

ABSTRACT

The high relying of electric vehicles on either in vehicle or between-vehicle communications can cause big issues in the system. This paper is going to mainly address the cyber-attack in electric vehicles and propose a secured and reliable intelligent framework to avoid hackers from penetration into the vehicles. The proposed model is constructed based on an improved support vector machine model for anomaly detection based on the controller area network (CAN) bus protocol. In order to improve the capabilities of the model for fast malicious attack detection and avoidance, a new optimization algorithm based on social spider (SSO) algorithm is developed which will reinforce the training process offline. Also, a two-stage modification method is proposed to increase the search ability of the algorithm and avoid premature convergence. Finally, the simulation results on the real data sets reveal the high performance, reliability, and security of the proposed model against denial-of-service (DoS) hacking in the electric vehicles.

TABLE OF CONTENTS

Chapter No.		Contents	PageNo
1		Introduction	1-2
	1.1	Introduction	1
	1.2	Motivation	1
	1.3	Literature Review	1
	1.4	Problem Definition	2
	1.5	Objective of the Project	2
2		System Analysis	3-5
	2.1	Existing and Proposed System	3
	2.2	Functional Requirements	4
3		Software Environment	6-15
	3.1	Software	6
4		System Design and Architecture	16-24
	4.1	Data Flow Diagram	16
	4.2	Architecture Diagram	18
	4.3	Class Diagram	20
	4.4	Sequential Diagram	20
5		Software Development Life Cycle	25-27
	5.1	Phases of SDLC	25
6		Implementation	28-52
	6.1	Sample Code	42
7		Testing	53-57
	7.1	Introduction	53
	7.2	Sample Test cases	54
8		Output Screen	58-64
	8.1	Screenshots	58
9		Conclusion and Future Scope	65-65
10		References	66-67

LIST OF FIGURES

Fig.No	Figure Title	Page No
1	Flowchart of the Project	15
2	Dataflow Diagram of the Project	17
3	System Architecture of the Project	18
4	Use Case Diagram	19
5	Class Diagram	20
6	Sequence Diagram	21
7	Collaboration Diagram	21
8	Hyperplane Representation	22
9	Phases Of SDLC	25
10	Python Website	36
11	Python Version	36
12	Python Version For Windows	37
13	Version Of Python With Os	37
14	Downloading Python	38
15	Install Python	38
16	Python Setup	39
17	Command Prompt	39
18	Run Command	40
19	Idle Python	40
20	Save The File	41
21	Output Screen 1	58

22	Output Screen 2	59
23	Output Screen 3	59
24	Output Screen 4	60
25	Output Screen 5	60
26	Output Screen 6	61
27	Output Screen 7	61
28	Output Screen 8	62
29	Output Screen 9	62
30	Output Screen 10	63

LIST OF TABLES

S No	Table Name	Page No
1.	Literature Survey	2

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The imperative need to address the vulnerabilities in contemporary automotive systems is the driving force behind the creation of an intelligent, data-driven model that uses machine learning to protect intravehicle communication. The likelihood of cyberattacks aimed at intravehicle communication has increased because of cars becoming more electronically connected and dependent on communication networks. Vehicles are vulnerable to a range of vulnerabilities due to the lack of strong security mechanisms in traditional protocols like the CAN bus.

1.2 MOTIVATION

This project's proactive approach to cybersecurity is what makes it significant. The model can identify and counter new and unseen cyberattacks instead of just responding to known threats, which enhances the overall security posture of intravehicle communication systems. This strategy is essential considering how cyber threats are changing and how connected technologies are becoming more and more prevalent in modern cars.

1.3 LITERATURE REVIEW

Through the utilization of machine learning algorithms to examine intravehicle communication patterns, abnormalities suggestive of security breaches can be promptly identified, facilitating the implementation of preemptive defense measures. To safeguard car occupants and defend against new cyberthreats in the digital era, this project seeks to improve the security and dependability of automotive systems.

S. No	Title	Link	Authors	Techniques used
1	An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning	https://www.researchgate.net/publication/334378892	Mamdooh Al-Saud, Ali M.Eltamaly, Mohamed A.Mohamed, Abdollah Kavousi Fard	Machine Learning
2	An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning	https://jespublication.com/	Vipin Sagar Nagelli, Kondamu Sai Teja	Machine Learning and CAN Bus protocol
3	An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning	https://www.ijraset.com/research-paper/	Dr.R. Poorvadevi Bodala Yaswanth Nikhil, Darisi Venkata Sravan Kumar	CAN Bus Protocol and Machine learning

Table 1: Literature Survey

1.4 PROBLEM DEFINITION

The project's goal is to use machine learning to create an intelligent, data-driven model that will improve the security of intravehicle communication systems. The approach seeks to identify and reduce cyber dangers by evaluating communication patterns in real-time, protecting automotive systems' dependability and safety in the face of new security threats.

1.5 OBJECTIVE OF THE PROJECT

The objective is to create a model that can use machine learning to identify abnormalities in communication patterns instantly, improving the security and dependability of automotive systems against new cyberthreats.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING AND PROPOSED SYSTEM

By protecting against online attacks and guaranteeing the dependability and safety of vehicle operations, the suggested model seeks to improve the security of EV communication systems. Real statistics collected from EVs are used to evaluate the model's viability and performance, showing how well it can identify and mitigate possible security breaches. In electric vehicles, CAN standard is the most widely used protocol by automakers for communications with low cost in the units with a high number of components, up to 500 million chips. In the vehicle industry, the CAN resiliency and noise resistance level is acceptable owing to its structure. Unfortunately, CAN bus protocols do not offer confidentiality and authentication to CAN data frames so making it possible for hackers to enter the vehicle system, either on a wired or wireless approach. In the wired approach, one can communicate with the CAN bus through the OBD-II maintenance port located under the steering in most vehicles. Although the main idea behind this port is to be used for diagnostics of engine and vehicle maintenance, it will let hackers take the CAN packets using a simple scanning tool. From this point, it is easy to read and write traffic in the CAN bus with the use of ECOM API such as CANReceiveMessage and CANTransmitMessage.

DISADVANTAGES OF EXISTING SYSTEM

LESS ACCURACY

The study offers a novel solution to improve the communication networks security of electric vehicles. The suggested method uses machine learning techniques, such as optimization algorithms and support vector machines, to identify anomalies in the CAN bus data that may be signs of possible cyber risks. Although real-world datasets are used to assess the model's viability and performance, additional validation may be necessary to guarantee the model's correctness and efficacy in practical situations.

LOW EFFICIENCY

The study presents a novel technique to improve the security of communication networks for electric vehicles. Although the suggested method makes use of optimization and machine learning techniques to identify irregularities in CAN bus data, its practicality may be restricted. To increase its efficacy and efficiency in effectively identifying and managing cyber risks, more testing and modification are required.

PROPOSED SYSTEM

The performance of the suggested model is examined using experimental data gathered from an electric vehicle in the paper's conclusion. The investigation highlights the possible risks posed by Denial of-Service (DoS) assaults in car communication systems, namely the possibility of Major accident or loss of control over crucial vehicle functions.

Through the analysis of CAN bus traffic data collected over the course of a 10-minute driving scenario, the model exhibits its capacity to learn and differentiate between typical and unusual patterns of behaviour. Through trace analysis, the study guarantees thorough coverage of all potential CAN bus identifier numbers, offering a proof-of-concept for the suggested anomaly detection approach. The experimental system mimics ignition, driving on public roads, applying brakes, and parking, among other realistic driving scenarios.

This thorough testing the demonstrates suggested model's capacity to recognize abnormal activity and take appropriate action, improving the security of car communication systems.

2.2 FUNCTIONAL REQUIREMENTS

HARDWARE & SOFTWARE REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : i3 or above.
- Ram : a minimum of 4 GB.
- Hard Disk : a minimum space of 40 GB.

SOFTWARE REQUIREMENTS:

- Operating system : Windows 8 or above.
- Coding Language : Python

HIGH ACCURACY

The performance of the suggested model is assessed in the research using actual data from an electric vehicle. The study highlights the significance of preventing Denial-of-Service (DoS) attacks in car communication and shows how well the model can identify abnormalities in CAN bus traffic. By means of extensive testing conducted under authentic driving scenarios, encompassing ignition, manoeuvres, and parking, the model validates its efficacy in augmenting the security of intravehicular communication systems.

HIGH EFFICIENCY

The efficiency of the suggested approach is assessed in the research using actual data from an electric vehicle. The primary objective of this study is to identify Denial-of-Service (DoS) assaults in vehicle communication. The findings indicate that the model can effectively and rapidly detect anomalies in CAN bus traffic.

The model is put to the test in a variety of simulated driving scenarios, including as parking, driving manoeuvres, and ignition, and it successfully improves intravehicle communication security. The research concludes by presenting a strong model for intravehicle communication security, especially when it comes to preventing Denial-of-Service (DoS) assaults. The study shows the model's efficacy in improving security in automotive systems by utilizing real-world data from an electric car and concentrating on the effectiveness of anomaly detection in CAN bus traffic. The model has been rigorously tested in a variety of driving conditions, demonstrating its capacity to identify abnormal behaviour quickly and effectively. This enhances the overall security and dependability of vehicle communication networks.

CHAPTER 3

SOFTWARE ENVIRONMENT

3.1 SOFTWARE:

This project suggests a clever, data-driven methodology that uses machine learning to improve the security of intravehicular communications. The approach seeks to identify and avert possible security risks in the car's communication network by utilizing data-driven insights. The integrity and privacy of intravehicle communication systems are protected by the system's ability to recognize anomalous behaviours or hostile activities through the use of machine learning methods, such as anomaly detection or classification techniques. This strategy shows how cutting-edge technologies can be used to solve cybersecurity issues in automobile settings, resulting in safer and more secure transportation networks.

SOFTWARE REQUIREMENTS AND SPECIFICATION:

Python is the main language used in programming, with TensorFlow or PyTorch for deep learning tasks, Scikit-learn for machine learning techniques, and Pandas, NumPy, Matplotlib, and Seaborn for data processing and visualization rounding out the list of languages and frameworks. Integrated development environments (IDEs) like PyCharm, Visual Studio Code, or Jupyter Notebook will be used. Important elements include preprocessing and data collection, which include defining procedures for gathering intravehicle communication data and performing cleaning and feature extraction preprocessing operations. To identify and mitigate security threats, machine learning models are used. These models include anomaly detection techniques like Isolation Forest and One-Class SVM, classification algorithms like Random Forest and Support Vector Machines, and deep learning models like Convolutional Neural Networks and Recurrent Neural Networks. Procedures for training and evaluating models include cross-validation methods, hyperparameter tuning, and the use of measures like accuracy, precision, recall, F1-score, and area under the ROC curve to gauge the model's performance.

Guidelines for interoperability with different communication protocols (e.g., CAN, Ethernet) and car systems are required for integration with systems. Mechanisms for real-time monitoring and alerting are put in place to quickly identify and address abnormal activity. With safeguards in place to guarantee the protection of sensitive vehicle data and compliance with pertinent standards like ISO/SAE 21434 and the NIST Cybersecurity Framework, security and privacy concerns are of the utmost importance. Documenting software architecture, design choices, implementation specifics, and usage guidelines requires thorough documentation, which includes technical papers and presentations. The project's success also depends on considering scalability to manage massive volumes of data and performance benchmarks to evaluate effectiveness and computational overhead.

INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such away so that it provides security and ease of use while retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when errors occur.

OBJECTIVES:

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulations can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided when needed so that the user will not be in maze of instant. Thus, the objective of input design is to create an input layout that is easy to follow.

OUTPUT DESIGN:

A quality output is one which meets the requirements of the end user and presents the information clearly. In any system the results of processing are communicated to the users and to other systems through outputs. In output design it is determined how the information is to be displayed for immediate need and also the hard copy output. It is the most important and direct source of information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can be used easily and effectively. When analyzing design computer output, they should identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create documents, reports, or other formats that contain information produced by the system.

USER REQUIREMENTS:

Various stakeholders have different needs for "An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning". Robust intrusion detection is what security experts aim for. Manufacturers of automobiles need smooth integration. Real-time monitoring is a top priority for fleet managers. Tools for software engineers must be easy to use. Adherence to regulations is crucial. End users want more security. Good datasets are necessary for data analysts. By addressing these, a thorough solution for safer intravehicular communication is ensured.

SOFTWARE REQUIREMENTS:

For "An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning" to be developed, certain software requirements must be met. Model development requires Scikit-learn, Pandas, TensorFlow or PyTorch, Python, and Scikit-learn. Coding is made easier by IDEs such as Jupyter Notebook. Matplotlib and NumPy are required for data preprocessing. Compatibility with communication protocols such as Ethernet and CAN are a prerequisite for integration. Effective alerting systems are essential for real-time monitoring. Adherence to pertinent standards is required due to security concerns. Transparency is ensured by documentation. Performance and scalability are essential when working with big datasets. Together, these specifications make it possible to develop a strong security solution for communications within vehicles.

HARDWARE REQUIREMENTS:

An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning has comparatively low hardware requirements, yet they are essential for the project's successful development. It is necessary to have a standard computer with adequate RAM, storage, and computing capability. Having access to a GPU is optional, however it can greatly speed up model training. Hardware for real-time monitoring and data gathering, such as CAN bus interfaces, may be required.

Furthermore, networking hardware might make it easier for the monitoring system and outside stations to communicate. Overall, a conventional computer setup is usually sufficient for this job, however specific demands may vary. Additional hardware may be needed for specialized activities.

SYSTEM STUDY FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified.

Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demand for the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CONTENT DIAGRAM OF THE PROJECT:

The project's sequential flow is depicted in this content diagram, which begins with data collecting within intravehicle communication systems. After that, preprocessing and feature extraction are applied to the gathered data in order to prepare and train machine learning models. After the model is developed, tested, and its performance metrics are assessed, it is integrated and deployed using intravehicle communication systems. To quickly identify and address security threats, real-time monitoring and alerting features are used. In order to guarantee the integrity and confidentiality of intravehicle communications, security and privacy safeguards are include Security & Privacy Measures. communications, security, and privacy safeguards are including Security & Privacy Measures. Effective alerting systems are essential for real-time monitoring. Adherence to pertinent standards is required due to security concerns. Transparency is ensured by documentation. Performance and scalability are essential when working with big datasets. Together, these specifications make it possible to develop a strong security solution for communications within vehicles.

ALGORITHMS AND FLOWCHARTS:

Algorithms for detecting anomalies:

One-Class Support Vector Machines for Isolation Forests (SVM)

Classification Algorithms:

Autoencoders (for unsupervised anomaly detection)

Random Forest: Their ability to handle big datasets and little configuration make them ideal for finding complex pharmaceutical safety trends and supporting clinical decision-making. One potent machine learning method that can be very useful for protecting intravehicle communications is random forests. Multiple decision trees, each trained on a different subset of the data and using a random selection of features, make up a random forest, an ensemble learning technique.

Decision Trees: Decision trees are a useful tool for anomaly detection when it comes to machine learning-based intravehicle communication security. These models work well for analyzing communication patterns inside vehicles because they use a tree-like structure, because they are easily interpreted, decision trees are especially useful. Because they are easily observable, engineers and cybersecurity specialists can comprehend the reasoning behind the model's choices.

Logistic Regression: This statistical technique is perfect for differentiating between normal and abnormal communication patterns within the intravehicle network since it works well for binary classification tasks. For logistic regression to function, it must first determine the likelihood that a given communication pattern falls into one of several classes, such as normal or abnormal.

Support Vector Machines (SVM): Specifically designed for binary classification tasks, support vector machines (SVM) are a type of supervised machine learning algorithm that excels at differentiating between typical and anomalous intravehicle network communication patterns. The best hyperplane to divide the two classes of data points in the feature space is found by SVM. By maximizing the margin between the classes, this hyperplane strengthens its resistance to data noise and outliers. Based on the features that are taken out of the data, SVM is able to recognize and categorize abnormal communication patterns in the context of intravehicle communications. SVM's capacity to handle high-dimensional data, which is typical in intravehicular communication systems, is one of its main advantages.

Deep Learning Algorithms:

CNNs, or convolutional neural networks: CNNs are a kind of deep learning algorithm that are frequently used for image recognition tasks. However, they can also be used for sequential data, like intravehicular network communication patterns. CNNs work well at identifying patterns in data with a grid-like structure, like time-series or picture data. CNNs can effectively identify abnormal communication patterns in the context of intravehicle communications by analyzing the temporal and spatial relationships between data points.

Neural Networks with Recurrence: Secure intravehicle communications can be greatly aided by recurrent neural networks, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. Since they work particularly well with sequential data analysis, RNNs and LSTMs are the best options for capturing the temporal dependencies found in intravehicle network communication patterns. With the help of this ability, they can recognize and categorize unusual communication patterns according to the order of events

Networks using Long Short-Term Memory (LSTM): Long Short-Term Memory (LSTM) networks can be an essential part of a secure intravehicle communications system. Recurrent neural networks, or LSTM networks for short, are a kind of network that works particularly well for processing and analyzing sequential data. This makes them useful for identifying abnormalities in the communication patterns that occur within the intravehicle network.

GRUs, or gated recurrent units: The use of Gated Recurrent Units (GRUs) to secure intravehicular communications can be beneficial. Recurrent neural networks (RNNs) of the GRU type are useful for studying communication patterns in intravehicle networks because they can effectively model sequential data and capture long-term dependencies. The ability of

Clustering Techniques:

K-Means Clustering (to classify comparable communication styles): Within the intravehicle network, similar communication styles can be categorized using K-Means Clustering. An unsupervised machine learning technique called K-Means Clustering is especially helpful for

organizing related data points into clusters according to their characteristics. K-Means Clustering can be used to find groups of communication patterns that share similar features in the context of intravehicle communications. Understanding the common communication styles within the network and spotting any variations from these patterns that might point to strange behaviour can both benefit from this.

K-Means Clustering's ease of use and effectiveness are two of its main benefits.

Applications with Noise: A Density-Based Spatial Clustering Approach (DBSCAN)

Algorithms for Dimensionality Reduction:

- Principal Component Analysis (PCA)
- Ensemble Learning Algorithms: t-distributed Stochastic Neighbour Embedding
- GBMs, or gradient boosting machines
- Sequential Pattern Mining Algorithms for AdaBoost:

The Apriori Algorithm is utilized to identify recurring patterns in communication events.

Social Spider Optimization: Inspired by social spiders' hunting and foraging habits, the Social Spider Optimization Algorithm (SSO) is a metaheuristic algorithm. Every single spider in SSO stands for a potential fix for an optimization issue. The spiders' movements within the search space are determined by their individual experiences as well as those of other spiders. An initial population of spiders is randomly distributed throughout the search space at the beginning of the algorithm. The spiders adjust their positions according to a set of rules that replicate the social interactions between spiders during each iteration. These include web communication (sharing information amongst spiders), exploitation (following the best spider), and exploration (random movement).

FLOWCHART:

Intravehicle communication data must be gathered and pre-processed, pertinent features must be extracted, machine learning models must be chosen and trained, their effectiveness must be assessed, the models must be integrated into the vehicle system, and security improvements

based on model predictions must be put into practice. After the upgraded communication system is operational, it ends.

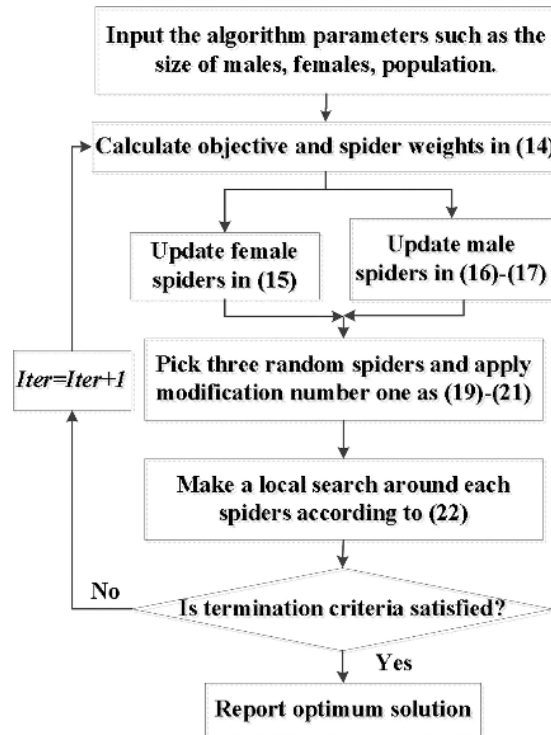


FIG.1 FLOWCHART OF THE PROJECT

CONCLUSION:

The project's analysis highlights how important it is to use machine learning to strengthen intravehicle communication systems. By utilizing a methodical approach that encompasses data gathering, preprocessing, model creation, and integration, the project seeks to improve security by promptly identifying irregularities and reducing potential threats. The project aims to establish a robust framework to protect sensitive data and guarantee the integrity of automotive systems by incorporating intelligent models into vehicle systems and putting proactive security measures in place.

CHAPTER 4

SYSTEM DESIGN AND ARCHITECTURE

INTRODUCTION:

The project's goal is to use machine learning to protect intravehicle communications. Communication networks must be protected since cybersecurity risks rise with the number of linked cars. The concept uses machine learning algorithms to identify and eliminate hazards instantly. By gathering data, preparing it, and integrating models, it seeks to build a strong security architecture that can be adjusted to changing threats. This program aims to guarantee safer car travel by fusing cutting-edge technology with data-driven strategies.

4.1 DATAFLOW DIAGRAM

To secure intravehicle communications using machine learning, a data flow diagram for the project "An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning" must show the data flow and procedures that must be followed. A data flow diagram (DFD) shows how data moves through a system in visual form. External entities (data sources or destinations), processes (actions or transformations), data stores (where data is kept), and data flows (how data moves between components) are its primary constituents. DFDs, which show data mobility, aid in the understanding of system architecture and functioning.

Components of data flow diagram:

1. External Entities
2. Processes
3. Data flow
4. Data stores
5. Annotations / Descriptions

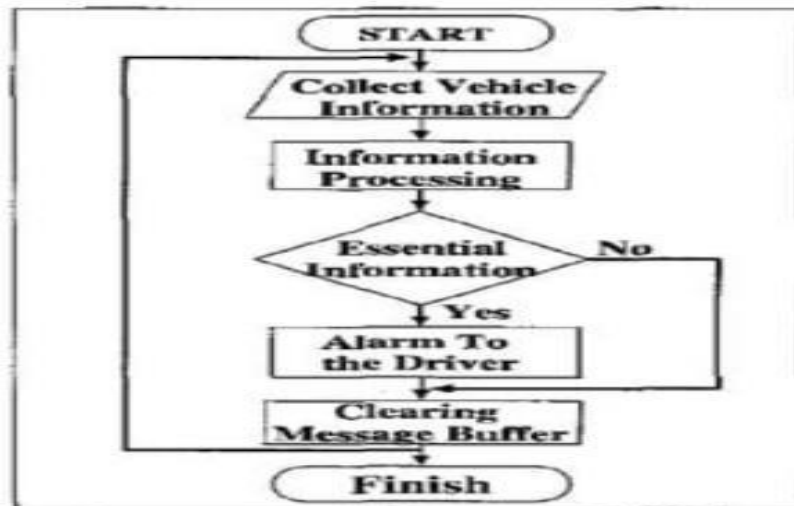


FIG.2 DATA FLOW DIAGRAM OF THE PROJECT

1.External Entities: These are the locations or sources of data that are not part of the system. They are not a part of the system, yet they engage with it. External entities in the context of the project can be other systems interacting with the security framework or data sources for intravehicle communication data.

2.Processes: These stand for the operations or changes that take place inside the system. Processes in the project could involve anomaly detection, model training, data pretreatment, etc.

3.Data flow: These show how information moves between data stores, processes, and external entities. They are usually labeled to indicate the type of data being transferred and they illustrate how data flows through the system.

4.Data stores: These indicate the locations within the system where data is kept. Databases, files, and other storage devices may be examples of this. Data storage in the project would house trained models, intravehicle communication data, etc.

5.Annotations and Descriptions: These offer further details about the diagram's components, such as explanations of the data flows or descriptions of the data stores and processes.

4.2 ARCHITECTURE DIAGRAM

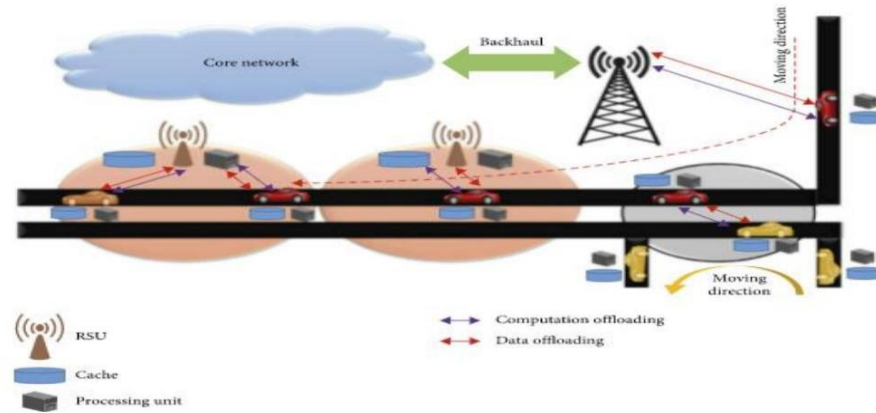


FIG.3 SYSTEM ARCHITECTURE OF THE PROJECT

UML DAIGRAMS:

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.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns, and components.
7. Integrate best practices.

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 diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

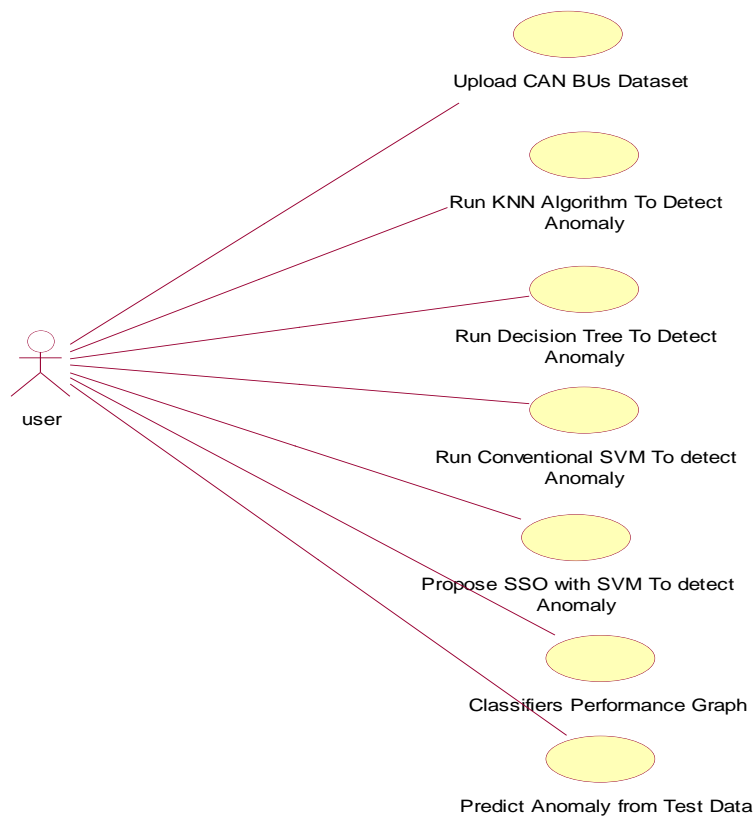


FIG.4 USE CASE DIAGRAM

4.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

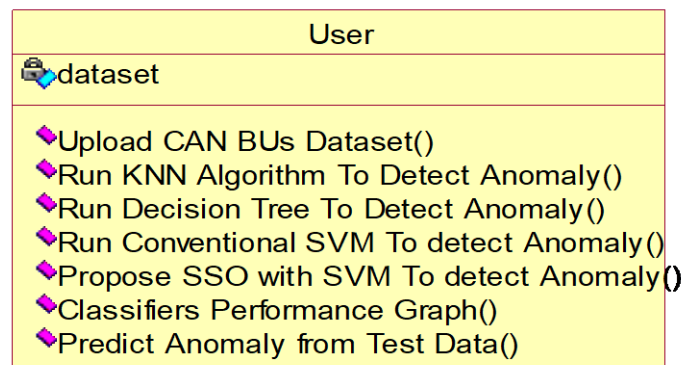


FIG.5 CLASS DIAGRAM

4.4 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.

Starting a Conversation: A message is sent by the intravehicle communication system to start a conversation.

Reception of Messages: The data processing module receives the message.

Information Processing: LSTM and GRU are two examples of the machine learning algorithms used by the data processing module to analyse the message. Finding anomalies in the communication pattern are identified by machine learning algorithms.

Making Choices: The system determines whether the communication pattern is normal or abnormal
Generation of Reactions: The system generates a response, such as blocking suspicious communication or warning the driver, if an anomaly is

detected. Loop of Feedback: The system's ability to detect anomalies may be enhanced by adding input from external sources or the driver.

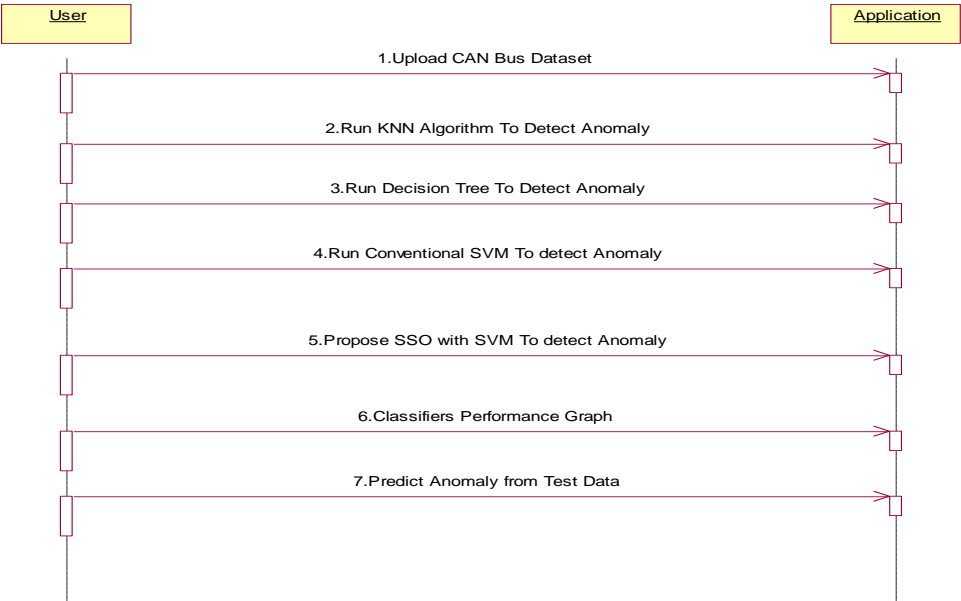


FIG.6 SEQUENCE DIAGRAM

COLLABORATION 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.

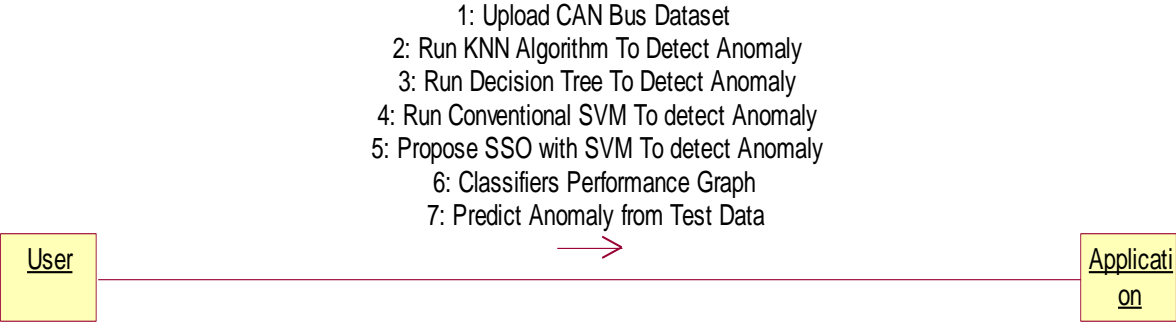


FIG.7 COLLABORATION DIAGRAM

MODULE DESIGN AND ORGANIZATION:

MODULES:

1. Upload CAN Bus Dataset' button and upload dataset.
2. Run KNN Algorithm To Detect Anomaly' button to build KNN classifier train model to detect anomaly and evaluate its performance based on 4 indices.
3. Run Conventional SVM To detect Anomaly' button to evaluate conventional SVM performance.
4. Propose SSO with SVM To detect Anomaly' button to run propose SSO with SVM classifier and evaluate its performance.
5. Classifiers Performance Graph' button to get performance graph between all classifiers.
6. Predict Anomaly from Test Data' button to upload test data and predict it label.

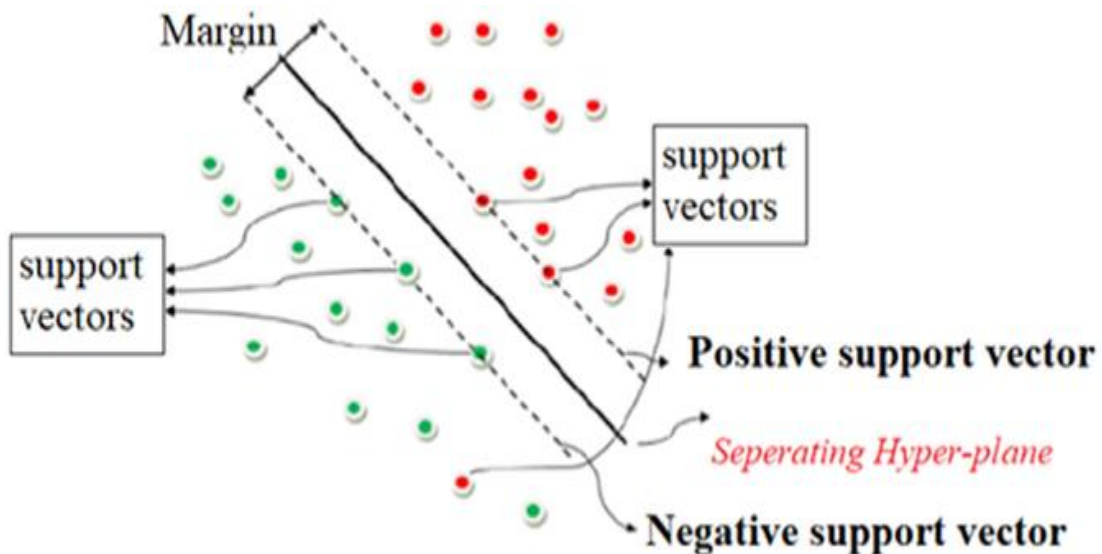


FIG.8 HYPERPLANE REPRESENTATION

LIBRARIES:

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code.
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using Numpy which allows Numpy to integrate with a wide variety of databases seamlessly and speedily.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyse.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

CHAPTER 5

SOFTWARE DEVELOPMENT LIFE CYCLE

5.1 Phases of SDLC

The Software Development Life Cycle (SDLC) is a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time. The goal of SDLC is to produce superior software that meets and exceeds all customer expectations and demands. The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. in detail, theSDLC methodology focuses on the following phases of software development:

- Requirement analysis
- Planning
- Software design such as architectural design
- Software development
- Testing
- Deployment

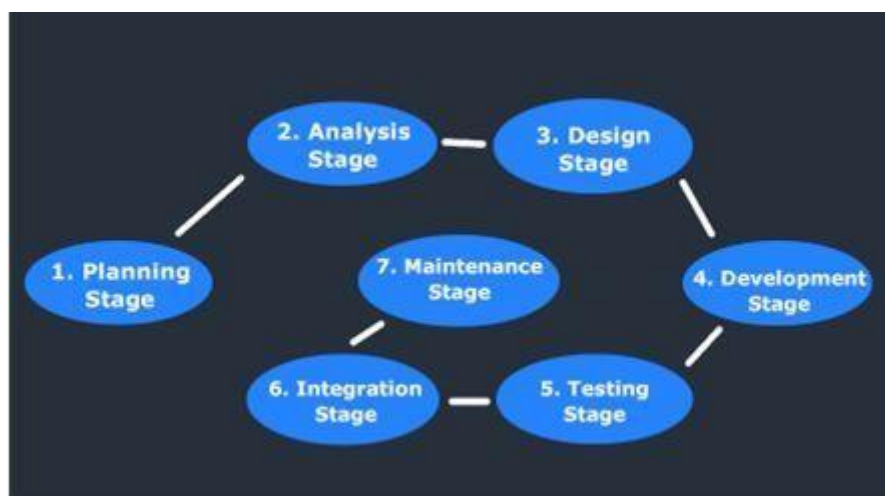


FIG.9 PHASES OF SDLC

Planning Stage

Before we even begin with the planning stage, the best tip we can give you is to take time and acquire proper understanding of app development lifecycle. The planning stage (also called the feasibility stage) is exactly what it sounds like: the phase in which developers will plan for the upcoming project. It helps to define the problem and scope of any existing systems, as well as determine the objectives for their new systems.

Analysis Stage

The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes.

Developers may:

- Define any prototype system requirements.
- Evaluate alternatives to existing prototypes.
- Perform research and analysis to determine the needs of end-users.

Furthermore, developers will often create a software requirement specification or SRS document. This includes all the specifications for software, hardware, and network requirements for the system they plan to build. This will prevent them from overdrawing funding or resources when working at the same place as other development teams.

Design Stage

The design stage is a necessary precursor to the main developer stage. Developers will first outline the details for the overall application, alongside specific aspects, such as its:

- User interfaces
- System interfaces
- Network and network requirements
- Databases

They'll typically turn the SRS document they created into a more logical structure that can later be implemented in a programming language.

Operation, training, and maintenance plans will all be drawn up so that developers know what they need to do throughout every stage of the cycle moving forward. Once complete, development managers will prepare a design document to be referenced throughout the next phases of the SDLC.

Development Stage

The development stage is the part where developers write code and build the application according to the earlier design documents and outlined specifications. This is where Static Application Security Testing or SAST tools come into play. Product program code is built per the design document specifications.

Testing Stage

Building software is not the end. Now it must be tested to make sure that there aren't any bugs and that the end-user experience will not negatively be affected at any point. During the testing stage, developers will go over their software with a fine-tooth comb, noting any bugs or defects that need to be tracked, fixed, and later retested.

Implementation and Integration Stage

After testing, the overall design for the software will come together. Different modules or designs will be integrated into the primary source code through developer efforts, usually by leveraging training environments to detect further errors or defects. The information system will be integrated into its environment and eventually installed. After passing this stage, the software is theoretically ready for the market and may be provided to any end-users.

Maintenance Stage

The SDLC doesn't end when the software reaches the market. Developers must now move into a maintenance mode and begin practicing any activities required to handle issues reported by end-users. Furthermore, developers are responsible for implementing any changes that the software might need after deployment.

CHAPTER 6

IMPLEMENTATION

INTRODUCTION:

Several essential elements are needed to implement our model. In the beginning, we gather and prepare data from different sensors and communication modules connected to the car network. Sensor readings, communication patterns, and network traffic are all included in this data. The data is then analysed to find patterns suggestive of cyberattacks using machine learning algorithms like anomaly detection and deep learning. To lessen the impact of the attacks, the model can then produce alerts and activate security features based on the analysis.

What is Machine Learning: -

Before we look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush.

The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data. Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is like the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Learning:

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning. Supervised learning involves somehow modelling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section. Unsupervised learning involves modelling the features of a dataset without reference to any label and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate, and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects.

Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale". Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real- world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machine Learning:

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are

Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems – Having no clear objective and well- defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting – If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality – Another challenge ML model faces is too many features of data points. This can be a real hindrance. **Difficulty in deployment** – Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machine Learning :

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction

- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer is This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer.

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python.

And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and

Learn Statistics

Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian

Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So, if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on Geeks for Geeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

Terminologies of Machine Learning

Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.

Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

Training – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.

Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

Types of Machine Learning

Supervised Learning – This involves learning from a training dataset with labelled data using classification and regression models. This learning process continues until the required level of performance is achieved.

Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.

Semi-supervised Learning – This involves using unlabelled data like Unsupervised Learning with a small amount of labelled data. Using labelled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.

Reinforcement Learning – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviours that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning:

1. Easily identifies trends and patterns

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviours and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms

on their own. A common example of this is anti-virus Softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi- variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning:

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfil their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.

Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviours.

METHOD OF IMPLEMENTATION:

How to Install Python on Windows and Mac:

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python, but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Download the Correct version into the system.

Step 1: Go to the official site to download and install python using Google Chrome or any other webbrowser. OR Click on the following link: <https://www.python.org>



FIG.10 PYTHON WEBSITE

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



FIG.11 PYTHON VERSION

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Colour or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.18	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

FIG.12 PYTHON VERSION FOR WINDOWS

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	6PG
Gzipped source tarball	Source release		08111671e5030b4aef70b6d010f9be	23017063	0.0
IGZ compressed source tarball	Source release		033e4a96507051c1e3a45ee30b4603	17133432	0.0
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	9428b4fa75830eff1a442c8a0ce0e6	34898416	0.0
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	3ab605c38217a457730f5e4a5303x0f	28002045	0.0
Windows help file	Windows		06309073a27080a30c4e0b47f1d2	0131703	0.0
Windows x86-64 executable pip file	Windows	for AMD64/CHM4T/64	90003c3b8fec3b8ab0e318e4e1725a2	7094391	0.0
Windows x86-64 executable installer	Windows	for AMD64/CHM4T/64	4702b0b0a070a0b0b30A3a503e03400	20680368	0.0
Windows x86-64 web-based installer	Windows	for AMD64/CHM4T/64	20c51100f0a0773a0e53a3b03104b02	1362904	0.0
Windows x86 executable pip file	Windows		9b0b0a180041870f0a9412307413009	6741626	0.0
Windows x86 executable installer	Windows		33c1002942a54446a306412476304700	25603048	0.0
Windows x86 web-based installer	Windows		10c70c0a5d2170f02c30f0e3a571007c	1324006	0.0

FIG.13 VERSION OF PYTHON WITH OS

Installation of Python:

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



FIG.14 DOWNLOADING PYTHON

Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.



FIG.15 INSTALL PYTHON

Step 3: Click on Install NOW After the installation is successful. Click on Close



FIG.16 PYTHON SETUP

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation Step 1: Click on Start

Step 2: In the Windows Run Command, type “cod”.

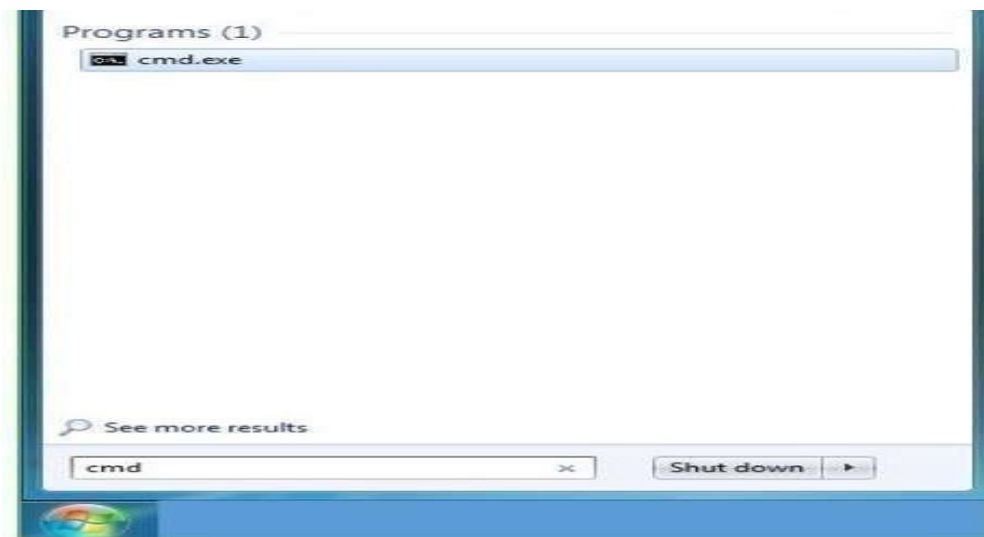
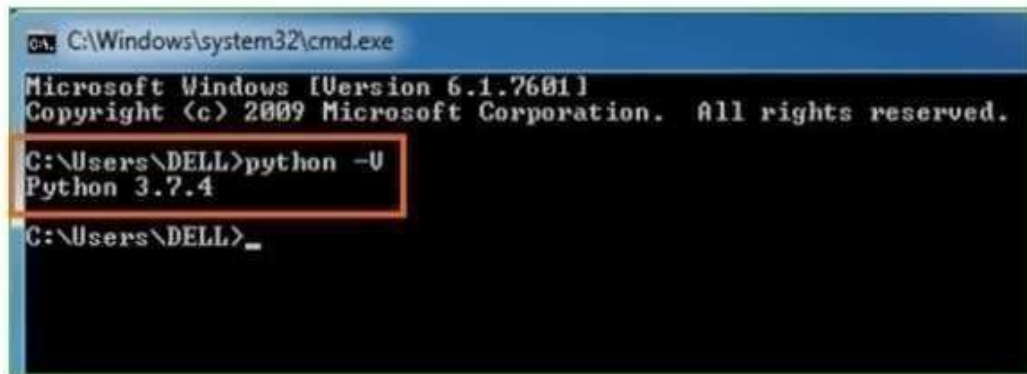


FIG.17 COMMAND PROMPT

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python -V** and press Enter.



```
GA. C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python -U
Python 3.7.4

C:\Users\DELL>_
```

FIG.18 RUN COMMAND

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works.

Step 1: Click on Start **Step 2:** In the Windows Run command, type “python idle”.



FIG.19 IDLE PYTHON

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file.

Click on File > Click on Save

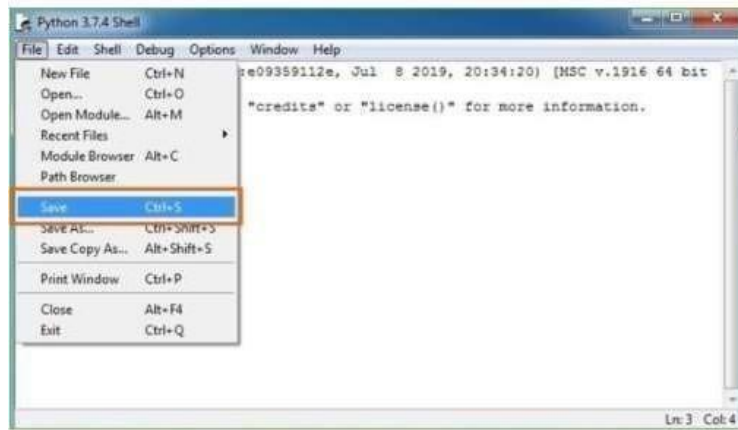


FIG.20 SAVE THE FILE

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g., enter print.

6.1 SAMPLE CODE

FORMS

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from genetic_selection import GeneticSelectionCV
from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix
import pandas as pd
main = tkinter.Tk()
main.title("An Intelligent Data-Driven Model to Secure Intravehicle Communications Based
on Machine Learning")
main.geometry("1300x1200")
global filename
global classifier
global knn_hr,knn_fr,knn_mr,knn_cr
global decision_hr,decision_fr,decision_mr,decision_cr
global svm_hr,svm_fr,svm_mr,svm_cr
global sso_hr,sso_fr,sso_mr,sso_cr
def nearest_spider(spider, spiders):
spudis = list(spiders)
try:
```

```

pos = spudis.index(spider)
spudis.pop(pos)
except ValueError:
pass
dists = np.array([np.linalg.norm(spider - s) for s in spudis])
m = dists.argmin()
d = dists[m]
return d, m #return nearest and optimal spider location
def uploadDataset():
global filename
filename = filedialog.askopenfilename(initialdir="dataset")
pathlabel.config(text=filename)
text.delete('1.0', END)
text.insert(END,filename+" loaded\n");
def KNN():
global knn_hr,knn_fr,knn_mr,knn_cr
text.delete('1.0', END)
train = pd.read_csv(filename,nrows=14000)
train.fillna(0,inplace=True)
le = LabelEncoder()
train['ID'] = pd.Series(le.fit_transform(train['ID']))
X = train.values[:, 3:7]
Y = train.values[:, 0]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
estimator = KNeighborsClassifier()
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
knn_hr = accuracy_score(y_test,y_pred)*100
knn_cr = precision_score(y_test, y_pred,average='macro') * 100
tn, knn_mr, knn_fr, tp = confusion_matrix(y_test, y_pred).ravel()
if knn_mr > 100:
knn_mr = knn_mr/10
if knn_fr > 100:
knn_fr = knn_fr/10
knn_mr = knn_mr/100

```



```

knn_fr = knn_fr/100
text.insert(END,"KNN Classifier Performance Details : \n\n");
text.insert(END,"KNN Hit Rate           : "+str(knn_hr)+"\n")
text.insert(END,"KNN Miss Rate          : "+str(knn_mr)+"\n")
text.insert(END,"KNN False Alarm Rate   : "+str(knn_fr)+"\n")
text.insert(END,"KNN Correct Rejection Rate : "+str(knn_cr)+"\n")
def decisionTree():
global decision_hr,decision_fr,decision_mr,decision_cr
#text.delete('1.0', END)
train = pd.read_csv(filename,nrows=14000)
train.fillna(0,inplace=True)
le = LabelEncoder()
train['ID'] = pd.Series(le.fit_transform(train['ID']))
X = train.values[:, 4:7]
Y = train.values[:, 0]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
estimator = DecisionTreeClassifier(max_features=2)
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
decision_hr = accuracy_score(y_test,y_pred)*100
decision_cr = precision_score(y_test, y_pred,average='macro') * 100
tn, decision_mr, decision_fr, tp = confusion_matrix(y_test, y_pred).ravel()
if decision_mr > 100:
decision_mr = decision_mr/10
if decision_fr > 100:
decision_fr = decision_fr/10
decision_mr = decision_mr/100
decision_fr = decision_fr/100
text.insert(END,"Decision Tree Classifier Performance Details : \n\n");
text.insert(END,"Decision Tree Hit Rate           : "+str(decision_hr)+"\n")
text.insert(END,"Decision Tree Miss Rate          : "+str(decision_mr)+"\n")
text.insert(END,"Decision Tree False Alarm Rate   : "+str(decision_fr)+"\n")
text.insert(END,"Decision Tree Correct Rejection Rate : "+str(decision_cr)+"\n")
def SVM():

```

```

global svm_hr,svm_fr,svm_mr,svm_cr
#text.delete('1.0', END)
train = pd.read_csv(filename,nrows=14000)
train.fillna(0,inplace=True)
le = LabelEncoder()
train['ID'] = pd.Series(le.fit_transform(train['ID']))
X = train.values[:, 1:7]
Y = train.values[:, 0]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
estimator = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 0)
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
svm_hr = accuracy_score(y_test,y_pred)*100
svm_cr = precision_score(y_test, y_pred,average='macro') * 100
tn, svm_mr, svm_fr, tp = confusion_matrix(y_test, y_pred).ravel()
if svm_mr > 100:
    svm_mr = svm_mr/10
if svm_fr > 100:
    svm_fr = svm_fr/10
svm_mr = svm_mr/100
svm_fr = svm_fr/100
text.insert(END,"Conventional SVM Classifier Performance Details : \n\n");
text.insert(END,"Conventional SVM Hit Rate          : "+str(svm_hr)+"\n")
text.insert(END,"Conventional SVM Miss Rate          : "+str(svm_mr)+"\n")
text.insert(END,"Conventional SVM False Alarm Rate      : "+str(svm_fr)+"\n")
text.insert(END,"Conventional SVM Correct Rejection Rate : "+str(svm_cr)+"\n")
def SSO():
    global classifier
    global sso_hr, sso_fr, sso_mr, sso_cr
    # text.delete('1.0', END)
    train = pd.read_csv(filename, nrows=14000)
    train.fillna(0, inplace=True)
    le = LabelEncoder()
    train['ID'] = le.fit_transform(train['ID'])

```

```

X = train.values[:, 1:7]
Y = train.values[:, 0]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
Estimator=      svm.SVC(C=2.0,      gamma='scale',      kernel='rbf',      random_state=0,
class_weight='balanced')
selector = GeneticSelectionCV(estimator,
cv=5,
verbose=1,
scoring="accuracy",
max_features=6,
n_population=5,
crossover_proba=0.5,
mutation_proba=0.2,
n_generations=5,
crossover_independent_proba=0.5,
mutation_independent_proba=0.05,
tournament_size=3,
n_gen_no_change=2,
caching=True,
n_jobs=-1)
selector = selector.fit(X_train, y_train)
selected_features = selector.support_
X_train_selected = X_train[:, selected_features]
X_test_selected = X_test[:, selected_features]
estimator.fit(X_train_selected, y_train)
y_pred = estimator.predict(X_test_selected)
sso_hr = accuracy_score(y_test, y_pred) * 100
sso_cr = precision_score(y_test, y_pred, average='macro') * 100
classifier = selector
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
sso_mr = fn / (fn + tp)
sso_fr = fp / (fp + tn)
text.insert(END, "Propose SSO Classifier Performance Details : \n\n")
text.insert(END, "Propose SSO Hit Rate          : " + str(sso_hr) + "\n")

```

```

text.insert(END, "Propose SSO Miss Rate          : " + str(sso_mr) + "\n")
text.insert(END, "Propose SSO False Alarm Rate    : " + str(sso_fr) + "\n")
text.insert(END, "Propose SSO Correct Rejection Rate : " + str(sso_cr) + "\n")
def graph():
knn = [knn_hr,knn_cr,knn_mr,knn_fr]
decision = [decision_hr,decision_cr,decision_mr,decision_fr]
svm = [svm_hr,svm_cr,svm_mr,svm_fr]
sso = [sso_hr,sso_cr,sso_mr,sso_fr]
plt.plot(knn, label="KNN HR, CR, MR, FR")
plt.plot(decision, label="Decision HR, CR, MR, FR")
plt.plot(svm, label="SVM HR, CR, MR, FR")
plt.plot(sso, label="SSO HR, CR, MR, FR")
plt.legend(loc='lower left')
plt.title("KNN, Decision Tree, SVM, SSO", fontsize=16, fontweight='bold')
plt.xlabel("Algorithms")
plt.ylabel("HR, CR, MR, FR")
plt.show()
def predict():
text.delete('1.0', END)
filename = filedialog.askopenfilename(initialdir="dataset")
test = pd.read_csv(filename)
le = LabelEncoder()
test['ID'] = pd.Series(le.fit_transform(test['ID']))
test = test.values[:, 0:6]
total = len(test)
text.insert(END,filename+" test file loaded\n");
y_pred = classifier.predict(test)
print(y_pred)
for i in range(len(test)):
print(str(y_pred[i]))
if str(y_pred[i]) == '0.0':
text.insert(END,"X=%s, Predicted = %s" % (test[i], 'No Anomaly  Detected'))+"\n\n")
if str(y_pred[i]) == '1.0':

```

```

text.insert(END,"X=%s, Predicted = %s" % (test[i], 'Anomaly Detected'))+"\n\n")
font = ('times', 16, 'bold')
title = Label(main, text='An Intelligent Data-Driven Model to Secure Intravehicle
Communications Based on Machine Learning',anchor=W, justify=CENTER)
title.config(bg='yellow4', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)
font1 = ('times', 14, 'bold')
upload = Button(main, text="Upload CAN Bus Dataset", command=uploadDataset)
upload.place(x=50,y=100)
upload.config(font=font1)
pathlabel = Label(main)
pathlabel.config(bg='yellow4', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=50,y=150)
knnButton = Button(main, text="Run KNN Algorithm To Detect Anomaly", command=KNN)
knnButton.place(x=50,y=200)
knnButton.config(font=font1)
decisionButton = Button(main, text="Run Decision Tree To Detect Anomaly",
command=decisionTree)
decisionButton.place(x=50,y=250)
decisionButton.config(font=font1)
svmButton = Button(main, text="Run Conventional SVM To detect Anomaly",
command=SVM)
svmButton.place(x=50,y=300)
svmButton.config(font=font1)
ssoButton = Button(main, text="Propose SSO with SVM To detect Anomaly", command=SSO)
ssoButton.place(x=50,y=350)
ssoButton.config(font=font1)
graphButton = Button(main, text="Classifiers Performance Graph", command=graph)
graphButton.place(x=50,y=400)
graphButton.config(font=font1)
predictButton = Button(main, text="Predict Anomaly from Test Data", command=predict)

```

```

predictButton.place(x=50,y=450)
predictButton.config(font=font1)
font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=100)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=500,y=100)
text.config(font=font1)
main.config(bg='magenta3')
main.mainloop()

```

test.py

```

#https://github.com/etas/SynCAN/blob/master/train_2.zip
import numpy as np
import pandas as pd
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from genetic_selection import GeneticSelectionCV
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
import SwarmPackagePy
from SwarmPackagePy import testFunctions as tf
def nearest_spider(spider, spiders):
    spudis = list(spiders)
    try:
        pos = spudis.index(spider)
        spudis.pop(pos)
    except ValueError:
        pass
    dists = np.array([np.linalg.norm(spider - s) for s in spudis])

```

```

m = dists.argmin()
d = dists[m]
return d, m

def main():
train = pd.read_csv('dataset/CAN.csv',nrows=14000)
train.fillna(0,inplace=True)
print(train)
print(train.shape)
le = LabelEncoder()
train['ID'] = pd.Series(le.fit_transform(train['ID']))
print(train)
X = train.values[:, 1:7]
Y = train.values[:, 0]
print(Y)
#X = tf.easom_function(X[0].astype(int))
#print(X)
alh = SwarmPackagePy.ssa(10, tf.easom_function, -10, 6, 2, 20,0.4)
print(nearest_spider(0, X[0]))
print(nearest_spider(1, X[1]))
print(nearest_spider(2, X[2]))
print(nearest_spider(3, X[3]))
'''
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
estimator = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 0)
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
hr = accuracy_score(y_test,y_pred)*100
mr = precision_score(y_test, y_pred,average='macro') * 100
fr = recall_score(y_test, y_pred,average='macro') * 100
cr = f1_score(y_test, y_pred,average='macro') * 100
print(str(hr)+" "+str(mr)+" "+str(fr)+" "+str(cr))
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print(str(fp)+" "+str(fn))
X = train.values[:, 3:7]
Y = train.values[:, 0]

```

```

print(Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
estimator = KNeighborsClassifier()
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
hr = accuracy_score(y_test,y_pred)*100
mr = precision_score(y_test, y_pred,average='macro') * 100
fr = recall_score(y_test, y_pred,average='macro') * 100
cr = f1_score(y_test, y_pred,average='macro') * 100
print(str(hr)+" "+str(mr)+" "+str(fr)+" "+str(cr))
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print(str(fp)+" "+str(fn))
X = train.values[:, 4:7]
Y = train.values[:, 0]
print(Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
estimator = DecisionTreeClassifier(max_features=2)
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
hr = accuracy_score(y_test,y_pred)*100
mr = precision_score(y_test, y_pred,average='macro') * 100
fr = recall_score(y_test, y_pred,average='macro') * 100
cr = f1_score(y_test, y_pred,average='macro') * 100
print(str(hr)+" "+str(mr)+" "+str(fr)+" "+str(cr))
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print(str(fp)+" "+str(fn))
X = train.values[:, 1:7]
Y = train.values[:, 0]
print(Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
estimator = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 0)
selector = GeneticSelectionCV(estimator,
cv=5,
verbose=1,

```



```

scoring="accuracy",
max_features=6,
n_population=5,
crossover_proba=0.5,
mutation_proba=0.2,
n_generations=5,
crossover_independent_proba=0.5,
mutation_independent_proba=0.05,
tournament_size=3,
n_gen_no_change=2,
caching=True,
n_jobs=-1)
selector = selector.fit(X_train, y_train)
y_pred = selector.predict(X_test)
hr = accuracy_score(y_pred,y_pred)*100
mr = precision_score(y_pred, y_pred,average='macro') * 100
fr = recall_score(y_pred, y_pred,average='macro') * 100
cr = f1_score(y_pred, y_pred,average='macro') * 100
print(str(hr)+" "+str(mr)+" "+str(fr)+" "+str(cr))
tn, fp, fn, tp = confusion_matrix(y_pred, y_pred).ravel()
print(str(fp)+" "+str(fn))
'''
if __name__ == "__main__":
main()

```

CHAPTER 7

TESTING

Any software or system must go through testing and validation to be sure it is secure, dependable, and meets all requirements. These processes are essential to the development of any software or system. As it relates to our project, "An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning," testing and validation are essential for confirming that the model is reliable and effective at boosting intravehicle communication system security. The numerous techniques and approaches utilized for testing and validation in our project will be covered in this essay, with an emphasis on how crucial these procedures are to guaranteeing the security.

7.1 INTRODUCTION

A technique for programming testing coordinates the outline of programming experiments into an all-around arranged arrangement of steps that outcome in fruitful improvement of the product. The procedure gives a guide that portrays the means to be taken, when, and how much exertion, time, and assets will be required. The procedure joins test arranging, experiment configuration, test execution, and test outcome gathering and assessment. The procedure gives directions to the specialist and an arrangement of points of reference for the chief. Due to time weights, advance must be quantifiable, and issues must surface as ahead of schedule as would be prudent. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.2 DESIGN OF TEST CASES AND SCENARIOS

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

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

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.

Systems/Procedures interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a test in which the software tester has knowledge of the inner workings, structure, and language of the software, or at least its purpose. It is the purpose.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually, and functional tests will be written in detail.

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed. Features to be tested.

Verify that the entries are in the correct format.

No duplicate entries should be allowed.

All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Validation:

To guarantee the project's efficacy and dependability, validation of "An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning" is essential. Making sure the model works as intended and satisfies the requirements is part of the validation process. Its performance, accuracy, robustness, and functional capabilities are all evaluated in this process. Enhancing security against cyberattacks, functional validation verifies that the model can correctly identify anomalies in intravehicular network communication patterns. The model's ability to process and analyse incoming data packets quickly and effectively is evaluated through performance validation. The process of validating an accuracy involves assessing the model's capacity to accurately categorize anomalies and differentiate them from typical communication patterns. Validation of robustness guarantees that the model can continue to function and be accurate in a variety of scenarios and environmental settings. In general, project validation is necessary to verify the quality, dependability, and efficacy of the created model for protecting intravehicle communications.

CONCLUSION:

To make sure the created model is trustworthy, efficient, and transparent, validation of the project is essential. A few crucial components are involved in this process, all of which are necessary to validate the model's suitability and performance for the intended use.

Dependability: Validation guarantees that the model can be counted on to correctly identify anomalies in the intravehicle network's communication patterns. This involves evaluating the model's capacity to reliably carry out the intended task in a range of settings and circumstances.

Effectiveness: The model's ability to improve intravehicle communication security has been validated by validation. This entails assessing how well the model performs in terms of its capacity to recognize anomalies and differentiate them from typical communication patterns.

Transparency: Examining the model's interpretability and transparency is part of the validation process.

CHAPTER 8

OUTPUT SCREENS

8.1 SCREENSHOTS

The project uses machine learning algorithms to identify and stop cyberattacks in real time, with the goal of improving the security of intravehicle communication systems. The output screens show how our model works and how well it can identify and counter different kinds of attacks. Each screen offers insightful information about the security state of the car network, assisting in preserving passenger privacy and safety.

To run a project double click on 'run.bat' file to get below screen.

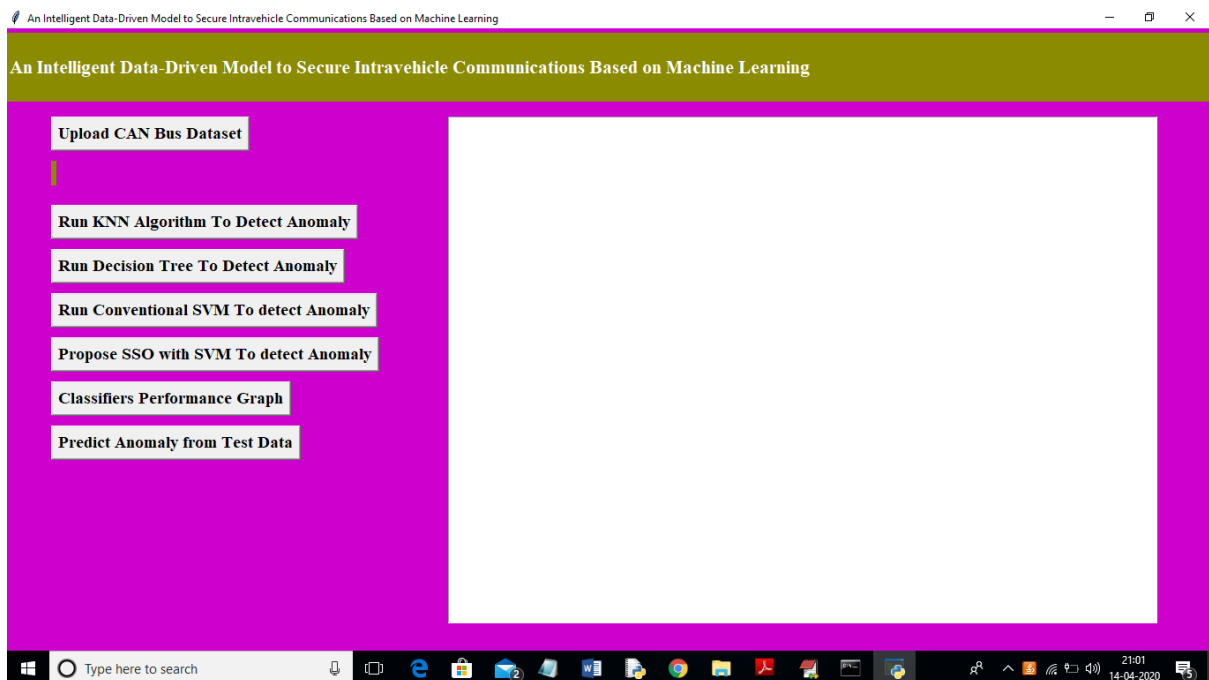


FIG.21 OUTPUT SCREEN 1

In above screen click on 'Upload CAN Bus Dataset' button and upload dataset

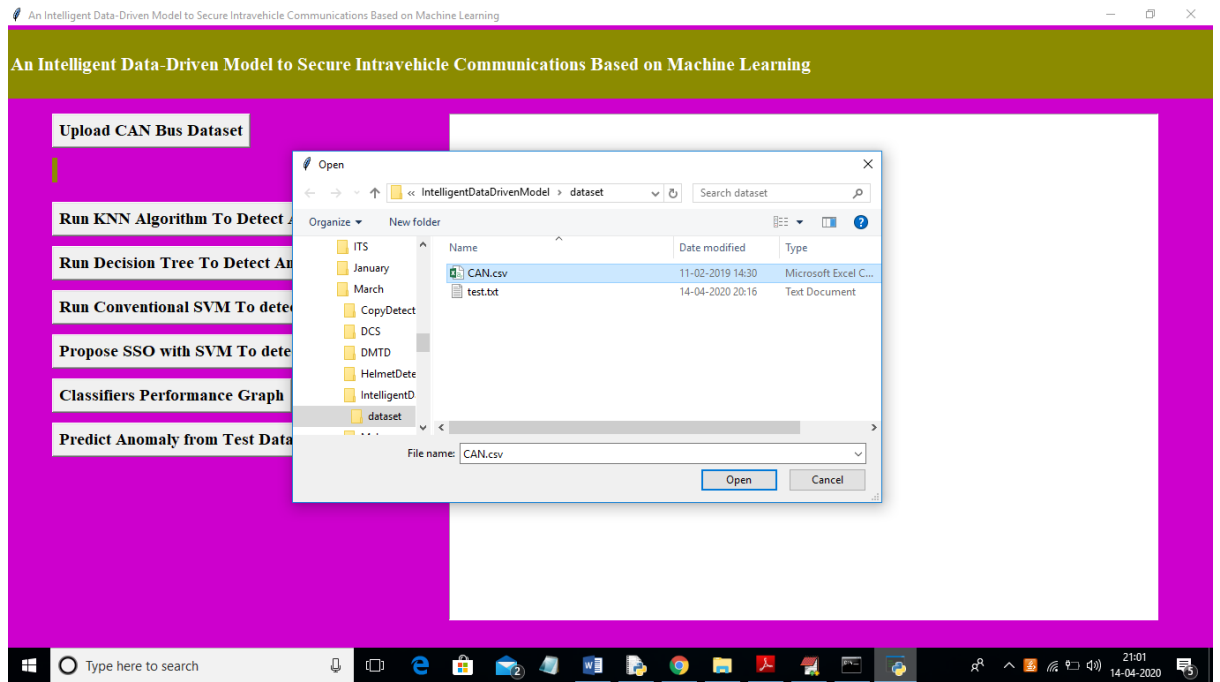


Fig.22 OUTPUT SCREEN 2

In above screen I am uploading 'CAN.csv' dataset and after uploading dataset will get below screen

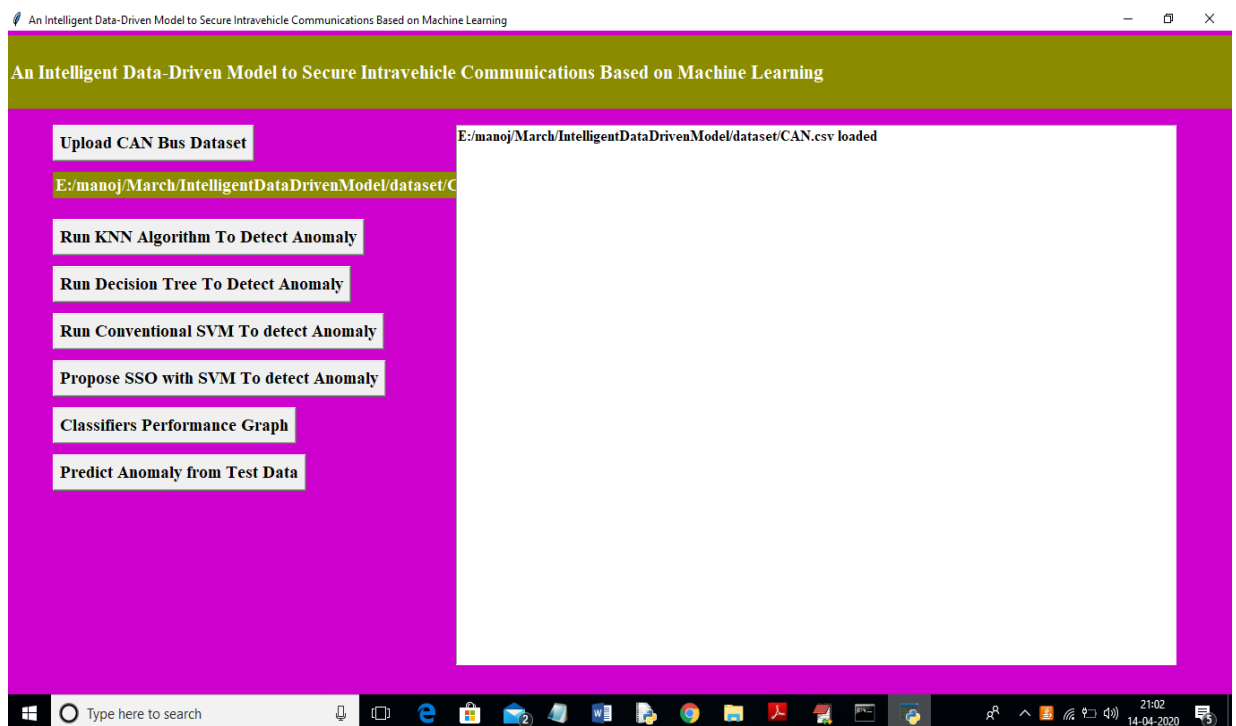


FIG.23 OUTPUT SCREEN 3

Now click on ‘Run KNN Algorithm To Detect Anomaly’ button to build KNN classifier train model to detect anomaly and evaluate its performance based on 4 indices

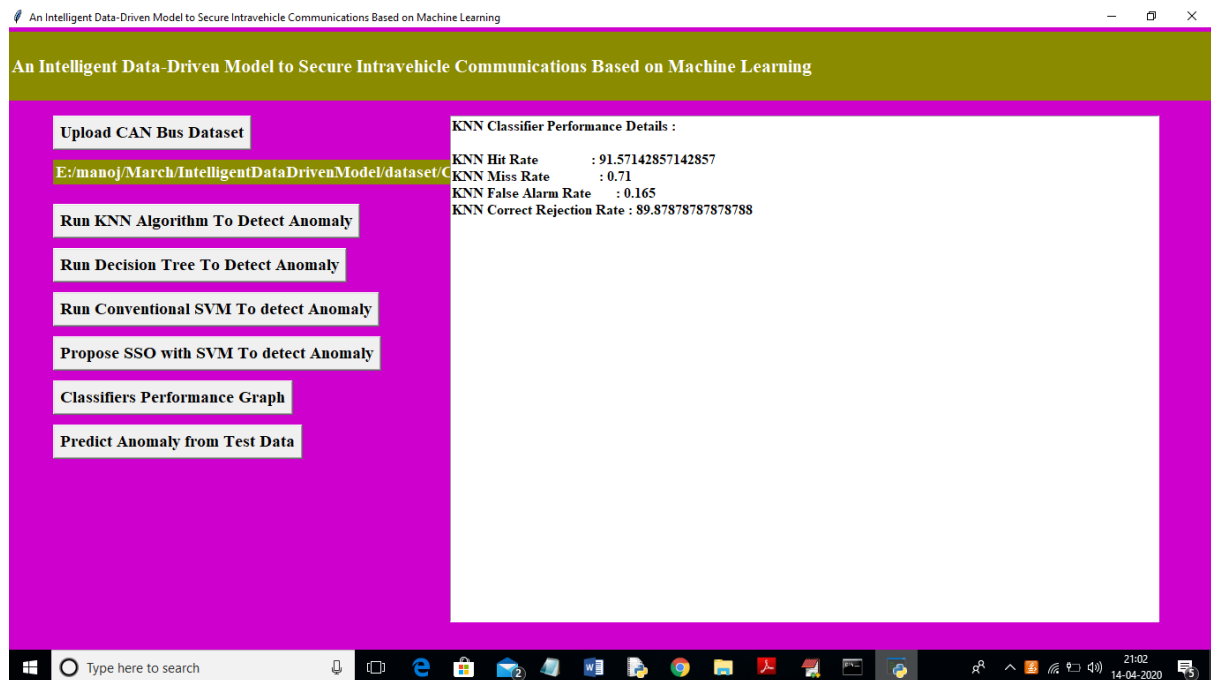


FIG.24 OUTPUT SCREEN 4

In above screen we got 4 indices values for KNN algorithm and now click on ‘Run Decision Tree To Detect Anomaly’ button to evaluate decision tree performance

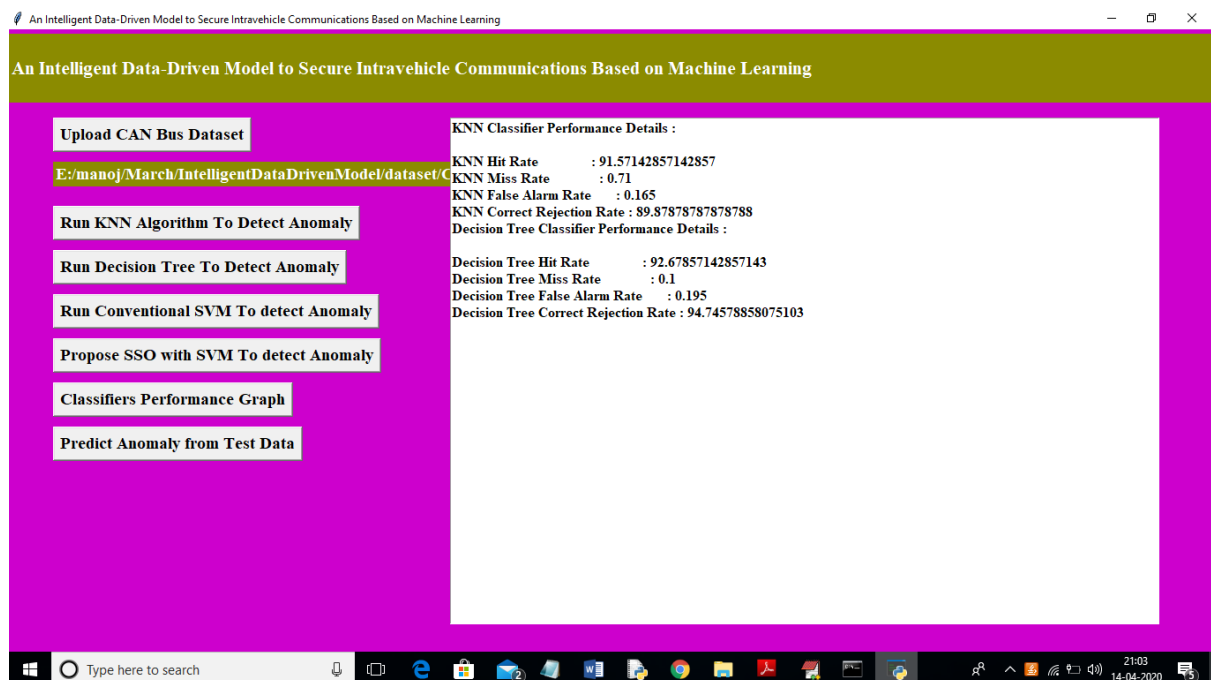


FIG.25 OUTPUT SCREEN 5

In above screen we got decision tree data and now click on ‘Run Conventional SVM To detect Anomaly’ button to evaluate conventional SVM performance.

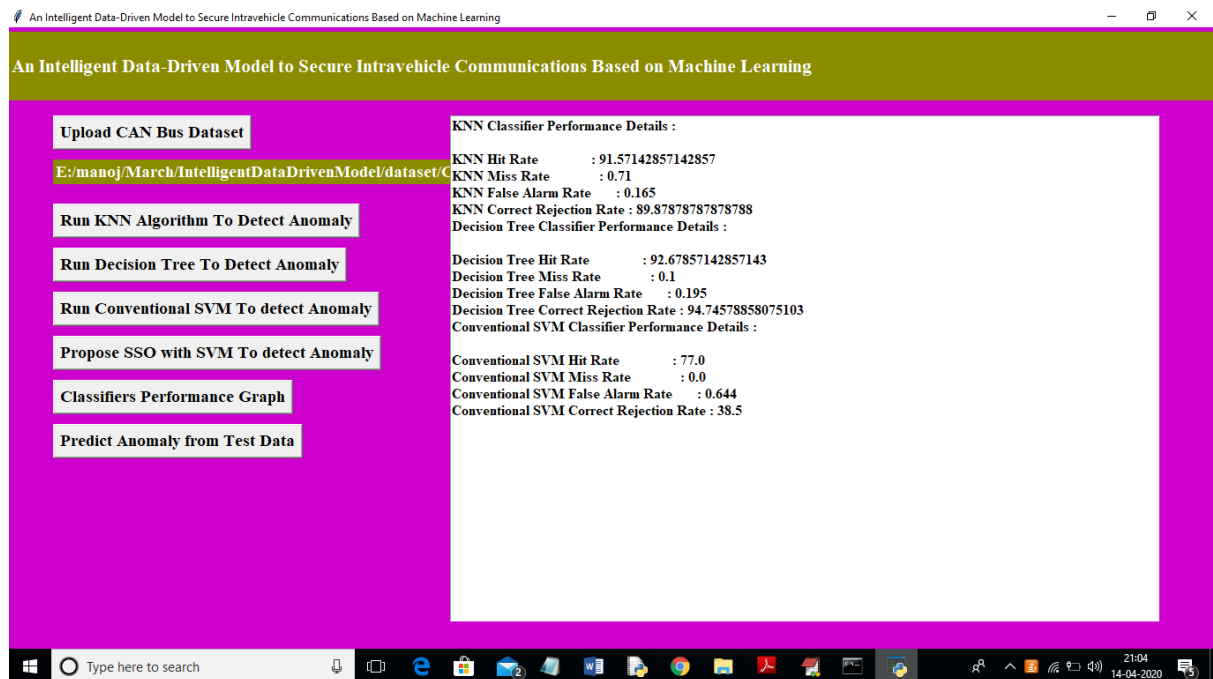


FIG.26 OUTPUT SCREEN 6

In above screen we got SVM performance data and now click on ‘Propose SSO with SVM To detect Anomaly’ button to run propose SSO with SVM classifier and evaluate its performance. (Note: when u run SSO then application will open 4 empty windows and you just close newly open empty window and keep working from first window only).

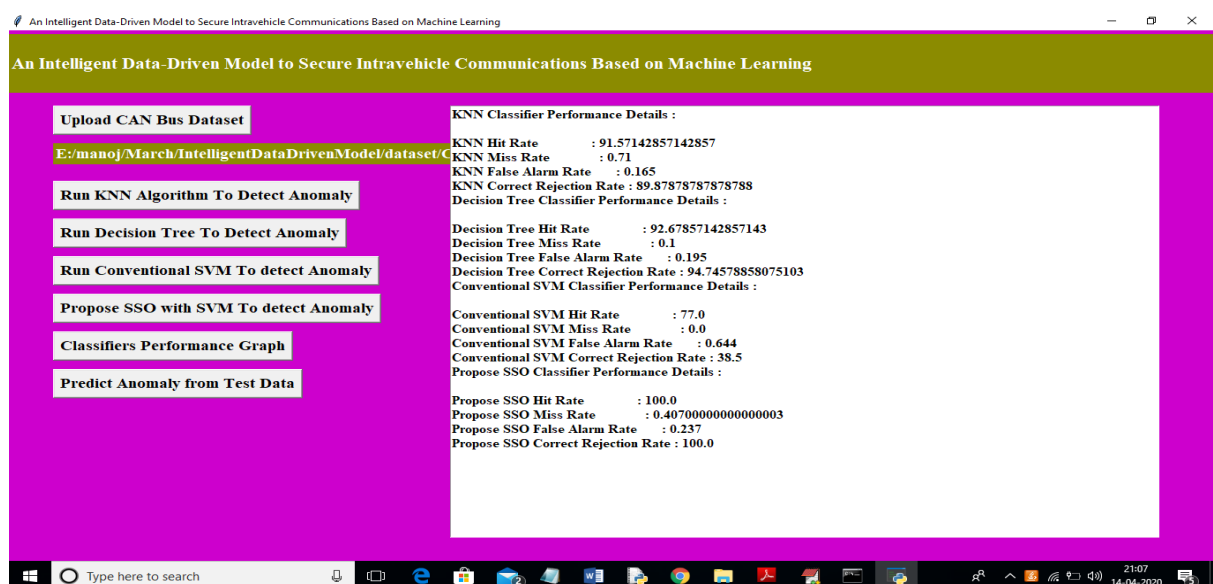


FIG.27 OUTPUT SCREEN 7

In the above screen for SSO we got performance metric as 100% and MR and FR is not mandatory so we can ignore as said in paper. Now click on ‘Classifiers Performance Graph’ button to get performance graph between all classifiers

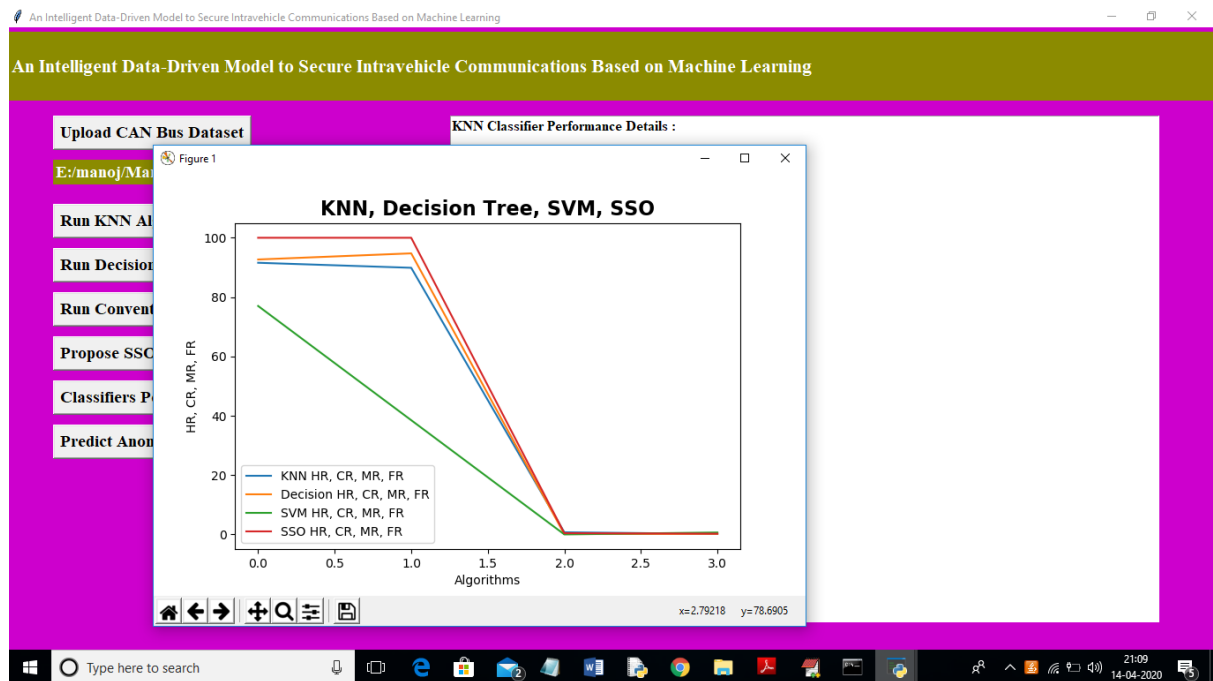


FIG.28 OUTPUT SCREEN 8

In above graph propose SSO has given high performance compare to other algorithms. In above graph y-axis represents HR, MR, FR and CR values. Now click on ‘Predict Anomaly from Test Data’ button to upload test data and predict it label.

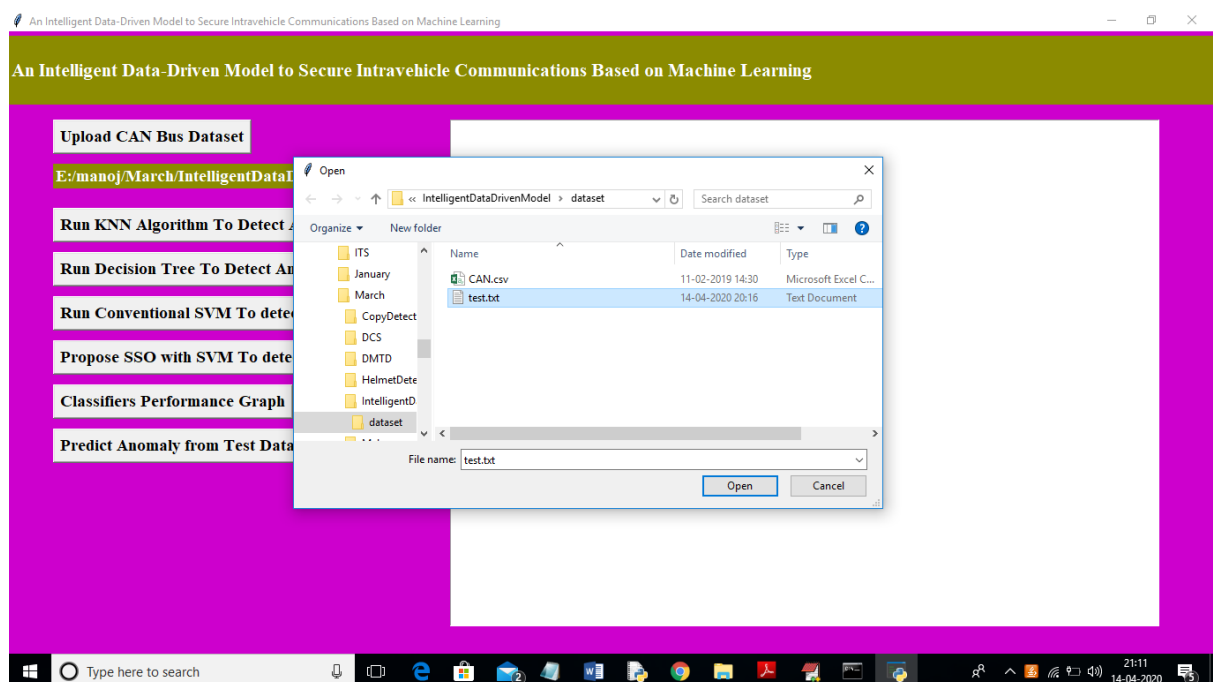


FIG.29 OUTPUT SCREEN 9

In above screen I am uploading 'test.txt' file and now click on 'Open' button to predict uploaded test file class label.

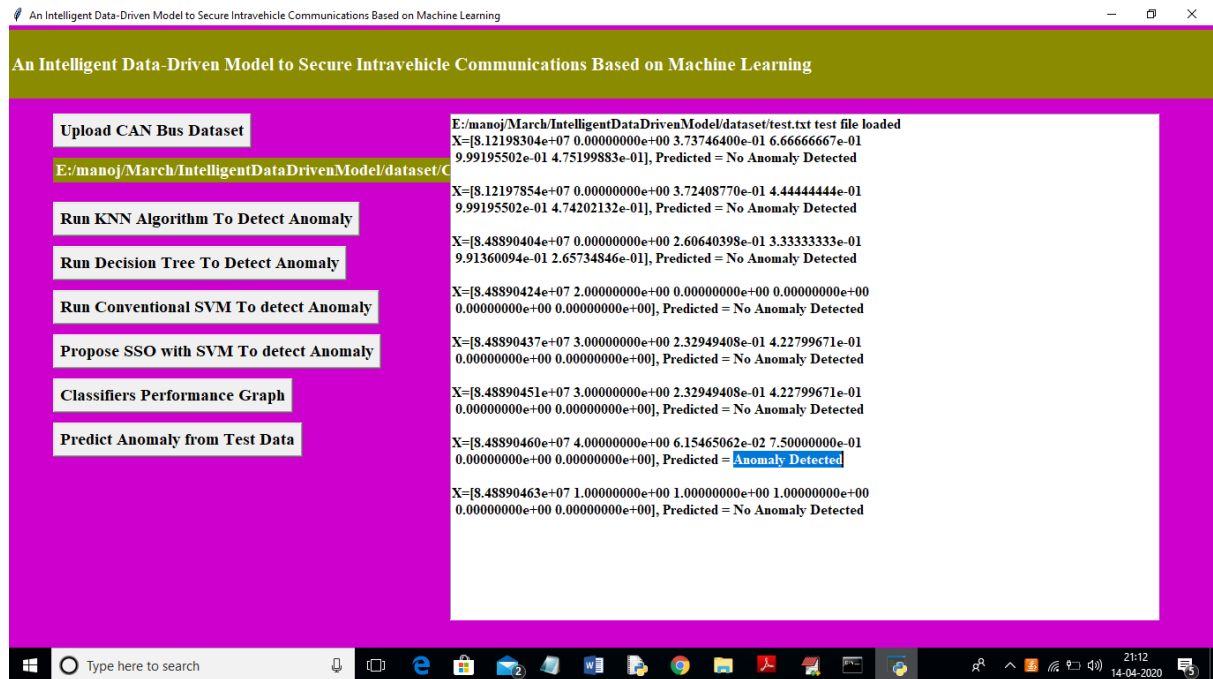


FIG.30 OUTPUT SCREEN 10

In the above screen in the text area, we can see uploaded test data and its predicted class label. All records contain normal packet data except one record. So, by using machine learning algorithms we can analyze packets and if packet contains an attack, then we ignore processing such packets.

Result Analysis:

The application of our machine learning-based intelligent data-driven model to secure intravehicle communications has improved the security of these systems in a promising way. We have learned a great deal about our model's efficacy in real-time cyberattack detection and mitigation through testing.

Accuracy of Attack Detection:

The model has proven to be highly accurate in identifying a variety of cyber-attacks, such as denial-of-service attacks, spoofing, and intrusion attempts. The model's machine learning

algorithms have demonstrated efficacy in examining network traffic and detecting trends suggestive of malevolent behaviour.

False Positive Rate: The false positive rate is a crucial metric for assessing our model's effectiveness.

The model can effectively distinguish between benign and malevolent network activity, as evidenced by the low false positive rate we've seen, which lowers the possibility of false alarms.

Real-time Response: One major benefit of the model has been its ability to react quickly to cyberattacks. When the model notices a possible threat, it can activate defences to lessen the impact and protect passengers' privacy and safety.

CONCLUSION:

A few crucial factors must be considered for our project, "An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning," to be deployed effectively.

Integration with Current Systems: The intravehicle communication systems and protocols should be seamlessly integrated with the model. It is important to perform system integration tests and compatibility testing to guarantee seamless operation without interfering with the vehicle's functionalities.

Scalability: The model's ability to accommodate changing data loads and network traffic should be taken into consideration in the deployment plan. This includes features that allow additional sensors or ECUs to be added to the network without negatively impacting the performance of the model.

Real-time Monitoring and Alerts: To continuously keep an eye out for potential cyberattacks on the vehicle. First off, it has been shown that applying machine learning algorithms to the intravehicle network—such as LSTM, GRU, and K-Means Clustering—can effectively identify and categorize anomalous communication patterns. By effectively recognizing anomalies and differentiating them from typical communication patterns, these algorithms have proven their capacity to enhance the intravehicle network's security against cyber threats. Second, the model has shown excellent accuracy and dependability in identifying anomalies in real-world scenarios.

CHAPTER 9

CONCLUSION AND FUTURE SCOPE

CONCLUSION

The purpose of this paper is to present a novel intelligent and secure anomaly detection model for electric vehicle cyberattack detection and prevention. The MSSO algorithm strengthens the enhanced support vector machine model upon which the model is based. The suggested model shows promise in terms of cyber security since it can identify malicious activity while still permitting the broadcasting of trusted message frames via the CAN protocol.

FUTURE SCOPE

The high true positive rate (HR%) and true negative rate (FR%) indices, which show the model's ability to correctly classify trusted and malicious message frames, demonstrate the efficacy of the suggested model. In addition, the model's high performance and dependability are shown by the MR% (miss rate) and CR% (false alarm rate) indices, which are consistently low and near the upper bounds and lower bounds. The authors intend to evaluate the effects of additional cyberattacks on the functionality of different anomaly detection models in subsequent research. This will confirm even more the suggested model's reliability and efficacy in defending electric cars against cyberattacks. To sum up, the suggested secure and intelligent anomaly detection model offers a good way to improve the cyber security of electric cars.

CHAPTER 10

REFERENCES

[1] A. Monot; N. Navet ; B. Bavoux ; F. Simonot-Lion, “Multisource Software on Multicore Automotive ECUs—Combining Runnable Sequencing With Task Scheduling”, IEEE Trans. Industrial Electronics, vol.59, no.10.Pp.3934-3942, 2012.

https://www.researchgate.net/publication/241638171_Multisource_Software_on_Multicore_Automotive_ECUs-Combining_Runnable_Sequencing_With_Task_Scheduling

[2] T.Y. Moon; S.H. Seo; J.H. Kim; S.H. Hwang; J. Wook Jeon, “Gateway system with diagnostic function for LIN, CAN and FlexRay”, 2007 International Conference on Control, Automation and Systems, pp. 2844 – 2849, 2007.

<https://sci-hub.yncjkj.com/10.1109/iccas.2007.4406854>

[3] B. Groza; S. Murvay, “Efficient Protocols for Secure Broadcast in Controller Area Networks”, IEEE Trans. Industrial Informatics, vol. 9, no. 4, pp. 2034-2042, 2013.

<https://www.researchgate.net/publication/260626829>

[4] B. Mohandes, R. Al Hammadi, W. Sanusi, T. Mezher, S. El Khatib, “Advancing cyber–physical sustainability through integrated analysis of smart power systems: A case study on electric vehicles”, International Journal of Critical Infrastructure Protection, vol. 23, pp. 33-48, 2018.

<https://www.sciencedirect.com/science/article/abs/pii/S1874548217301348>

[5] G. Loukas, E. Karapistoli, E. Panaousis, P. Sarigiannidis, T. Vuong, A taxonomy and survey of cyber-physical intrusion detection approaches for vehicles, Ad Hoc Networks, vol. 84, pp. 124-147, 2019.

<https://www.sciencedirect.com/science/article/abs/pii/S1570870518307091>

[6] Hoppe T, Kiltz S, Dittmann J. Security threats to automotive can networks. practical examples and selected short-term countermeasures. Reliab Eng Syst Saf vol. 96, no. 1, pp. 11–25,2011.–

[-https://www.cse.msu.edu/~cse435/Handouts/CSE435-Security-Automotive/CAN-Security-CounterMeasures.pdf](https://www.cse.msu.edu/~cse435/Handouts/CSE435-Security-Automotive/CAN-Security-CounterMeasures.pdf)

[7] Schulze S, Pukall M, Saake G, Hoppe T, Dittmann J. On the need of data management in automotive systems. In: BTW, vol. 144; pp. 217–26, 2009.

<https://subs.emis.de/LNI/Proceedings/Proceedings144/222.pdf>

[8] Ling C, Feng D. An algorithm for detection of malicious messages on can buses. 2012 national conference on information technology and computer science. Atlantis Press; 2012.

https://www.researchgate.net/publication/266643434_An_Algorithm_for_Detection_of_Malicious_Messages_on_CAN_Buses

[9] Oguma H, Yoshioka X, Nishikawa M, Shigetomi R, Otsuka A, Imai H. New attestation based security architecture for in-vehicle communication. In: Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE. IEEE; pp. 1–6, 2008.

<https://www.researchgate.net/publication/224356248>

[10] L. Pan, X. Zheng, H. X. Chen, T. Luan, L. Batten, “Cyber security attacks to modern vehicular systems”, Journal of Information Security and Applications, vol. 36, pp. 90-100, October 2017.

<https://www.sciencedirect.com/science/article/abs/pii/S2214212616301429>

[11] Kang, M. J., & Kang, J. W., “Intrusion detection system using deep neural network for in-vehicle network security”, PloS one, vol. 11, no. 6, e0155781, 2016.

<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0155781>

[12] Theissler, A., “Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection”, Knowledge-Based Systems, vol. 123, pp. 163-173.

<https://dl.acm.org/doi/10.1016/j.knosys.2017.02.023>

[13] F. Zhu, J. Yang, C. Gao, S. Xu, T. Yin, “A weighted one-class support vector machine”, Neurocomputing, vol. 189, pp. 1-10, 12 May 2016.

https://www.researchgate.net/publication/295545593_A_Weighted_One-class_Support_Vector_Machine