A FIELD PROJECT REPORT

on

“Mitigating False Data Injection Attacks Using Machine Learning Models”

Submitted

by

221FA04138	221FA04155
J Vani Akhila	M Manikanta
221FA04004	221FA04031
S Sai Lakshmi	N Phanindra Raja Mithra

Under the guidance of

Maridu Bhargavi

Assistant Professor

Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH Deemed to be UNIVERSITY

Vadlamudi, Guntur.

ANDHRA PRADESH, INDIA, PIN-522213.

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Field Project entitled "Mitigating False Data Injection Attacks Using Machine Learning Tools" that is being submitted by 221FA04138 (Jannavarapu Vani Akhila), 221FA04155 (Mondem Manikanta), 221FA04004 (Srigakolapu Sai Lakshmi),221FA04031(Nagulapati Phanindra Raja Mithra) for partial fulfilment of Field Project is a bonafide work carried out under the supervision of Ms. Maridu Bhargavi, M.Tech., Assistant Professor, Department of CSE.

Guide name& Signature	[illegible]	[illegible]
Assistant Professor, CSE	Dr. S. V. Phani Kumar	
	HOD,CSE	**Dr. K. V. Krishna Kishore**
		Dean, soCI

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that the Field Project entitled “Mitigating False Data Injection Attacks Using Machine Learning Tools” is being submitted by 221FA04138 (Jannavarapu Vani Akhila), 221FA04155 (Mondem Manikanta), 221FA04004 (Srigakolapu Sai Lakshmi),221FA04031(Nagulapati Phanindra Raja Mithra) in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of Ms. Maridu Bhargavi, M.Tech., Assistant Professor, Department of CSE.

By
221FA04138 (Jannavarapu Vani Akhila),
221FA04155 (Mondem Manikanta),
221FA04004 (Srigakolapu Sai Lakshmi),
221FA04031 (Nagulapati Phanindra Raja Mithra)

Date:

ABSTRACT

A vehicle that functions purely on electricity, as opposed to normal gasoline or diesel automobiles, is known as an electric vehicle (EV). Electric vehicles are becoming extremely popular. EVs come equipped with a built-in connection that allows for features like remote control via smart phone apps, over-the-air software upgrades, and navigation. Smart charging stations are available for electric vehicles (EVs) and are capable of utilizing energy consumption and cost data to improve charging times through communication with the car. However, there are cyber security problems nowadays everywhere. Monthly pricing is one option for charging at electric charging stations. False Data Injection Attacks are a frequent form of attack in which hackers alter data to their benefit. We are developing a machine learning algorithm to identify the site of the deceptive data injection in order to solve this issue. Random Forest Algorithms, RBF kernel, linear kernel, polynomial kernel, and Support Vector Machine were utilized for training and assessing the outcomes. Keywords: False Data Injection Attack (FDIA), Electric Vehicle Charging Stations, SVM with RBF Kernel, Machine Learning.

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER – 1

INTRODUCTION

1. Introduction

Cars and other vehicles that run on electricity rather than gasoline or diesel are known as electric vehicles, or EVs. They are growing in popularity since they may lessen pollutants and are healthier for the environment. Electric motors in EVs are powered by batteries, which allow the vehicle to move. In addition, EVs are often smoother and quieter to drive than cars with internal combustion engines. Because they have fewer moving components, they can be more dependable over time and require less maintenance. In addition, as EVs may be charged at the comfort of home, a normal electrical socket, or at the community charging stations they are simple to use on a regular schedule. But there are certain drawbacks to electric vehicles as well. One issue is that certain electric cars have a limited range, which means they can only go so far before needing to be recharged. Long excursions or locations with a lack of charging infrastructure may have this problem. Both bodily harm and financial damages may result from such attacks. The price of EVs, which might be more up front than that of conventional automobiles, is another factor to take into account. Nevertheless, over time, fuel and maintenance savings frequently outweigh this expense. In addition, costs are anticipated to decrease as EV adoption increases and technology progresses. Due to their many technological features, electric cars are susceptible to hackers.[1] Consider an EV charging station located in a gated neighborhood. To utilize this station for charging their EVs, users must pay a monthly charge. The charging station keeps track of data, such as the quantity of power consumed by each user and the accompanying payments. Unauthorized access to the charging station’s data system is obtained by an attacker. This may occur as a result of software flaws, weak passwords, or other security holes. The objective of the attacker is to use data manipulation to their advantage covertly. The charge data that is kept in the system is altered by the attacker. For instance, they can make it seem as though they consumed more power than they actually did by increasing the documented energy consumption for their own EV. Alternatively, they might limit the consumption data for other customers, lowering their prices. By inserting fraudulent data, the attacker can profit monetarily or disrupt the charging station’s operations. The False Data Injection Attack caused numerous implications, including income loss for the charging station owner owing to inaccurate invoicing. The station’s efficiency has been weakened, impacting overall operations. Alternatively, they might re duce usage data for other customers, lowering their fees. By inserting fraudulent data, the attacker can profit monetarily or disrupt the charging station’s operations. To address this issue, we proposed applying a machine learning approach called False Data Injection to identify the cyberattack. Our major objective is to detect cyber attacks with Machine Learning models. Our approach was implemented using the Electric Vehicle Charging Dataset from Kaggle. Techniques for machine learning, such support vector machines (SVM), can detect anomalous charging patterns in data. These algorithms use previous data to identify potentially suspicious transactions. Regular audits, secure authentication, and encryption are crucial for preventing such crimes. False Data Injection Attacks modify charge data for personal advantage, potentially resulting in financial losses and operational problems. Detecting and blocking such attacks is critical for the safety of EV charging stations.

1.1 Cyber Attacks

A Cyber attack refers to an attempt by cybercriminals, hackers, or other digital adversaries to access a computer network or system with the intention of altering, stealing, destroying, or exposing information. These attacks can target a wide range of victims, from individual users to enterprises and even governments. Let’s delve into some common types of cyber attacks:

1. **Malware:**
    - **Description:** Malware, short for malicious software, encompasses various subsets such as ransomware, trojans, spyware, viruses, worms, and more.
    - **Ransomware**: In a ransomware attack, the adversary encrypts a victim’s data and demands payment in exchange for a decryption key. These attacks often arrive via phishing emails, but unpatched vulnerabilities and policy misconfigurations can also be exploited.
    - **Fileless Malware:** Unlike traditional malware, fileless malware uses native, legitimate tools within a system to execute attacks. It doesn’t require installing any code on the target system, making it harder to detect.
2. **Denial-of-Service (DoS) Attacks:**
    - **Description:** These attacks overwhelm a system’s resources, rendering it unable to respond to legitimate service requests. The goal is to disrupt normal operations.
    - **Distributed Denial-of-Service (DDoS) Attacks:** Similar to DoS attacks, but orchestrated from multiple sources, making them more potent. Attackers flood the target with traffic, causing service disruptions.
3. **Phishing:**
    - **Description:** Phishing involves tricking users into revealing sensitive information (such as login credentials) by posing as a trustworthy entity via emails, messages, or fake websites.

4. **Spoofing:**
    - **Description:** Spoofing disguises the origin of communication. Examples include IP spoofing (faking an IP address) or email spoofing (sending emails that appear to come from a legitimate source).
5. **Identity-Based Attacks:**
    - **Description:** These attacks target user identities, aiming to steal personal information or compromise accounts. Examples include credential stuffing, brute force attacks, and man-in-the-middle attacks.
6. **Code Injection Attacks:**
    - **Description:** Attackers inject malicious code into a system, exploiting vulnerabilities. Examples include SQL injection and cross-site scripting(XSS) attacks.
7. **Supply Chain Attacks:**
    - **Description:** These attacks compromise software or hardware during the development or distribution process. Attackers exploit trust in the supply chain to introduce malicious components.
8. **Insider Threats:**
    - **Description:** Threats from within an organization, where employees or contractors misuse their access to cause harm. Examples include data theft or sabotage.
9. **IoT-Based Attacks:**
    - **Description:** Internet of Things (IoT) devices are vulnerable to attacks due to weak security. Attackers can compromise smart devices, leading to privacy breaches or network infiltration.

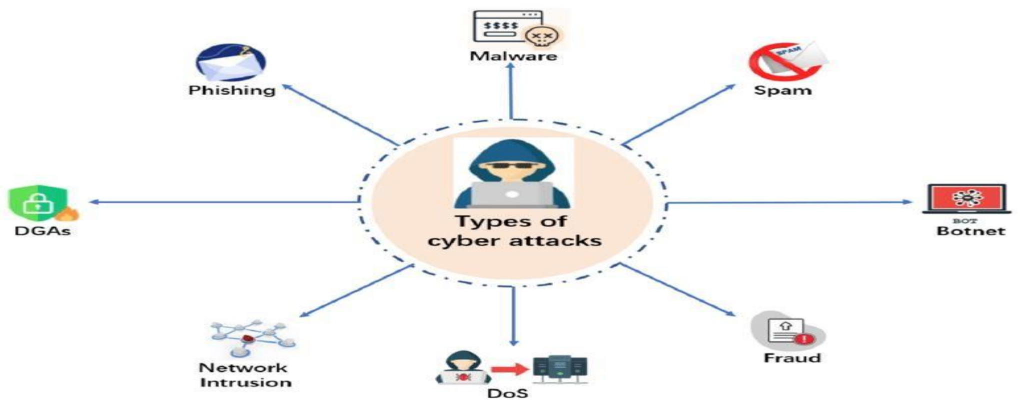


Figure 1.1 Cyber Attacks

Figure 1.1 describes the Cyber Attacks

1.2 Injection Attacks

An injection attack refers to a broad class of attack vectors where an attacker supplies untrusted input to a program. This input gets processed by an interpreter as part of acommand or query, which then alters the execution of that program. These attacks areparticularly dangerous for web applications and can lead to various security risks. Let’s exploresome common types of injection attacks:

1. **Code Injection:**
   - **Description:** In a code injection attack, the attacker injects application code written in the application language. This injected code may be used to execute operating system commands with the privileges of the user who is running the web application.
   - **Potential Impact:** In advanced cases, code injection can lead to full system compromise, allowing the attacker to take control of the entire web server.
2. **CRLF Injection:**
   - **Description:** The attacker injects an unexpected CRLF (Carriage Return and Line Feed) character sequence. This sequence is used to split an HTTP response header and write arbitrary contents to the response body. It can also be combined with Cross-site Scripting (XSS) attacks.
   - **Potential Impact:** CRLF injection can lead to account impersonation, defacement, or allow the attacker to run arbitrary JavaScript in the victim’s browser.
3. **Cross-site Scripting (XSS):**
   - **Description:** In an XSS attack, the attacker injects an arbitrary script (usually in JavaScript) into a legitimate website or web application. This script is then executed inside the victim’s browser.
   - **Potential Impact**: XSS can lead to information disclosure, session hijacking,and defacement of web pages.
4. **Email Header Injection:**
   - **Description:** Similar to CRLF injections, the attacker sends IMAP/SMTP commands to a mail server that is not directly available via a web application.
   - **Potential Impact**: Email header injection can be exploited for spam relay or other malicious purposes.

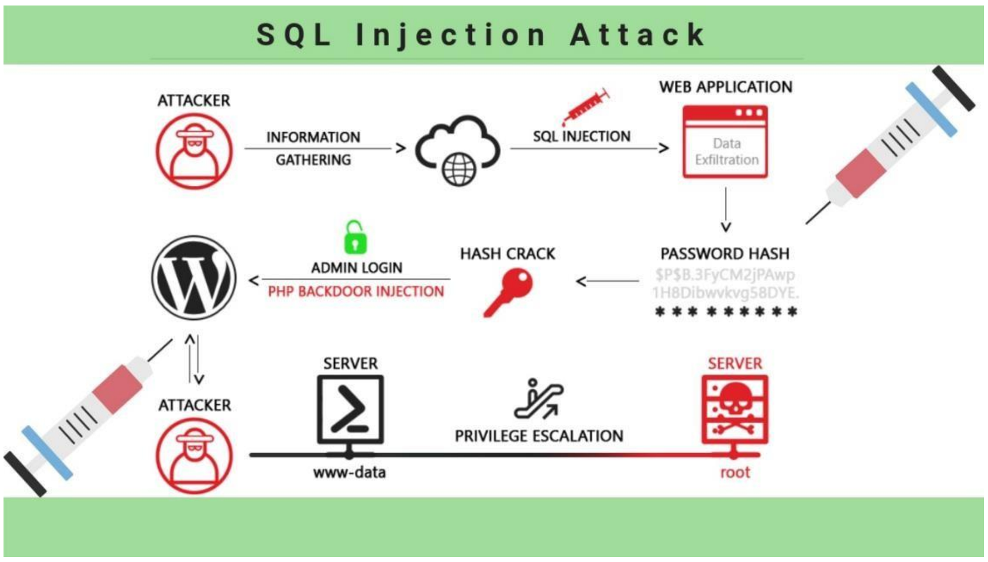


Figure 1.2 Injection Attacks

Figure 1.2 describes the Injection Attacks

1.3 False Data Injection Attack

A false data injection attack (FDIA) is a specific type of cyber attack originally introduced in the context of smart grids. Although the term might sound commonplace, its implications are critical. Let’s delve into the details:

1. Definition:
   - An FDIA occurs when an attacker manipulates sensor readings in a clever way that introduces undetected errors into calculations of state variables and values.
   - Specifically, the attacker compromises the integrity of data by injecting false information into the system.
2. Smart Grid Context:
   - In the smart grid domain, FDIA can have severe consequences. Imagine an attacker subtly altering electricity consumption data or tampering with voltage measurements.
   - These seemingly minor changes can lead to incorrect decisions by grid regulators, affecting the stability and security of the entire power grid.
3. Expanding Threat Landscape:
   - Beyond smart grids, cyber attackers are interested in similar attacks across various domains: healthcare, finance, defense, governance, and more.
   - As our interconnected world grows, FDIA becomes a top-priority issue.
4. Countermeasures:
   - Defending against FDIA requires robust countermeasures.

- Existing approaches include:
    1. Data validation: Rigorous validation of sensor data to detect anomalies.
    2. Intrusion detection systems: Monitoring for suspicious patterns.
    3. Secure communication protocols: Ensuring data integrity during transmission.
    4. Machine learning-based techniques: Leveraging ML models toidentify abnormal data patterns.

5. Challenges:
    - Benchmark datasets for evaluating FDIA detection techniques are scarce.
    - Researchers need to develop better evaluation metrics to assess the effectiveness of countermeasures.

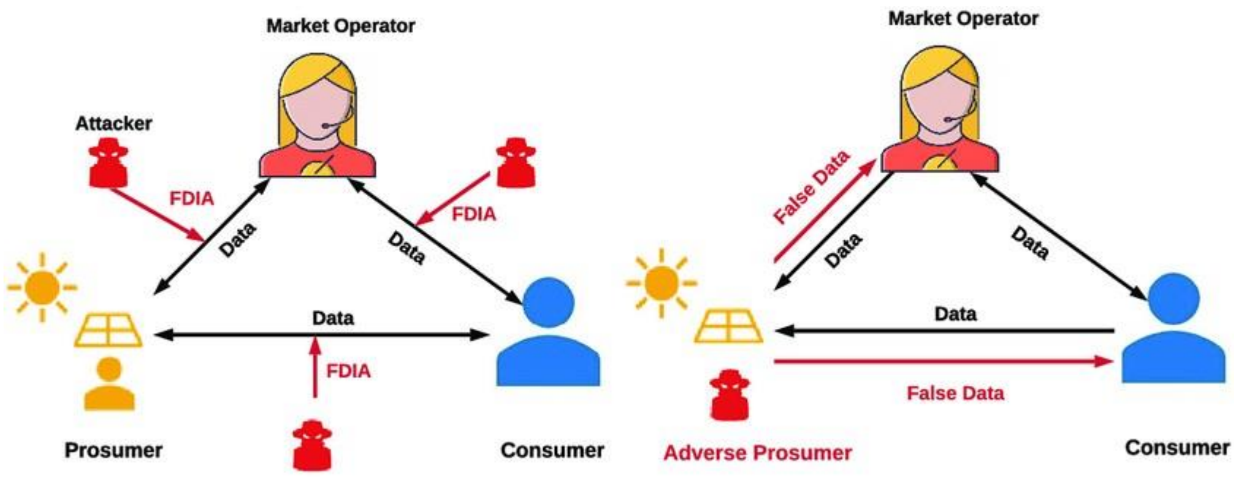


Figure 1.3 False Data Injection Attack

Figure 1.3 describes the False Data Injection Attack

1.4 Machine Learning

Machine Learning is a system of computer algorithms that can learn from example through self-improvement without being explicitly coded by a programmer. Machine learning is a partof artificial Intelligence which combines data with statistical tools to predict an output which can be used to make actionable insights.
Machine learning is also used for a variety of tasks like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

1.4.1 Machine Learning Working

Machine learning is a method of learning that resembles human learning, where machines learn from experience and predict outcomes based on similar examples. The core objective of machine learning is to learn and infer from data. Machines discover patterns using feature vectors, which are subsets of data used to solve problems. The learning stage simplifies the reality and transforms this into a model. The model is then used to make predictions on new data, without the need to update rules or train again. The life of machine learning programs is straightforward, involving defining a question, collecting data, visualizing it, training the algorithm, testing it, collecting feedback, refining the algorithm, and applying the model to make predictions

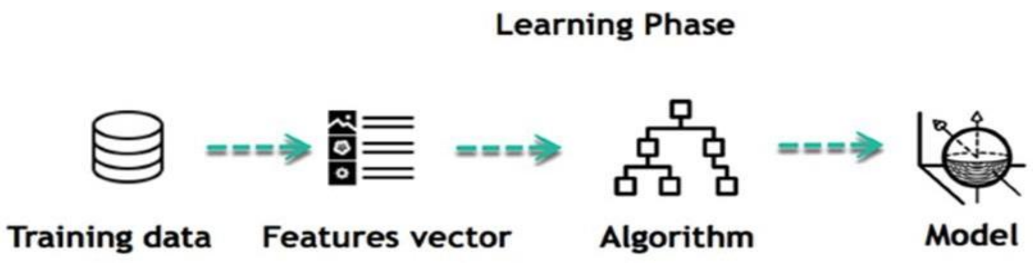


Figure 1.4.1 Learning Phase

Figure 1.4.1 describes the Learning Phase in Machine Learning

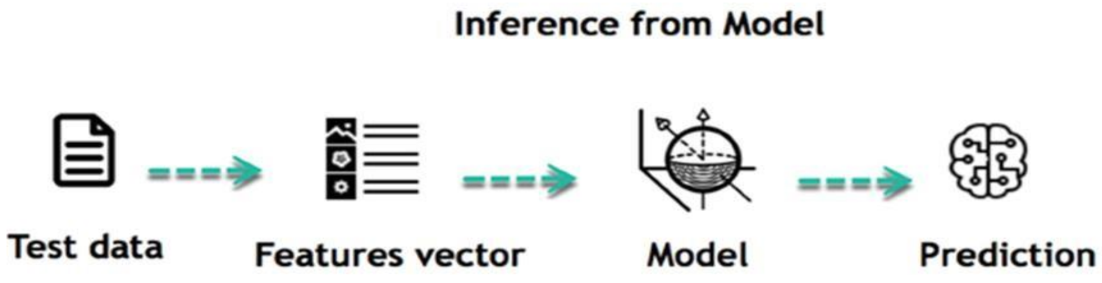


Figure 1.4.2 Inference from Model

Figure 1.4.2 describes the Inference from Model in Machine Learning

The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Use the model to make a prediction

1.5 Supervised Learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

- Classification task
- Regression task

1.5.1 Classification

To predict a customer's gender in a commercial, data on height, weight, job, salary, etc. is collected. The classifier's objective is to assign a probability of being male or female based on this information. Once the model learns to distinguish genders, new data can be used for predictions. For example, if the classifier predicts a 70% chance of male, it means there's a 70% certainty. This machine learning scenario involves two classes (male and female), but classifiers can predict multiple classes like objects such as glass, table, or shoes, where each object represents a distinct class.

1.5.2 Regression

A regression task involves forecasting stock values based on features like equity, previous performance, and macroeconomics index, with a trained system aiming for the lowest possible error

1.6 Unsupervised learning

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

1.7 Challenges and Limitations of Machine Learning

The primary challenge of machine learning is the lack of data or the diversity in the dataset. A machine cannot learn if there is no data available. Besides, a dataset with a lack of diversity gives the machine a hard time. A machine needs to have heterogeneity to learn meaningful insight. It is rare that an algorithm can extract information when there are no or few variations. It is recommended to have at least 20 observations per group to help the machine learn. This constraint leads to poor evaluation and prediction.

1.7 Support Vector Machines

Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and outlier detection tasks. It can be applied to various tasks like text classification, image classification, spam detection, gene expression analysis, and anomaly detection. SVMs are efficient in managing high-dimensional data and nonlinear relationships by finding the maximum separating hyperplane between different classes. In classification and regression problems, SVM aims to find the optimal hyperplane in an N-dimensional space to separate data points, ensuring maximum margin between closest points of different classes. The dimension of the hyperplane depends on the number of features, with a line for two features and a 2-D plane for three features.

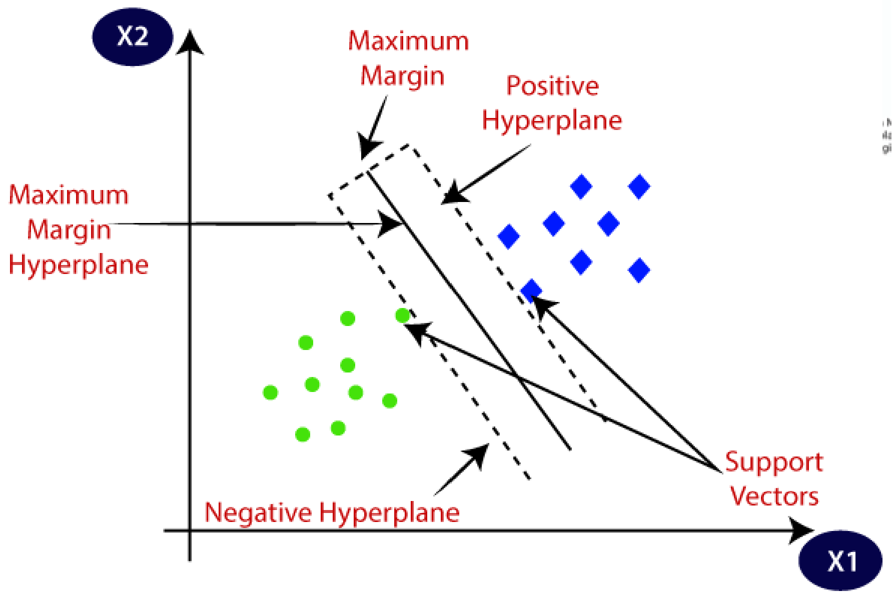


Figure 1.8 Support Vector Machine

Figure 1.8 describes the Support Vector Machine

1.9 Kernel Methods

Kernel methods are a set of techniques used in machine learning for classification, regression, and prediction. They involve functions called kernels that measure the similarity between data points in a high-dimensional space. Kernel methods convert input data into this space to differentiate between classes or make predictions. Support Vector Machines (SVMs) utilize kernel methods to find the best hyperplane for separating groups by mapping data into a higher-dimensional space. The kernel function, such as the Gaussian or radial basis function (RBF) kernel, determines the decision boundary between classes. Other types of kernels, like polynomial and sigmoid kernels, can also be used in SVMs depending on the data characteristics. Kernel methods are preferred in machine learning due to their adaptability, computational efficiency, and ability to handle complex data structures like strings and graphs. They are resilient to noise, outliers, and can capture nonlinear relationships in data. Overall, kernel methods in SVMs offer a powerful solution for classification and regression tasks.

1.9.1 Linear Kernel

A linear kernel is a basic kernel function used in machine learning, particularly in SVMs. It calculates the dot product between input vectors in the original feature space.

The linear kernel can be expressed as K(x, y) = x . y

where x and y are input feature vectors. When applied in an SVM, a linear hyperplane is used as the decision boundary to separate classes in the feature space. This approach is suitable for data already separable by a linear boundary or when dealing with high-dimensional data to prevent overfitting.

1.9.2 Polynomial Kernel

The polynomial kernel in SVM incorporates input vectors x,y, constant term c, and polynomial degree d to create a decision boundary capturing complex correlations. This nonlinear hyperplane can detect both linear and nonlinear correlations, with the degree of nonlinearity determined by the polynomial degree. Choosing the appropriate degree is crucial to prevent overfitting or underrepresentation of data relationships. The polynomial kernel effectively transforms input data into a higher-dimensional feature space to better capture nonlinear correlations among input characteristics.

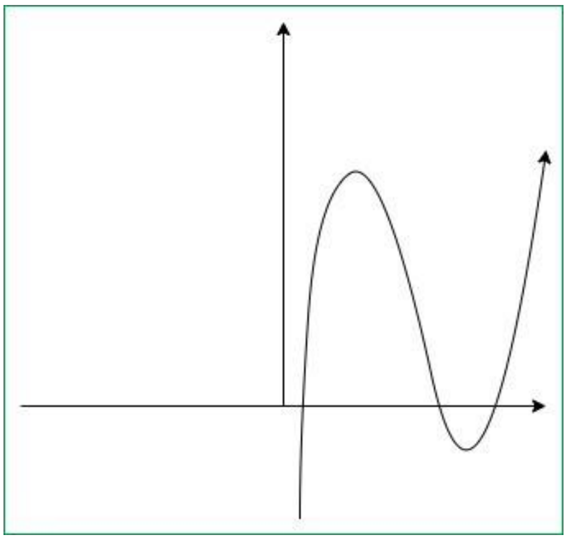


Figure 1.9.2 Polynomial Kernel

1.9.3 Gaussian Kernel Radial Basis Function (RBF) Kernel

The Gaussian kernel, or radial basis function (RBF) kernel, is commonly used in machine learning, especially in SVMs. It is a nonlinear function that maps data to a higher-dimensional space with a Gaussian function.

The Gaussian kernel can be defined as:K(x, y) = exp(-gamma * ||x - y||^2

Where x and y are the input feature vectors, gamma is a parameter that controls the width of the Gaussian function, and ||x - y||^2 is the squared Euclidean distance between the input vectors. The Gaussian kernel in SVM creates a nonlinear decision boundary capturing complex relationships between input features. Gamma parameter controls the width of the Gaussian function and the degree of nonlinearity. It eliminates the need for explicit feature engineering.

However, the choice of the gamma parameter can be challenging, as a smaller value may resultin under fitting, while a larger value may result in over fitting.

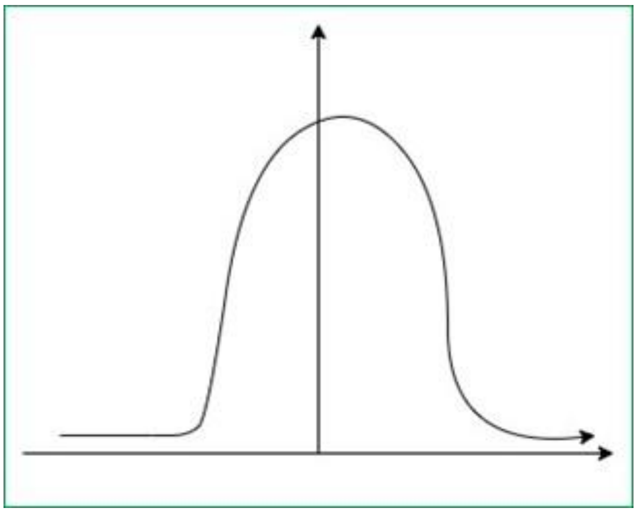


Figure 1.9.3 Gaussian RBF Kernel

Figure 1.9.3 shows the Gaussian RBF Kernel

1.10 Random Forest Classifier

Machine learning, a combination of computer science and statistics, has made great strides, with the Random Forest algorithm as a standout. Random Forest is a group of decision trees that collaborate to produce a single output, developed by Leo Breiman in 2001. This algorithm is a key tool for machine learning enthusiasts, as it creates multiple decision trees during training, each using random subsets of data and features. This randomness helps prevent overfitting and improves predictive performance. Random Forest is effective in handling complex data, reducing overfitting, and providing accurate forecasts for classification and regression tasks. Through collective decision-making by multiple trees, it generates stable and precise results.

CHAPTER – 2

LITERATURE SURVEY

LITERATURE SURVEY

False Data Injection Attacks (FDIAs) pose a serious threat to the integrity of cyber-physical systems by altering sensor or measurement data to disrupt operations, especially in critical infrastructures like smart grids and industrial control systems. Recent research has focused on leveraging machine learning (ML) models to detect and mitigate these attacks. Supervised learning approaches, such as support vector machines (SVM), random forests, and neural networks, have shown promise in identifying known attack patterns, while unsupervised methods like clustering, anomaly detection, and autoencoders help in recognizing previously unseen or evolving threats.

No	Author(s)	Model/Approach	Accuracy/Results	Limitation
1	Hongyang Li Department of Automation, Tsinghua University, Beijing, China ; Xiao He; Yufeng Zhang; Wenyuan Guan	Particle Filter algorithm Sliding-Time Window Technique. Resampling Algorithms.	99.901%.	Computational complexity Degradation
2	Elisabeth Drayer; Tirza Routtenberg	1) Random Forest 2) Convolutional Neural Networks (CNNs) 3) K- means clustering 4) SVM	0.99	1) Random Forest 2)Convolutional Neural Networks (CNNs) 3) K- means clustering 4) SVM
3	Abdelrahman Ayad; Hany E. Z. Farag; Amr Youssef; Ehab F. El-Saadany	MapReduce Algorithm, Which is commonly used for large-scale data processing.	99.901%	Iterative Algorithms, Communication overhead, Limited expressiveness.
4	D. Said, M.	Support Vector	91.29%	Limited by data

	Elloumi and L. Khokhi	Machine (SVM)		separability and requires careful tuning of parameters to avoid overfitting in noisy environments.
5	Gaoqi Liang ; Junhua Zhao; Fengji Luo; Steven R. Weller; Zhao Yang Dong	Various FDIA construction methods, including state estimation and residual tests.	Identifying all attack instances with minimal false positives and a slight delay in detection.	Limited focus on AC model-based FDIAs; primarily based on DC state estimation models, which are less realistic for practical systems.
6	Yuancheng Li; Yuanyuan Wang	Kernel Independent Component Analysis (KICA), Lagrangian function for attack vector formation.	when the information incompleteness is less than 64%. It also can successfully pass the DBN based detector 2 when the information incompleteness is less than 38%.	Limited by the amount of data and topology information available; effectiveness decreases with higher incomplete rates.
7	A. Kumar, N. Saxena and B. J. Choi	Machine learning techniques such as Naive Bayes, Random Forest, Decision Tree, Support Vector Machine (SVM), k-Nearest Neighbors (kNN), Logistic Regression, AdaBoost, and Gradient Boosting were used.	92%	Overfitting Time Complexity
8	MD Jainul Abudin, Surmila Thokchom	Support Vector Machine (SVM) Decision Tree Logistic Regression Auto encoder	0.99	Isolation Forest and Feed-Forward Neural Network (FFNN) Computation Overhead
9	Md. Ashfaqur Rahman; Hamed Mohsenian-Rad	Attack models for FDIAs, including perfect and imperfect attack scenarios	Highlighting the complexity and potential vulnerabilities of these systems.	Nonlinear attacks require online data collection, which can be challenging compared to the offline.
10	Kjonath Kwizera ; Liu Zhaohui.	BERT (Bidirectional Encoder Representations from Transformers) is a pre-	Cyber-attack detection using data mining techniques to enhance preparedness against	BERT is computationally expensive during training and

		trained language model that captures contextual information from both left and right context in a sentence.	future cyber threats.	inference. It has a fixed context window, which may limit capturing very long ranges dependencies.
11	Xiao Lu; Jiangping Jing; Yi Wu.	Multi-label Decision Tree (ML-DT): This ensemble model classifies the state of each meter to detect the exact location of FDIA without needing power topology information	Accurately detect the location of false data injection attacks in smart grids without needing power topology information.	1)Noise Sensitivity 2)Complexity
12	M. Esmalifalak et al. (2017)	Support Vector Machine (SVM)	Achieved 92.5% detection accuracy in identifying FDIA in smart grids	Limited scalability in large networks with dynamic conditions
13	A. Rahman et al. (2019)	Random Forest Classifier	Achieved over 90% accuracy in IoT-based industrial networks	High computational complexity with real-time detection systems
14	L. Xie et al. (2018)	Deep Learning (CNN-LSTM hybrid)	95% accuracy in detecting stealthy FDIAs in smart grids	Vulnerable to adversarial examples when the system is under attack
15	J. Kim and S. Choi (2020)	Autoencoder-based anomaly detection	Achieved 93% anomaly detection rate in power systems	Difficulty in detecting low-magnitude attacks
16	D. Wang et al. (2019)	Reinforcement Learning for adaptive FDIA detection	89% accuracy in adaptive attack mitigation	Time-consuming training and lack of scalability to larger systems
17	H. He et al. (2021)	Federated learning combined with anomaly detection	91% detection accuracy across distributed power grids	High communication overhead in distributed environments

18	P. Zhu et al. (2020)	Hybrid Deep Learning (LSTM + GNN)	Achieved 94% detection accuracy in detecting complex attacks	Susceptible to training data noise
19	X. Lu et al. (2019)	Graph-based anomaly detection (GNN)	88% attack detection in large-scale networks	High computational cost when applied to dynamic grid systems
20	Y. Hu et al. (2021)	LSTM-based time-series prediction model	92% accuracy in smart grid attack detection	Model underperforms with noisy sensor data
21	J. Zhang et al. (2020)	PCA-based dimensionality reduction with SVM	Achieved 90% detection accuracy in energy management systems	Inability to handle complex, stealthy attacks
22	C. Cheng et al. (2020)	K-means clustering with anomaly detection	87% detection accuracy for FDIA in industrial IoT	Struggles with detecting sophisticated multi-point attacks
23	R. Li et al. (2021)	Ensemble learning combining decision trees and boosting	93% detection accuracy in smart grids	Increased training time with large datasets
24	N. Yu et al. (2019)	CNN-based image classification for grid state estimation	89% accuracy in detecting FDIA in grid images	Limited applicability to non-visual data streams
25	L. Sun et al. (2020)	SVM with kernel trick for time-series analysis	90% detection rate for FDIAs in SCADA systems	High computational demand for real-time analysis

2.1 Exisiting System

We have a system where electric vehicles (CEVs) can participate in an open electricity market in a parking lot. The parking lot is equipped with a block chain server, and we're using a consortium blockchain based on Ethereum. CEVs can connect to the system as buyers or sellers. We're using a predictive bidding approach (PBA) based on stochastic bids, which are processed by a smart contract. It is decentralized approach and results regulatory challenges and during the detection of FDIA also results less accuracy when they use linear SVM,KNN,CNN Algorithms.

2.1 Limitations of Existing System

- Complexity and Standardization: Implementing the decentralized way often involvesthe high level of complexity especially in terms of energy exchange. if there aren't clear standards for how they communicate or exchange energy, it could create problems in making them all work together easily.

- Regulatory Challenges: The rules and laws we have now might not be fully ready for the new way of sharing energy directly between electric vehicles. As this cool technology grows, the government rules need to change to make sure everything is fair, people are protected, and the power grid stays stable. But, figuring out how to fit this new idea into the old rules can be hard and take a lot of time.
- Scalability Issues: Scaling a decentralized peer-to-peer energy exchange system to a large number of users and vehicles can pose challenges. As the number of participants increases, the system needs to handle a higher volume of transactions and communication, which may lead to latency issues and increased demands on the underlying infrastructure.
- Market Dynamics: The peer-to-peer energy exchange market could face challenges related to pricing mechanisms and economic incentives. Determining fair and transparent pricing models that incentivize participants while ensuring affordability for consumers can be complex and may require continuous adjustments.
- Limited Adoption: The success of decentralized peer-to-peer energy exchange relies on widespread adoption by EV users and stakeholders. Resistance to change, lack of awareness, or skepticism about the benefits may slow down the adoption rate, limiting the effectiveness of the system.

CHAPTER – 3

PROPOSED METHOD

3.1 Proposed System

The proposed system describes about the False Data Injection Attack (FDIA) in electric vehicles where the energy is transferred through the charging station. In some communities there are some charging stations where the amount of selling price is paid to the incharge of charging station.

There might be chances of FDIA by the attackers on the data they have. To detect this type of FDIA attack we proposed a Machine learning model using Support vector machine Algorithm. We suppose that an attacker injects a false data vector ’a’ into our training data which will the network and deliver a false results.

In this context, FDIA detection is treated as a supervised binary classification problem. Based on the research work done in the SVM is more efficient than CNN and KNN in anomaly detection with 91.29 % of precision. Moreover, the SVM is a popular practice for training a decision boundary that divides data into several classes.The SVM is based on a hyperplane that maximizes the separation margin between two classes.

The training points which are close to the limit defining this division margin are called support vectors. So, in this case we are seeking a hyper-plane that separates attacked (1) and secure data (0) in a N dimensional feature

3.2 Advantages of Proposed System

1. Enhanced Security:Detecting and preventing false data injection attacks ensures the integrity of data used for voltage regulation in electric charging stations.By identifying and mitigating these attacks, the overall security of the charging infrastructure is strengthened.
2. Accurate Voltage Regulation:False data injection attacks can lead to incorrect voltage regulation capacity estimations.Detecting such attacks allows for precise estimation of the capacity that electric vehicle charging stations can provide for voltage regulation.Accurate estimations help maintain grid stability and prevent overestimation or underestimation of available resources.
3. Cost Optimization:Proper estimation of voltage regulation capacity avoids wastage of surplus resources.Detecting false data ensures that the right amount of capacity is allocated,

minimizing operational costs.Overestimation can lead to unnecessary expenses, while underestimation affects network stability.

4. Network Stability:Reliable voltage regulation is crucial for maintaining a stable distribution network.Detecting false data injection attacks prevents mispredictions that could compromise network stability.Ensuring accurate capacity estimations contributes to a robust and resilient grid.
5. Efficient EV Aggregation:Electric vehicle aggregators play a significant role in voltage regulation.Detecting false data ensures that EV aggregators provide the expected capacity, optimizing their participation in the VR market.Efficient aggregation benefits both the grid and EV owners.

3.3 Methodologies

Dataset:

In the first module, we developed the system to get the input dataset. Data collection process is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get, the better our model will perform. There are several techniques to collect the data, like web scraping, manual interventions. Our dataset is placed in the project and it's located in the model folder. The dataset is referred from the popular standard dataset repository kaggle where all the researchers refer it. The dataset consists of 3395 instances. The following is the URL for the dataset referred from Kaggle.

Link: **https://www.kaggle.com/datasets/michaelbryantds/electric-vehicle-charging-dataset**

Features of Dataset:

Session ID: A unique identifier for each charging session that can be used for tracking purposes.

Kwh Total: The total amount of energy consumed by the EV during the charging session, which isa crucial data point for analyzing energy consumption patterns.

Dollar: The cost of the charging session, which can be used to analyze the economic feasibility ofP2P energy trading.

Created: The time and date when the charging session started, which can be used to analyze temporal patterns in EV charging behavior.

Ended: The time and date when the charging session ended, which can also be used to analyze temporal patterns in EV charging behavior.

Starting time: The starting time of the charging session in hours and minutes, which can be used to further analyze temporal patterns in EV charging behavior.

Ending time: The ending time of the charging session in hours and minutes, which can also be used to further analyze temporal patterns in EV charging behavior.

Charging time: The duration of the charging session in minutes, which can be used to calculate the charging speed and to analyze the duration of EV charging sessions.

Weekday: The day of the week when the charging session took place, which can be used to analyze weekly patterns in EV charging behavior.

User ID: The unique identifier for the EV driver using the charging station, which can be used to track individual behavior and to analyze the energy consumption behavior of different user groups.

Station ID: The unique identifier for the charging station, which can be used to analyze the energy consumption behavior of different charging stations.

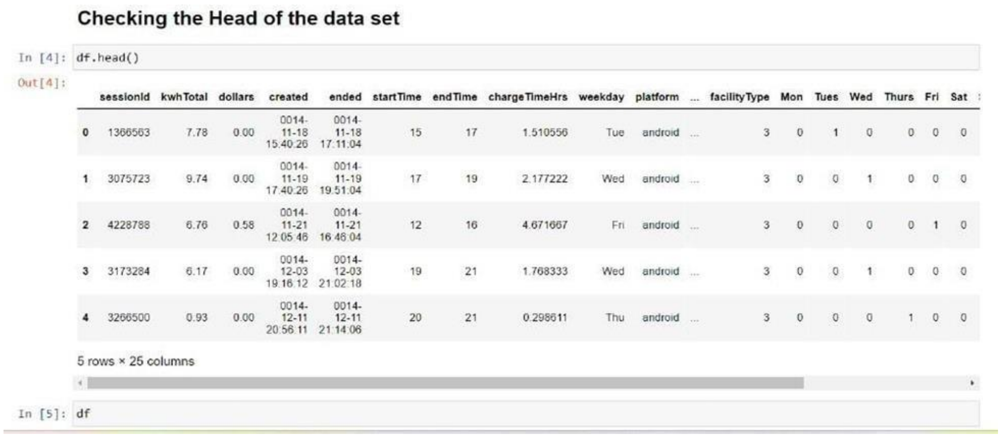
Location ID: The unique identifier for the location of the charging station, which can be used to analyze the energy consumption behavior of different locations and to evaluate the feasibility of P2P energy trading based on the availability of renewable energy sources.

False Data Injection: This is the key point where cyber attack is occuring and if the value is 0 then it does not occur and 1 cyber attack was successful.

Independent Variables: kwhTotal,dollars,chargeTimeHrs,Distance,Week days,manager vehicle

Dependent Variables:

False Data Injection



Checking the Head of the data set

```
In [4]: df.head()
Out[4]:
```

	sessionId	kwhTotal	dollars	created	ended	startTime	endTime	chargeTimeHrs	weekday	platform	...	facilityType	Mon	Tues	Wed	Thurs	Fri	Sat
0	1366563	7.78	0.00	0014-11-18 15:40:26	0014-11-18 17:11:04	15	17	1.510556	Tue	android	...	3	0	1	0	0	0	0
1	3075723	9.74	0.00	0014-11-19 17:40:26	0014-11-19 19:51:04	17	19	2.177222	Wed	android	...	3	0	0	1	0	0	0
2	4228788	6.76	0.58	0014-11-21 12:05:46	0014-11-21 16:46:04	12	16	4.671667	Fri	android	...	3	0	0	0	0	1	0
3	3173284	6.17	0.00	0014-12-03 19:16:12	0014-12-03 21:02:18	19	21	1.768333	Wed	android	...	3	0	0	1	0	0	0
4	3266500	0.93	0.00	0014-12-11 20:56:11	0014-12-11 21:14:06	20	21	0.298611	Thu	android	...	3	0	0	0	1	0	0

5 rows × 25 columns

```
In [5]: df
```

Figure 3.3 Dataset

Figure 3.3 describes the Dataset

Importing the necessary libraries:

We will be using Python language for this. First we will import the necessary libraries such as svm for building the main model, sklearn for splitting the training and test data, PIL for converting the images into array of numbers and other libraries such as pandas, numpy, matplotlib,pickle,sys, and seaborn.

Splitting the dataset:

In this module, the image dataset will be divided into training and testing sets. Split the dataset into Train and Test. 80% train data and 20% test data. This will be done to train the model on a subset of the data, validate the model's performance, and test the model on unseen data to evaluate its accuracy. Split the dataset into train and test. 80% train data and 20% test data

Building the model:

Apply SVM Model

Changing the kernel of SVM can impact the model's performance. The choice of kernel depends on the data and problem being solved. The provided code snippet uses the 'rbf' kernel, which is often used for non-linearly separable data. Other kernels such as linear, polynomial, and sigmoid can also be used based on the specific problem.

It's generally a good idea to experiment with different kernels and compare their performance using metrics such as accuracy, precision, recall, and F1-score. This can help you choose the best kernel for your particular problem.

SVM Model with RBF Kernel: Confusion Matrix & Accuracy Score

This code is an extension of the previous code that trains an SVM classifier on the dataset and evaluates its performance using cross-validation. It adds a confusion matrix visualization to further analyze the classifier's performance.

After selecting the features for SVM and splitting the data into features (X) and target (y), a SVM classifier with an RBF kernel is created. Then, 5-fold cross-validation is used to evaluate the classifier's performance, and the mean accuracy and standard deviation of the cross-validation scores are printed.

Next, the SVM classifier is trained on the entire dataset, and its predictions on the target variable (y_pred) are obtained. Finally, the confusion matrix is calculated using the true labels (y) and predicted labels (y_pred), and a heatmap visualization of the matrix is plotted using the seaborn library.

The confusion matrix helps to visualize the performance of the classifier by showing the number of true positives, true negatives, false positives, and false negatives. The heatmap visualization makes it easier to interpret the matrix by highlighting the values using different colors.

CHAPTER – 4

SYSTEM DESIGN

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

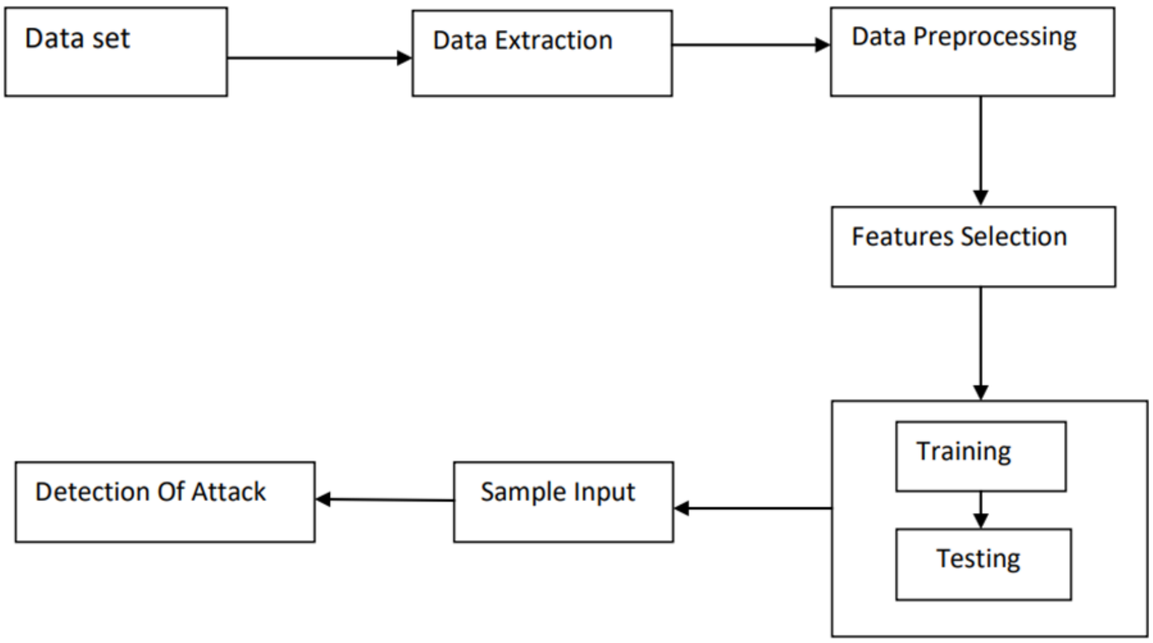


Figure 4.1 System Architecture

The system architecture can be described using various views or perspectives, including the following:

1. Functional View: This view describes the functions of the system and how they interact with each other to achieve the system's goals.

2. Structural View: This view describes the physical components and modules of the system and how they are organized and connected.

3. Behavioral View: This view describes how the system responds to different inputs and how it Interacts with external systems and users.

4. Data View: This view describes the data elements and data flows within the system, including the storage, retrieval, and manipulation of data.

For every organization to protect the security of its systems and data, security threat detection is a crucial duty. The effectiveness and precision of security attack detection can be greatly enhancedby the use of artificial intelligence (AI) and machine learning (ML.) approaches. Here is a suggested approach for identifying security attacks using Al and ML techniques

1. Data gathering Gather all the necessary information, including network traffic, system logs, and security events, and store it in a central database.

2. Data pre-processing: Pre-process the data to remove noise, inconsistencies, and unnecessary information that could compromise the detection system's accuracy.

3. Feature Extraction: Take pertinent features out of the preprocessed data. For instance, the source and destination IP addresses, ports, and other factors can be used to evaluate network traffic

4. Model Development: Using the features gathered, create an AI/ML. model that can recognize security assaults. To create the model, a number of methods can be utilized, including anomaly detection, clustering, and classification.

5. Model Training: Using the pre-processed and feature-extracted data, train the model.

6. Model Testing: Model testing involves putting the trained model to the test using a set of data that it has never seen before. Various performance indicators, including precision, recall, and F1 score, can be used to assess the model's correctness.

7. Model Deployment: Implementing the trained model in the production environment and integrating it with the security infrastructure will enable you to continuously monitor for and identify security assaults.

8. Model Maintenance: Maintaining the model involves tracking its performance over time and updating it with new information.

4.2 DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

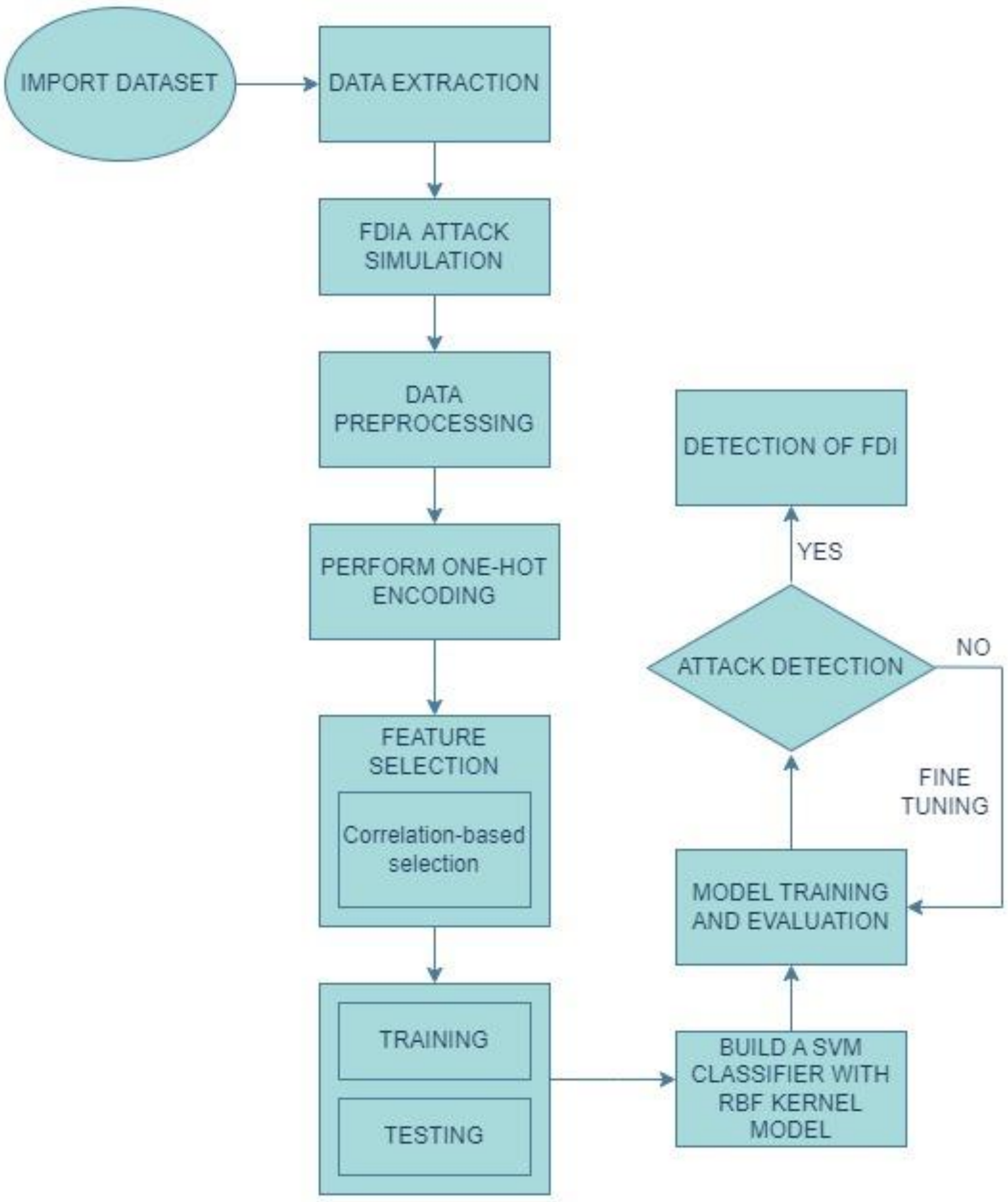


Figure 4.2 Data Flow Diagram

4.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language (UML) is a standard language used for specifying, visualizing, constructing, and documenting software and business models. It is based on best engineering practices for modeling large and complex systems. UML is crucial in developing object-oriented software, using graphical notations to express software project designs.

4.3.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case overview of the functionality provided by a system in terms of actors, their goals (represented asuse cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

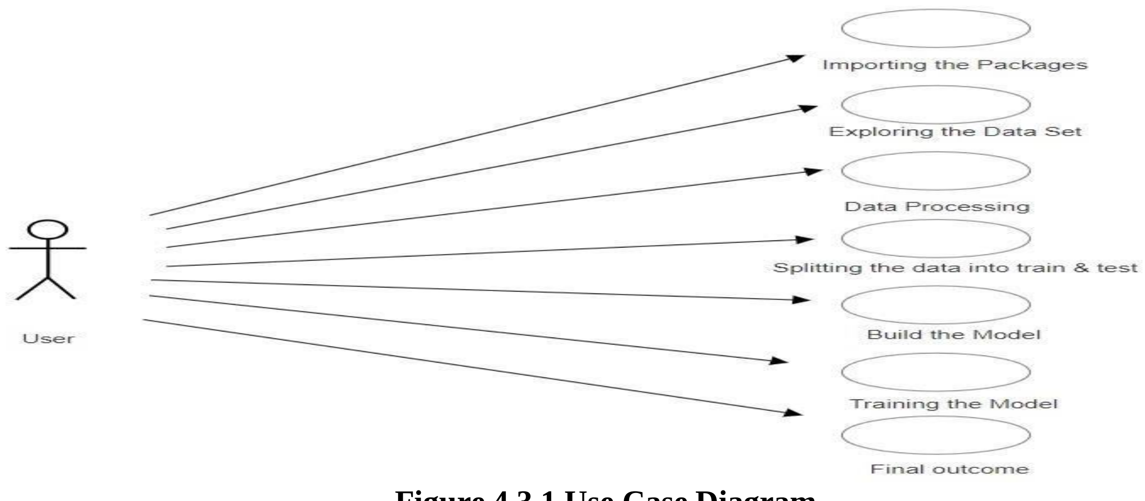


Figure 4.3.1 Use Case Diagram

4.3.2 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

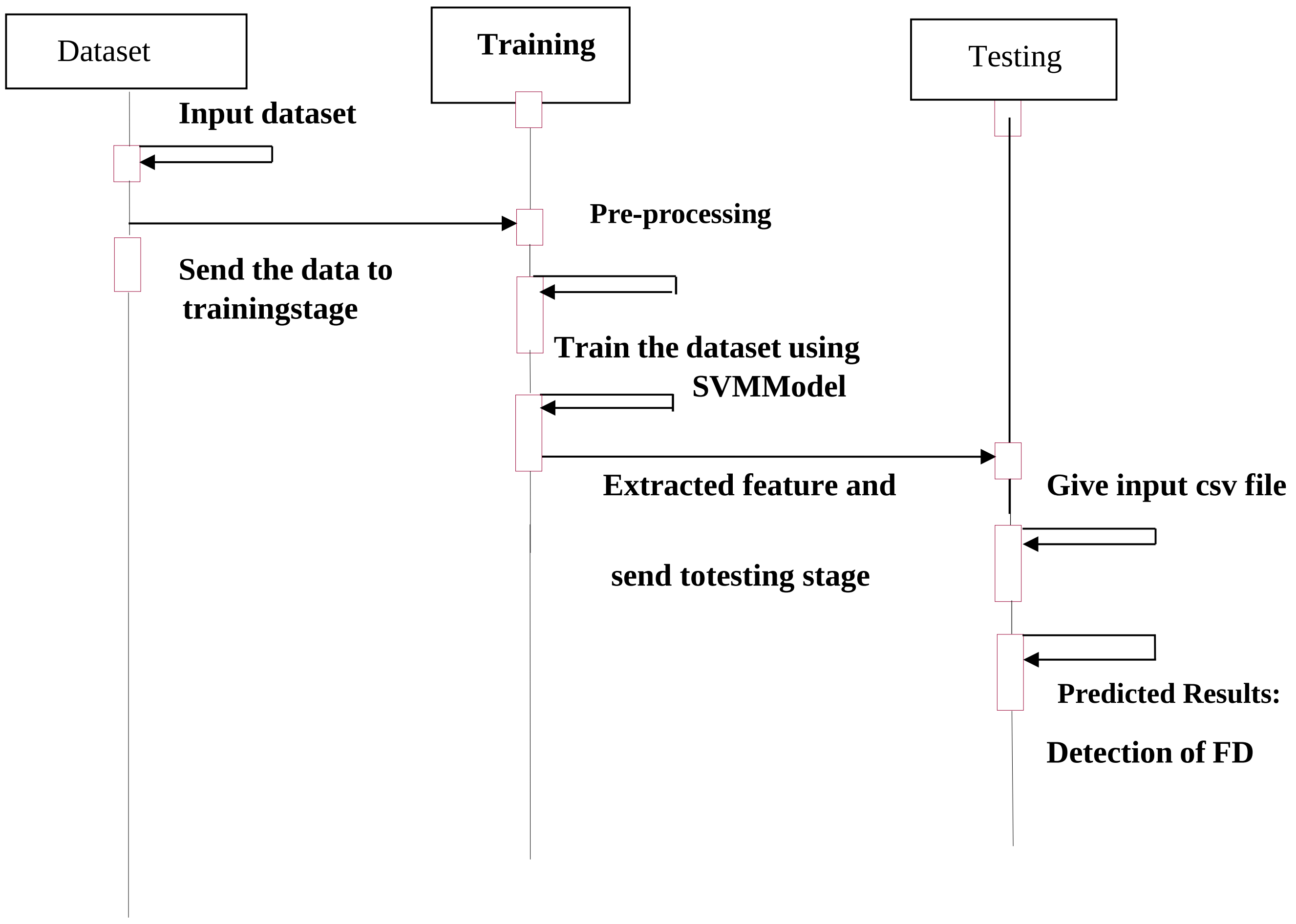


Figure 4.3.2 Sequence Diagram

4.3.3 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

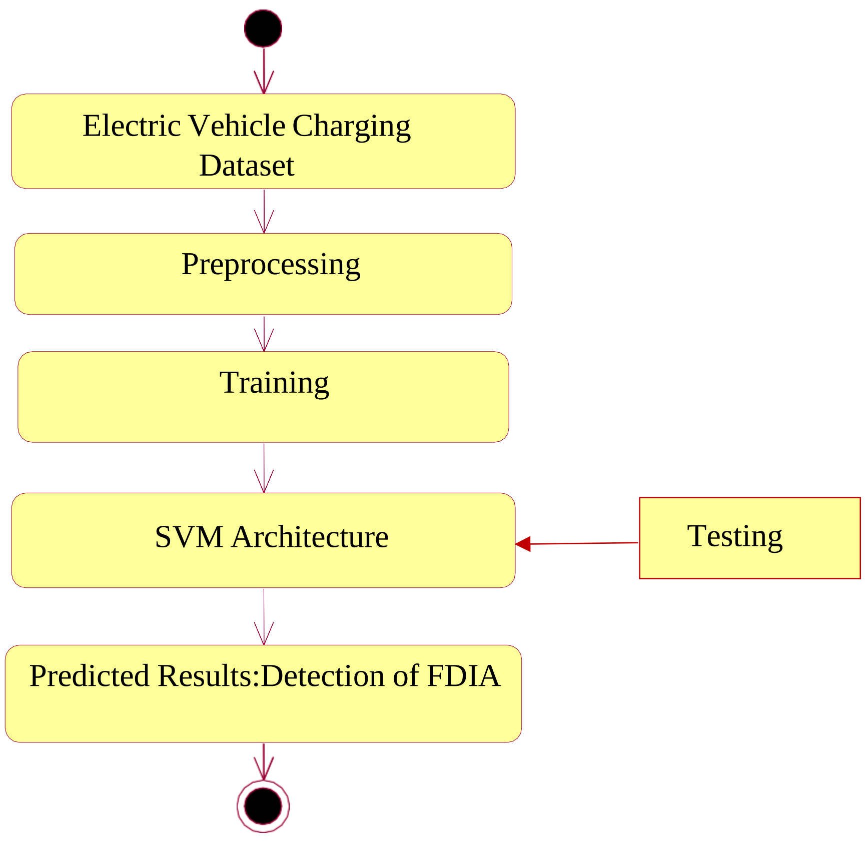

Electric Vehicle Charging Dataset
Preprocessing
Training
SVM Architecture
Testing
Predicted Results:Detection of FDIA

Figure 4.3.3 Activity Diagram

4.3.4 Deployment Diagram

A deployment diagram in Unified Modeling Language (UML) is a type of diagram that models the physical deployment of software components onto hardware nodes in a system. It shows how software artifacts (such as components, executables, libraries) are deployed across hardware nodes (such as servers, computers, devices) and the connections between them.

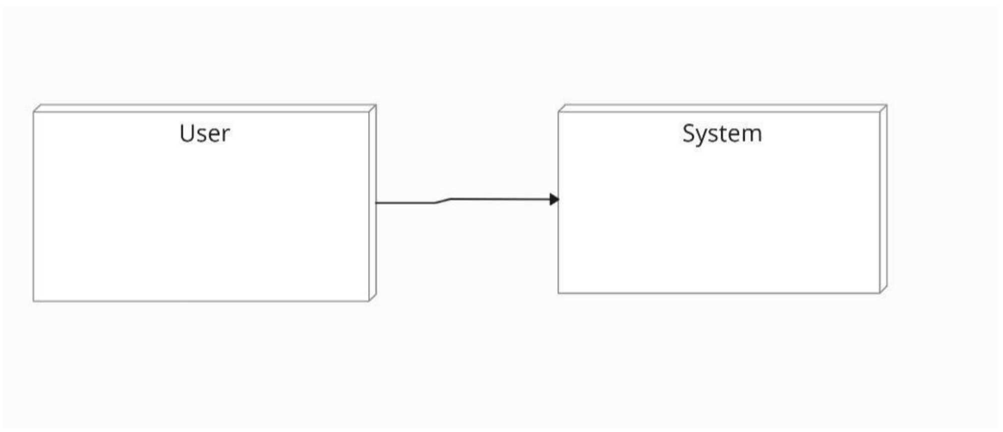


Figure 4.3.4 Deployment Diagram

CHAPTER – 5

DEPLOYEMENT

DEPLOYEMENT

5.1 Python

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- Web development (server-side),
- software development,
- mathematics,
- system scripting.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way. Good to know
- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

5.2 Modules

Pandas

Pandas is a Python module for data manipulation and analysis, introducing Series and DataFrame data structures. It simplifies data handling tasks, handles missing data, and integrates with various formats. It also enables statistical analysis and data visualization through libraries like NumPy and Matplotlib. Pandas is widely used in data science, machine learning, and research.

Numpy

NumPy is a Python library that simplifies numerical computations by introducing array objects and functions. Its core component, ndarray, allows efficient storage and manipulation of large datasets. NumPy offers mathematical functions like linear algebra, Fourier transforms, and random number generation. It enhances performance with optimized C algorithms, making it suitable for scientific and engineering applications.

Scikit-learn

Scikit-learn is a popular Python library for machine learning tasks, offering efficient tools for data preprocessing, model selection, training, evaluation, and deployment. It offers a wide range of algorithms, including classification, regression, clustering, dimensionality reduction, and model ensemble techniques. Its user-friendly interface and consistent API make it accessible to beginners and experts, and integrates with other libraries like NumPy, Pandas, and Matplotlib.

Matplotlib

Matplotlib is a Python library that creates static, animated, and interactive visualizations. It offers various plotting functions, including line plots, bar charts, histograms, and scatter plots. It offers precise control over plot customization and supports multiple output formats like PNG, PDF, SVG, and Jupyter Notebook. Matplotlib is highly extensible and integrates with other libraries like NumPy and Pandas.

Seaborn

Seaborn is a Python statistical data visualization library that simplifies the creation of visually appealing and informative statistical plots. It offers a high-level interface for creating various types of plots, customizable color palettes, themes, and styling options. Seaborn integrates with Pandas data structures, making it easy to work with dataframes and includes functionalities for statistical estimation like linear regression and distribution fitting.

Pickle

Pickle is a Python module that converts Python objects into byte streams, allowing efficient storage and transfer over networks. It is commonly used for persisting machine learning models, caching data, and transferring objects between processes. Pickle supports binary and text protocols and offers compression for efficient storage. It is versatile, easy to use, and widely used in Python applications for data persistence tasks.

5.3 Streamlit Framework

Streamlit is an open-source Python framework for creating interactive web applications for data science and machine learning projects. It simplifies the process by providing a high-level API for widgets, charts, and interactive components. It supports real-time updates, data caching, and integration with popular libraries like Pandas, Matplotlib, and Plotly. Streamlit applications are easily shared and accessed through web browsers.

Here are some key points about Streamlit:

1. Simplicity: Streamlit allows you to convert data scripts into shareable web apps in minutes, all using pure Python. You don’t need to write HTML, CSS, or JavaScript.
2. Principles:
    - o Embrace Scripting: You can build an app with just a few lines of code using Streamlit’s simple API. As you iteratively save the source file, the app automatically updates. o Weave in Interaction: Adding widgets (like sliders, date pickers, and radio buttons) is as easy as declaring a variable. No need to handle backend, routes, or HTTP requests.
    - o Deploy Instantly: Share, manage, and deploy your apps directly from Streamlit.It’s free and Efficient.

Source Code:

```
cyber.py
import pandas as pd
import numpy as np
from sklearn import svm
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import confusion_matrix, precision_score
import sys
import pickle
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Load the dataset into a Pandas DataFrame
df = pd.read_csv('station_data_dataverse.csv')

# Print all of the features in the dataset
print(df.columns)
# read in dataset
df = pd.read_csv('station_data_dataverse.csv')

# add FalseDataInjection column with random 0 or 1 values
df['FalseDataInjection'] = np.random.randint(2, size=len(df))

# save new dataset to CSV file
df.to_csv('dataset_with_FDI.csv', index=False)
df.head()
df
# Check for any null or missing values in the DataFrame
print(df.isnull().sum())
# Drop all rows with null or empty values
df = df.dropna()
```

```
# Save the cleaned dataset to a new csv file
df.to_csv('cleaned_dataset.csv', index=False)
# read cleaned dataset
df = pd.read_csv('cleaned_dataset.csv')

# print all features
print(df.columns)
# Perform one-hot encoding on the 'Weekday' feature
one_hot = pd.get_dummies(df['weekday'], prefix='weekday')

# Add the new one-hot encoded features to the original dataset
df = pd.concat([df, one_hot], axis=1)

# Drop the original 'Weekday' feature since it's no longer needed
df.drop('weekday', axis=1, inplace=True)

# Print the updated dataset
print(df.head())
df
# Select the features for SVM
features = ['kwhTotal', 'dollars', 'chargeTimeHrs', 'distance', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri',
'Sat', 'Sun', 'managerVehicle']

# Split the data into X (features) and y (target)
X = df[features]
y = df['FalseDataInjection']

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

# Create SVM classifier with RBF kernel
clf = svm.SVC(kernel='rbf')
```

```
# Train the SVM classifier on the training data
clf.fit(X_train, y_train)

# Predict the target variable for the testing data
y_pred = clf.predict(X_test)
print(y_pred)
# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: %.2f" % accuracy)

# Plot the confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='g')
with open('svm_model.pkl', 'wb') as file:
    pickle.dump(clf, file)
import os
print(os.getcwd())
from sklearn.metrics import accuracy_score, precision_score, confusion_matrix

# Select the features for SVM
features = ['kwhTotal', 'dollars', 'chargeTimeHrs', 'distance', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri',
'Sat', 'Sun', 'managerVehicle']

# Split the data into X (features) and y (target)
X = df[features]
y = df['FalseDataInjection']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the Random Forest classifier on the training data
rf_classifier.fit(X_train, y_train)
```

```
# Predict the target variable for the testing data
rf_y_pred = rf_classifier.predict(X_test)

# Evaluate the accuracy of the Random Forest model
rf_accuracy = accuracy_score(y_test, rf_y_pred)
print("Random Forest Accuracy: %.2f" % rf_accuracy)
# Select the features for SVM
features = ['kwhTotal', 'dollars', 'chargeTimeHrs', 'distance', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri',
'Sat', 'Sun', 'managerVehicle']

# Split the data into X (features) and y (target)
X = df[features]
y = df['FalseDataInjection']

# Create SVM classifier with linear kernel
clf = svm.SVC(kernel='linear')

# Evaluate performance using 5-fold cross-validation
scores = cross_val_score(clf, X, y, cv=5)

# Print the mean accuracy and standard deviation of the cross-validation scores
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

# Train and test the SVM classifier on the data
clf.fit(X, y)
y_pred = clf.predict(X)

# Plot the confusion matrix
cm = confusion_matrix(y, y_pred)
sns.heatmap(cm, annot=True, fmt='g')
# Select the features for SVM
features = ['kwhTotal', 'dollars', 'chargeTimeHrs', 'distance', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri',
'Sat', 'Sun', 'managerVehicle']
```

```
# Split the data into X (features) and y (target)
X = df[features]
y = df['FalseDataInjection']

# Create SVM classifier with polynomial kernel
clf = svm.SVC(kernel='poly')

# Evaluate performance using 5-fold cross-validation
scores = cross_val_score(clf, X, y, cv=5)

# Print the mean accuracy and standard deviation of the cross-validation scores
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

# Train and test the SVM classifier on the data
clf.fit(X, y)
y_pred = clf.predict(X)

# Plot the confusion matrix
cm = confusion_matrix(y, y_pred)
sns.heatmap(cm, annot=True, fmt='g')
```

```
app.py
import streamlit as st
from PIL import Image
import pandas as pd
import pickle
from sklearn.impute import SimpleImputer
model=pickle.load(open('svm_model.pkl','rb'))
back_ground=Image.open("Cyberattack.jpeg")
def main():
    st.title("Detection of False Data Injection")
    st.image(back_ground,use_column_width=True)
    # Upload CSV file
    uploaded_file = st.file_uploader("Upload CSV file", type=["csv"])

    if uploaded_file is not None:
        # Read CSV file
        df = pd.read_csv(uploaded_file)
        features = ['kwhTotal', 'dollars', 'chargeTimeHrs', 'distance', 'Mon', 'Tues', 'Wed', 'Thurs', 
'Fri', 'Sat', 'Sun', 'managerVehicle']
        new_x=df[features]
        x_session=df[['sessionId']]
        if new_x.isnull().values.any():
            # Impute NaN values
            imputer = SimpleImputer(strategy='mean')  # You can choose a different strategy as 
needed
            new_x_imputed = imputer.fit_transform(new_x)

            # Predict using the imputed data
            y_pred = model.predict(new_x_imputed)
        else:
            # Predict directly if there are no NaN values
            y_pred = model.predict(new_x)
        # Display the dataframe
        result= x_session.assign(predict=y_pred)
        false_detect=result.loc[result['predict']==1]
        print(len(false_detect))
        print(false_detect)
        st.write(false_detect)
        st.write(len(false_detect))

if _name_ == "_main_":
    main()
```

CHAPTER – 6

RESULTS

6.1 RESULTS

Our System is deployed as web application which can be implemented in Visual Studio Code. The SVM Architecture plays an important role in detecting the cyber attack that is FDIA in Electric vehicle Charging Station.

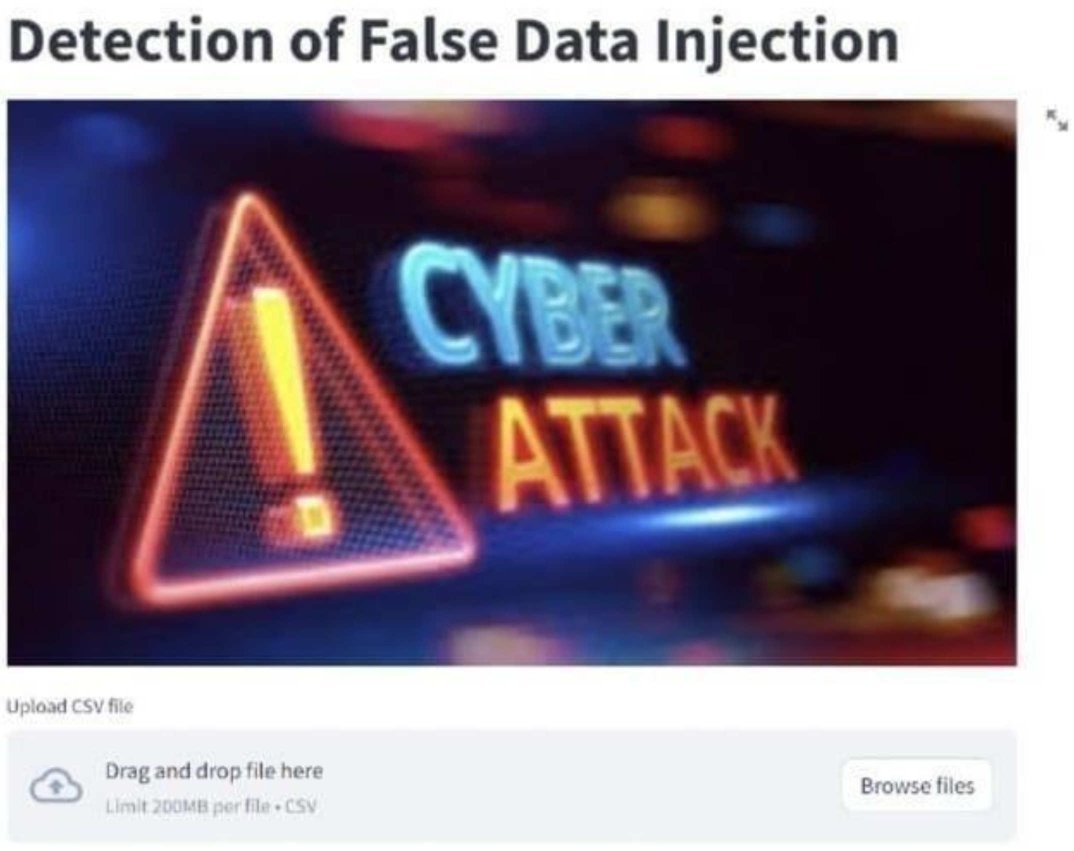
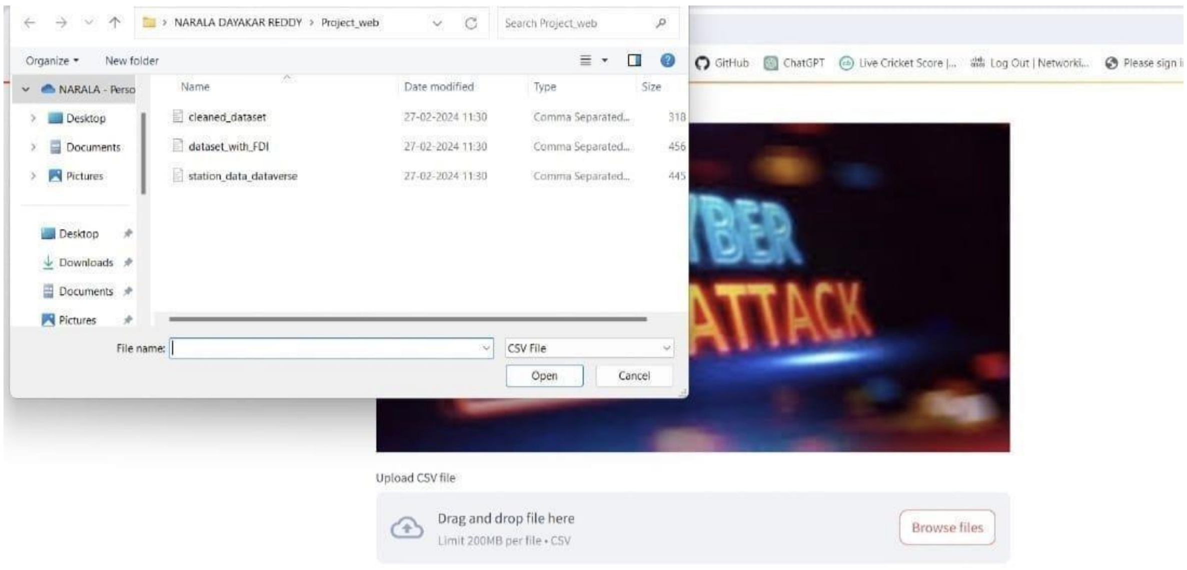


Figure 6.1 Home Page

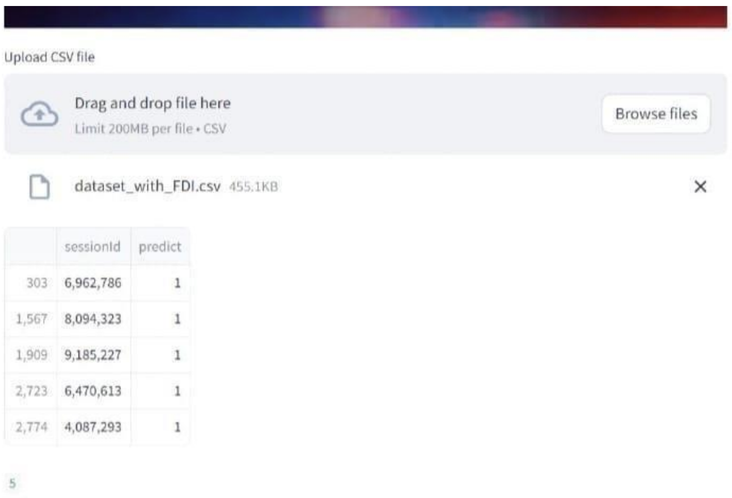
The above figure 6.1 describes about Home Page which shows the interface which can be easily understand by the user.And you can the option Drag and drop file here where you can drop the dataset of the Electric vehicle charging station dataset.Here we have to upload the .csv file to get the result.



Name	Date modified	Type	Size
cleaned_dataset	27-02-2024 11:30	Comma Separated...	318
dataset_with_FDI	27-02-2024 11:30	Comma Separated...	456
station_data_dataverse	27-02-2024 11:30	Comma Separated...	445

Figure 6.2 Selecting File Page

The figure 6.2 describes the selecting file page where it will displays the page of selecting the .csv file



	sessionId	predict
303	6,962,786	1
1,567	8,094,323	1
1,909	9,185,227	1
2,723	6,470,613	1
2,774	4,087,293	1

5

Figure 6.3 Result Page

The above figure 6.3 shows the results page that it detects the FDIA .Here ‘0’ represents that NO FDIA happened and ‘1’ represents that FDIA happened.

CHAPTER – 7

CONCLUSION

CONCLUSION

Our research successfully developed a robust system aimed at improving the detection of False Data Injection Attacks (FDIA) in Electric Vehicle (EV) charging station datasets. We proposed a model that achieved an accuracy of 81%, significantly surpassing prior attempts that reached only around 60%. Unlike existing systems that offered minimal accuracy gains, our approach demonstrated a substantial improvement.

During our exploration of machine learning techniques, we experimented with different models, including various Support Vector Machine (SVM) kernels such as RBF, linear, and polynomial, along with other algorithms like Random Forest. Through rigorous testing, we found that the SVM with an RBF kernel outperformed the others, delivering the highest accuracy in detecting false data injected into the dataset.

The SVM model classifies instances into two categories: whether false data has been injected (represented by 1) or not (represented by 0). This classification is based on several key features, such as total energy consumption (‘kwh Total’), charge time (‘chargeTimeHrs’), and the day of the week, among others.

Our project’s success in identifying false data ensures the security of EV charging stations, laying the foundation for further advancements in detecting FDIAs and other cyber-attacks. By employing advanced machine learning techniques and leveraging an extensive dataset, we provide a scalable and adaptable solution. The ability to detect such attacks in real-time helps safeguard critical infrastructure and paves the way for future research in anomaly detection and secure communication protocols within cyber-physical system.

CHAPTER – 8

REFERENCES

REFERENCES

[1] [D. Said, M. Elloumi and L. Khoukhi, "Cyber-Attack on P2P Energy Transaction Between Connected Electric Vehicles: A False Data Injection Detection Based Machine Learning Model," in IEEE Access, vol. 10, pp. 63640-63647, 2022, doi: 10.1109/ACCESS.2022.3182689.keywords: {Support vector machines;Computer security;Data models;Detectors;Blockchains;Training data;Phasor measurement units;Blockchain;connected electric vehicles;false data injection attack;machine learning;shortvector machine;smart contract},

[2] L. Garza and P. Mandal, "Detection and Classification of False Data Injection Attacks in Power Grids Using Machine Learning and Hyperparameter Optimization Methods," 2023 IEEE Industry Applications Society Annual Meeting (IAS), Nashville, TN, USA, 2023, pp. 1-8, doi: 10.1109/IAS54024.2023.10406838. keywords: {Support vector machines;Analytical models;Toxicology;Hyperparameter optimization;Feature extraction;Power grids;Principal component analysis;Classification algorithms;cyberattack;data analysis;data integrity;feature extraction;machine learning algorithms;optimization},

[3] A. Abdelfatah, A. Ali, M. F. Shaaban and A. Osman, "Optimal False Data Injection Attackon EV Chargers and DGs in Active Distribution Networks," 2023 3rd International Conferenceon Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Tenerife, Canary Islands, Spain, 2023, pp. 1-6, doi: 10.1109/ICECCME57830.2023.10253138.keywords: {Vehicle-to-grid;Costs;Mechatronics;Power distribution;Voltage;Wind power generation;Software;electric vehicles;vehicle-to-grid;cyber-physical attacks;false data injection;distribution system;distributed energy resources;optimization},

[4] Y. Liu, O. Ardakanian, I. Nikolaidis and H. Liang, "False Data Injection Attacks on Smart Grid Voltage Regulation With Stochastic Communication Model," in IEEE Transactions on Industrial Informatics, vol. 19, no. 5, pp. 7122-7132, May 2023, doi: 10.1109/TII.2022.3209287.

[5] keywords: {Voltage measurement;Stochastic processes;Real-time systems;Phasor measurement units;Informatics;Charging stations;Batteries;Cyberattacks;distribution system state estimation;electric vehicles (EVs);stochastic optimization},

[6] A. Kumar, N. Saxena and B. J. Choi, "Machine Learning Algorithm for Detection of False Data Injection Attack in Power System," 2021 International Conference on Information Networking (ICOIN), Jeju Island, Korea (South), 2021, pp. 385-390, doi: 10.1109/ICOIN50884.2021.9333913. keywords: {Machine learning algorithms;Feature extraction;Smart grids;Power systems;Information and communication technology;Time factors;Time complexity;Data Injection Attack;Smart Grid;Machine Learning;Power System},

[7] R. Nawaz, M. A. Shahid, I. M. Qureshi and M. H. Mehmood, "Machine learning based falsedata injection in smart grid," 2018 1st International Conference on Power, Energy and Smart Grid (ICPESG), Mirpur Azad Kashmir, Pakistan, 2018, pp. 1-6, doi: 10.1109/ICPESG.2018.8384510. keywords: {Transmission line measurements;Smart grids;Time measurement;Jacobian matrices;Power transmission lines;Support vector machines;Linear regression;Smart Grid;Cyber Attacks;Stealthy Attack;Machine Learning;SVM;Linear Regression},

[8] P. L. Bhattar, N. M. Pindoriya, A. Sharma and R. T. Naayagi, "False Data Injection Attack Detection with Feedforward Neural Network in Electric Vehicle Aggregator Bidding Price," 2022 IEEE PES Innovative Smart Grid Technologies - Asia (ISGT Asia), Singapore, Singapore,2022, pp. 665-669, doi: 10.1109/ISGTAsia54193.2022.10003474. keywords: {Transactive energy;Training;Asia;Process control;Electric vehicles;Feedforward neural networks;Smart grids;Anomaly detection;electric vehicle aggregator;electricity price bidding;false data injection attacks;transactive energy management},

[9] D. Said and M. Elloumi, "A New False Data Injection Detection Protocol based Machine Learning for P2P Energy Transaction between CEVs," 2022 IEEE International Conference onElectrical Sciences and Technologies in Maghreb (CISTEM), Tunis, Tunisia, 2022, pp. 1-5, doi: 10.1109/CISTEM55808.2022.10044067. keywords: {Support vector machines;Protocols;Simulation;Urbanareas;Transportation;Training data;Electric vehicles;Connected Electric Vehicles;Machine Learning;Short Vector Machine;False Data Injection Attack;Blockchain;Smart Contract},

[10]A. Kumar, N. Saxena and B. J. Choi, "Machine Learning Algorithm for Detection of False Data Injection Attack in Power System," 2021 International Conference on Information Networking (ICOIN), Jeju Island, Korea (South), 2021, pp. 385-390, doi: 10.1109/ICOIN50884.2021.9333913. keywords: {Machine learning algorithms;Feature extraction;Smart grids;Power systems;Information and communication technology;Time factors;Time complexity;Data Injection Attack;Smart Grid;Machine Learning;Power System},

[11]X. Tong and W. Qi, "False Data Injection Attack on Power System Data- Driven Methods Based on Generative Adversarial Networks," 2021 IEEE Sustainable Power and Energy Conference (iSPEC), Nanjing, China, 2021, pp. 4250-4254, doi:10.1109/iSPEC53008.2021.9735442 keywords: {Analytical models;Conferences;Predictive models;Generative adversarial networks;Generators;Power systems;machinelearning;generative adversarial network;critical clearing time;false data injection attack},

[12]D. Said, A decentralized electricity trading framework (DETF) for con nected EVs: A blockchain and machine learning for pro t margin opti mization, IEEE Trans. Ind. Informat., vol. 17, no. 10, pp. 65946602, Oct. 2021, doi: 10.1109/TII.2020.3045011.

[13]L. Liu, M. Esmalifalak, Q. Ding, V. A. Emesih, and Z. Han, Detect ing false data injection attacks on power grid by sparse optimization, IEEE Trans. Smart Grid, vol. 5, no. 2, pp. 612621, Mar. 2014, doi: 10.1109/TSG.2013.2284438.

[14]S. Zheng, T. Jiang, and J. S. Baras, Robust state estimation under false data injection in distributed sensor networks, in Proc. IEEE Global Telecommun. Conf. (GLOBECOM), Dec. 2010, pp. 15, doi: 10.1109/GLOCOM.2010.5685223.

[15]C. Pei, Y. Xiao, W. Liang, and X. Han, PMU placement protec tion against coordinated false data injection attacks in smart grid, IEEE Trans. Ind. Appl., vol. 56, no. 4, pp. 43814393, Aug. 2020, doi: 10.1109/TIA.2020.2979793.

[16]B. Tang, J. Yan, S. Kay, and H. He, Detection of false data injec tion attacks in smart grid under colored Gaussian noise, in Proc. IEEE Conf. Commun. Netw. Secur. (CNS), Oct. 2016, pp. 172179, doi: 10.1109/CNS.2016.7860483.

[17]K.Huang,Z.Xiang,W.Deng,C.Yang,andZ.Wang, Falsedatainjection attacks detection in smart grid: A structural sparse matrix separation method, IEEE Trans. Netw. Sci. Eng., vol. 8, no. 3, pp. 25452558, Jul. 2021, doi: 10.1109/TNSE.2021.3098738.

[18]J. Zhao and L. Mili, Vulnerability of the largest normalized residual statistical test to leverage points, IEEE Trans. Power Syst., vol. 33, no. 4, pp. 46434646, Jul. 2018.

[19]X. Liu, Y. Guan, and S. W. Kim, Bayesian test for detecting false data injection in wireless relay networks, IEEE Commun. Lett., vol. 22, no. 2, pp. 380383, Feb. 2018.

[20]N. M. M. Mohamed, H. M. Sharaf, D. K. Ibrahim, and A. Elgharably, Proposed ranked strategy for technical and economical enhancement of EVs charging with high penetration level, IEEE Access, vol. 10, pp. 4473844755, 2022, doi: 10.1109/ACCESS.2022.3169342.

[21]S. U. Jeon, J.-W. Park, B.-K. Kang, and H.-J. Lee, Study on battery charging strategy of electric vehicles considering battery capacity, IEEE Access, vol. 9, pp.8975789767, 2021, doi: 10.1109/ACCESS.2021.3090763.

[22]D. Said and H. T. Mouftah, A novel electric vehicles charg ing/discharging management protocol based on queuing model, IEEE Trans. Intell. Vehicles, vol. 5, no. 1, pp. 100111, Mar. 2020, doi: 10.1109/TIV.2019.2955370.

[23]E.Braco, I. S. Martín, A. Berrueta, P. Sanchis, and A. Ursúa, Experimen tal assessment of rst- and second-life electric vehicle batteries: Perfor mance, capacity dispersion, and aging, IEEE Trans. Ind. Appl., vol. 57, no. 4, pp. 41074117, Jul./Aug. 2021, doi: 10.1109/TIA.2021.3075180.

[24]S. K. Singh, K. Khanna, R. Bose, B. K. Panigrahi, and A. Joshi, Joint transformation-based detection of false data injection attacks in smart grid, IEEE Trans. Ind. Informat., vol. 14, no. 1, pp. 8997, Jan. 2018, doi: 10.1109/TII.2017.2720726.

[25]D. Xue, X. Jing, and H. Liu, Detection of false data injection attacks in smart grid utilizing ELM-based OCON framework, IEEE Access, vol. 7, pp. 3176231773, 2019, doi: 10.1109/ACCESS.2019.2902910.