

DETECTION OF SMART GRID ATTACKS USING MACHINE LEARNING TECHNIQUES

A Project Report Submitted in partial fulfillment of the requirements for the award of
the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

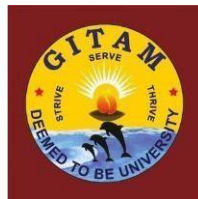
Submitted by

B. Pranitha Reddy, 121810306011

Under the esteemed guidance of

Dr. P. ANURADHA

Assistant professor-CSE



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

(Deemed to be University)

VISAKHAPATNAM

MARCH 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM INSTITUTE OF TECHNOLOGY
GITAM
(Deemed to be University)



DECLARATION

We hereby declare that the project report entitled “**DETECTION OF SMART GRID ATTACKS USING MACHINE LEARNING TECHNIQUES**” is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 31-03-2022

Registration No(s).

Name(s)

Signature(s)

121810306011

B. PRANITHA REDDY

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM INSTITUTE OF TECHNOLOGY
GITAM**

(Deemed to be University)



CERTIFICATE

This is to certify that the project report entitled "**DETECTION OF SMART GRID ATTACKS USING MACHINE LEARNING TECHNIQUES**" is a bonafide record of work carried out by **B. PRANITHA REDDY (121810306011), K. KIRAN SAI (121810306004), B. ANIL (121810306036)** submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

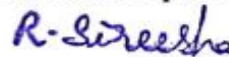
Project Guide



Dr. P. ANURADHA

ASSOCIATE PROFESSOR

Head of the Department



Dr. R. SIREESHA

PROFESSOR

Head of the Department
Department of Computer Science & Engineering
GITAM Institute of Technology
Gandhi Institute of Technology and Management (GITAM)
(Deemed to be University)
Visakhapatnam-520 045

TABLE OF CONTENTS

1.	Abstract	1
2.	Introduction	2-3
3.	Literature Review	4-5
4.	Problem Identification & Objectives	6-7
5.	System Methodology	8-13
6.	Overview of Technologies	14
7.	Implementation	15-21
	8.1 Coding	22-24
	8.2 Testing	
8.	Results & Discussions	25-27
9.	Conclusion & Future Scope	28
10.	References	29

1. ABSTRACT

As cybersecurity is becoming a major concern in Smart Grid Technology, it is important to reduce the vulnerabilities in it. Experiments reveal that machine learning techniques outperform traditional attack detection algorithms in detecting attacks. Machine learning approaches were used to examine malicious activity and intrusion detection challenges at the network layer of smart grid communication systems. In this project, machine learning algorithms are used to classify the measurements as either being attacked or secured. The proposed system aims to predict the false data injections in the smart grid by using various machine learning algorithms such as Perceptron, Logistic Regression, Support Vector Machine and KNN algorithm.

2. INTRODUCTION

Machine learning methods for power system monitoring and control have been frequently proposed in the literature on smart grids. Rudin presents an intelligent system design paradigm that use machine learning techniques to predict system component failures. Anderson uses machine learning techniques to control energy in smart grid networks by managing demands and sources. Machine learning approaches were used to examine malicious activity and intrusion detection challenges at the smart grid communication system's network layer.

This project will focus on identifying fake data injection attacks on the smart grid's physical layer. In this model, the attack is guided by introducing erroneous data into the observed local measurements by either a local carrier in a hierarchical network or a smart phasor measurement unit (PMU). The measurements are organized into clusters in this way.

The system's state is determined initially from the observed data in attack detection approaches that use State Vector Estimation (SVE). The exact recovery of state vectors in sparse networks, where the Jacobian measurement matrix is sparse, is a challenge for SVE-based techniques. The problem can be solved using sparse reconstruction methods, the sparsity of the state vectors, however, limits the performance of this technique.

Ozay, who predicted fake data injection attacks using supervised learning methods, provides a full review of the methodologies. The smart grid's validity of statistical learning theory's major assumptions is also investigated. Semi-supervised and online learning techniques, as well as fusion techniques at the decision and feature levels, are then offered in a generic attack construction framework for use in hierarchical and topological networks in a variety of attack scenarios.

3. LITERATURE REVIEW

1. "Machine Learning Methods for Attack Detection in the Smart Grid"- Mete Ozay, IEEE, Member, Inaki Esnaola, IEEE, Member

Machine learning methods are used to determine if measurements are safe or vulnerable. In the suggested technique, an attack detection framework is proposed that makes advantage of any past system information and overcomes limits imposed by the problem's sparse nature. To simulate the attack detection problem, with decision- and feature-level fusion, well-known batch and online learning approaches (supervised and semi-supervised) are integrated.

2. A. Boulanger, R. N. Anderson, W. Scott, and W. B. Powell, "Adaptive stochastic control for the smart grid," Proc. IEEE, vol. 99

Concerns about electricity production and distribution, the traditional electric power system has evolved into a smart grid due to efficiency, dependability, economics, and sustainability. Throughout the electrical system, there is two-way communication which is a fundamental enabler of the smart grid, allowing an advanced information system to make the best judgments possible about power system performance. Due to the expected extensive penetration of renewable energy sources, energy storage devices, demand side management (DSM) technologies, and electric vehicles (EVs) in the future smart grid, there are significant technological challenges in power system planning and operation. Effective stochastic information management strategies should be developed to meet the unpredictability in renewable energy generation, the buffering impact of energy storage devices, consumer behavior patterns in the context of DSM, and the high mobility of EVs.

3. Y. Nozaki, N. Kato, M. M. Fouda, X. Shen, and Z. M. Fadlullah, “An early warning system against malicious activities for smart grid communications,”

In today's Machine-to-Machine (M2M) sector, smart grid (SG) has the most potential for growth. Thanks to recent advancements in M2M technology, smart meters/sensors utilized in smart grid are expected to eliminate the need for human engagement in specifying power demand and energy distribution. The controller may receive data such as power usage and other monitoring signals from these numerous sensors.

4. L. Wang, Y. Zhang, W. Sun, M. Alam, and R. C. Green, “Distributed intrusion detection system in a multi-layer network architecture of smart grids”.

Several control methods for smart grids have been developed, based on centralized, decentralized, or hybrid approaches. The Secure Overlay Communications and Control Architecture is a decentralized, peer-to-peer control and communications paradigm that connects a physical power system to its communications and control systems. It has its own set of communication protocols as well as intrusion detection systems.

4. PROBLEM IDENTIFICATION AND OBJECTIVES

EXISTING SYSTEM

In certain smart cities, the smart grid controls the use of power, the opening and closing of doors, and other operations. Some malicious users may attempt to target this smart grid system in order to propagate or inject fake information, and the smart grid may be activated based on this false information, potentially resulting in significant financial loss. Existing techniques for detecting such attacks used state vector estimation to check if values are within a given threshold; if values are outside of that threshold, it is considered an attack. However, this technique is not reliable for detecting attacks when values are mixed with genuine and attack data, as it is insufficient to detect unobserved data mixed with observe data.

DISADVANTAGES OF EXISTING SYSTEM

Existing techniques for detecting attacks used state vector estimation to check if values are within a given threshold, and if they are outside of that threshold, it was considered an attack. However, this technique is not reliable for detecting attacks when values are mixed with genuine and attack data, as it is insufficient to detect unobserved data mixed with observe data.

PROPOSED SYSTEM

We use a variety of machine learning algorithms to detect harmful attacks from both observed and unobserved data in order to avoid dangerous assaults. Because machine learning algorithms are good at noticing unobserved data mixed with observe data, they can predict fake injection attacks combined with observed data.

ADVANTAGES OF PROPOSED SYSTEM

The alarm system allows the smart grid control center to predict various hostile occurrences, allowing SG to react ahead of time and limit the potential consequences of the malicious behavior. We use computer simulations to test the usefulness of the suggested early warning system.

5. SYSTEM METHODOLOGY

SYSTEM DESIGN

The technical heart of the software engineering process is software design, which is employed regardless of development methodology or application area. The first step in the creation of any technical product or system is design. The goal of the designer is to build a model or representation of entity which will subsequently be developed.

The aim of design is identifying the components of software and their interactions. By considering the software structure and offering the blue print, one can provide a blueprint for the phase of the document. One of the desired characteristics of large systems is modularity. It signifies that the system is divided into several parts. In this way, the relationship between the parts is presented in the simplest possible way.

Developers are responsible for bridging the gap between the requirements specification established during requirements elicitation and analysis and the system delivered to the user during system design activities. Design is the area where excellence is promoted through progress. Software design is the process of translating the requirements into the software's visual representation.

SYSTEM ARCHITECTURE

Figure 1

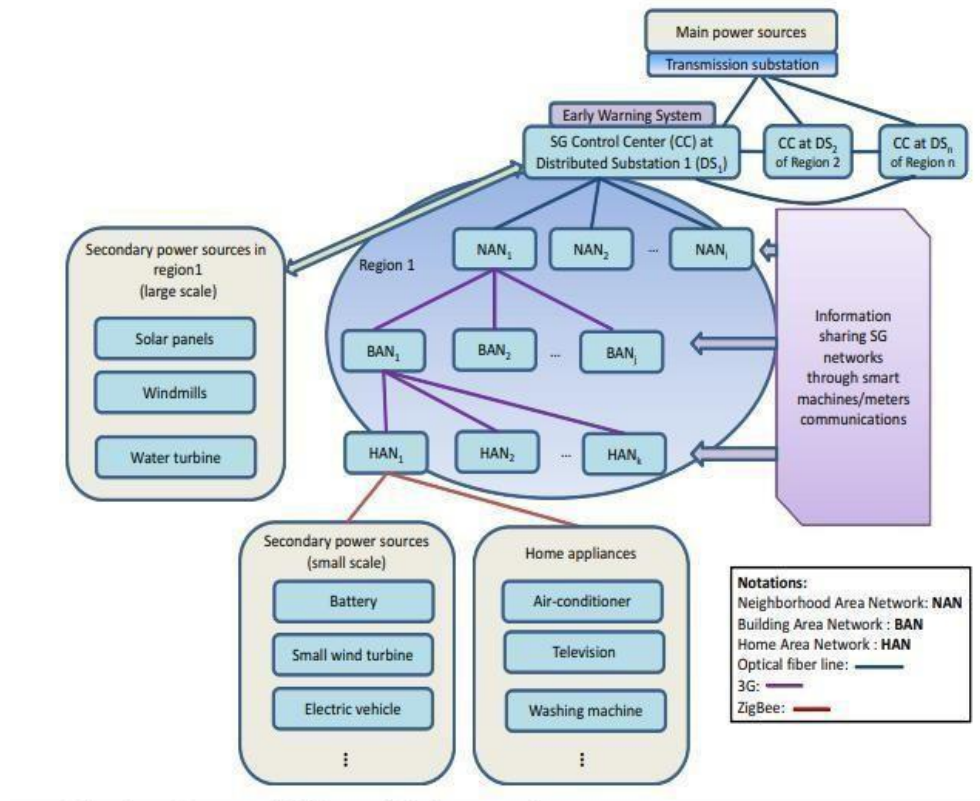
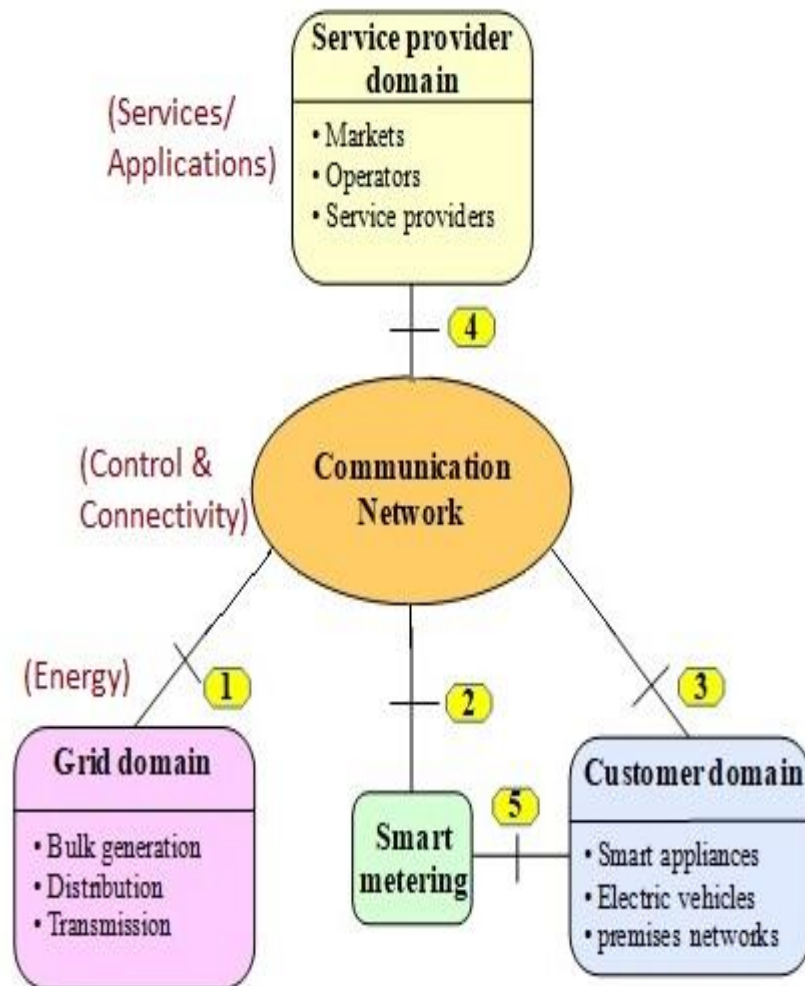


Figure 2

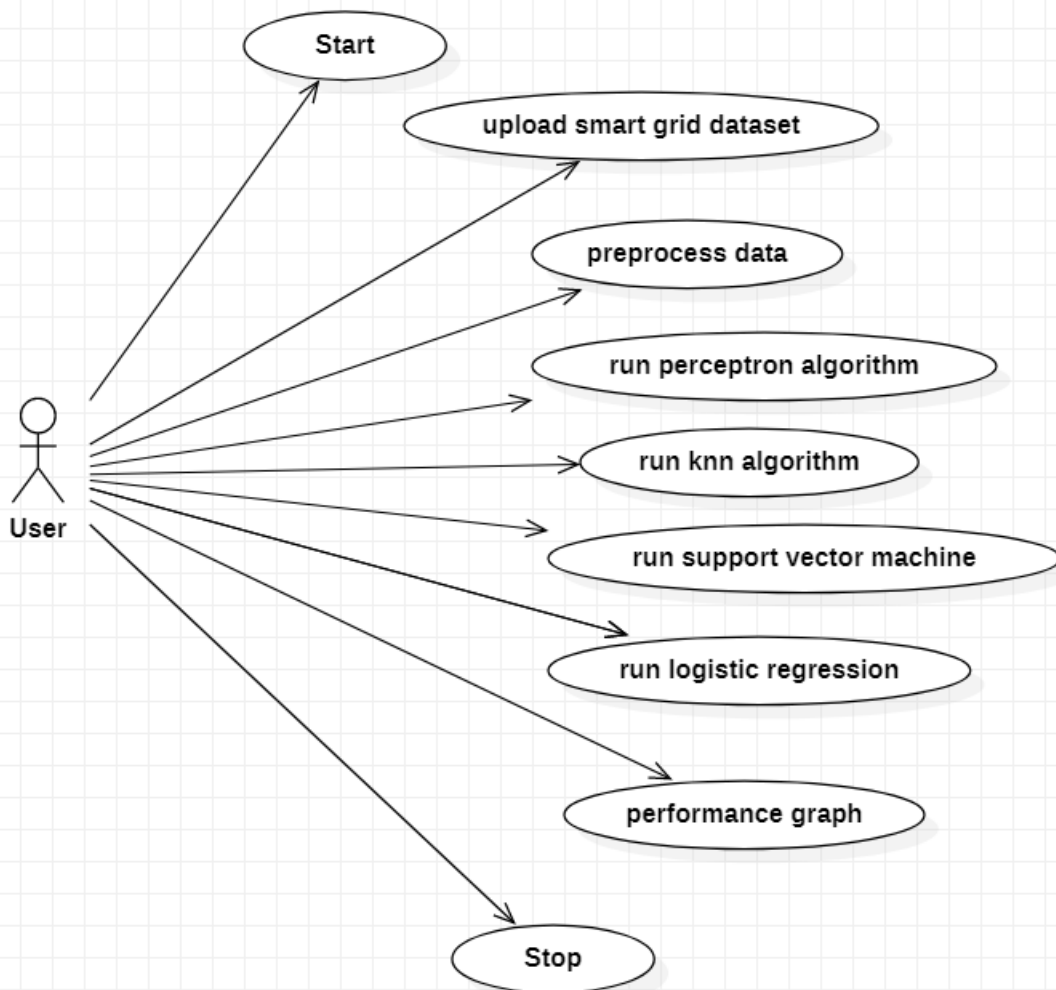


UML Diagrams:

1. USE CASE DIAGRAM

Use case diagrams are used to depict and document a system's behavior and requirements. A system's high-level functionality and scope are depicted in use-case diagrams.

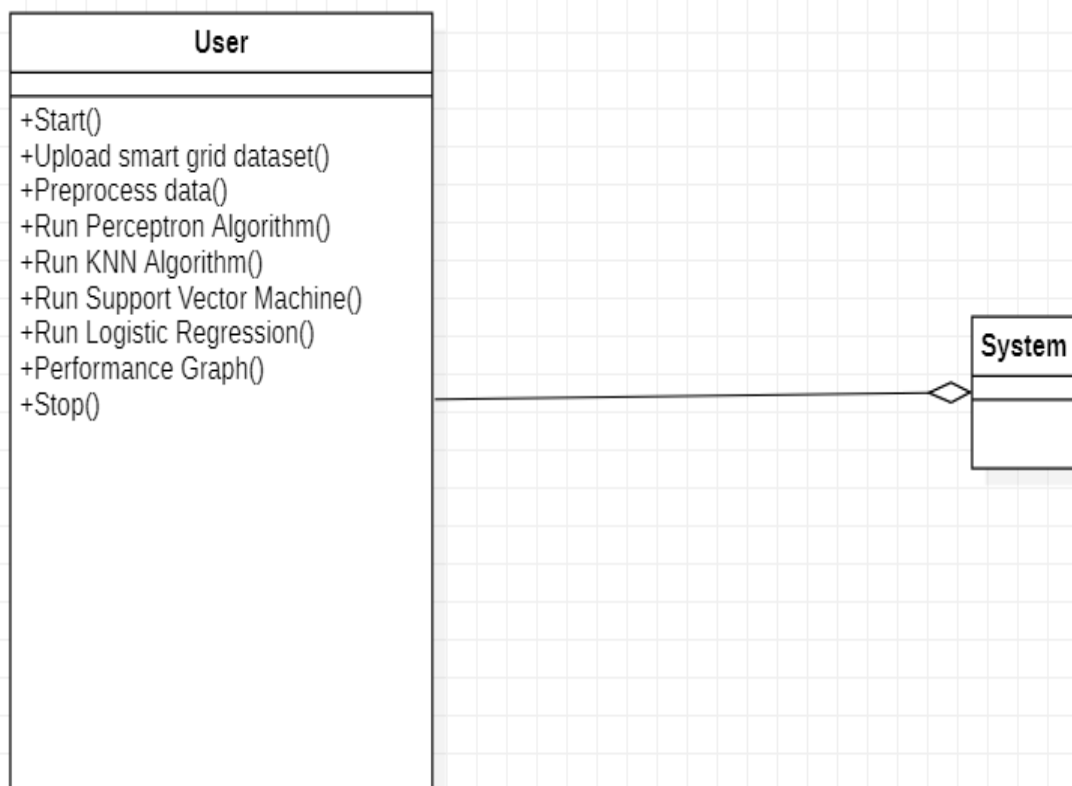
Figure 3



2. CLASS DIAGRAM:

The class diagram represents the attributes of class and its activities, as well as system constraints. Class diagrams are the UML diagrams that are translated directly into an object oriented language, they are employed frequently for modelling object oriented systems.

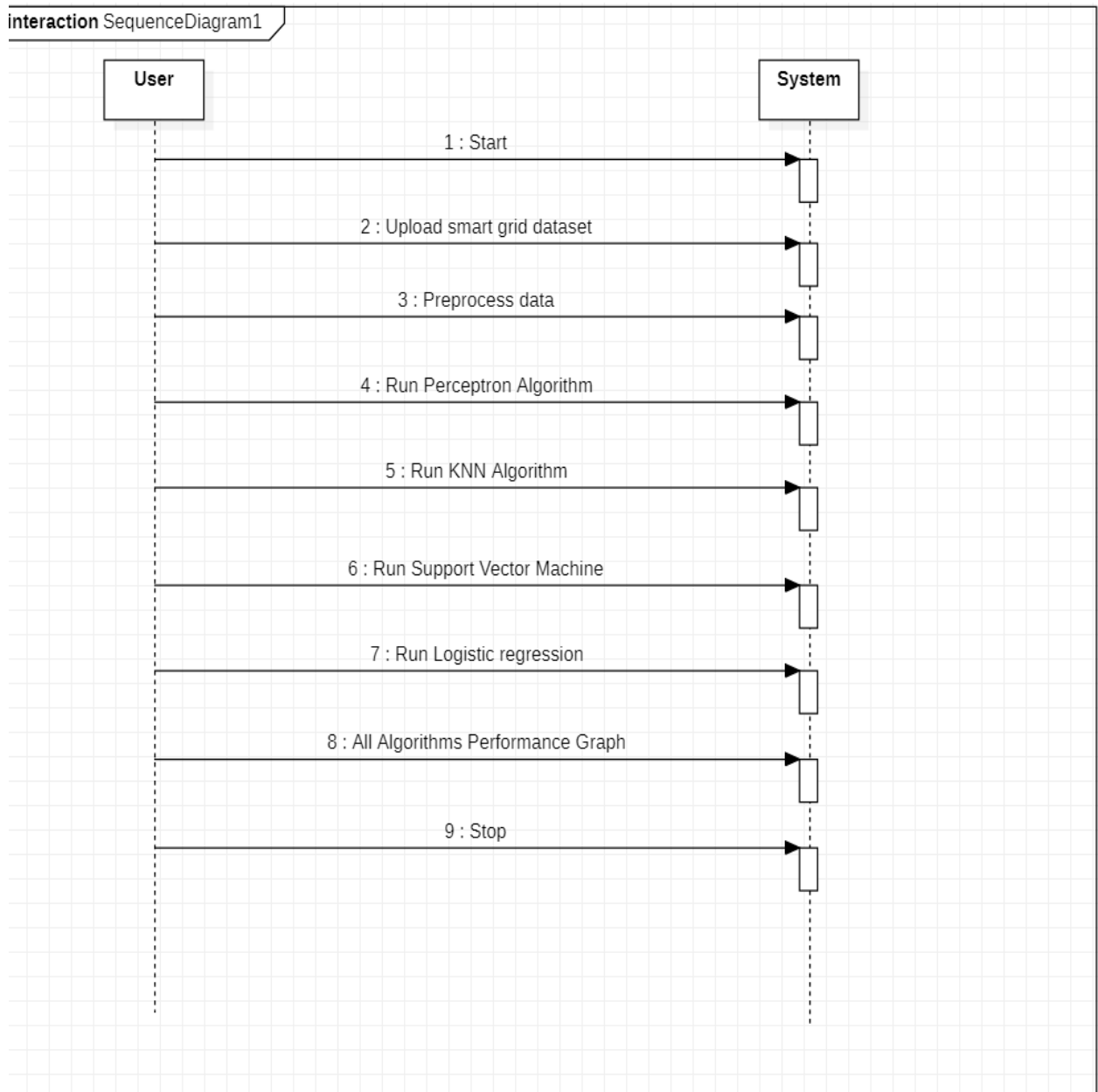
Figure 4



3. SEQUENCE DIAGRAM:

The sequence diagram depicts the messages flow through the system. It may be used to look at various dynamic settings. The communication between the lifelines is shown as events sequence that occurs in an order.

Figure 5



6. OVERVIEW OF TECHNOLOGIES

SOFTWARE REQUIREMENTS

Operating system	:Windows 7,8,10 Ultimate, Linux, Mac
Front-End	:Python
Software Environment	:Sublime Text and Anaconda (Jupyter)
Coding Language	:Python

HARDWARE REQUIREMENTS

System	: Intel I-5, 3, 7 processor
Floppy Drive	: 1.44 Mb
Hard Disk	: 500 GB
Ram	: 4Gb

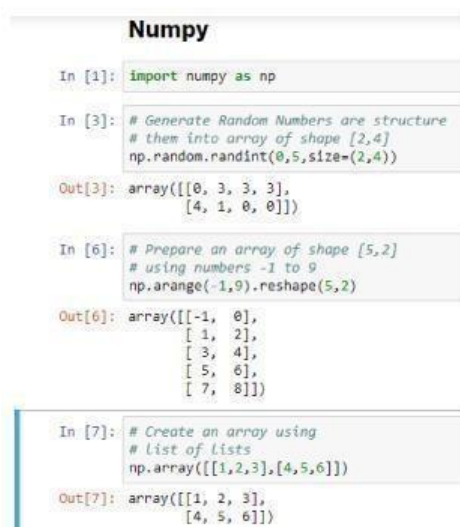
7. SYSTEM IMPLEMENTATION

MODULES

1. Numpy

Python provides a wide range of data types and data structures from which to pick. It wasn't, however, designed with Machine Learning in mind. Python is a computer language that uses NumPy (pronounced as num-pee). Numpy is a data management package that lets us work with multi-dimensional arrays and a wide range of mathematical functions.

Figure 6



```
Numpy

In [1]: import numpy as np

In [3]: # Generate Random Numbers are structure
# them into array of shape [2,4]
np.random.randint(0,5,size=(2,4))

Out[3]: array([[0, 3, 3, 3],
               [4, 1, 0, 0]])

In [6]: # Prepare an array of shape [5,2]
# using numbers -1 to 9
np.arange(-1,9).reshape(5,2)

Out[6]: array([[ -1,  0],
               [  1,  2],
               [  3,  4],
               [  5,  6],
               [  7,  8]])

In [7]: # Create an array using
# list of lists
np.array([[1,2,3],[4,5,6]])

Out[7]: array([[1, 2, 3],
               [4, 5, 6]])
```

2. Pandas

Pandas usually writes and reads the data from different formats including CSVs and other file systems. It includes features such as adding, updating, and deleting columns, merging or splitting data frames/series, working with datetime objects, imputation null/missing values, working with time series data, and converting to and from NumPy objects, among others.

Figure 7

Pandas

```
In [8]: import pandas as pd

In [10]: pd_series = pd.Series(data=['Val1', 'Val2', 'Val3'], index=range(0,3), name='Series_object')

In [11]: pd_series
Out[11]: 0    Val1
         1    Val2
         2    Val3
         Name: Series_object, dtype: object

In [14]: df = pd.DataFrame(data={'col_1': [1,2,3,4],
                                   'col_2': ['A', 'B', 'C', 'D']})

In [15]: df
Out[15]:
```

	col_1	col_2
0	1	A
1	2	B
2	3	C
3	4	D

3. Matplotlib

Matplotlib, another component of the SciPy stack, is a visualization package. It's compatible with NumPy objects (and its high-level derivatives like pandas). Matplotlib is a plotting environment similar to MATLAB that may be used to create high-quality figures/charts for papers, notebooks, web applications, and other applications.

Figure 8



4. Scikit-Learn

Scikit learn provides a suite of tools for machine learning and statistical learning that includes clustering, regression and dimensionality reduction using a python interface. The foundations of sklearn are mostly written in python.

5. Tkinter

Tkinter is used to create Graphical User Interfaces (GUI) and mostly commonly used. It is the fastest and easiest way to create a GUI .

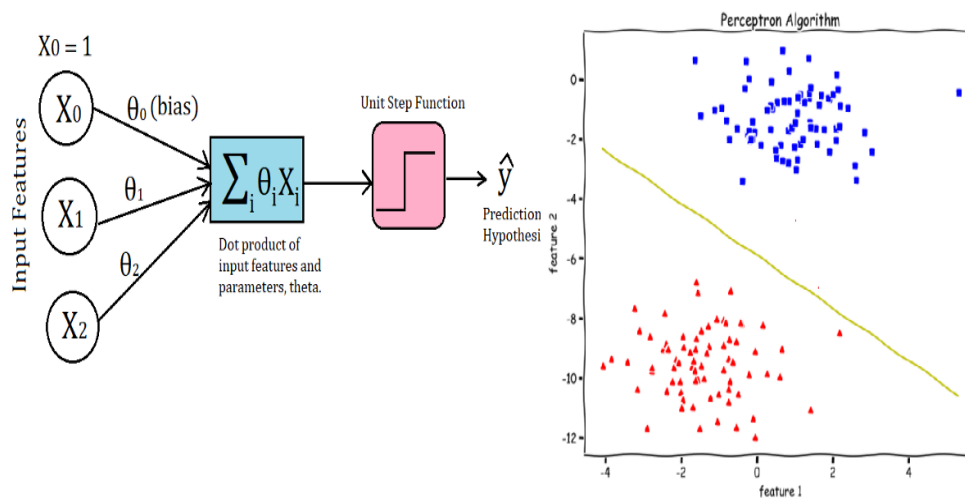
PERCEPTRON LEARNING ALGORITHM

The Perceptron is a basic ANN unit inspired by the information processing of neurons. Before being transmitted to the activation function, the Perceptron, like the neuron, receives input signals from instances of training data that are weighted and blended in a linear equation (transfer function). It returns 1 if the result exceeds a particular threshold; else, it returns -1. In other words, if perceptron is given inputs ranging from x_1 to x_n , perceptron's output is

$$o(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

where the contribution of input x_i to the perceptron output is determined by each w_i , which is a weight.

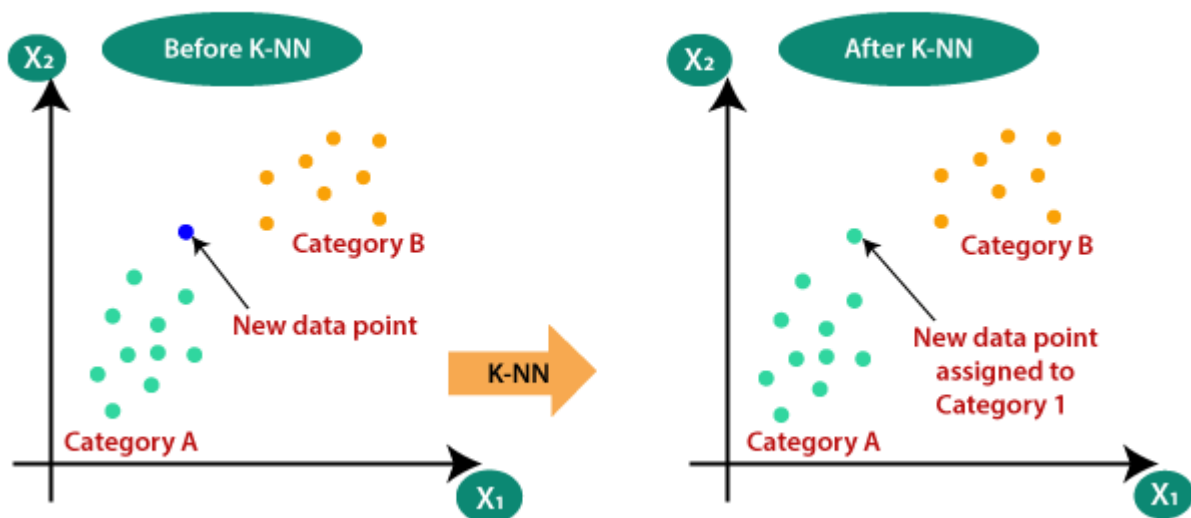
Figure 9



K- Nearest Neighbor (KNN) Algorithm

K nearest neighbor is a supervised learning approach that is used to resample datasets. It is used to predict the class of a new datapoint.

Figure 10

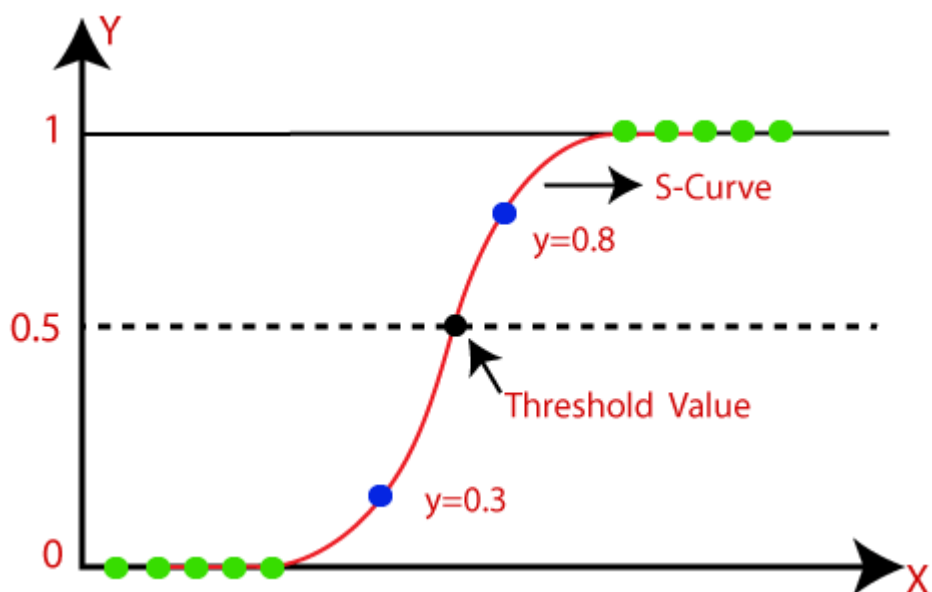


Logistic Regression

Logistic regression comes under the supervised learning approach and is a popular algorithm. This algorithm is used to predict the dependent variable that is categorical from various independent variables.

The outcome can either be a discrete value or categorical value. The output may be no or yes, 1 or zero, false or true, etc.

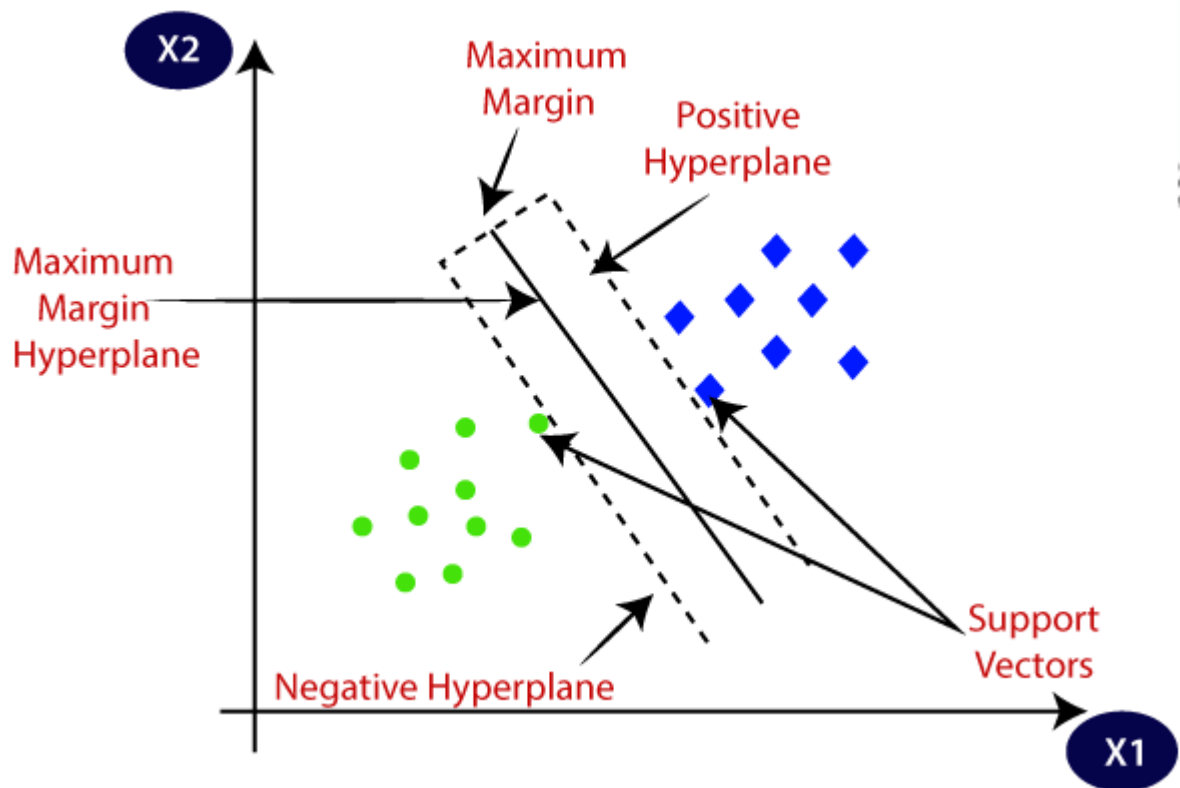
Figure 11



Support Vector Machine Algorithm:

The main aim of the SVM algorithm is to decide the optimal line for classifying n dimensional space into the classes so that the subsequent data points may be placed easily in the relevant category. Most ideal decision boundary is known as a hyperplane.

Figure 12



CODE:

```
def processDataset():
    global X, Y
    global dataset
    global X_train, X_test, y_train, y_test
    dataset['marker'] = pd.Series(1e.fit_transform(dataset['marker']))
    #dataset = dataset.fillna(dataset.mean())
    temp = dataset
    dataset = dataset.values
    X = dataset[:,0:dataset.shape[1]-1]
    Y = dataset[:,dataset.shape[1]-1]
    X = X.round(decimals=4)
    #X = normalize(X)
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    print(Y)
    #text.insert(END,Y)
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.1)
    print(y_test)
    #text.insert(END,y_test)
    print(y_train)
    #text.insert(END,y_train)
    text.insert(END,"Total records found in dataset are : "+str(X.shape[0])+"\n")
    text.insert(END,"Total records used to train machine learning algorithms are : "+str(X_train.shape[0])+"\n")
    text.insert(END,"Total records used to test machine learning algorithms are : "+str(X_test.shape[0])+"\n\n")
    text.insert(END,str(temp)+"\n\n")
```

```
def runPerceptron():
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    accuracy.clear()
    precision.clear()
    recall.clear()
    fscore.clear()

    cls = Perceptron(class_weight='balanced')
    cls.fit(X_train,y_train)
    predict = cls.predict(X_test)
    a = accuracy_score(y_test,predict) * 100
    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
    text.insert(END,"Perceptron Precision : "+str(p)+"\n")
    text.insert(END,"Perceptron Recall : "+str(r)+"\n")
    text.insert(END,"Perceptron FScore : "+str(f)+"\n")
    text.insert(END,"Perceptron Accuracy : "+str(a)+"\n\n")
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
```

```

def runKNN():
    global classifier
    global X_train, X_test, y_train, y_test
    cls = KNeighborsClassifier(n_neighbors = 3)
    cls.fit(X_train,y_train)
    predict = cls.predict(X_test)
    a = accuracy_score(y_test,predict) * 100
    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
    text.insert(END,"KNN Precision : "+str(p)+"\n")
    text.insert(END,"KNN Recall : "+str(r)+"\n")
    text.insert(END,"KNN FScore : "+str(f)+"\n")
    text.insert(END,"KNN Accuracy : "+str(a)+"\n\n")
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
    classifier = cls

```

```

def logisticRegression():
    global X_train, X_test, y_train, y_test
    cls = LogisticRegression(class_weight='balanced')
    cls.fit(X_train,y_train)
    predict = cls.predict(X_test)
    a = accuracy_score(y_test,predict) * 100
    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
    text.insert(END,"Logistic Regression Precision : "+str(p)+"\n")
    text.insert(END,"Logistic Regression Recall : "+str(r)+"\n")
    text.insert(END,"Logistic Regression FScore : "+str(f)+"\n")
    text.insert(END,"Logistic Regression Accuracy : "+str(a)+"\n\n")
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)

```

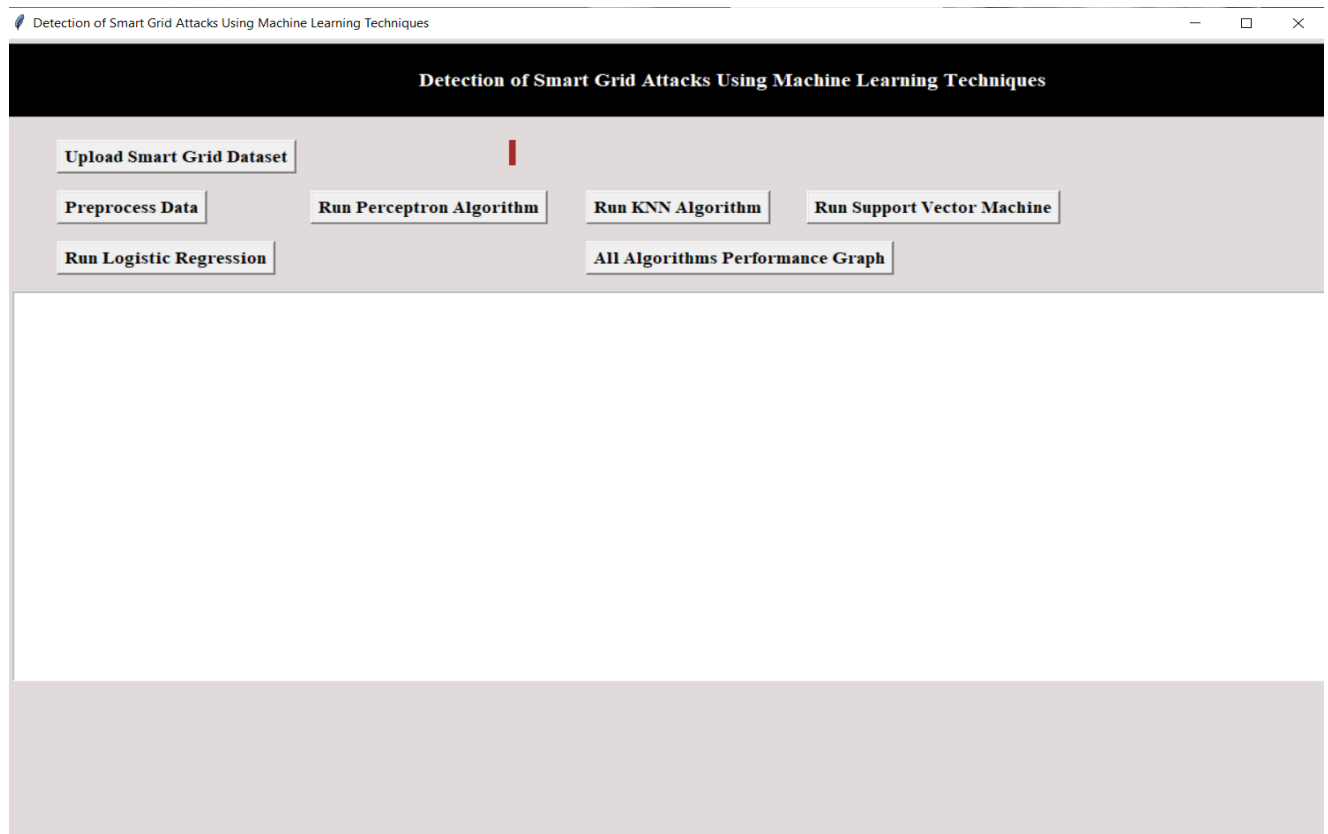
```

def runSVM():
    global X_train, X_test, y_train, y_test
    cls = svm.SVC(class_weight='balanced')
    cls.fit(X_train, y_train)
    predict = cls.predict(X_test)
    a = accuracy_score(y_test, predict) * 100
    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
    text.insert(END, "SVM Precision : "+str(p)+"\n")
    text.insert(END, "SVM Recall : "+str(r)+"\n")
    text.insert(END, "SVM FScore : "+str(f)+"\n")
    text.insert(END, "SVM Accuracy : "+str(a)+"\n\n")
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)

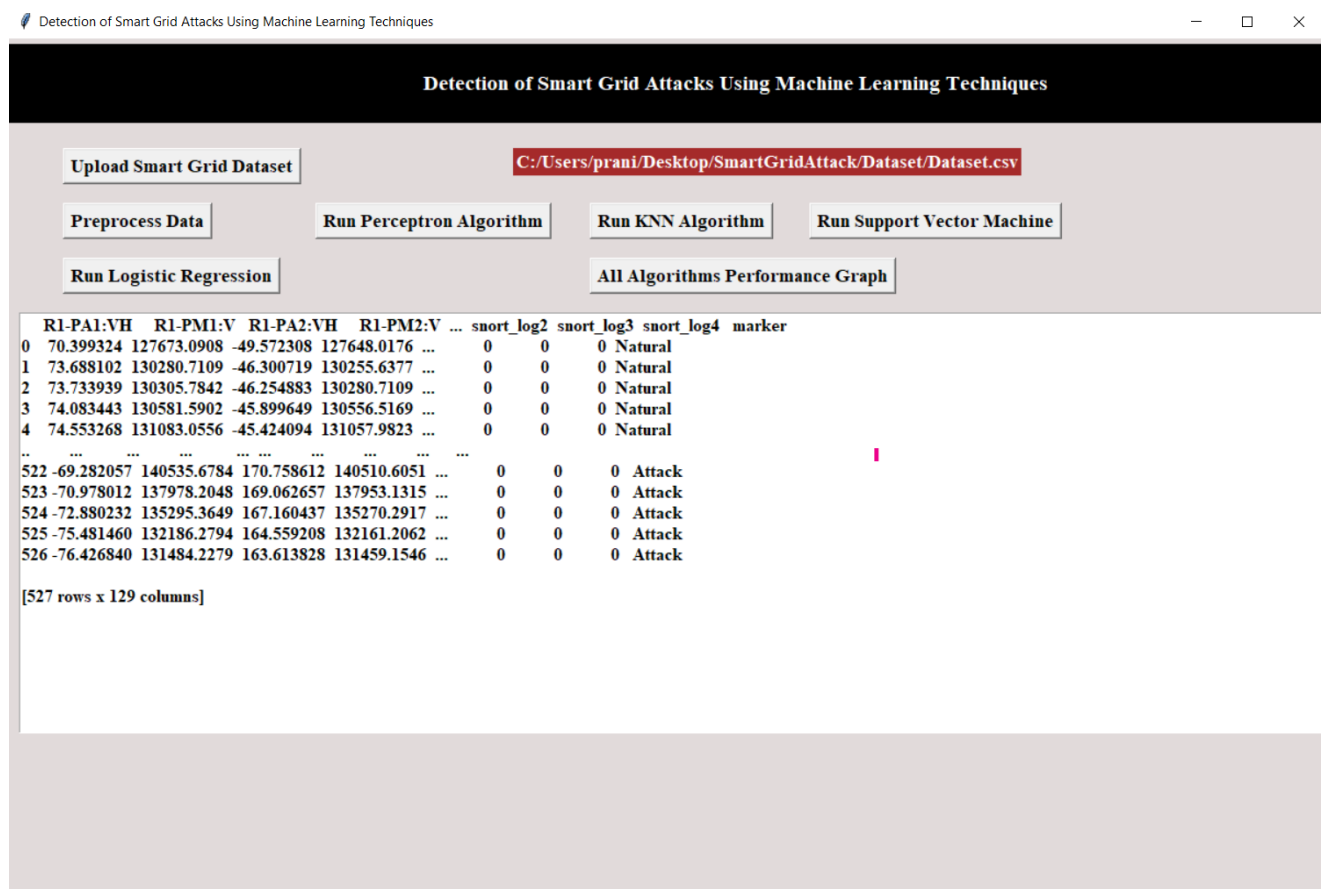
def graph():
    df = pd.DataFrame([['Perceptron', 'Precision', precision[0]], ['Perceptron', 'Recall', recall[0]], ['Perceptron', 'F1 Score', fscore[0]], ['Perceptron', 'Accuracy', accuracy[0]],
                        ['KNN', 'Precision', precision[1]], ['KNN', 'Recall', recall[1]], ['KNN', 'F1 Score', fscore[1]], ['KNN', 'Accuracy', accuracy[1]],
                        ['SVM', 'Precision', precision[2]], ['SVM', 'Recall', recall[2]], ['SVM', 'F1 Score', fscore[2]], ['SVM', 'Accuracy', accuracy[2]],
                        ['Logistic Regression', 'Precision', precision[3]], ['Logistic Regression', 'Recall', recall[3]], ['Logistic Regression', 'F1 Score', fscore[3]], ['Logistic Regression', 'Accuracy', accuracy[3]]],
                        columns=['Parameters', 'Algorithms', 'Value'])
    df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')
    plt.show()

```

8. RESULTS



After uploading Smart Grid dataset



Data Preprocessing:

Detection of Smart Grid Attacks Using Machine Learning Techniques

Detection of Smart Grid Attacks Using Machine Learning Techniques

Upload Smart Grid Dataset

C:/Users/prani/Desktop/SmartGridAttack/Dataset/Dataset.csv

Preprocess Data

Run Perceptron Algorithm

Run KNN Algorithm

Run Support Vector Machine

Run Logistic Regression

All Algorithms Performance Graph

[527 rows x 129 columns]

Total records found in dataset are : 527

Total records used to train machine learning algorithms are : 474

Total records used to test machine learning algorithms are : 53

	R1-PA1:VH	R1-PM1:V	R1-PA2:VH	R1-PM2:V	...	snort_log2	snort_log3	snort_log4	marker
0	70.399324	127673.0908	-49.572308	127648.0176	...	0	0	0	1
1	73.688102	130280.7109	-46.300719	130255.6377	...	0	0	0	1
2	73.733939	130305.7842	-46.254883	130280.7109	...	0	0	0	1
3	74.083443	130581.5902	-45.899649	130556.5169	...	0	0	0	1
4	74.553268	131083.0556	-45.424094	131057.9823	...	0	0	0	1
...
522	-69.282057	140535.6784	170.758612	140510.6051	...	0	0	0	0
523	-70.978012	137978.2048	169.062657	137953.1315	...	0	0	0	0
524	-72.880232	135295.3649	167.160437	135270.2917	...	0	0	0	0
525	-75.481460	132186.2794	164.559208	132161.2062	...	0	0	0	0
526	-76.426840	131484.2279	163.613828	131459.1546	...	0	0	0	0

[527 rows x 129 columns]

Algorithms Output:

Detection of Smart Grid Attacks Using Machine Learning Techniques

Detection of Smart Grid Attacks Using Machine Learning Techniques

Upload Smart Grid Dataset

C:/Users/prani/Desktop/SmartGridAttack/Dataset/Dataset.csv

Preprocess Data

Run Perceptron Algorithm

Run KNN Algorithm

Run Support Vector Machine

Run Logistic Regression

All Algorithms Performance Graph

Perceptron Precision : 63.46153846153846

Perceptron Recall : 51.28205128205128

Perceptron FScore : 23.71212121212121

Perceptron Accuracy : 28.30188679245283

KNN Precision : 93.75

KNN Recall : 97.43589743589743

KNN FScore : 95.35087719298245

KNN Accuracy : 96.22641509433963

SVM Precision : 36.79245283018868

SVM Recall : 50.0

SVM FScore : 42.39130434782608

SVM Accuracy : 73.58490566037736

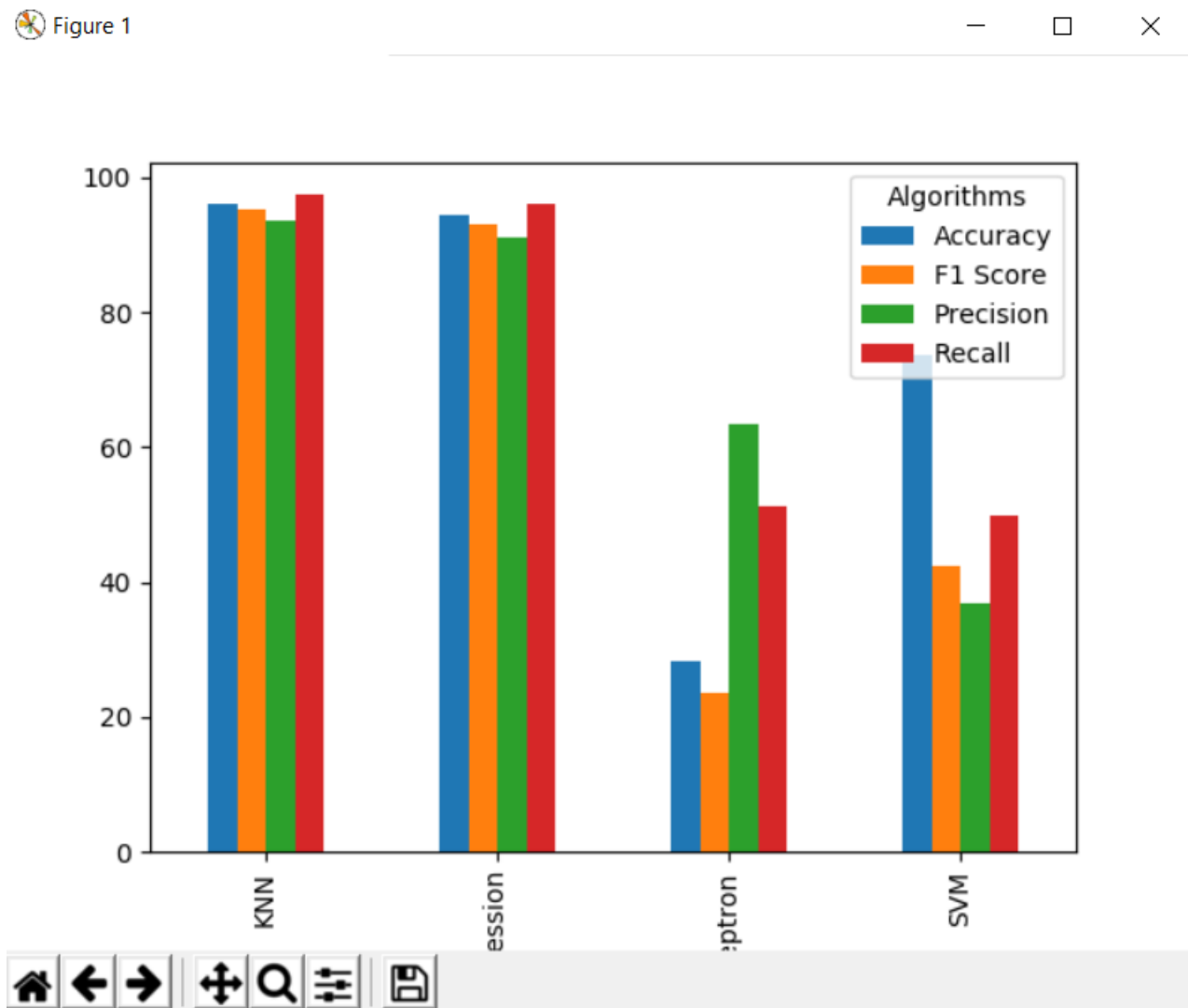
Logistic Regression Precision : 91.17647058823529

Logistic Regression Recall : 96.15384615384616

Logistic Regression FScore : 93.16129032258064

Logistic Regression Accuracy : 94.33962264150944

Performance Graph of All Algorithms:



9. CONCLUSION AND FUTURE SCOPE

It is found that the perceptron is less susceptible to system size and the k-NN approach has a higher sensitivity than the other methods. The performance of K-NN's is harmed by the imbalanced data problem. As a result, when compared to other algorithms, in small systems, k-NN may perform better, but in big systems, it may perform worse. SVM outperforms the other techniques in the large scale systems. We've presented a system for predicting malicious assaults that could occur in the future of smart grids. A probabilistic distribution is used by the system to predict if an irregular style of operation would cause smart grid connections to be interrupted. The results of simulations reveal that the proposed system is capable of forewarning damaging threats. The suggested approach may also detect other hostile threats and anomalies allowing the control center to swiftly instruct smart meters to respond to such irregularities. We must first establish a baseline for calculating mistakes in smart grid networks with background traffic to identify actual aberrant behaviors. These are the topics that will be investigated more in the future.

10. REFERENCES

1. R. Lu, X. Li, X. Lin, X. Liang, and X. Shen, "GRS: The green, reliability, and security of emerging machine to machine communications," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 28–35, Apr. 2011.
2. C. W. Gellings, "The Smart Grid: Enabling Energy Efficiency and Demand Response," published by CRC Press, Aug. 2009.
3. Z. M. Fadlullah, M. M. Fouda, N. Kato, A. Takeuchi, N. Iwasaki, and Y. Nozaki, "Towards intelligent machine-to-machine communications in smart grid," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 60–65, Apr. 2011.
4. M. M. Fouda, Z. M. Fadlullah, N. Kato, R. Lu, and X. Shen, "A lightweight message authentication scheme for smart grid communications," *IEEE Transactions on Smart Grid*, 2011, to appear.
5. M. Alizadeh, A. Scaglione, and Z. Wang, "On the impact of smartgrid metering infrastructure on load forecasting," in Invited article in Proc. 48th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois, Sept. 2010.
6. X. Wang, H. Wang, and L. Hou, "Electricity demand forecasting based on three-point gaussian quadrature and its application in smart grid," in Proc. 6th Int. Wireless Communications Networking and Mobile Computing (WiCOM) Conf., Chengdu, China, Sep. 2010, pp. 1–4.
7. O. Kramer, B. Satzger, and J. Laessig, "Power prediction in smart grids with evolutionary local kernel regression," in Proc. Hybrid Artificial Intelligence Systems (HAIS'10), San Sebastian, Spain, Jun. 2010.
8. H. K. Alfares and M. Nazeeruddin, "Electric load forecasting: literature survey and classification of methods," *International Journal of Systems Science*, vol. 33, no. 1, pp. 23–34, Dec. 2001.
9. T. I. Alecu, S. Voloshynovskiy, and T. Pun, "The gaussian transform of distributions: definition, computation and application," vol. 54, no. 8, pp. 2976–2985, Aug. 2006.
10. D. Niyato, P. Wang, Z. Han, and E. Hossain, "Impact of packet loss on power demand estimation and power supply cost in smart grid," in *IEEE Wireless Communications and Networking Conference (WCNC'11)*, Cancun, Quintana Roo, Mexico, Mar. 2011, pp. 2024–2029.