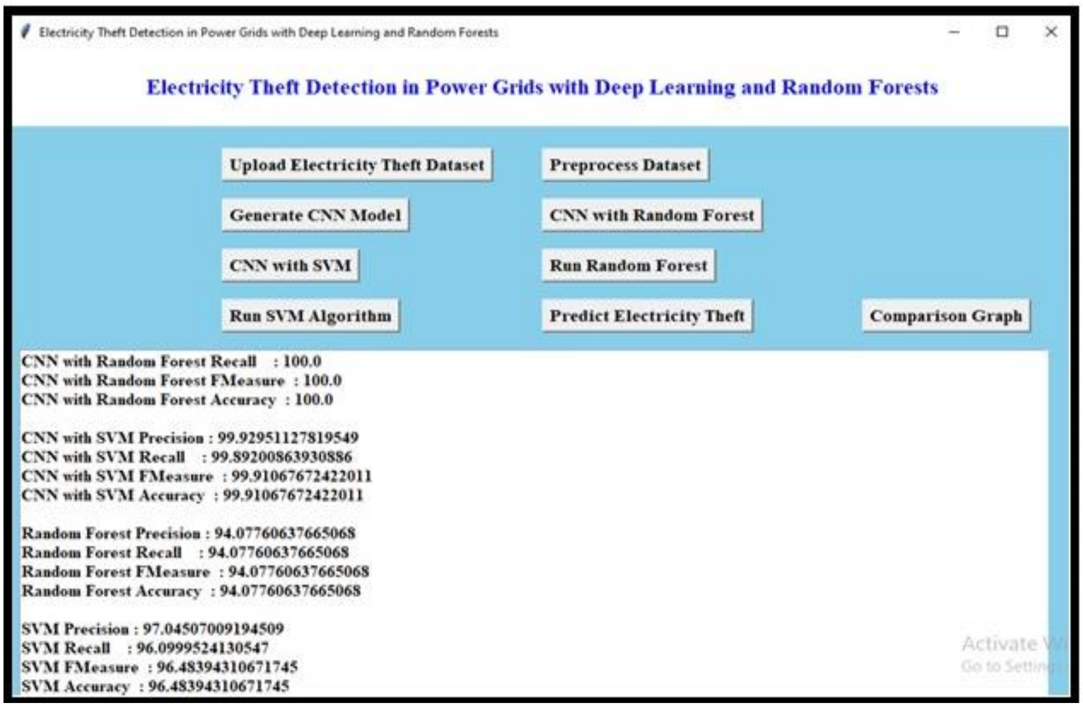
In above screen with CNN-SVM we got 99% accuracy and now click on ‘Run Random Forest’ button to train alone RF on dataset


Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Upload Electricity Theft Dataset
Preprocess Dataset
Generate CNN Model
CNN with Random Forest
CNN with SVM
Run Random Forest
Run SVM Algorithm
Predict Electricity Theft
Comparison Graph

```
CNN Precision : 95.00998602464743
CNN Recall    : 93.90312091049688
CNN FMeasure  : 94.33413831996754
CNN Accuracy  : 94.33413831996754

CNN with Random Forest Precision : 100.0
CNN with Random Forest Recall    : 100.0
CNN with Random Forest FMeasure  : 100.0
CNN with Random Forest Accuracy  : 100.0

CNN with SVM Precision : 99.92951127819549
CNN with SVM Recall    : 99.89200863930886
CNN with SVM FMeasure  : 99.91067672422011
CNN with SVM Accuracy  : 99.91067672422011

Random Forest Precision : 94.07760637665068
Random Forest Recall    : 94.07760637665068
Random Forest FMeasure  : 94.07760637665068
```

Fig 7.8 RANDOM FOREST METRICS

In above screen with alone Random Forest we got 94% accuracy and now click on ‘Run SVM Algorithm’ button to train alone SVM with above dataset


Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Upload Electricity Theft Dataset
Preprocess Dataset
Generate CNN Model
CNN with Random Forest
CNN with SVM
Run Random Forest
Run SVM Algorithm
Predict Electricity Theft
Comparison Graph

```
CNN with Random Forest Recall   : 100.0
CNN with Random Forest FMeasure  : 100.0
CNN with Random Forest Accuracy  : 100.0

CNN with SVM Precision : 99.92951127819549
CNN with SVM Recall   : 99.89200863930886
CNN with SVM FMeasure  : 99.91067672422011
CNN with SVM Accuracy  : 99.91067672422011

Random Forest Precision : 94.07760637665068
Random Forest Recall   : 94.07760637665068
Random Forest FMeasure  : 94.07760637665068
Random Forest Accuracy  : 94.07760637665068

SVM Precision : 97.04507009194509
SVM Recall   : 96.0999524130547
SVM FMeasure  : 96.48394310671745
SVM Accuracy  : 96.48394310671745
```

Fig 7.9 SVM METRICS

In above screen with alone SVM we got 96% accuracy and now click on 'Predict Electricity Theft' button to upload test dataset.

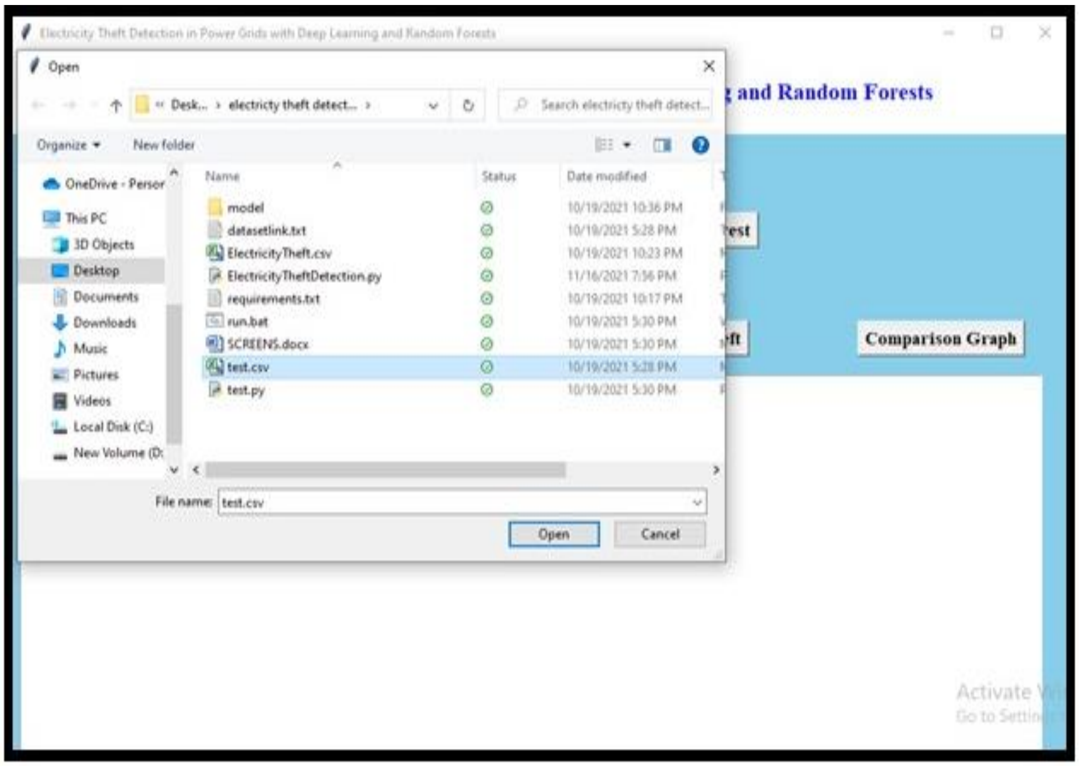
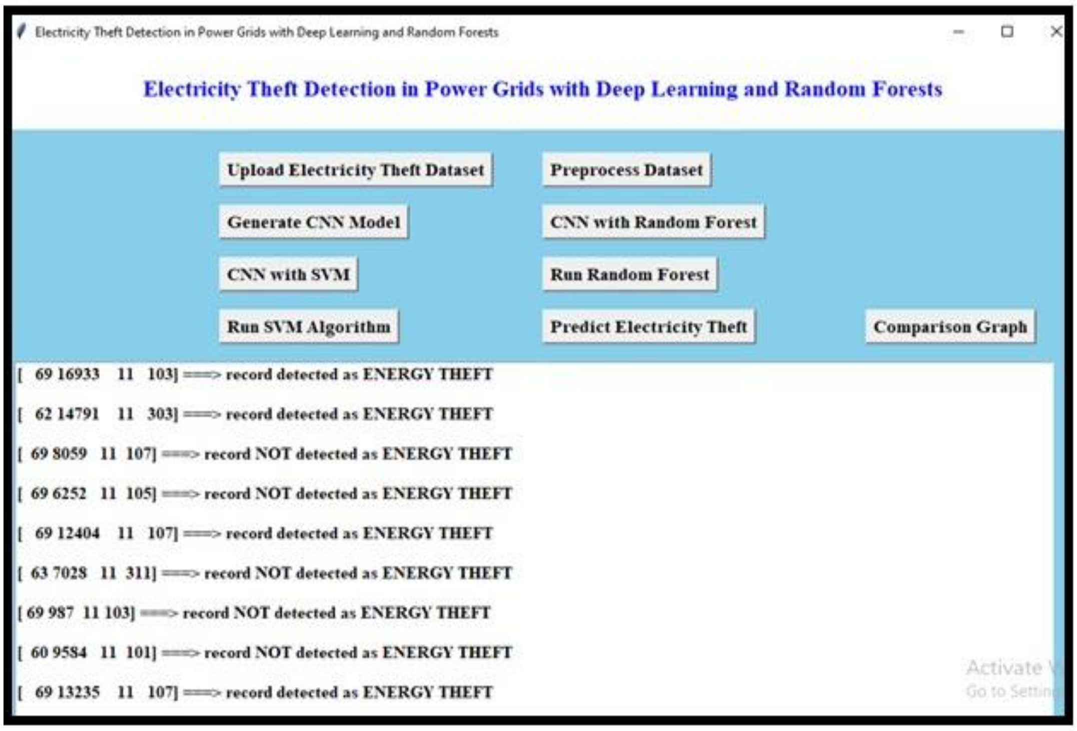


Fig 7.10 UPLOADING TEST DATASET

In above screen selecting and uploading 'test.csv' file and then click on 'Open' button to load test data and to get below prediction result


Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Electricity Theft Detection in Power Grids with Deep Learning and Random Forests
Upload Electricity Theft Dataset
Preprocess Dataset
Generate CNN Model
CNN with Random Forest
CNN with SVM
Run Random Forest
Run SVM Algorithm
Predict Electricity Theft
Comparison Graph

```
[  69 16933   11   103] ===> record detected as ENERGY THEFT
[  62 14791   11   303] ===> record detected as ENERGY THEFT
[ 69 8059   11  107] ===> record NOT detected as ENERGY THEFT
[ 69 6252   11  105] ===> record NOT detected as ENERGY THEFT
[  69 12404   11   107] ===> record detected as ENERGY THEFT
[ 63 7028   11  311] ===> record NOT detected as ENERGY THEFT
[ 69 987  11 103] ===> record NOT detected as ENERGY THEFT
[ 60 9584   11  101] ===> record NOT detected as ENERGY THEFT
[  69 13235   11   107] ===> record detected as ENERGY THEFT
```

Fig 7.11 STATUS OF ENERGY THEFT

In above screen in square brackets we can see test data and after square bracket we can see prediction result as ‘record detected as ENERGY THEFT’ or ‘record NOT detected as ENERGY THEFT’. Now click on ‘Comparison Graph’ button to get below graph

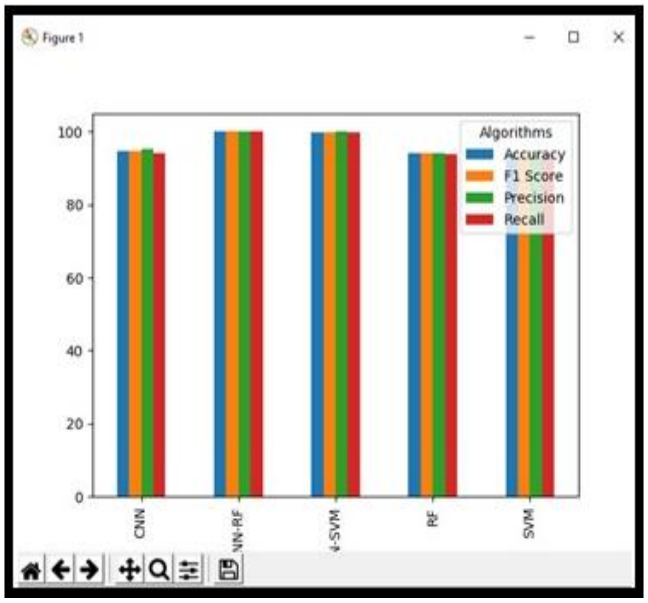


Fig 7.12 COMPARISON GRAPH

In above graph x-axis represents algorithm names and y-axis represents precision, recall, FSCORE and Accuracy for each algorithm and in all algorithms CNN-RF is giving 100% accuracy.

CHAPTER 8
CONCLUSION AND FUTURE ENHANCEMENTS

8.1 CONCLUSION:

In this paper, a novel CNN-RF model is presented to detect electricity theft. In this model, the CNN is similar to an automatic feature extractor in investigating smart meter data and the RF is the output classifier. Because a large number of parameters must be optimized that increase the risk of overfitting, a fully connected layer with a dropout rate of 0.4 is designed during the training phase. In addition, the SMOT algorithm is adopted to overcome the problem of data imbalance. Some machine learning and deep learning methods such as SVM, RF, GBDT, and LR are applied to the same problem as a benchmark, and all those methods have been conducted on SEAI and LCL datasets. -e results indicate that the proposed CNN-RF model is quite a promising classification method in the electricity theft detection field because of two properties: The first is that features can be automatically extracted by the hybrid model, while the success of most other traditional classifiers relies largely on the retrieval of good hand-designed features which is a laborious and time-consuming task. The second lies in that the hybrid model combines the advantages of the RF and CNN, as both are the most popular and successful classifiers in the electricity theft detection field.

8.2 Future Enhancement:

Since the detection of electricity theft affects the privacy of consumers, the future work will focus on investigating how the granularity and duration of smart meter data might affect this privacy. Extending the proposed hybrid CNN-RF model to other applications (e.g., load forecasting) is a task worth investigating.

REFERENCES

JOURNALS:

[1] S. S. S. R. Depuru, L. Wang, and V. Devabhaktuni, “Electricity theft: overview, issues, prevention and a smart meter based approach to control theft,” Energy Policy, vol. 39, no. 2, pp. 1007–1015, 2011.

[2] J. P. Navani, N. K. Sharma, and S. Sapra, “Technical and nontechnical losses in power system and its economic consequence in Indian economy,” International Journal of Electronics and Computer Science Engineering, vol. 1, no. 2, pp. 757–761, 2012.

[3] S. McLaughlin, B. Holbert, A. Fawaz, R. Berthier, and S. Zonouz, “A multi-sensor energy theft detection framework for advanced metering infrastructures,” IEEE Journal on Selected Areas in Communications, vol. 31, no. 7, pp. 1319–1330, 2013.

[4] P. McDaniel and S. McLaughlin, “Security and privacy challenges in the smart grid,” IEEE Security & Privacy Magazine, vol. 7, no. 3, pp. 75–77, 2009.

[5] T. B. Smith, “Electricity theft: a comparative analysis,” Energy Policy, vol. 32, no. 1, pp. 2067–2076, 2004.

[6] J. I. Guerrero, C. Leon, I. Monedero, F. Biscarri, and ´ J. Biscarri, “Improving knowledge-based systems with statistical techniques, text mining, and neural networks for nontechnical loss detection,” Knowledge-Based Systems, vol. 71, no. 4, pp. 376–388, 2014.

[7] C. C. O. Ramos, A. N. Souza, G. Chiachia, A. X. Falcão, and J. P. Papa, “A novel algorithm for feature selection using harmony search and its application for non-technical losses detection,” Computers & Electrical Engineering, vol. 37, no. 6, pp. 886–894, 2011.

[8] P. Glauner, J. A. Meira, P. Valtchev, R. State, and F. Bettinger, “-e challenge of non-technical loss detection using artificial intelligence: a surveyficial intelligence: a

survey," International Journal of Computational Intelligence Systems, vol. 10, no. 1, pp. 760–775, 2017.

[9] S.-C. Huang, Y.-L. Lo, and C.-N. Lu, "Non-technical loss detection using state estimation and analysis of variance," IEEE Transactions on Power Systems, vol. 28, no. 3, pp. 2959–2966, 2013.

[10] O. Rahmati, H. R. Pourghasemi, and A. M. Melesse, "Application of GIS-based data driven random forest and maximum entropy models for groundwater potential mapping: a case study at Mehran region, Iran," CATENA, vol. 137, pp. 360–372, 2016.

TEXTBOOKS:

- Programming Python, Mark Lutz
- Head First Python, Paul Barry
- Core Python Programming,R. Nageswara Rao
- Learning with Python, Allen B. Downey

WEBSITES:

1. https://www.w3schools.com/python/
2. https://www.tutorialspoint.com/python/index.htm
3. https://www.javatpoint.com/python-tutorial
4. https://www.learnpython.org/
5. https://www.pythontutorial.net/