Detection and Removal of Class Imbalance from VANET Traffic

A Project Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNVERSITY ANANTAPUR.

In Partial Fulfillment of the Requirements for the Award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
BY

M. ABHIRAM	**(18121A05F8)**
M. MAHESH CHOWDARY	**(18121A05E5)**
M. ATHAULLA	**(18121A05F5)**
N. HANEESHA	**(18121A05G3)**
R. LIKHITH KUMAR	**(18121A05K2)**

Under the Guidance of

Dr. S. Sreenivasa Chakravarthi, M.Tech
Associate Professor
Dept. of CSE, SVEC

Department of Computer Science and Engineering
SREE VIDYANIKETHAN ENGINEERING COLLEGE
(Affiliated to JNTUA, Anantapuramu)
Sree Sainath Nagar, Tirupathi – 517 102

2018-2022

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Affiliated to Jawaharlal Nehru Technological University Anantapur)
Sree Sainath Nagar, A. Rangampet, Tirupati – 517 102, Chittoor Dist., A.P.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project Work entitled

"Detection and Removal of Class Imbalance from VANET Traffic "

is the bonafide work done by

M. ABHIRAM	**(18121A05F8)**
M. MAHESH CHOWDARY	**(18121A05E5)**
M. ATHAULLA	**(18121A05F5)**
N. HANEESHA	**(18121A05G3)**
R. LIKHITH KUMAR	**(18121A05K2)**

In the Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College, A. Rangampet. is affiliated to JNTUA, Anantapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering during 2018-2022.
This is work has been carried out under my guidance and supervision.

The results embodied in this Project report have not been submitted in any University or Organization for the award of any degree or diploma.

Internal Guide

Dr. S. Sreenivasa Chakravarthi
Associate Professor
Dept. of CSE
Sree Vidyanikethan Engineering College
Tirupathi

Head

Dr.Narendra Kumar Rao
Prof & Head
Dept. of CSE
Sree Vidyanikethan Engineering College
Tirupathi

INTERNAL EXAMINER

EXTERNAL EXAMINER

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION AND MISSION

VISION

To become a Centre of Excellence in Computer Science and Engineering by imparting high quality education through teaching, training and research.

MISSION

The Department of Computer Science and Engineering is established to provide undergraduate and graduate education in the field of Computer Science and Engineering to students with diverse background in foundations of software and hardware through a broad curriculum and strongly focused on developing advanced knowledge to become future leaders.

Create knowledge of advanced concepts, innovative technologies and develop research aptitude for contributing to the needs of industry and society.

Develop professional and soft skills for improved knowledge and employability of students.

Encourage students to engage in life-long learning to create awareness of the contemporary developments in computer science and engineering to become outstanding professionals.

Develop attitude for ethical and social responsibilities in professional practice at regional, National and International levels.

Program Educational Objectives (PEO's)

1. Pursuing higher studies in Computer Science and Engineering and related disciplines

2. Employed in reputed Computer and I.T organizations and Government or have established startup companies.

3. Able to demonstrate effective communication, engage in team work, exhibit leadership skills, ethical attitude, and achieve professional advancement through continuing education

Program Specific Outcomes (PSO's)

1. Demonstrate knowledge in Data structures and Algorithms, Operating Systems, Database Systems, Software Engineering, Programming Languages, Digital systems, Theoretical Computer Science, and Computer Networks. (PO1)

2. Analyze complex engineering problems and identify algorithms for providing solutions (PO2)

3. Provide solutions for complex engineering problems by analysis, interpretation of data, and development of algorithms to meet the desired needs of industry and society. (PO3, PO4)

4. Select and Apply appropriate techniques and tools to complex engineering problems in the domain of computer software and computer based systems (PO5)

Program Outcomes (PO's)

1. Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems (**Engineering knowledge**).

2. Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences (**Problem analysis**).

3. Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations (**Design/development of solutions**).

4. Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions (**Conduct investigations of complex problems**).

5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations (**Modern tool usage**)

6. Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice (**The engineer and society**)

7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development (**Environment and sustainability**).

8. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice (**Ethics**).

9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings (**Individual and team work**).

10. Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions (**Communication**).

11. Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments (**Project management and finance**).

12. Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change (**Life-long learning**).

Course Outcomes

CO1. Knowledge on the project topic (PO1)

CO2. Analytical ability exercised in the project work.(PO2)

CO3. Design skills applied on the project topic. (PO3)

CO4. Ability to investigate and solve complex engineering problems faced during the project work. (PO4)

CO5. Ability to apply tools and techniques to complex engineering activities with an understanding of limitations in the project work. (PO5)

CO6. Ability to provide solutions as per societal needs with consideration to health, safety, legal and cultural issues considered in the project work. (PO6)

CO7. Understanding of the impact of the professional engineering solutions in environmental context and need for sustainable development experienced during the project work. (PO7)

CO8. Ability to apply ethics and norms of the engineering practice as applied in the project work.(PO8)

CO9. Ability to function effectively as an individual as experienced during the project work. (PO9)

CO10. Ability to present views cogently and precisely on the project work. (PO10)

CO11. Project management skills as applied in the project work. (PO11)

CO12. Ability to engage in life-long leaning as experience during the project work. (PO12)

CO-PO Mapping

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3	PSO 4
CO1	3												3			
CO2		3												3		
CO3			3												3	
CO4				3											3	
CO5					3											3
CO6						3										
CO7							3									
CO8								3								
CO9									3							
CO10										3						
CO11											3					
CO12												3				

DECLARATION

We hereby declare that this project report titled **"Detection and Removal of Class Imbalance from VANET Traffic"** is a genuine project work carried out by us, in **B.Tech *(Computer Science and Engineering)*** degree course of **Jawaharlal Nehru Technological University Anantapur** and has not been submitted to any other course or University for the award of any degree by us.

Signature of the students

ACKNOWLEDGEMENT

We are extremely thankful to our beloved Chairman and founder **Dr. M. Mohan Babu** who took keen interest to provide us the infrastructural facilities for carrying out the project work.

We are highly indebted to **Dr. B. M. Satish,** Principal of Sree Vidyanikethan Engineering College for his valuable support and guidance in all academic matters.

We are very much obliged to **Dr. S. Sreenivasa Chakravarthi,** Associate Professor, Department of CSE, for providing us the guidance and encouragement in completion of this project.

We would like to express our indebtedness to the project coordinator,**Dr**. **K. Kannan**, Professor, Department of CSE for his valuable guidance during the course of project work.

We would like to express our deep sense of gratitude to **Dr. B. Narendra Kumar Rao**, Professor & Head, Department of CSE, for the constant support and invaluable guidance provided for the successful completion of the project.

We are also thankful to all the faculty members of CSE Department, whohave cooperated in carrying out our project. We would like to thank our parents and friends who have extended their help and encouragement either directly or indirectly in completion of our project work.

1. M. ABHIRAM
2. M. MAHESH CHOWDARY
3. M. ATHAULLA
4. N. HANEESHA
5. R. LIKHITH KUMAR

ABSTRACT

The intrinsic features of Internet networks lead to imbalanced class distributions when datasets are conformed, phenomena called Class Imbalance. This class imbalance is also present in network traffic coming from VANETs. We propose the application of a novel Difficult Set Sampling Technique (DSSTE) algorithm to tackle the class imbalance problem. First, use the Edited Nearest Neighbor (ENN) algorithm to divide the imbalanced training set into the difficult set and the easy set. Next, use the K-Means algorithm to compress the majority samples in the difficult set to reduce the majority. Zoom in and out the minority samples continuous attributes in the difficult set synthesize new samples to increase the minority number. Finally, the easy set, the compressed set of majority in the difficult, and the minority in the difficult set are combined with its augmentation samples to make up a new training set. The algorithm reduces the imbalance of the original training set and provides targeted data augment for the minority class that needs to learn. It enables the classifier to learn the differences in the training stage better and improve classification performance. To verify the proposed method, we conduct experiments on the Vehicular Reference Misbehavior (VeReMi) dataset, a dataset for the evaluation of misbehavior detection mechanisms for VANETs. This dataset consists of message logs of on-board units, including a labeled ground truth, generated from a simulation environment. The dataset also includes malicious messages intended to trigger incorrect application behavior.

KEYWORDS: Class Imbalance, data resampling, Vehicular ad-hoc networks, Classification

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

With the enormous traffic that is being generated there is a huge need of classification of this complex data for better understanding and with this arises the class imbalance problem. The traffic that is received is complex and diverse and the classification models applied on this type of traffic often tend to leave out the minute differences which leads to class imbalance problem. This problem exists in all kinds of traffic in real world and the one area that we are dealing. With is class imbalance in Vehicular Reference Misbehavior (VeReMi) dataset generated from a simulation environment dataset for the evaluation of misbehavior detection mechanisms for VANETs. This dataset consists of message logs of on- board units, including a labeled ground truth, The dataset also includes malicious messages intended to trigger incorrect application behavior. generally this problem is tackled with SMOTE algorithm but the efficiency of this lacks compared to novel DSSTE algorithm that is proposed andused here.

We use VeReMi dataset and conduct detailed analysis and data cleaning Use the Edited Nearest Neighbor (ENN) algorithm to divide the imbalanced training set into the difficult set and the easy set. Novel DSSTE algorithm used for reducing the majority samples and augmenting the minority samples in the difficult set, tackling the class imbalance problem in intrusion detection so that the classifier learns the differences better in training. we propose the Difficult Set Sampling Technique (DSSTE) algorithm to compress the majority samples and augment the number of minority samples in difficult samples, reducing imbalance in the training set to achieve better classification accuracy.

1.2 STATEMENT OF THE PROBLEM

Developing an Algorithm for Detection and Removal of Class Imbalance. The previous datasets with class imbalance problem use SMOTE and, in our work, we are developing a framework DSSTE (Difficult Set Sampling Technique) that maximizes minority set and minimizes majority set. This is performed on VANET Traffic.

1.3 OBJECTIVE

1. Converting imbalance to balanced network
2. To observe the accuracy of the DSSTE algorithm
3. Data Cleaning and preprocessing of VeReMi dataset
4. To design the framework in such a way that classes are balanced properly.

1.4 SCOPE

Class Imbalance in VeReMi dataset is a problem where unequal distribution of attack types are present. We try to remove theimbalance and decrease the difference to classify attack and non- attack types.

1.5 LIMITATIONS

Malfunctioning of sensors can pose a threat to vehicular grid in vehicular ad- hoc networks. In such situations, the entire grid may break down.

CHAPTER 2

LITERATURE SURVEY

Piyasak proposed a method to improve the accuracy of minority classification. This method combines the Synthetic Minority Over-sampling Technique (SMOTE) and Complementary Neural Network (CMTNN) to solve imbalanced data classification. Experiments on the UCI dataset show that the proposed combination technique can improve class imbalance problems. Yan proposed an improved local adaptive composite minority sampling algorithm (LA-SMOTE) to deal with the network traffic imbalance problem and then based on the deep learning GRU neural network to detect the network traffic anomaly [3].

Abdul- hammed et al. deal with the imbalanced dataset CIDDS- 001 using data Up-sampling and Down-sampling methods, and by Deep Neural Networks, Random Forest, Voting, Variational Auto encoder, and Stacking Machine Learning classifiers to evaluate datasets [10]. In their proposed method, the accuracy can reach 99.99%.

Recently, Chuang and Wu trained the depth automatic encoder to establish a data generation model to generate reasonable data needed to form a balanced dataset. His experiments show that the generation of balanced datasets helps to deal with the problem of over fitting caused by imbalanced data, and it can prevent the training model from misjudging new data types, including those not in the training dataset[11].

Bedi et al. proposed a new type of IDS based on Siamese Neural Network (Siamese-NN), the proposed Siam-IDS can detect R2L and U2R attacks without using traditional class balancing techniques, such as over-sampling and random under-sampling. The performance of Siam-IDS was com- pared

with Deep Neural Network (DNN) and CNN, Siam-IDS can achieve a higher recall value for R2L and U2R attack categories compared with similar products [12].

David Cieslak proposed a model for effective evaluation and comparison of sampling methods, including oversampling by replication, SMOTE (Synthetic Minority Over-sampling Technique), and under sampling, on real-world intrusion datasets; comparisons with RIPPER's loss ratio implementation for re-weighting the costs of false positives vs. false negatives and a clustering-based implementation of SMOTE (Cluster-SMOTE) that further improves the performance over all the sampling methods [2].

To counter imbalance in data, he implemented a combination of oversampling (both by replication and synthetic generation) and under sampling techniques. Cieslak also proposed a clustering based methodology for oversampling by generating synthetic instances. Where he evaluate approaches on two intrusion datasets destination and actual packets based — constructed from actual Notre Dame traffic, giving a flavor of real- world data with its idiosyncrasies.

Novel inverse random under sampling (IRUS) method is proposed by Atif Tahir for the class imbalance problem. The main idea is to severely under sample the majority class thus creating a large number of distinct training sets. For each training set we then find a decision boundary which separates the minority class from the majority class. By combining the multiple designs through fusion, we construct a composite boundary between the majority class and the minority class. The proposed methodology is applied on 22 UCI data sets and experimental results indicate a significant increase in performance when compared with many existing class-imbalance learning methods. We also present promising results for multi-label classification, a challenging research problem in many modern applications such as music, text and image categorization [6].

The proposed model by P. Sharma focuses on the deep learning-based

intrusion detection schemes and puts forward an in-depth survey and classification of these schemes. It first presents the primary background concepts about IDS architecture and various deep learning techniques.
It then classifies these schemes according to the type of deep learning methods utilized in each of them. It describes how deep learning networks are utilized in the intrusion detection process to recognize intrusions accurately [23].

Alexander Liu introduced an easily implementable resampling technique (generative oversampling) which creates new data points by learning from available training data. Empirically, he demonstrated that generative oversampling outperforms other well-known resampling methods on several datasets in the example domain of text classification [7].

Existing IDMs cannot solve the imbalance of traffic distribution, while ignoring the temporal relationship within traffic, which result in the reduction of the detection performance of the IDM and increase the false alarm rate, especially for low-frequency attacks. Binghao Yan proposed a new model for combined IDM called LA-GRU based on a novel imbalanced learning method and gated recurrent unit (GRU) neural network [9].

In the proposed model, a modified local adaptive synthetic minority oversampling technique (LA-SMOTE) algorithm is provided to handle imbalanced traffic, and then the GRU neural network based on deep learning theory is used to implement the anomaly detection of traffic. The experimental results evaluated on the NSL-KDD dataset confirm that, compared with the existing state-of-the-art IDMs, the proposed model not only obtains excellent overall detection performance with a low false alarm rate but also more effectively solves the learning problem of imbalanced traffic distribution.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Hardware Requirements:

Operating Systems: Windows 10
Processors: Any Intel or AMD x86-64
RAM: 52 GB
Harddisk or SSD: More than 500 GB
Camera: Internal or external

3.2 Software Requirements:

Languages: Python 3.6 or high version.
Packages: NumPy, Pandas, Matplotlib**.**

CHAPTER 4

ANALYSIS

4.1 EXISTING SYSTEM

In the field of machine learning, the problem of category imbalance has always been a challenge. Therefore, classification also faces enormous challenges in network traffic with extremely imbalanced categories. Therefore, many scholars have begun to study how to improve the accuracy of imbalanced network traffic data.

Piyasak proposed a method to improve the accuracy of minority classification. This method combines the Synthetic Minority Over-sampling Technique (SMOTE) and Complementary Neural Network (CMTNN) to solve imbalanced data classification. Experiments on the UCI dataset show that the proposed combination technique can improve class imbalance problems [3].

Yan proposed an improved local adaptive composite minority sampling algorithm (LA-SMOTE) to deal with the network traffic imbalance problem and then based on the deep learning GRU neural network to detect the network traffic anomaly [9].

Most scholars use interpolation, oversampling, encoder synthesis data, and other data augmentation methods, balance the training set, and achieve better experimental performance results. Although their method synthetic close to real data and effectively expand the minority class, the test data distribution may exceed the range. The Existing System has performed on NSL-KDD dataset and CSECIC-IDS2018 dataset and combines the Synthetic Minority Over-Sampling Technique (SMOTE) and Complementary Neural Network (CMTNN) to solve imbalanced data classification.

Veremi dataset consists of several folders each corresponding to different combinations of traffic densities, attack densities, attacker types and repetitions. Each folder consists of a single ground truth and several json logs as shown in fig 4.1

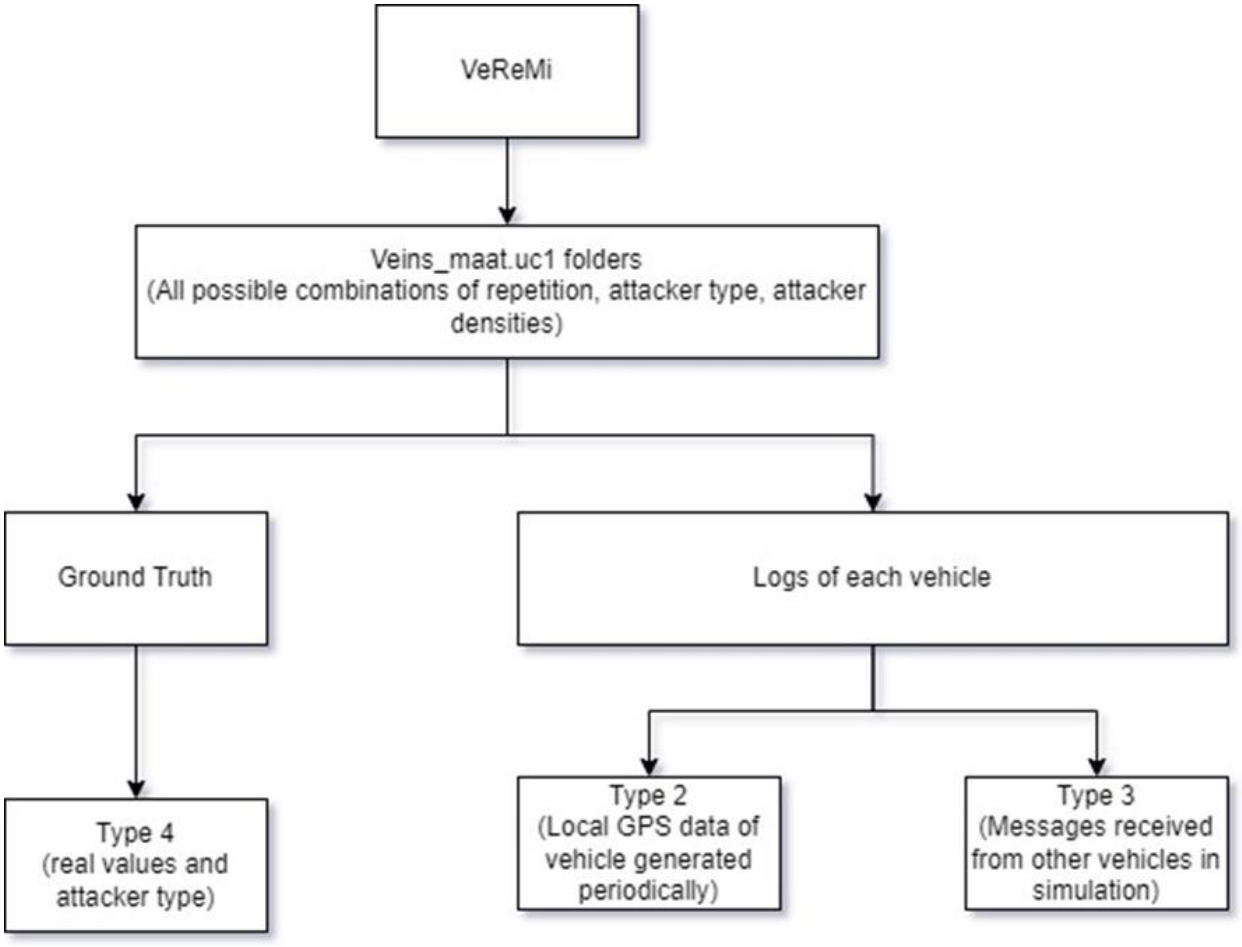


fig 4.1 VeReMi Folder Hierarchy

There is a json log for every vehicle in the simulation (JSONlog-0-7-A0 indicates log of 0th vehicle with OMNeT++ module ID 7. A0 indicates that this vehicle is not an attacker). Json logs consists of two types of data(type 2 and type 3). Type 2 data consists of logs generated periodically by

onboard GPS units. It has different parameters like receive time speed, position, noise. Type 3 data consists of BSM messages received from other vehicles through DSRC in the simulation. Type 3 data additionally has sent time, senderID, messageID and RSSI values.

In ground truth file, there is an entry for all the type 3 logs of all vehicles. The corresponding entry in ground truth consists of attacker type(attacker type 0 indicates the message is not malicious), original speed and positions(if attack type is non zero). There are five different types of attacks. They are Constant, Constant offset, Random, Random and Eventual stop.

Accurate detection of different types of attacks in VANET [36] is a challenging task. Most intrusion detection approaches can be divided into node-centric detection, where the detection of misbehaviour depends on the credibility of the sender or data-centric detection, where detection is based on contents of the message [21]. In [22], the authors introduced a framework called Maat that uses subjective logic to build a fusion and data management system to determine the trustworthiness of data. s. In [23], the authors proposed integrating plausibility checks and a machine learning framework for misbehaviour detection using the sender-receiver pair approach in the VeReMi dataset. They added six features, including two plausibility checks capable of detecting fake location and four quantitative metrics used to describe vehicle’s behaviour in the network. In paper [35], authors proposed Intrusion Detection System which can efficiently identify a fake information attack using statistical techniques, as well as other forms of attacks without relying on trust or reputation scores legitimate vehicles and misbehaving nodes, using supervised classification algorithms [30], such as K-Nearest Neighbour algorithm [31], Decision Tree algorithm [32], Random Forest algorithm [33], and Naïve Bayes classification algorithm [34].

4.2 PROPOSED SYSTEM

This work proposes a novel DSSTE algorithm, reducing the majority samples and augmenting the minority samples in the difficult set, tackling the class imbalance problem in intrusion detection so that the classifier learns the differences better in training.

Preprocessing the VeReMi dataset and removing the null and unwanted features and we scale the dataset. The important parameters like attackertype, spdX, spdY, posX, posY, RSSI are taken into consideration and after this step we divide the scaled dataset into training and test set and perform the algorithm.

As shown in the fig 4.2, the imbalanced training set to divide into near-neighbor set and far-neighbor set by Edited Nearest Neighbor (ENN) algorithm. The samples in the near-neighbor set are highly similar, making it very difficult for the classifier to learn the differences between the categories, so we refer to the samples in the near-neighbor set as difficult samples and the far-neighbor set as easy samples. Next, we zoom in and out the minority samples in difficult set. Finally, the easy set and minority in difficult set are combined with its augmentation samples to make up a new training set. We use the *K* neighbors in the ENN algorithm as the scaling factor of the entire algorithm. When scaling factor *K* increases, the number of difficult samples increases, and the compression rate of the majority of samples and the synthesis rate of the minority of class also increase.

In imbalanced network traffic, different traffic information types have comparative portrayals, particularly minority assaults can tuck away among a lot of typical traffic, making it hard for the classifier to get familiar with the distinctions between them during the preparation cycle. In the comparable examples of the imbalanced preparation set, the larger part class is excess commotion information. The number is a lot bigger than the minority class, making the classifier unfit to gain proficiency with the distribution of the minority class, so we pack the larger part class.

The minority class discrete traits stay steady, what's more, there are contrasts in constant ascribes. Therefore, the minority class persistent properties are zoomed to create information that adjusts to the genuine dispersion. Thus, we propose the DSSTE calculation to decrease the lopsidedness.

To begin with, the imbalanced preparation set to separate into close neighbor set and far-neighbor set by Edited Nearest Neighbor (ENN) calculation. The examples in the close neighbor set are profoundly comparable, making it truly challenging for the classifier to get familiar with the distinctions between the classifications, so we allude to the examples in the close neighbor set as troublesome examples furthermore, the far-neighbor set as simple examples. Then, we zoom in and out the minority tests in troublesome set.

At last, the simple set and minority in troublesome set are joined with its increase tests to make up another preparation set. We use the K neighbors in the ENN calculation as the scaling factor of the whole calculation. While scaling factor K increments the quantity of troublesome examples increments, and the compression pace of most of tests and the union pace of the minority of class likewise increment.

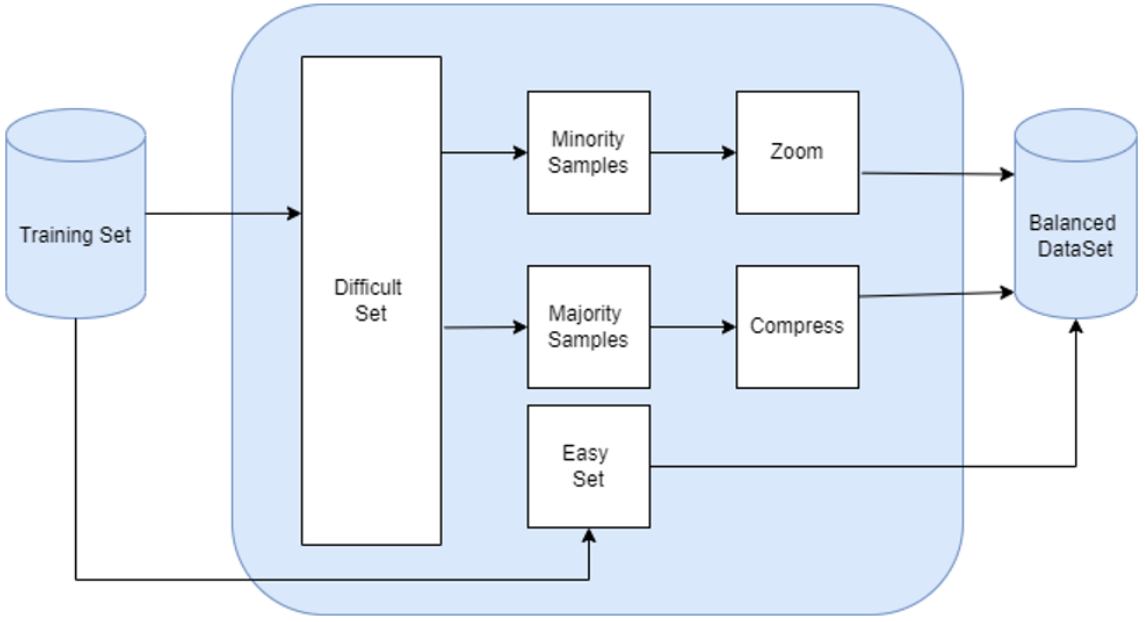


fig 4.2 DSSTE Algorithm

Project Architecture:

For balancing the VeReMi dataset, we first take the dataset and perform preprocessing to suit our algorithm. We perform cleaning like removing null values, outliers and repeating values. After this the next step would be dividing the dataset into training and test set and apply ENN algorithm on the training set to separate training set into difficult and easy set. In difficult set we take majority samples and compress them and zoom out minority samples. For obtaining new balanced dataset we combine majority samples, minority samples and easy set. The figure below fig 4.3 depicts the architecture.

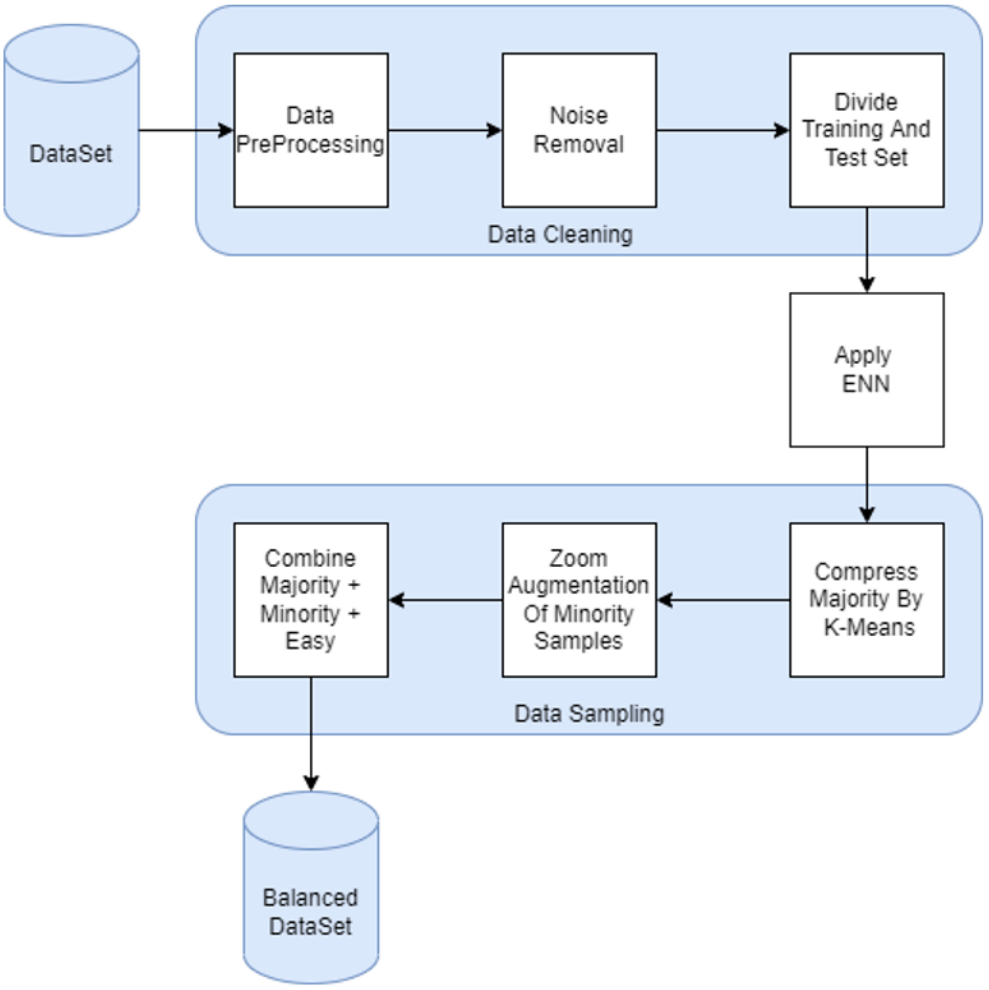


fig 4.3 VeReMi Class Balancing

CHAPTER 5

DESIGN

The most creative and hard component of system development is system design. System design is the process of establishing a device, architecture, modules, interfaces, and data for a system to meet specified criteria utilizing various approaches and concepts.

Building Blocks of UML:

The UML vocabulary includes three types of building blocks. They are:

1.Things

2.Relationships

3.Diagrams

Things:

Things refer to anything that is a physical entity or object. There are four different categories of things. Structural Things, Behavioral Things, Grouping Things, and Annotational Things are the different types of things.

Relationships:

It shows how things are connected in significant ways. It depicts the relationships between things and defines an application's functionality. Dependency, Association, Generalization, and Realization are the four forms of relationships.

Diagrams:

The diagrams are the visual representations of the models, which include symbols and text. In the context of the UML diagram, each symbol has a particular meaning. Structural Diagrams, Behavioral Diagrams, and Interaction Diagrams are the three types of UML diagrams.

Use Case Diagram

A use case diagram is used to represent the dynamic behaviour of a system. In use case diagrams the encapsulation of system's functionality takes place by incorporating the aspects like actors, use cases, and their connections. It refers to the activities, functionalities that a system/subsystem of an application requires and their services.

The functionalities for this use case diagram are Data Preprocessing,Noise Removal, Division of Dataset and the actor here is Training user. These are depicted in fig 5.1 below.

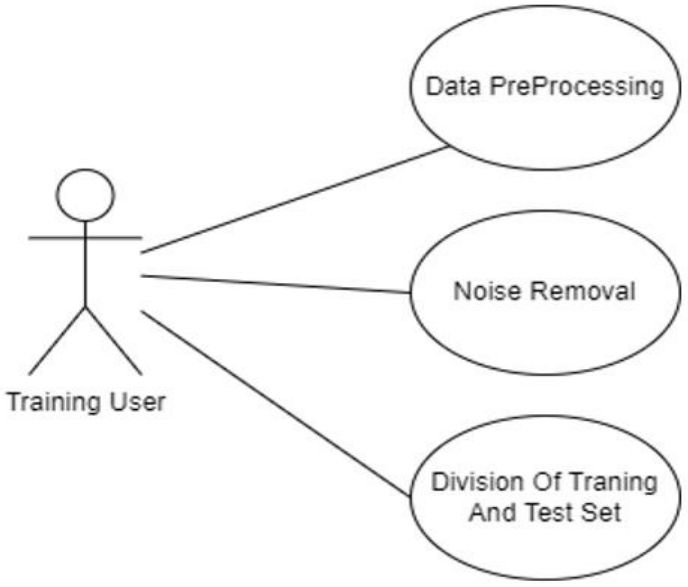


fig 5.1 Use Case diagram for training user

The DSSTE use case diagram fig 5.2 depicted below has actor DSSTE and the use cases areDivide training set, Compress Majority samples, Zoom out Minority samples, New Training set.

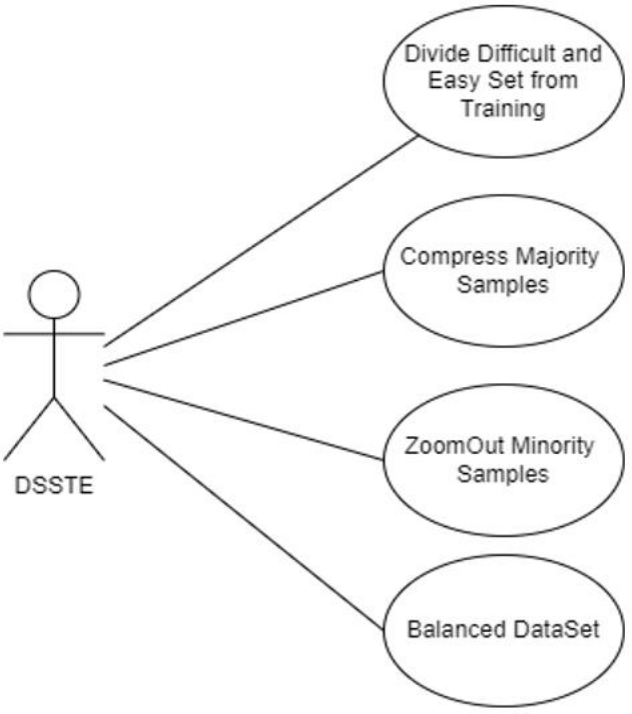


fig 5.2 Use Case diagram for DSSTE

Class Diagram

A class diagram represents the system characteristics, limitations and its activities. Another name for the class diagram is structural diagram. Each class is represented by a rectangle that is divided into three compartments: name, attributes, and operation. Modifiers are used to determine the visibility of attributes and operations.

Here in fig 5.3 we are representing the system with two classes that are Training User and DSSTE. The class Training User has operations Preprocess and divide Dataset and the class DSSTE has operations Zoom in Minority, Zoom out Minority.

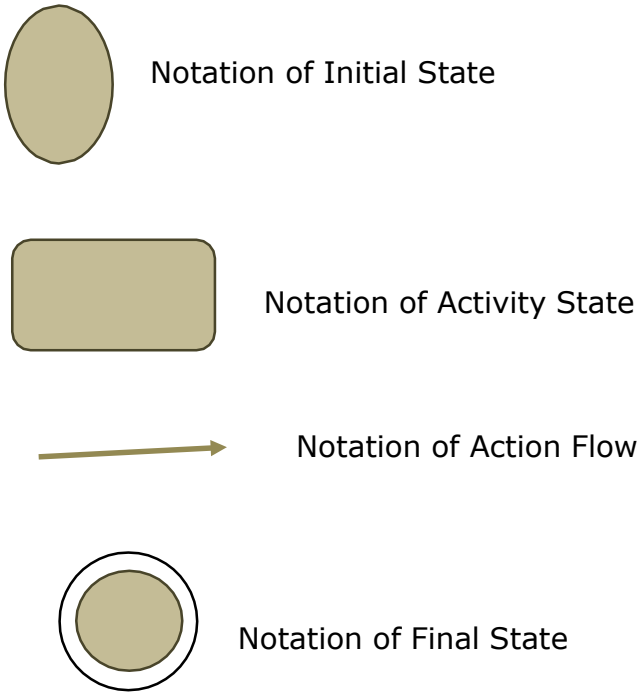
fig 5.3 Class diagram

Activity Diagram

Instead of showing the implementation, the activity diagram is used in UML to show the system's flow of control. It can imitate both concurrent and sequential activities. The flow of work from one action to the next is visualized using the activity diagram. It concentrated on the state of flow as well as the order in which it occurs.

A flowchart and an activity diagram are extremely similar.

Activity Diagram Notations –



The activity diagram here in fig 5.4 represents the flow of control and the actions that are performed on the dataset. First, we take input of dataset and wepreprocess that dataset to remove any unwanted changes and then in next step we divide the training set and compress and zoom out majority and minority samples.

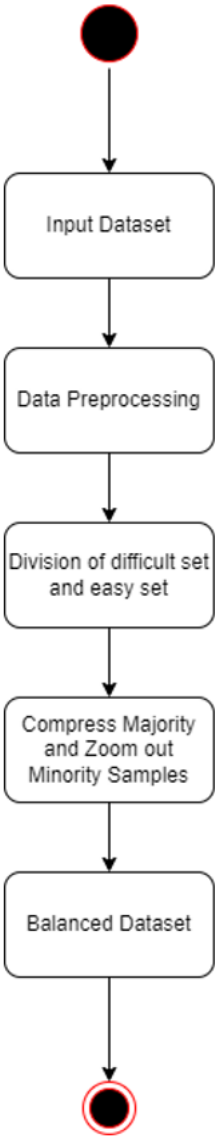


fig 5.4 Activity diagram

Data Flow Diagram

A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

This dataflow diagram fig 5.5 represents the flow of dataset through the system and in the below diagram the dataset is taken input and preprocessed andchanges are done to this preprocessed dataset accordingly to form the required dataset.

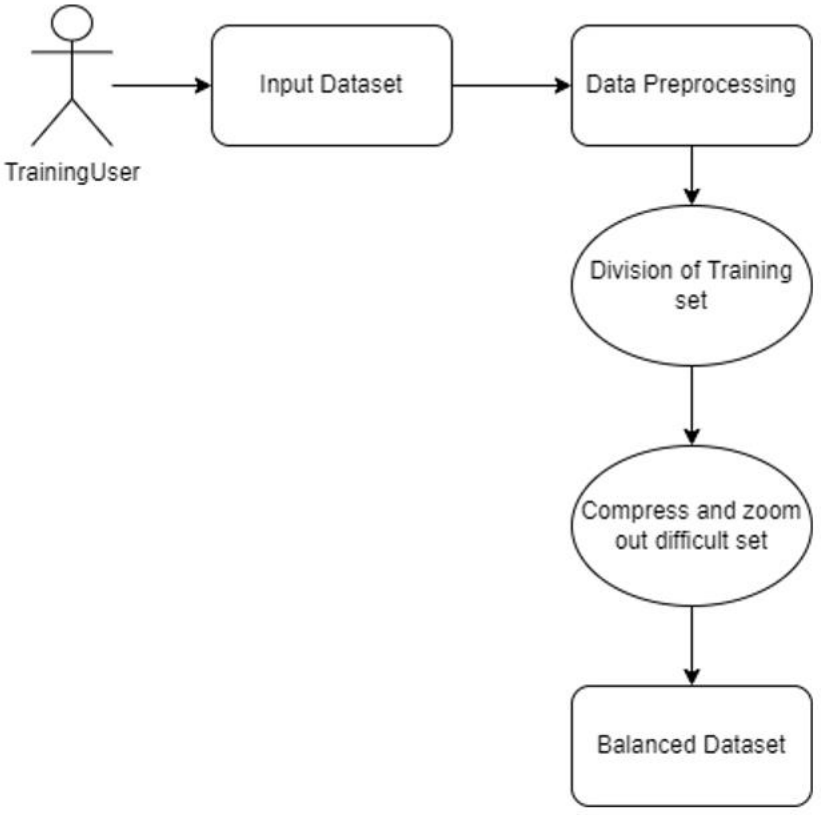


fig 5.5 Data flow diagram

CHAPTER 6

IMPLEMENTATION

```
import pandas as pd
df = pd.read_csv("Veremi.csv")
df.shape
df.head()
```

#dropping columns that are not needed

```
df=df.drop(['type','sender','rcvTime','sendTime','messa
geID','spdZ','posZ','pos_noise','spd_noise','RSSI','rep
etition','attackerDensity'],axis=1)
```

#display distribution with graphs

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(20,20))
sns.countplot(df['attackerType'])
plt.xticks(rotation = 45)
plt.show()
```

#applying KNN on raw dataset

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
X =df.drop("attackerType",axis =1)
y=df.attackerType
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = \
train_test_split(X,y,test_size=0.3,random_state=10)
y_train.head()
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix,
classification_report
model = KNeighborsClassifier()
model.fit(X_train,y_train)
y_predict = model.predict(X_test)
```

```
print(classification_report(y_test, y_predict))

from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_predict))
pd.crosstab(y_test,y_predict)
```

#applying KNN on dataset with smote

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 5)
X_train_res, y_train_res = sm.fit_resample(X_train,
y_train)
model1 = KNeighborsClassifier()
model1.fit(X_train_res,y_train_res)
y_predict_res = model1.predict(X_test)

from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_predict_res))
pd.crosstab(y_test,y_predict_res)
print(classification_report(y_test, y_predict_res))
```

#DSSTE code

```
import csv
import pandas as pd
import numpy as np
import math
import random
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.under_sampling import ClusterCentroids
from imblearn.under_sampling import
EditedNearestNeighbours
import matplotlib.pyplot as plt
```

#seperating easy set and difficult set

```
enn = EditedNearestNeighbours(sampling_strategy='all',
n_neighbors=20, kind_sel='mode', n_jobs=-1)
```

```
X_res, y_res = enn.fit_resample(X_train, y_train)
X_res = pd.DataFrame(X_res)
y_res = pd.DataFrame(y_res)

X_res = X_res.reset_index()
y_res = y_res.reset_index()
easy_df = pd.concat([X_res, y_res], axis=1)
easy_df = easy_df.drop(['index'], axis=1)
print('easy samples %s' % Counter(easy_df.iloc[:,-1:]))
easy_df.shape
data_df = pd.concat([X_train, y_train], axis=1)
data_df.shape
data_df['attackerType'].value_counts()
difficult_df = easy_df.append(data_df)
difficult_df = difficult_df.drop_duplicates(keep=False)
data = easy_df
```

#minority set consists of all datatypes with allattacks apart from 0 attack

```
minority_df = data[data['attackerType'] >= 1]
data = easy_df
data.shape
X, y = data.iloc[:,:-1], data.iloc[:,-1:]
```

#compressing the majority dataset

```
cc = ClusterCentroids(sampling_strategy='not minority',
random_state=None, estimator=None, voting='auto',
n_jobs='deprecated')

X_res, y_res = cc.fit_resample(X, y)

X_res = pd.DataFrame(X_res)
y_res = pd.DataFrame(y_res)

X_res = X_res.reset_index()
y_res = y_res.reset_index()
```

```
compress_df = pd.concat([X_res, y_res], axis=1)
compress_df = compress_df.drop(['index'], axis=1)
compress_df.shape
```

#here we generate synthetic samples. value has to beadjusted according to imbalance level

```
data = minority_df
value =50000
data.shape
import pandas as pd
pd.options.mode.chained_assignment = None
multiple = value / data.shape[0]
print(multiple)
weight_list = []
n = 20

# weight list

for x in range(n, (int(multiple/2))+n):
    weight1 = 1 + (1/x)
    weight2 = 1 - (1/x)
    weight_list.append(weight1)
    weight_list.append(weight2)
    weight_list.sort()
```

synthete samples

```
synthetic_data = pd.DataFrame()
for weight in weight_list:
    data_sample = data.iloc[:,:-
1].sample(n=random.randint(int(len(data.columns)*0.25),
int(len(data.columns)*0.75)), random_state=None,
axis=1)*(weight)
    data[data_sample.columns] = data_sample
    synthetic_data = synthetic_data.append(data,
ignore_index=True)

print('synthete samples %s' %
```

```
Counter(synthetic_data.iloc[:,-1:]))
synthetic_data.shape
f_data= pd.concat([easy_df, synthetic_data], axis=0,
ignore_index=True)
f_data.shape
f_data = pd.concat([f_data, compress_df], axis=0,
ignore_index=True)
f_data.shape
f_data = pd.concat([f_data, minority_df], axis=0,
ignore_index=True)
f_data.shape
f_data.head()
f_data['attackerType'].value_counts()
X =f_data.drop("attackerType",axis =1)
y=f_data.attackerType
```

#here we apply KNN on dataset with DSSTE

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = \
train_test_split(X,y,test_size=0.3,random_state=10)

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix,
classification_report
modelf = KNeighborsClassifier()
modelf.fit(X_train,y_train)
y_predict = modelf.predict(X_test)
print(classification_report(y_test, y_predict))
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_predict))
pd.crosstab(y_test,y_predict)
```

CHAPTER 7

EXECUTION PROCEDURE AND TESTING

7.1 EXECUTION PROCEDURE

We first prepare the VeReMi dataset and take the necessary data. We extract information like speed and position only as they contribute moreto separation. We then apply the proposed DSSTE algorithm on the extracted dataset. We make use of Google CoLab Pro Plus to apply classification as running this algorithm is very resource intensive. We apply KNN algorithm on the processed dataset.

7.2 TESTING

Software testing is a very important part of any software quality assurance as it is the final check of the requirement specification, design, analysis and code. The main goal of testing is to track the errors and also the mistakes. Testing is the process of determining all possible flaws or mistakes in a work product which can test the functionality of different components in a system, sub-systems, and also a whole product. There are several different forms of testing. Each test type is designed to satisfy a distinct testing need. The main goal of testing is to find errors and mistakes. Testing is the practice of finding all possible flaws or mistakes ina system. It allows you to test the functionality of individual components, subsystems, and also the final product. It is the method of validating software to ensure that it complies with its specifications and satisfies user expectations, as well as that it doesn't fail in an improper way.

Objectives of Testing:

- Testing is the process of evaluating the software with the goal of finding errors.

- A successful test itself has a significant chance of revealing a previously undetected flaw.

- Testing should identify multiple types of faults in a methodical manner in the shortest amount of time and with the least amount of work possible. A supplementary advantage of testing is that it verifies that the program appears to function as intended by the specifications.

- The information gathered during testing can also be used to determine the software's dependability and quality. Testing, on the other hand, cannot prove the absence of faults; it can only indicate the existence of software defects.

Principles of Testing:

- All tests must be related to the needs of the end user.

- Testing normally begins on a modest small scale and it integrates its way up to large-scale type of testing.

Testing Strategies:

A software testing strategy incorporates software test cases into a set of well-planned procedures that lead to a successful build. Verification and validation are two terms for the same thing in software testing. Verification verifies that the software performs a specific function appropriately. Validation is a set of operations that ensures that the software that has been produced can be traced back to the needs of the customer. We implement our algorithm on VeReMi dataset. to verify whether the problem of class imbalance is reduced and it is generating the desired results or not.

CHAPTER 8

RESULTS AND PERFORMANCE EVALUATION

Here we see that most of attack type 5 are identified as type 0. This is due to the reason that vehicles act normally in the beginning and later starts broadcasting malicious messages. The precision, recall and f1-scores of KNN on raw data, KNN with SMOTE and KNN with DSSTE are shown in fig 8.1, 8.2 and 8.3. To solve this, a new extension dataset has been produced that labels vehicle as attacker even before it starts transmitting malicious data. The confusion matrix is displayed in figure 8.4.

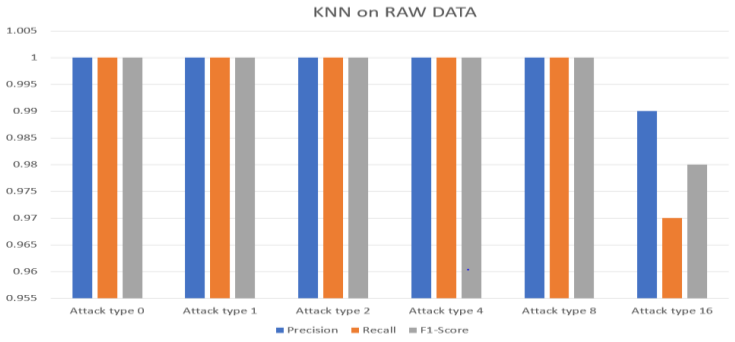


fig 8.1 Classification Report of Raw Data

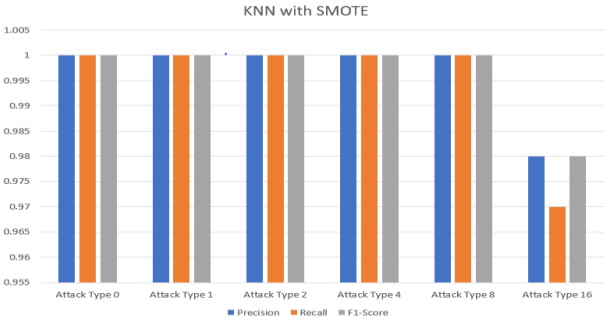


fig 8.2 Classification Report of SMOTE

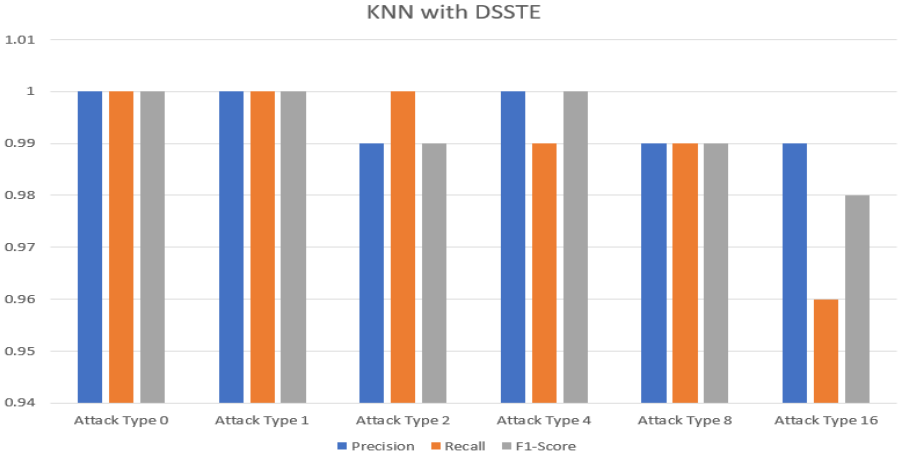


fig 8.3 Classification Report of DSSTE

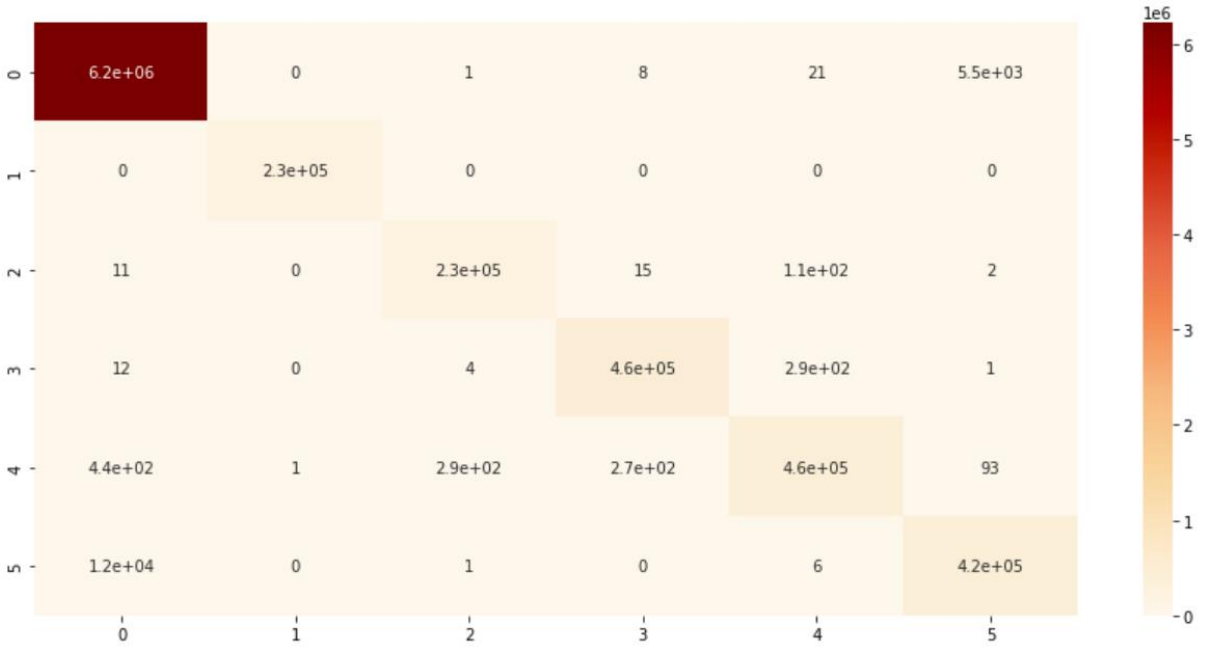


fig 8.4 Confusion Matrix

We achieved an accuracy of 99.6%. However, Attack type-16 had low precision and recall as compared to other attack types. Vehicles acts as genuine for some time and then transmits malicious data. As the same vehicle is transmitting original and malicious content, it is hard for the classifier to identify this type of attacks.

CHAPTER 9

CONCLUSION AND FUTURE WORK

The proposed DSSTE algorithm was able to successfully classify Attack types 1, 2, 3 and 4. But attack type 5 was not accurately predicted. We achieved an accuracy of 99.6%. The precision values are 1.00, 1.00, 0.99, 1.00, 0.99 and 0.99 and the recall values are 1.00, 1.00, 1.00, 1.00, 0.99 and 0.96 for attack types 0, 1, 2, 4, 8 and 16.

A further analysis of type 5 attack and new neural network is needed to improve the accuracy even further. An extension of VeReMi dataset has been proposed that consists of modification of type 16 attack and malfunctioning of sensors are included. We can try and further apply data balancing methods like DSSTE and SMOTE on extension dataset to generate more accurate predictions.

Different approaches that are proposed on VeReMi can also be applied on extension dataset. The above dataset is generated using Luxemburg city traffic data. We can apply the same generation techniques on Indian traffic conditions in cities like Hyderabad, Chennai and Mumbai where traffic densities will be much higher and detection of malicious attacks will be much harder.

CHAPTER 10

APPENDIX

10.1 List of Figures

CHAPTER 11
REFERENCES

1. Lan Liu , Pengcheng Wang , Jun Lin and Langzhou Liu, "Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning", *Proc. IEEE Int. Conf. Granular Comput.*, Vol-9, Dec 2020..
2. D. A. Cieslak, N. V. Chawla and A. Striegel, "Combating imbalancein network intrusion datasets", *Proc. IEEE Int. Conf. Granular Comput.*, pp. 732-737, May 2006.
3. P. Jeatrakul, K. W. Wong and C. C. Fung, "Classification of imbalanced data by combining the complementary neural network and smote algorithm", *Proc. Int. Conf. Neural Inf. Process.*, pp. 152-159, 2010.
4. P. Bedi, N. Gupta and V. Jindal, "Siam-IDS: Handling class imbalance problem in intrusion detection systems using siamese neural network", *Procedia Comput. Sci.*, vol. 171, pp. 780-789, 2020.
5. L. Feng, L. Yu, L. Xueqiang and L. Zhuo, "Research on query topic classification method", *Data Anal. Knowl. Discovery*, vol. 31, no. 4, pp. 10-17, 2015.
6. M. A. Tahir, J. Kittler and F. Yan, "Inverse random under sampling for class imbalance problem and its application to multi-label classification", *Pattern Recognit.*, vol. 45, no. 10, pp. 3738-3750, Oct. 2012.
7. A. Liu, J. Ghosh and C. E. Martin, "Generative oversampling for mining imbalanced datasets", *Proc. DMIN*, pp. 66-72, 2007.
8. N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique", *J. Artif. Intell. Res.*, vol. 16, pp. 321-357, Jun. 2002.

9. B. Yan and G. Han, ``LA-GRU: Building combined intrusion detection model based on imbalanced learning and gated recurrent unit neural net-work,'' *Secur. Commun. Netw.*, vol. 2018, pp. 1_13, Aug. 2018.
10. R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, Deep and machine learning approaches for anomaly- based intrusion detection of imbalanced network traf_c,'' *IEEE sensors Lett.*, vol. 3, no.1,Jan. 2019, Art. no. 7101404.
11. P.-J. Chuang and D.-Y.Wu, ``Applying deep learning to balancing network intrusion detection datasets,'' in *Proc. IEEE 11th Int. Conf. Adv. Infocomm Technol. (ICAIT)*, Oct. 2019, pp. 213_217.
12. P. Bedi, N. Gupta, and V. Jindal, ``Siam-IDS: Handling class imbalance problem in intrusion detection systems using siamese neural network,'' *Procedia Comput. Sci.*, vol. 171, pp. 780_789, 2020.
13. S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular Ad Hoc network," J. Netw. Comput. Appl., vol. 37, pp. 380–392, 2014.
14. S.-h. An, B.-H. Lee, and D.-R. Shin, "A survey of intelligent transportation systems," in Proc. 3rd Int. Conf. Comput. Intell., Commun. Syst. Netw., 2011, pp. 332–337.
15. S. Zeadally, R. Hunt, Y.-S. Chen, A. Irwin, and A. Hassan,"Vehicular ad hoc networks (VANETs): Status, results, and challenges," Telecommun. Syst., vol. 50, no. 4, pp. 217–241, 2012.
16. H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, "VANET security challenges and solutions: A survey," Veh. Commun., vol. 7, pp. 7–20, 2017

17. M. Arif, G. Wang, M. Z. A. Bhuiyan, T. Wang, and J. Chen, "A survey on security attacks in VANETs: Communication, applications and challenges," Veh. Commun., vol. 19, 2019, Art. no. 100179

18. S. S. Manvi and S. Tangade, "A survey on authentication schemes in VANETs for secured communication," Veh. Commun.,vol. 9, pp. 19–30, 2017

19. M. A. Hezam et al., "Classification of security attacks in VANET: A review of requirements and perspectives," in Proc. MATEC Web Conf 150 06038, 2018, doi: 10.1051/matecconf/201815006038.

20. P. Tyagi and D. Dembla, "A taxonomy of security attacks and issues in vehicular Ad-Hoc networks (VANETs)," Int. J. Comput. Appl., vol. 91, no. 7, pp. 22–29, 2014.

21. R. W. van der Heijden, S. Dietzel, and F. Kargl, "Misbehavior detection in vehicular Ad-hoc networks," in Proc. 1st Inter- Veh. Commun. Conf. (FG-IVC 2013), 2013, pp. 23–25.

22. R. W. van der Heijden, T. Lukaseder, and F. Kargl, "VeReMi: A dataset for comparable evaluation of misbehavior detection in VANETs," in Security and Privacy in Communication Networks, R. Beyah et al. Eds., New York, NY, USA: Springer, 2018, pp. 318–337.

23. S. So, P. Sharma, and J. Petit, "Integrating plausibility checks and machine learning for misbehavior detection in VANET," in Proc. 17th IEEE Int. Conf. Mach. Learn. Appl., 2018, pp. 564–571.

24. M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," Veh. Commun., vol. 14, pp. 52–63, 2018.

25. X. Xue, N. Lin, J. Ding, and Y. Ji, "A trusted neighbor table based location verification for VANET routing," in IET 3rd Int. Conf. Wireless, Mobile Multimedia Netw., 2010, pp. 1–5, doi: 10.1049/cp.2010.0603.

26. K. Zaidi, M. B. Milojevic, V. Rakocevic, A. Nallanathan, and M. Rajarajan, "Host-based intrusion detection for VANETs: A statistical approach to rogue node detection," IEEE Trans. Veh. Technol., vol. 65, no. 8, pp. 6703–6714, Aug. 2016

27. L. Liang, H. Ye, and G. Y. Li, "Toward intelligent vehicular networks: A machine learning framework," IEEE Internet Things J., vol. 6, no. 1, pp. 124–135, Feb. 2 2019.

28. J. Grover, V. Laxmi, and M. S. Gaur, "Misbehavior detection based on ensemble learning in VANET," in Proc. Int. Conf. Adv. Comput., Netw. Secur. Berlin, Heidelberg: Springer, 2011, pp. 602– 611.

29. P. K. Singh, R. R. Gupta, S. K. Nandi, and S. Nandi, "Machine learning based approach to detect wormhole attack in VANETs," in Proc. Workshops Int. Conf. Adv. Inf. Netw. Appl. Cham: Springer, 2019, pp. 651–661.

30. P. C. Sen, M. Hajra, and M. Ghosh, "Supervised classification algorithms in machine learning: A survey and review," Emerging Technology in Modelling and Graphics. Singapore: Springer, 2020, pp. 99–111.

31. A. Mucherino, P. J. Papajorgji, and P. M. Pardalos, "K-nearest neighbor classification," in Data Mining in Agriculture. New York, NY, USA: Springer, 2009, pp. 83–106.

32. A. Priyam, G. Abhijeeta, A. Rathee, and S. Srivastava, "Comparative analysis of decision tree classification algorithms," Int. J. Curr. Eng.

Technol., vol. 3, no. 2, pp. 334–337, 2013. [25] A. Liaw et al., "Classification and regression by randomforest," R News, vol. 2, no. 3, pp. 18–22, 2002

33. A. Liaw et al., "Classification and regression by randomforest," R News, vol. 2, no. 3, pp. 18–22, 2002.

34. K. M. Leung, "Naive bayesian classifier," Polytech. Univ. Dept. Comput. Sci./Finance Risk Eng., vol. 2007, pp. 123–156, 2007.

35. K. Zaidi, M. B. Milojevic, V. Rakocevic, A. Nallanathan, and M. Rajarajan, "Host-based intrusion detection for VANETs: A statistical approach to rogue node detection," IEEE Trans. Veh. Technol., vol. 65, no. 8, pp. 6703–6714, Aug. 2016

36. H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, "VANET security challenges and solutions: A survey," Veh. Commun., vol. 7, pp. 7–20, 2017.