

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor
of Engineering Degree in Computer Science and Engineering

By

NERELLA PAVAN KALYAN GOUD
REG. NO. 38110363
&
G. SAI CHARAN REDDY
REG. NO. 38110181



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

MARCH 2022



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI– 600119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **NERELLA PAVAN KALYAN GOUD (Reg. No: 38110363), G. SAI CHARAN REDDY (Reg. NO: 38110181)** who carried out the project entitled “**TRAFFIC CLASSIFICATION OF SOFTWARE DEFINED NETWORKS USING MACHINE LEARNING ALGORITHMS**” under my supervision from **November 2021 to March 2022**.

Internal Guide

Dr. A. Mary Posonia, M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

InternalExaminer

ExternalExaminer

DECLARATION

NERELLA PAVAN KALYAN GOUD (Reg. No: 38110363), G. SAI CHARAN REDDY (Reg. NO: 38110181) here by declare that the project report entitled **“TRAFFIC CLASSIFICATION OF SOFTWARE DEFINED NETWORKS USING MACHINE LEARNING ALGORITHMS”** done by us under the guidance of **Dr.A.Mary Posonia .M.E.,Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

1)Nerella pavan kalyan goud

2)G.Sai charan reddy

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D**, Dean, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D.** and **Dr. L. Lakshmanan, M.E., Ph.D.**, Heads of the Department of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A.Mary Posonia , M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Traffic classification is becoming an important research area especially with the day-to-day machine learning and software defined networking advancements. It is used in network management, service measurements, network design, security monitoring and advertising. The proposed model uses two machine learning models Logistic regression and k-means clustering were used to classify ping, telnet,dns ,voice traffic flows by using a tool called distributed internet traffic generator-(D-ITG) . Host in this network are connected through an overlay network to an open vswitch(Ovs). This Ovs is connected to a Ryu (Ryu means “flow” in Japanese language. Ryu is pronounced as “ree-yooh”) controller which is used to collect flow statistics between hosts. These statistics are parsed by a Python traffic classification script which periodically outputs the learned traffic labels. The proposed system will classify the type of the network traffic flowing in live between two hosts.

TABLE OF CONTENTS

CHAPTE R No.	TITLE	PAGE No
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	LIST OF ABBREVIATIONS	iv
1	INTRODUCTION	1
2	LITERATURE REVIEW	2
3.	EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USE	4
	3.1 Proposed Architecture	4
	3.2 Logistic regression algorithm	6
	3.3 K-means clustering	7
4.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	9
5	SUMMARY AND CONCLUSIONS	12
	REFERENCES	13
	APPENDIX	14
	A. SOURCE CODE	14
	B. SCREENSHOTS	24
	C.PUBLISHED PAPER	25

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO.
1	System architecture	4
2	Topology used in the proposed work	4
3	Accuracy Proof	6
4	Logistic Regression confusion matrix	7
5	Four clusters	7
6	learned cluster labels	7
7	K-means algorithm using confusion matrix.	8
8	network traffic as ping	9
9	network traffic as telnet	10
10	network traffic as voip	10
11	network traffic as dns	10

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
1	Comparing the Accuracy III	9

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
D-ITG	Distributed internet traffic generator
Ovs	Open vSwitch
Vswitch	A virtual switch
Ryu	Ryu means “flow”
Qos	Quality of service
Dns	Domain Name System
Voip	Voice over Internet Protocol
AI	Artificial Intelligence

CHAPTER 1

INTRODUCTION

A significant piece of examination is the constant improvement of AI and programming that characterizes the standards of coordination of exercises that assistance to sort out the progression of traffic in the internet protocol line. As it was done in the good old days, for example, distinguishing the port number and really taking a look at the heap because of the roadway and the put away things, isn't working as expected, and the venture will assist with utilizing automated and automated AI machines to decrease the quantity of vehicles utilizing their required and custom code ground floor. The requesting of vehicles utilizing AI is expanding in the organization investigation. An illustration of this is Skype an application used for video and Voice call application, which generates VoIP traffic. where even experts find it challenging to classify and dissect huge volumes of movement data [5]. Another illustration is aha a video streaming application in this quality of service is important. To maintain quality the traffic flowing to the user should be classified to give a best streaming experience. These Machine learning sequencing and connecting traffic can assist with characterizing network availability through Qos and broadband. This is significant in picking the future and the requirements of the organization, and it is exceptionally energizing to give web, government, and business administrations to get to the street. Posting of vehicles in light of controlled preparing to decide the phases of the street utilizing known wellsprings of data. The technique for AI is depicted in [11], [12] [13], [14], and the creators of [13] present the development of SDNs, which makes street information in view of OpenFlow. Many learning calculations are currently utilized for information gathered to be executed in an assortment of ways. The study was led on an organization of organizations.

CHAPTER 2

LITERATURE SURVEY

There has been a ton of examination on programming traffic, and a great deal of exploration has been done on this thought a good survey can be found here: Chen, Joy long Zong, and P. Hengjinda. "Early Prediction of Coronary Artery Disease (CAD) by Machine Learning Method-A Comparative Study." Journal of Artificial Intelligence in this an image pre-processing technique is used that helped in provoking an idea of classification based on the data packet size parameter [16]. Another study by Balasubramaniam, Vivekanadam. "Artificial Intelligence Algorithm with SVM Classification using Dermoscopic Images for Melanoma Diagnosis." Journal of Artificial Intelligence and Capsule Networks the syudy is all about predicting skin cancer. K- means clustering algorithm is used for achieving the goal.

From this a clear working visualization of K-means clustering is achieved [17]. And one of which was finished by Zhongmin Kwon, Daeung Jung, and Hyung Park on the "Traffic Data Classification utilizing AI Amahugur Road Traffic Classification" "Figure out the product pipeline by 2020. In this review, they attempted to show the primary aftereffects of a mechanized information arranging calculation in view of AI calculations." Obstacles to the investigation of genuine organization-based AI, notwithstanding the surprising data gave in network the board. Investigating the gathered street information and the product that depicts the ecological organization, involving a portion of the calculations in the rush hour gridlock data preparing [1].

This study clarifies that it is feasible to observe an AI calculation for breaking down data utilizing constant correspondence networks gathered with a SDN regulator in two unique ways. The proposed concentrate on intends to expand the precision of AI calculations to accomplish this objective, and to test it on an immediate organization to more readily decide the degree of traffic.

Concentrates by Jungmin Kwon, Daeun Jung, and Hynggon show that AI machines can be utilized in the organization. Foster an organization of scientific AI calculations for future work. This is a point-by-point depiction of the exploration exercises in the field of street reporting utilizing AI "[1].

As the requests on them in a bigger and more mind-boggling manner increment, pipeline the board and checking are a significant stage in taking care of the issue. This should be possible with the assistance of street the board, otherwise called vehicle forecast, which requires the foundation of a street network that interfaces various sorts of pipelines to guarantee the appropriate preparation of the street.

Posting is an illustration of an instructional class that is controlled and used to decide the phase of another bearing connected with the data gave. For instance, contingent upon the configuration of the email, for example, the first IP address, the length of the text, and the text, the email can be carried out as spam or spam. [2]

The conventional strategies for positioning IP traffic that have been utilized for a long-time incorporate port-based techniques, profundity estimations, and that's just the beginning. The port-based technique empowers the grouping of vehicles or applications utilizing the port number of the exchange control and the client of the datagram convention. Albeit numerous applications or other port numbers have been utilized, this technique doesn't work. An inside and out test-pack is a pricey test and uses a ton of framework hardware. DPI doesn't work on the grounds that the substance of this bundle can't be utilized or gotten to.

Analysts have been attempting to involve a more top to bottom review technique. One of the analysts proposed a SAE-based way to deal with distinguishing stowed away data. In any case, the data gave was not open to general society, and the work that showed the secret street necessities didn't give an exhibition examination [6]. Late exploration has zeroed in on distinguishing and choosing the attributes of the stream to recognize the stream. Programming combination shows that a few things are a higher priority than meters that open the market and travel well, and that it makes them more proficient [8].

Networks characterized by programming give great standards on the best way to oversee parks in view of data in the head. It can likewise give rules to sending numbers, like groups and bytes. OpenFlow can be utilized in an assortment of ways of recovering genuine data. The idea of this regulation is void, and it rehashes the law after a timeframe without appropriate data [7].

AI calculations are utilized to foster the utilization of the Internet to accomplish energy saving and organization the board. Customary security techniques, for example, the utilization of handheld gadgets are the primary explanations behind inadequate security and space and reality. Complete vehicle classes can lessen the gamble [9].

CHAPTER 3

EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USE

Proposed Architecture:

The architecture diagram of figure – 2 explains about the steps involved in order to follow for the successful completion the predefined abstract. These steps are explained one after the other briefly.



Fig: 1 gives a clear idea of the steps involved in achieving the goal explained in abstract

A. VirtualBox Network Topology: -

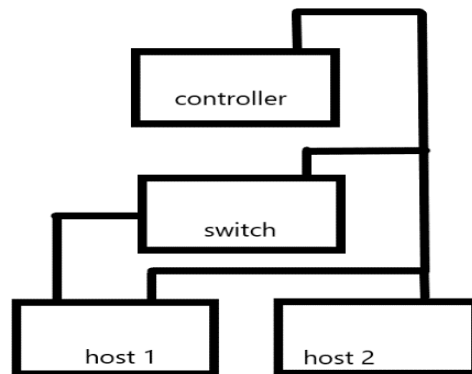


Fig: 2 Helps in Visualizing Topology used in the proposed work

In real life process a controller, switch two host are required to simulate the network traffic flow. To achieve this virtually an instant virtual network is needed there comes the importance of Mininet. Mininet is an impressionist for deploying large networks on the finite resources of a simple virtual machine. To visualize the topology of Mininet the above figure helps. After the successful setup of Mininet topology the controller should be conscious of the traffic packets flowing through the Switch between any of the host. The python script and simple_monitor_python.py will be updated to display the below flow information each second. To get everything rolling, utilize the accompanying order `sudo mn - regulator = remote, Ip = 127.0.0.1 - - top single, 3` this order will begin the regulator and open the Mininet terminal. Presently utilize the xterm h1 and xterm h2 orders to open the new Mininet Terminal, contrasting h1 and h2 as two.

B. Simulation of traffic flows:

The kind of organization traffic we use to group ping, DNS, Telnet, and standard voice. Ping traffic can be produced from a command line interface from one host to the other. In order to generate the remaining traffic flows D-ITG is utilized to make different flows of the Internet. Web circulation is classified "a website that can successfully make ipv4 and IPv6 by copying the usefulness of existing Internet applications." D-ITG can create traffic utilizing inaccurate model of park size (PS) and inter departure time (IDT) that emulates conduct levels. D-ITG is equipped for creating traffic insights for various exercises, for example, telnet and voip. Further work and limits depict the selection of boundaries to mimicked gadgets and the difficulties of utilizing D-ITG.

Steps to create traffic through conveyed Internet dispersion:

Stage 1: Telnet traffic comprises of telnet capacities. It will work with different setup convention control conventions overlooked.

```
Order utilized: cd/Desktop/D-ITG-2.8.1-r1023/bash telnet
```

Stage 2: DNS traffic comprises of DNS capacities that work with the datagram client convention and the control convention.

```
Order utilized: cd/Desktop/D-ITG-2.8.1-r1023/bash dns.sh
```

Stage 3: VOICE generate traffic which consists of voice characteristics this will work with user datagram protocol and different settings were ignored.

```
Order utilized: cd/Desktop/D-ITG-2.8.1-r1023/bash voice
```

C.IMPLEMENTATION OF TRAINING MODELS:

Sequence of command involved for collecting the dataset

```
Order utilized: Sudo python3 traffic_classifier_python3.py train [ping/telnet/dns/voice]
```

```
f system. Arguments [1] == "train": // checking the command
argument open (order, shell = True, stdout = subprocess. PIPE,
stderr = subprocess.STDOUT)//Start the interaction
open (traffic_type + '_ exercise_data.csv', 'w')//open result
record sign. (Image. SIGALRM, UI)//begin the person cycle
sign. Notice (TIME)//Set inside 15 minutes
Operating system. erase way (os.getpgid (p.pid), image.
SIGTERM)//Delete Ryu way and
exit close ()//Close numbers
```

Implementing Network traffic classification using Logistic Regression:

Calculated relapse is the principal control of the AI calculations utilized in research. The result indicator is a parallel worth, yet assuming there are multiple objectives, pick an objective that has whatever number prospects as could be expected under the circumstances. In the program, I utilized a calculated relapse that functioned admirably with an exactness of ~ close to 99%. This is a design with the initial two standards whose fundamental limits are genuine.

Formula for accuracy = $2((\text{precision} * \text{recall}) / (\text{precision} + \text{recall}))$.

It is somewhere in the range of 0 and 1. Over the worth, this is a genuine model. In the immediate test, the example rate was 0.90.

```
[23] print('Accuracy: %.2f%%' % (accuracy_score(predictions,y_test)*100))
```

Accuracy: 98.89%

We see the model has a **98.89%** accuracy

Fig: 3 Accuracy obtained by logistic regression algorithm. Benefiting as much as possible from the reason for the disarray framework is the right decision. A deception framework is a grid marked as accepted characters on the y-hub and valid on the x-pivot. Any model that endeavors to foresee street segments for another example will be explained utilizing Figure 3, which outlines the above mentioned. To accurately decide the target a confusion matrix is a perfect choice.

Confusion matrix is a matrix with predicted labels on the y-axis and true labels on the x-axis. Any model which is trying to predict certain traffic classes on another class is more of then it will clearly evident using confusion Matrix figure 3 explain the above.

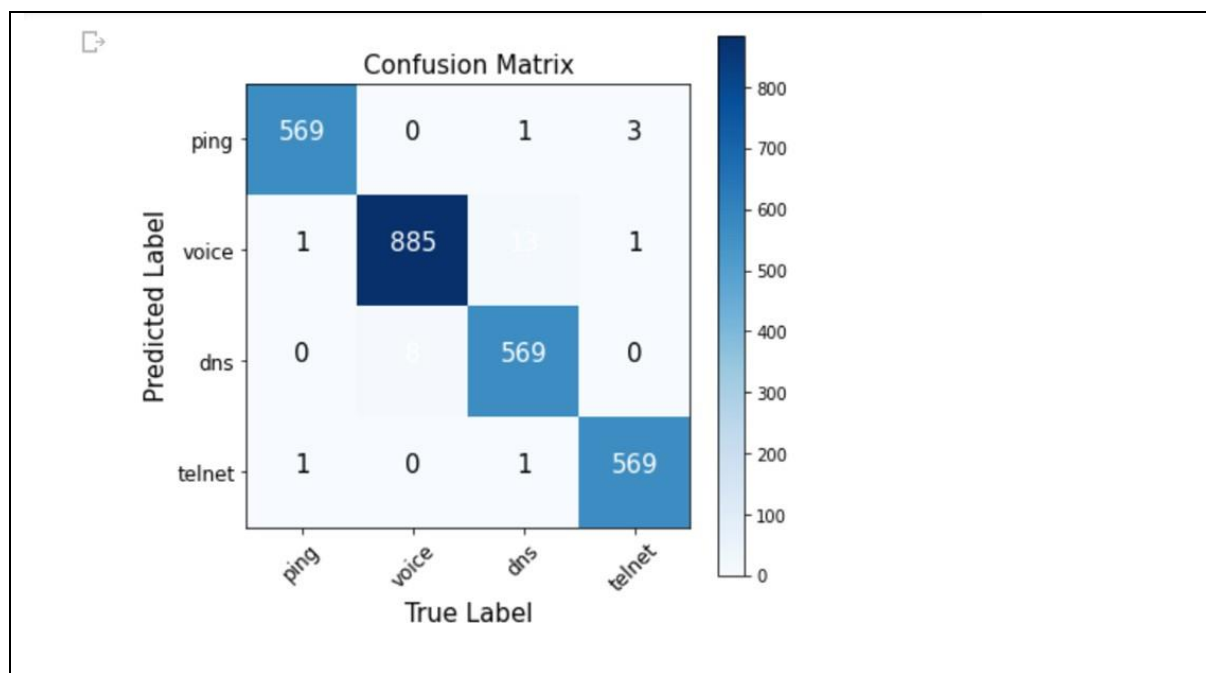


Fig: 4 To accurately decide the target the above confusion matrix is a used.

6

Implementation of Network traffic classification using K-means clustering:

K-means clustering is an unsupervised machine learning algorithm that groups data points in more than two-dimensional space together to form clusters and labels each data point in the cluster. Four clusters were used as an initial input parameter to the model for the four traffic types figure- 4 will explain to visualise each centroid of the four clusters. Where each square represents the dimension. If the dimension colour is dark then the higher the value this will display to a 12-dimensional vector space where the cluster centroids lie.

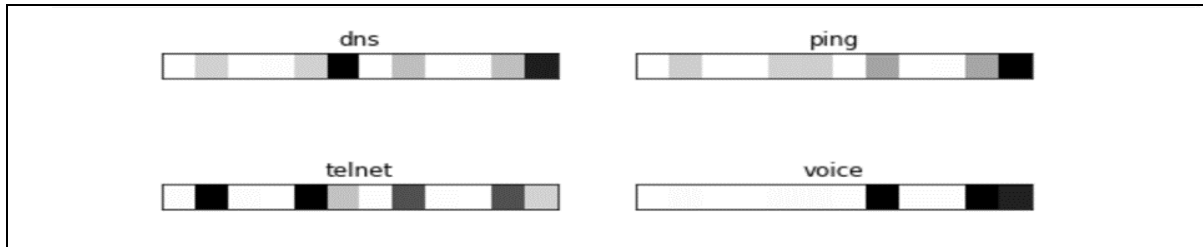


Fig: 5 Four clusters were used as an initial input parameter to the model.

For better visualization of model performance, I used principle component analysis where the reduction of number of dimensions from 12 to 2 and then plot in two- dimensional space Because of the non-auricularly groups of data the performance of the model was only around ~46% which is very poor .

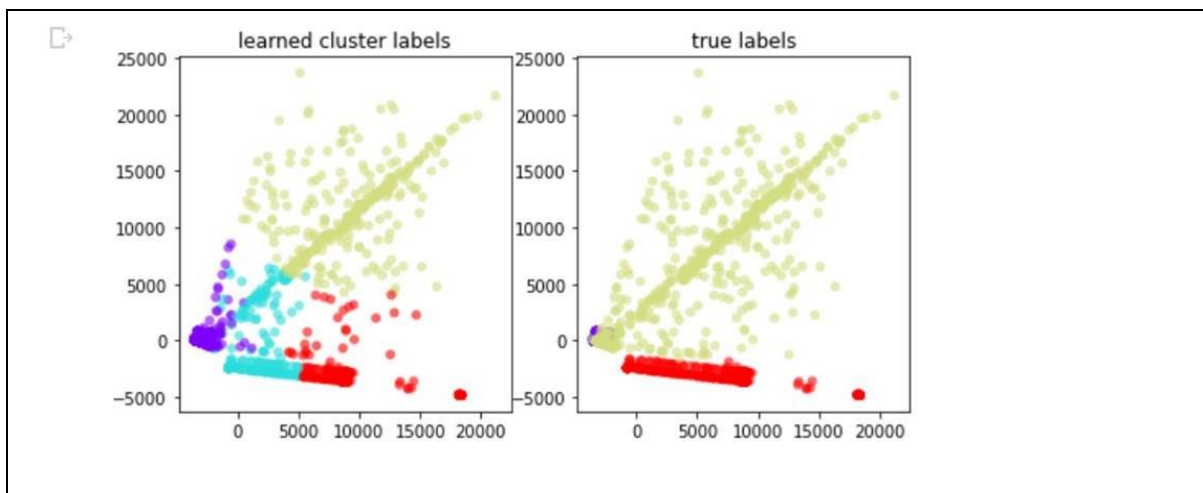


Fig: 6 explains about learned cluster labels from the training data. Right side graph explains about true labels from the testing data.

The above can be confirmed by visualizing confusion Matrix which explains a very disbalanced matrix

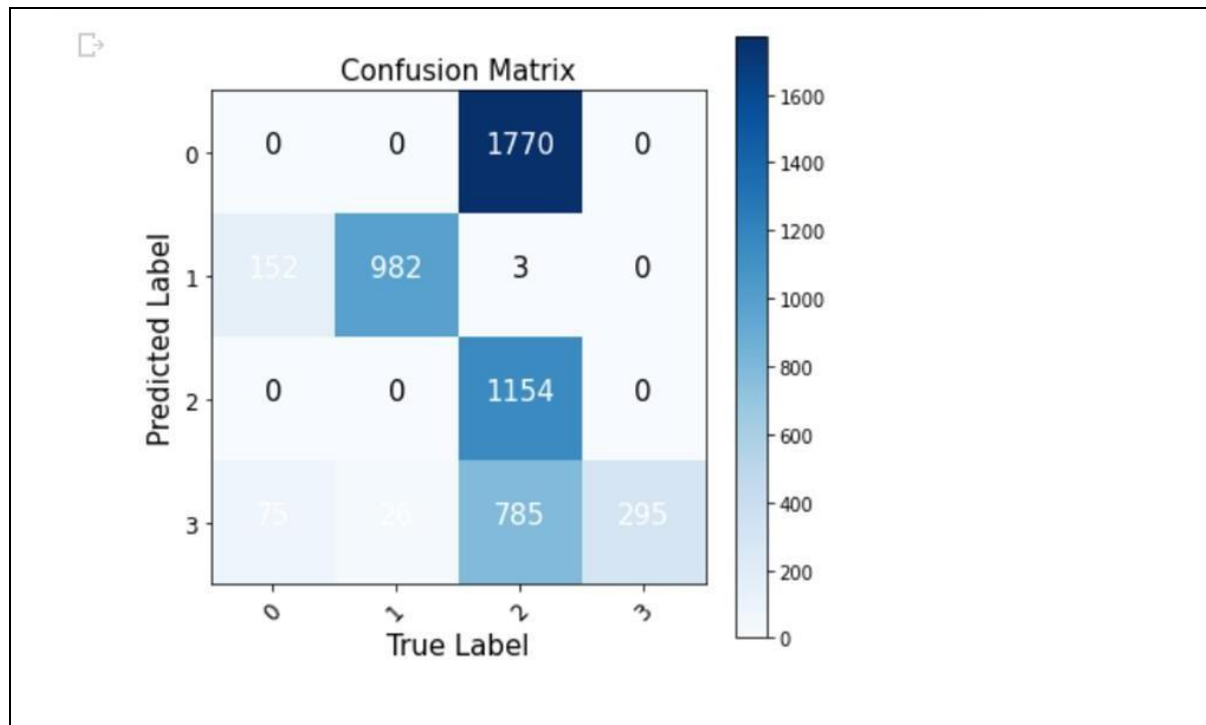


Fig: 7 Explaining about the disbalanced clustering of K-means algorithm using confusion matrix.

CHAPTER 4

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

Input : Collection of training data :

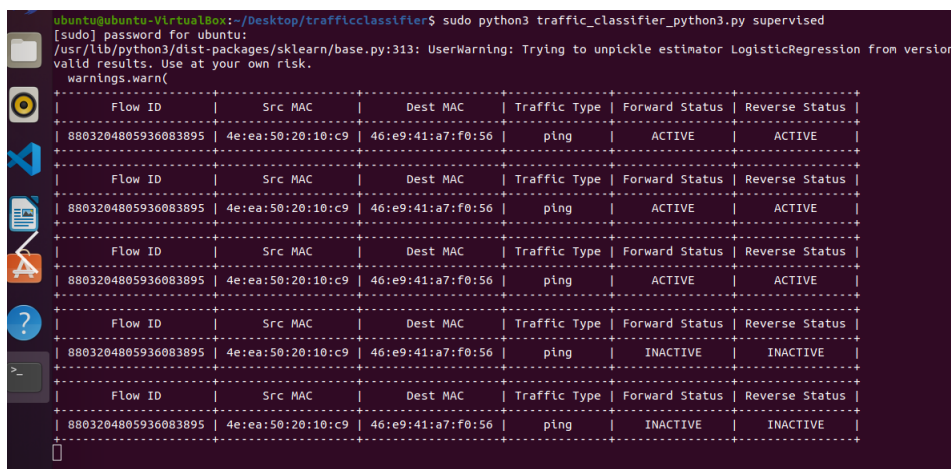
The novelty of proposed work is creating a graphical user interface (GUI) for displaying the traffic type and secondly increasing the accuracy by logistic regression algorithm. The procedure to correct training data for the machine learning models are primarily to simulate the traffic flow of the specified traffic between two hosts using D-ITG tool. The secondary thing is to start network_traffic_classifier.py script. The training data generated from simple_monitor_ak.py script is collected and transfer to update the ascribe the traffic flows objects. The ascribe of the traffic flow objects are then regularly outputted to a csv file. After the completion of collection of generated traffic flow ascribes in the csv file, they all are combined in a Pandas data frame will be taken for further process of model training and testing.

COMPARING THE ACCURACY OF THE ALGORITHMS USED.

Algorithm	Training Accuracy	Test Accuracy
Logistic regression	98.89%	84.16%
K-Means Clustering	45.98%	39.32%

The results obtained from the proposed work are displayed below and they are discussed to get a clear picture to the users and validators. Below figures explain about the live classification of network traffic. [Fig 7] classify the traffic flow as ping traffic and display the status of the traffic as 'active'. After stopping the flow between the hosts it will update to 'Inactive'. [Fig 8] classify the traffic flow as telnet traffic and display the status of the traffic as 'active'. After stopping the flow between the hosts it will update to 'Inactive'. [fig 9] classify the traffic flow as voice traffic and display the status of the traffic as 'active'. After stopping the flow between the hosts it will update to 'Inactive'. [Fig 10] classify the traffic flow as ping traffic and display the status of the traffic as 'active'. After stopping the flow between the hosts it will update to 'Inactive'.

The below figures 8,9,10,11 are the proof of classification.



```
ubuntu@ubuntu-VirtualBox:~/Desktop/trafficclassifier$ sudo python3 traffic_classifier_python3.py supervised
[sudo] password for ubuntu:
/usr/lib/python3/dist-packages/sklearn/base.py:313: UserWarning: Trying to unpickle estimator LogisticRegression from version 
valid results. Use at your own risk.
  warnings.warn(
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | ACTIVE | ACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | ACTIVE | ACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | ACTIVE | ACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | INACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | INACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
```

Fig:8 classification of network traffic as **ping** and updating the status of the traffic flow as

Active/Inactive 9

```

ubuntu@ubuntu-VirtualBox:~/Desktop/trafficClassifier$ sudo python3 traffic_classifier_python3.py supervised
[sudo] password for ubuntu:
/usr/lib/python3/dist-packages/sklearn/base.py:313: UserWarning: Trying to unpickle estimator LogisticRegression from version 1
valid results. Use at your own risk.
warnings.warn(

```

Flow ID	Src MAC	Dest MAC	Traffic Type	Forward Status	Reverse Status
-650268477036915217	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	ping	ACTIVE	INACTIVE
-650268477036915217	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	telnet	ACTIVE	ACTIVE
-650268477036915217	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	telnet	ACTIVE	ACTIVE
-650268477036915217	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	telnet	INACTIVE	INACTIVE
-650268477036915217	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	telnet	INACTIVE	INACTIVE

Fig:9 classification of network traffic as **telnet** and updating the status of the traffic flow as Active/Inactive

Flow ID	Src MAC	Dest MAC	Traffic Type	Forward Status	Reverse Status
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	voip	ACTIVE	ACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	voip	ACTIVE	INACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	voip	ACTIVE	INACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	voip	ACTIVE	INACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	voip	ACTIVE	INACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	voip	ACTIVE	INACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	voip	ACTIVE	INACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	voip	INACTIVE	INACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	voip	INACTIVE	INACTIVE

Fig :10 classification of network traffic as **voip**(voice traffic) and updating the status of the traffic flow as Active/Inactive

Flow ID	Src MAC	Dest MAC	Traffic Type	Forward Status	Reverse Status
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	dns	ACTIVE	INACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	dns	ACTIVE	INACTIVE
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	dns	ACTIVE	INACTIVE

Fig:11 classification of network traffic as **dns** and updating the status of the traffic flow as Active/Inactive

The attributes used in the classification are

- i. Flow Id: For each new flow generated a unique “flow id” is generated such that the same id is calculated for both directions of flow.
- ii. Src Mac: Source media access control address is the point where the traffic starts flowing (host1).
- iii. Dest Mac: Destination media access control address is the final point for traffic flow (host2).
- iv. Traffic Type: Classified Network traffic that is flowing between two hosts.
- v. Forward Status: Active indicates that particular network traffic is classified at the Source mac address. Inactive indicates classified network is not reached at the Source mac address.
- vi. Reverse Status: Active indicates that particular network traffic is classified at the Destination mac address. Inactive indicates classified network is not reached at the Destination mac address.

CHAPTER 5

SUMMARY AND CONCLUSIONS

The proof-of-Concept and design of the problem is working well but it is limited in the below cases. This is unable to classify multiply flows between the same source and designation nodes. For instance: - Enter ping from have A to have B and afterward add Telnet. Traffic from getting A to getting B accept ping and Telnet 1 as a private course and carry out it as ping. After beginning and halting this way and that, the calculation doesn't erase the past one, yet the requesting segments will be refreshed with the most recent ones, in light of the normal park size and the general number of bytes. This will do the above mentioned. Assuming traffic is light during this season of night, these at least two gridlocks will be diminished, and the street sign will never again exist.

D-ITG tool was unable to generate traffic flows for video or game play because these are the frequent internet traffic this will be helpful for real world classification situations. As we gone through K means clustering performance was too poor this model needs to be improved by further investigation using of different organisms will also help.

REFERENCES

- [1] Kwon, Jungmin, Daeun Jung, and Hyunggon Park. "Traffic Data Classification using Machine Learning Algorithms in SDN Networks." In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1031-1033. IEEE, 2020.
- [2] Jiang, Wei, Mathias Strufe, and Hans D. Schotten. "Experimental results for artificial intelligence-based self-organized 5G networks." In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1-6. IEEE, 2017.
- [3] Sun, Songlin, Liang Gong, Bo Rong, and Kejie Lu. "An intelligent SDN framework for 5G heterogeneous networks." *IEEE Communications Magazine* 53, no. 11 (2015): 142-147.
- [4] Owusu, Ampratwum Isaac, and Amiya Nayak. "An Intelligent Traffic Classification in SDN-IoT: A Machine Learning Approach." In *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1-6. IEEE, 2020.
- [5] He, Ying, F. Richard Yu, Nan Zhao, Victor CM Leung, and Hongxi Yin. "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach." *IEEE Communications Magazine* 55, no. 12 (2017): 31-37.
- [6] Amaral, Pedro, Paulo F. Pinto, Luis Bernardo, and Amineh Mazandarani. "Application aware SDN architecture using semi-supervised traffic classification." In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1-6. IEEE, 2018.
- [7] Wang, Pan, Feng Ye, Xuejiao Chen, and Yi Qian. "Datanet: Deep learning based encrypted network traffic classification in sdn home gateway." *IEEE Access* 6 (2018): 55380-55391.
- [8] Hayes, Matthew, Bryan Ng, Adrian Pekar, and Winston KG Seah. "Scalable architecture for SDN traffic classification." *IEEE Systems Journal* 12, no. 4 (2017): 3203-3214.
- [9] Amaral, Pedro, Joao Dinis, Paulo Pinto, Luis Bernardo, Joao Tavares, and Henrique S. Mamede. "Machine learning in software defined networks: Data collection and traffic classification." In *2016 IEEE 24th International conference on network protocols (ICNP)*, pp. 1-5. IEEE, 2016.
- [10] Shu, Fei, ShuTing Chen, Feng Li, JianYe Zhang, and Jia Chen. "Research and implementation of network attack and defense countermeasure technology based on artificial intelligence technology." In *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 475-478. IEEE, 2020.
- [11] Moore, Andrew W., and Denis Zuev. "Internet traffic classification using bayesian analysis techniques." In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 50-60. 2005.
- [12] Xiao, Peng, Wenyu Qu, Heng Qi, Yujie Xu, and Zhiyang Li. "An efficient elephant flow detection with cost-sensitive in SDN." In *2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom)*, pp. 24-28. IEEE, 2015.
- [13] Amaral, Pedro, Joao Dinis, Paulo Pinto, Luis Bernardo, Joao Tavares, and Henrique S. Mamede. "Machine learning in software defined networks: Data collection and traffic classification." In *2016 IEEE 24th International conference on network protocols (ICNP)*, pp. 1-5. IEEE, 2016.
- [14] Li, Yunchun, and Jingxuan Li. "MultiClassifier: A combination of DPI and ML for application-layer classification in SDN." In *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)*, pp. 682-686. IEEE, 2014.
- [15] Rossi, Dario, and Silvio Valenti. "Fine-grained traffic classification with netflow data." In *Proceedings of the 6th international wireless communications and mobile computing conference*, pp. 479-483. 2010.

APPENDIX

A. SOURCE CODE

```
#!/pip install numpy
#!/pip install pandas
#!/pip install
matplotlib #!/pip
install seaborn #!/pip
install sklearn import
numpy as np import
pandas as pd from
scipy import stats
import pickle

# Plotting libraries
import seaborn as
sns
import matplotlib.pyplot as plt

# Sklearn libraries
from sklearn.preprocessing import
StandardScaler from sklearn.preprocessing
import label_binarize from
sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.linear_model import
LogisticRegression from sklearn.cluster import
DBSCAN, KMeans
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Filter warnings
warnings.filterwarnings('ignore') #filter
warnings # Show plots inline
%matplotlib inline
ping_df = pd.read_csv('ping_training_data.csv',
delimiter='\\t') voice_df =
pd.read_csv('voice_training_data.csv', delimiter='\\t') dns_df
= pd.read_csv('dns_training_data.csv', delimiter='\\t')
telnet_df = pd.read_csv('telnet_training_data.csv',
delimiter='\\t')
df = pd.concat([ping_df, voice_df, dns_df, telnet_df], ignore_index=True)
df.dropna(inplace=True)
df.drop('Forward Packets', axis=1,
inplace=True) df.drop('Forward Bytes',
axis=1, inplace=True) df.drop('Reverse
Packets', axis=1, inplace=True)
```

```
df.drop('Reverse Bytes', axis=1,  
inplace=True)
```

```
print(df.shape  
)
```

14

```

df.describe
() df.info()
df['Traffic Type'] = df['Traffic
Type'].astype('category') df['Traffic
Type'].cat.categories
df['Traffic Type'].cat.codes.head()
print('Features:',df.columns[:-
1].values) print ('Value counts:')
df['Traffic
Type'].value_counts() X =
df.drop('Traffic Type',axis=1)
y = df['Traffic Type']
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.5, random_state=101)
model =
LogisticRegression()
model.fit(X_train,y_train)
idx = 2590 #random
number
single_x_test = [df.iloc[idx].drop('Traffic
Type').tolist()] single_y_test = df.iloc[idx]['Traffic
Type'] single_prediction =
model.predict(single_x_test)
print('For this sample, our model predicted %s and it was actually %s' %
(single_prediction[0], single_y_test))
predictions =
model.predict(X_test) resultsDF
= pd.DataFrame({
    'true':y_test,
    'predicted':predictio
ns
})
resultsDF.head()
print('Accuracy: %.2f%%' %
(accuracy_score(predictions,y_test)*100))
print(pickle.format_version)
pickle.dump(model,open('LogisticRegression','wb'))
cm = confusion_matrix(predictions,y_test,
labels=y.cat.categories) print(cm)
cmDF = pd.DataFrame()

for i, row in
    enumerate(y.cat.categories): temp
    = {}
    for j, col in
        enumerate(y.cat.categories):

```

```
temp[col]=cm[i,j]  
cmDF = cmDF.append(pd.DataFrame.from_dict({row:temp},  
orient='index')) print(cmDF)
```

```
plt.figure(figsize=(6,6  
))
```

```

plt.imshow(cm, interpolation='nearest',
cmap=plt.cm.Blues) plt.title('Confusion Matrix',
fontsize=15)
plt.colorbar()
tick_marks = np.arange(len(y.unique()))
plt.xticks(tick_marks, y.unique(), rotation=45,
fontsize=12) plt.yticks(tick_marks, y.unique(),
fontsize=12) plt.xlabel('True Label', fontsize=15)
plt.ylabel('Predicted Label', fontsize=15)

for i in range(len(cm)):
    for j in range(len(cm[i])):
        color = 'black'
        if cm[i][j] > 5:
            color =
            'white'
        plt.text(j, i, format(cm[i][j]),
            horizontalalignment='cente
            r', color=color, fontsize=15)
df.drop('Traffic Type',axis=1).values[0:5]
df.drop('Traffic
Type',axis=1).values.mean(axis=0)
df.drop('Traffic
Type',axis=1).values.std(axis=0) scaler =
StandardScaler() scaler.fit(df.drop('Traffic
Type',axis=1))
scaled_data = scaler.transform(df.drop('Traffic
Type',axis=1)) scaled_data[0:5]
scaled_data.mean(axis
=0)
scaled_data.std(axis=0)
pca =
PCA(n_components=2)
pca.fit(scaled_data)
scaled_data.shape
x_pca = pca.transform(scaled_data)
x_pca.shape
pca.explained_variance_ratio_
pca.explained_variance_ratio_.sum()*
100 fig = plt.figure(figsize=(10,6))
sns.scatterplot(x_pca[:,0], x_pca[:,1], hue=df['Traffic
Type']) plt.xlabel('First Principal Component', fontsize=15)
plt.ylabel('Second Principal Component', fontsize=15)
plt.legend(fontsize=15)
X_train, X_test, y_train, y_test = train_test_split(x_pca,y.cat.codes,test_size=0.3,
random_state=101)

```



```
model = LogisticRegression()  
model.fit(X_train,y_train)  
predictions =  
model.predict(X_test)
```

```

print('Accuracy: %.2f%%' %
      (accuracy_score(predictions,y_test)*100)) x_min = x_pca[:,0].min()
x_max = x_pca[:,0].max()
y_min = x_pca[:,1].min()
y_max =
x_pca[:,1].max()
spacing = 0.01
xx, yy = np.meshgrid(np.arange(x_min, x_max, spacing), np.arange(y_min,
y_max, spacing))
Z = model.predict(np.c_[xx.ravel(),
yy.ravel()]) Z = Z.reshape(xx.shape)
plt.figure(figsize=(10,6))
plt.contourf(xx, yy, Z, cmap=plt.cm.Spectral)
sns.scatterplot(x_pca[:,0], x_pca[:,1], hue=df['Traffic Type'],
cmap=plt.cm.Spectral) plt.title('Decision Boundaries', fontsize=15)
plt.xlabel('First Principal Component',
fontsize=15) plt.ylabel('Second Principal
Component', fontsize=15) plt.tight_layout()
plt.xlim([x_min,x_max
])
plt.ylim([y_min,y_max
])
plt.legend(fontsize=1
5)
X = df.drop('Traffic
Type',axis=1) y = df['Traffic
Type']
model =
KMeans(n_clusters=len(y.cat.categories))
clusters = model.fit_predict(X)
model.cluster_centers_.shape
fig = plt.figure(figsize=(8, 3))
for i in range(len(y.cat.categories)):
    ax = fig.add_subplot(2, 2, 1 + i, xticks=[], yticks=[])
    ax.set_title(str(y.cat.categories[i]))
    ax.imshow(model.cluster_centers_[i].reshape((1, 12)),
    cmap=plt.cm.binary)
df['Traffic
Type'].value_counts()
y_codes.tolist().count(0)
#DNS
y_codes.tolist().count(1)
#Ping
y_codes.tolist().count(2)
#Telnet
y_codes.tolist().count(3)

```

```
#Voice strlabels = [""]*len(y)
for i in range(len(clusters)):
    if clusters[i]==0: strlabels[i] =
        'dns' elif clusters[i]==1: strlabels[i]
        = 'ping'
    elif clusters[i]==2: strlabels[i] = 'telnet'
```

```
elif clusters[i]==3: strlabels[i] = 'voice'
```

17

```

from sklearn.metrics import
accuracy_score accuracy_score(y,
strlabels)*100.0
from sklearn.decomposition import
PCA X = PCA(2).fit_transform(X)
y_codes = np.asarray(y.cat.codes)
labels = np.zeros_like(clusters) # Create an array of 0s with equal length of
the number of clusters

```

```

# Set labels based on the modes of the
target for i in
range(len(y.cat.categories)):
    mask = (clusters == i)
    labels[mask] =
stats.mode(y_codes[mask])[0] print (labels)
kwargs = dict(cmap =
    plt.cm.get_cmap('rainbow', 4),
    edgecolor='none', alpha=0.6)
fig, ax = plt.subplots(1, 2, figsize=(8, 4))
ax[0].scatter(X[:, 0], X[:, 1], c=clusters,
**kwargs) ax[0].set_title('learned cluster
labels')

```

```

ax[1].scatter(X[:, 0], X[:, 1], c=y_codes,
**kwargs) ax[1].set_title('true labels');
cm=confusion_matrix(y, strlabels,
labels=['ping', 'voice', 'dns', 'telnet']) cm
plt.figure(figsize=(6,6))
plt.imshow(cm, interpolation='nearest',
cmap=plt.cm.Blues) plt.title('Confusion Matrix',
fontsize=15)
plt.colorbar()
tick_marks = np.arange(len(y.unique()))
plt.xticks(tick_marks, rotation=45,
fontsize=12) plt.yticks(tick_marks,
fontsize=12) plt.xlabel('True Label',
fontsize=15) plt.ylabel('Predicted Label',
fontsize=15)

```

```

for i in range(len(cm)):
    for j in range(len(cm[i])):
        color = 'black'
        if cm[i][j] > 5:
            color =
            'white'
        plt.text(j, i, format(cm[i][j]),

```

horizontalalignment='cente
r',

```
color=color, fontsize=15)  
pickle.dump(model,open('KMeans_Clustering',  
wb'))
```

```

from prettytable import PrettyTable #to display output from ML
model
import subprocess, sys #to handle the Ryu
output import signal #for timer
import os #for process handling
import numpy as np #for model
features import pickle #to use ML
model real-time ## command to run
##
cmd = "sudo ryu run /usr/local/lib/python2.7/dist-
packages/ryu/app/simple_monitor_AK.py"
flows = {} #empty flow dictionary
TIMEOUT = 15*60 #15 min #how long to collect training data

class Flow:
    def __init__(self, time_start, datapath, inport, ethsrc, ethdst,
        output,
        packets, bytes):
        self.time_start = time_start
        self.datapath = datapath
        self.inport = inport
        self.ethsrc = ethsrc
        self.ethdst = ethdst
        self.outport = output

        #attributes for forward flow direction (source -> destination)
        self.forward_packets = packets
        self.forward_bytes = bytes
        self.forward_delta_packets = 0
        self.forward_delta_bytes = 0
        self.forward_inst_pps = 0.00
        self.forward_avg_pps = 0.00
        self.forward_inst_bps = 0.00
        self.forward_avg_bps = 0.00
        self.forward_status = 'ACTIVE'
        self.forward_last_time = time_start

        #attributes for reverse flow direction (destination -> source)
        self.reverse_packets = 0
        self.reverse_bytes = 0
        self.reverse_delta_packets = 0
        self.reverse_delta_bytes = 0

```



```
self.reverse_inst_pps = 0.00
```

```
self.reverse_avg_pps =
```

```
0.00
```

19

```
self.reverse_inst_bps = 0.00
```

```
self.reverse_avg_bps = 0.00
```

```
self.reverse_status = 'INACTIVE'
```

```
self.reverse_last_time = time_start
```

#updates the attributes in the forward flow direction

```
def updateforward(self, packets, bytes, curr_time):
```

```
    self.forward_delta_packets = packets - self.forward_packets
```

```
    self.forward_packets = packets
```

```
    if curr_time != self.time_start: self.forward_avg_pps =  
packets/float(curr_time-self.time_start)
```

```
    if curr_time != self.forward_last_time: self.forward_inst_pps =  
self.forward_delta_packets/float(curr_time-self.forward_last_time)
```

```
    self.forward_delta_bytes = bytes - self.forward_bytes
```

```
    self.forward_bytes = bytes
```

```
    if curr_time != self.time_start: self.forward_avg_bps =  
bytes/float(curr_time-self.time_start)
```

```
    if curr_time != self.forward_last_time: self.forward_inst_bps =  
self.forward_delta_bytes/float(curr_time-self.forward_last_time)
```

```
    self.forward_last_time = curr_time
```

```
    if (self.forward_delta_bytes==0 or  
self.forward_delta_packets==0): #if the flow did not receive any  
packets of bytes
```

```
        self.forward_status = 'INACTIVE'
```

```
    else:
```

```
        self.forward_status = 'ACTIVE'
```

#updates the attributes in the reverse flow direction

```
def updatereverse(self, packets, bytes, curr_time):
```

```
    self.reverse_delta_packets = packets - self.reverse_packets
```

```
    self.reverse_packets = packets
```

```
    if curr_time != self.time_start: self.reverse_avg_pps =  
packets/float(curr_time-self.time_start)
```

```
    if curr_time != self.reverse_last_time: self.reverse_inst_pps =  
self.reverse_delta_packets/float(curr_time-self.reverse_last_time)
```

```
    self.reverse_delta_bytes = bytes - self.reverse_bytes
```

```
    self.reverse_bytes = bytes
```

```

        if curr_time != self.time_start: self.reverse_avg_bps =
bytes/float(curr_time-self.time_start)
        if curr_time != self.reverse_last_time: self.reverse_inst_bps =
self.reverse_delta_bytes/float(curr_time-self.reverse_last_time)
        self.reverse_last_time = curr_time

```

20

```

        if (self.reverse_delta_bytes==0 or
self.reverse_delta_packets==0): #if the flow did not receive any
packets of bytes
            self.reverse_status = 'INACTIVE'
        else:
            self.reverse_status = 'ACTIVE'

```

#function to print flow attributes and output of ML model to classify the flow

```

def printclassifier(model):
    x = PrettyTable()
    x.field_names = ["Flow ID", "Src MAC", "Dest MAC", "Traffic
Type","Forward Status","Reverse Status"]

    for key,flow in flows.items():
        features =
np.asarray([flow.forward_delta_packets,flow.forward_delta_bytes,flo
w.forward_inst_pps,flow.forward_avg_pps,flow.forward_inst_bps,
flow.forward_avg_bps,
flow.reverse_delta_packets,flow.reverse_delta_bytes,flow.reverse_in
st_pps
,flow.reverse_avg_pps,flow.reverse_inst_bps,flow.reverse_avg_bps]
).reshape(1,-1) #convert to array so the model can understand the
features properly

        label = model.predict(features.tolist()) #if model is supervised
(logistic regression) then the label is the type of traffic

```

#if the model is unsupervised, the label is a cluster number. Refer to Jupyter notebook to see how cluster numbers map to labels

```

        if label == 0: label = ['dns']
        elif label == 1: label = ['ping']
        elif label == 2: label = ['telnet']
        elif label == 3: label = ['voice']

```

```

x.add_row([key, flow.ethsrc, flow.ethdst,
label[0],flow.forward_status,flow.reverse_status])
print(x)#print output in pretty mode (i.e. formatted table)

```

#function to print flow attributes when collecting training data

```

def
printflows(traffic_type,f):

```

21

```

for key,flow in flows.items():
    outstring = '\t'.join([
        str(flow.forward_packets),
        str(flow.forward_bytes),
        str(flow.forward_delta_packets),
        str(flow.forward_delta_bytes),
        str(flow.forward_inst_pps),
        str(flow.forward_avg_pps),
        str(flow.forward_inst_bps),
        str(flow.forward_avg_bps),
        str(flow.reverse_packets),
        str(flow.reverse_bytes),
        str(flow.reverse_delta_packets),
        str(flow.reverse_delta_bytes),
        str(flow.reverse_inst_pps),
        str(flow.reverse_avg_pps),
        str(flow.reverse_inst_bps),
        str(flow.reverse_avg_bps),
        str(traffic_type)])
    f.write(outstring+'\n')

```

```

def run_ryu(p,traffic_type=None,f=None,model=None):

```

```

    ## run it ##

```

```

    time = 0

```

```

    while True:

```

```

        #print 'going through loop'

```

```

        out = p.stdout.readline()

```

```

        if out == " and p.poll() != None:

```

```

            break

```

```

        if out != " and out.startswith(b'data'): #when Ryu

```

'simple_monitor_AK.py' script returns output

```

            fields = out.split(b'\t')[1:] #split the flow details

```

```

            fields = [f.decode(encoding='utf-8', errors='strict') for f in
fields] #decode flow details

```

```

        unique_id = hash(".".join([fields[1],fields[3],fields[4]])) #create
        unique
ID for flow based on switch ID, source host,and destination host
        if unique_id in flows.keys():

flows[unique_id].updateforward(int(fields[6]),int(fields[7]),int(fields[0]))
#update forward attributes with time, packet, and byte count
                22

        else:
            rev_unique_id = hash(".".join([fields[1],fields[4],fields[3]]))
            #switch
            source and destination to generate same hash for src/dst and dst/src
            if rev_unique_id in flows.keys():

flows[rev_unique_id].updatereverse(int(fields[6]),int(fields[7]),int(fields[
0])) #update reverse attributes with time, packet, and byte count
            else:
                flows[unique_id] = Flow(int(fields[0]), fields[1], fields[2],
fields[3], fields[4], fields[5], int(fields[6]), int(fields[7])) #create new flow
object
                if not model is None:
                    if time%10==0: #print output of model every 10
                        seconds
                        printclassifier(model)
                else:
                    printflows(traffic_type,f) #for training data
            time += 1

#print help output in case of incorrect options
def printHelp():
    print("Usage: sudo python traffic_classifier.py [subcommand]
[options]")
    print("\tTo collect training data for a certain type of traffic, run: sudo
python traffic_classifier.py train [voice|video|ftp]")
    print("\tTo start a near real time traffic classification application using
unsupervised ML, run: sudo python traffic_classifier.py unsupervised")
    print("\tTo start a near real time traffic classification application using
supervised ML, run: sudo python traffic_classifier.py supervised")
    return

#for timer to collect flow training data
def alarm_handler(signum, frame):
    print("Finished collecting data.")
    raise Exception()

```



```

if __name__ == '__main__':
    SUBCOMMANDS = ('train', 'unsupervised', 'supervised')

    if len(sys.argv) < 2:
        print("ERROR: Incorrect # of args")
        print()
        printHelp()
        sys.exit(
            );

    else:
        if len(sys.argv) == 2:
            if sys.argv[1] not in SUBCOMMANDS:
                print("ERROR: Unknown subcommand argument.")
                print("    Currently subaccepted commands are: %s" %
str(SUBCOMMANDS).strip('()'))
                print()
                printHelp()
                sys.exit();

            if len(sys.argv) == 1:
                # Called with no arguments
                printHelp()
            elif len(sys.argv) >= 2:
                if sys.argv[1] == "train":
                    if len(sys.argv) == 3:
                        p = subprocess.Popen(cmd, shell=True,
                        stdout=subprocess.PIPE,
stderr=subprocess.STDOUT) #start Ryu process
                        traffic_type = sys.argv[2]
                        f = open(traffic_type+'_training_data.csv', 'w') #open training
                        data
                        output file
                        signal.signal(signal.SIGALRM, alarm_handler) #start signal
                        process
                        signal.alarm(TIMEOUT) #set for 15 minutes
                        try:
                            headers = 'Forward Packets\tForward Bytes\tDelta Forward
Packets\tDelta Forward Bytes\tForward Instantaneous Packets per
Second\tForward Average Packets per second\tForward
Instantaneous Bytes per Second\tForward Average Bytes per
second\tReverse Packets\tReverse Bytes\tDelta Reverse
Packets\tDelta Reverse Bytes\tDeltaReverse Instantaneous
Packets per Second\tReverse Average Packets per
second\tReverse Instantaneous Bytes per Second\tReverse
Average Bytes per second\tTraffic Type\n'
                            f.write(headers)
                            run_ryu(p,traffic_type=traffic_type,f=f)

```

```

        except Exception:
            print('Exiting')
            os.killpg(os.getpgid(p.pid), signal.SIGTERM) #kill ryu
            process on
exit
        f.close()
    else:
        print("ERROR: specify traffic type.\n")
        printHelp(
            )
23
    else:
        p = subprocess.Popen(cmd, shell=True,
            stdout=subprocess.PIPE,
            stderr=subprocess.STDOUT) #start ryu process
        if sys.argv[1] == 'supervised':
            infile = open('LogisticRegression','rb')
            elif sys.argv[1] == 'unsupervised':
                infile = open('KMeans_Clustering','rb')
                model = pickle.load(infile) #unload previously trained ML model
                (refer to Jupyter notebook for details)
                infile.close()
                run_ryu(p,model=model)
        sys.exit();

```


B. SCREEN SHOTS

```
ubuntu@ubuntu-VirtualBox:~/Desktop/trafficclassifier$ sudo python3 traffic_classifier_python3.py supervised
[sudo] password for ubuntu:
/usr/lib/python3/dist-packages/sklearn/base.py:313: UserWarning: Trying to unpickle estimator LogisticRegression from version
valid results. Use at your own risk.
warnings.warn(
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | ACTIVE | ACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | ACTIVE | ACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | ACTIVE | ACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | INACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| 8803204805936083895 | 4e:ea:50:20:10:c9 | 46:e9:41:a7:f0:56 | ping | INACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
```

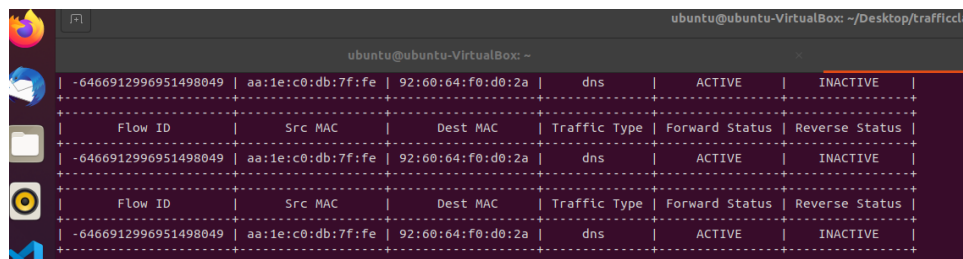
classification of network traffic as **ping** and updating the status of the traffic flow as Active/Inactive

```
ubuntu@ubuntu-VirtualBox:~/Desktop/trafficclassifier$ sudo python3 traffic_classifier_python3.py supervised
[sudo] password for ubuntu:
/usr/lib/python3/dist-packages/sklearn/base.py:313: UserWarning: Trying to unpickle estimator LogisticRegression from version 1
valid results. Use at your own risk.
warnings.warn(
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -650268477036915217 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | ping | ACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -650268477036915217 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | telnet | ACTIVE | ACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -650268477036915217 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | telnet | ACTIVE | ACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -650268477036915217 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | telnet | INACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -650268477036915217 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | telnet | INACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
```

classification of network traffic as **telnet** and updating the status of the traffic flow as Active/Inactive

```
+-----+-----+-----+-----+-----+-----+
| -6466912996951498049 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | voip | ACTIVE | ACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -6466912996951498049 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | voip | ACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -6466912996951498049 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | voip | ACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -6466912996951498049 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | voip | ACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -6466912996951498049 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | voip | ACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -6466912996951498049 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | voip | ACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -6466912996951498049 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | voip | ACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -6466912996951498049 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | voip | INACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
| Flow ID | Src MAC | Dest MAC | Traffic Type | Forward Status | Reverse Status |
+-----+-----+-----+-----+-----+-----+
| -6466912996951498049 | aa:1e:c0:db:7f:fe | 92:60:64:f0:d0:2a | voip | INACTIVE | INACTIVE |
+-----+-----+-----+-----+-----+-----+
```

classification of network traffic as **voip**(voice traffic) and updating the status of the traffic flow as Active/Inactive



The screenshot shows a terminal window with a table of network traffic data. The table has columns for Flow ID, Src MAC, Dest MAC, Traffic Type, Forward Status, and Reverse Status. The data is as follows:

Flow ID	Src MAC	Dest MAC	Traffic Type	Forward Status	Reverse Status
-6466912996951498049	aa:1e:c0:db:7f:fe	92:60:64:f0:d0:2a	dns	ACTIVE	INACTIVE

classification of network traffic as **dns** and updating the status of the traffic flow as Active/Inactive

TRAFFIC CLASSIFICATION OF SDN NETWORK USING MACHINE LEARNING ALGORITHMS

26

NERELLA PAVAN KALYAN GOUD
B.E - CSE
Sathyabama Institute of Science and
Technology, Chennai, India.
pavankalyan34201@gmail.com

G.SAI CHARAN REDDY
B.E - CSE
Sathyabama Institute of Science
and
Technology, Chennai, India.

A.MARY POSONIA
Associate professor , Dept of CSE
Sathyabama Institute of Science and
Technology, Chennai, India.
maryposonia.cse@sathyabama.ac.in

Abstract — Traffic classification is becoming an important research area especially with the day-to-day machine learning software defined networking advancements. It is utilized for networking executives, administration estimation, network plan, security observing, and publishing. The proposed model used two machine learning models logistic regression and kmeans are used to classify ping, telnet, dns, and voice flows (VoIP) with the utilization of an Internet-based motor (DITG) that utilizes two techniques for Artificial Intelligence (AI). The host on these organizations is associated with the open vswitch (Ovs) over the two organizations. The Ovs are joined by a reviewer named Ryu (signifying "streaming" in Japanese. Ryu is designated "ree-yooh") and gathers street figures between the beneficiaries. These figures are examined by Python's robotized archive arranging framework and learned street signs are carried out. The proposed work will classify the type of network traffic flowing between two hosts.

Keywords — D-ITG, Ovs, Vswitch, Ryu, Qos,Dns,Voip,AI

INTRODUCTION

A significant piece of examination is the constant improvement of AI and programming that characterizes the standards of coordination of exercises that assistance to sort out the progression of traffic in the internet protocol line. As it was done in the good old days, for example, distinguishing the port number and really taking a look at the heap because of the roadway and the put away things, isn't working as expected, and the venture will assist with utilizing automated and automated AI machines to decrease the quantity of vehicles utilizing their required and custom code ground floor. The requesting of vehicles utilizing AI

illustration of

this is Skype an application used for video and Voice call application, which generates VoIP traffic. where even experts find it challenging to classify and dissect huge volumes of movement data [5]. Another illustration is aha a video streaming application in this quality of service is important. To maintain quality the traffic flowing to the user should be classified to give a best streaming experience. These Machine learning sequencing and connecting traffic can assist with characterizing network availability through Qos and broadband. This is significant in picking the future and the requirements of the organization, and it is exceptionally energizing to give web, government, and business administrations to get to the street. Posting of vehicles in light of controlled preparing to decide the phases of the street utilizing known wellsprings of data. The technique for AI is depicted in [11], [12] [13], [14], and the creators of [13] present the development of SDNs, which makes street information in view of OpenFlow. Many learning calculations are currently utilized for information gathered to be executed in an assortment of ways. The study was led on an organization of organizations.

is expanding in the organization investigation. An

LITERATURE SURVEY

There has been a ton of examination on programming traffic, and a great deal of exploration has been done on this thought a good survey can be found here: Chen, Joy long Zong, and P. Hengjinda. "Early Prediction of Coronary Artery Disease (CAD) by Machine Learning Method-A Comparative Study." Journal of Artificial Intelligence in this an image pre-processing technique is used that helped in provoking an idea of classification based on the data packet size parameter [16]. Another study by Balasubramaniam, Vivekanadam. "Artificial Intelligence Algorithm with SVM Classification using Dermoscopic Images for Melanoma Diagnosis." Journal of Artificial Intelligence and Capsule Networks the study is all about predicting skin cancer. Kmeans clustering algorithm is used for achieving the goal. From this a clear working visualization of K-means clustering is achieved [17]. And one of which was finished by Zhongmin Kwon, Daeung Jung, and Hyung Park on the "Traffic Data Classification utilizing AI Amahugur Road Traffic Classification" "Figure out the product pipeline by 2020. In this review, they attempted to show the primary aftereffects of a mechanized information arranging calculation in view of AI calculations." Obstacles to the investigation of genuine organization-based AI, notwithstanding the surprising data gave in network the board. Investigating the gathered street information and the product that depicts the ecological organization, involving a portion of the calculations in the rush hour gridlock data preparing [1]. This study clarifies that it is feasible to observe an AI calculation for breaking down data utilizing constant correspondence networks gathered with a SDN regulator in two unique ways. The proposed concentrate on intends to expand the precision of AI calculations to accomplish this objective, and to test it on an immediate organization to more readily decide the degree of traffic. Concentrates by Jungmin Kwon, Daeun Jung, and Hynggon show that AI machines can be utilized in the organization. Foster an organization of scientific AI calculations for future work. This is a point-by-point depiction of the exploration exercises in the field of street reporting utilizing AI "[1]. As the requests on them in a bigger and more mind-boggling manner increment, pipeline the board and checking are a significant stage in taking care of the issue. This should be possible with the assistance of street the board, otherwise called

vehicle forecast, which requires the foundation of a street network that interfaces various sorts of pipelines to guarantee the appropriate preparation of the street. Posting is an illustration of an instructional class that is controlled and used to decide the phase of another bearing connected with the data gave. For instance, contingent upon the configuration of the email, for example, the first IP address, the length of the text, and the text, the email can be carried out as spam or spam. [2] The conventional strategies for positioning IP traffic that have been utilized for a long-time incorporate port-based techniques, profundity estimations, and that's just the beginning. The port-based technique empowers the grouping of vehicles or applications utilizing the port number of the exchange control and the client of the datagram convention. Albeit numerous applications or other port numbers have been utilized, this technique doesn't work. An inside and out test-pack is a pricey test and uses a ton of framework hardware. DPI doesn't work on the grounds that the substance of this bundle can't be utilized or gotten to.

Analysts have been attempting to involve a more top to bottom review technique. One of the analysts proposed a SAE-based way to deal with distinguishing stowed away data. In any case, the data gave was not open to general society, and the work that showed the secret street necessities didn't give an exhibition examination [6]. Late exploration has zeroed in on distinguishing and choosing the attributes of the stream to recognize the stream. Programming combination shows that a few things are a higher priority than meters that open the market and travel well, and that it makes them more proficient [8].

Networks characterized by programming give great standards on the best way to oversee parks in view of data in the head. It can likewise give rules to sending numbers, like groups and bytes. OpenFlow can be utilized in an assortment of ways of recovering genuine data. The idea of this regulation is void, and it rehashes the law after a timeframe without appropriate data [7].

AI calculations are utilized to foster the utilization of the Internet to accomplish energy saving and organization the board. Customary security techniques, for example, the utilization of handheld gadgets are the primary explanations behind inadequate security and space and reality. Complete vehicle classes can lessen the gamble [9].

PROPOSED SYSTEM

The architecture diagram of figure-2 explains about the steps involved in order to follow for the successful completion the predefined abstract. These steps are explained one after the other briefly.



Fig: 1 gives a clear idea of the steps involved in achieving the goal explained in abstract

A. VirtualBox Network Topology: -

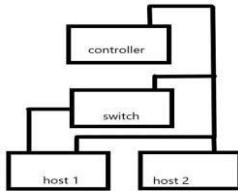


Fig: 2 Helps in Visualizing Topology used in the proposed work.

In real life process a controller, switch two host are required to simulate the network traffic flow. To achieve this virtually an instant virtual network is needed there comes the importance of Mininet. Mininet is an impressionist for deploying large networks on the finite resources of a simple virtual machine. To visualize the topology of Mininet the above figure helps. After the successful setup of Mininet topology the controller should be conscious of the traffic packets flowing through the Switch between any of the host. The python script and simple_monitor_python.py will be updated to display the below flow information each second. To get everything rolling, utilize the accompanying order `sudo mn - regulator = remote, lp = 127.0.0.1 - - top single, 3` this order will begin the regulator and open the Mininet terminal. Presently utilize the `xterm h1` and `xterm h2` orders to open the new Mininet Terminal, contrasting h1 and h2 as two.

```
f system. Arguments [1] == "train": // checking the command argument
open (order, shell = True, stdout = subprocess.PIPE, stderr = subprocess.STDOUT)//Start the interaction
open (traffic_type + '_ exercise_data.csv', 'w')//open result record
sign. (Image. SIGALRM, UI)//begin the person cycle sign. Notice (TIME)//Set inside 15 minutes
Operating system. erase way (os.getpgid (p.pid), image. SIGTERM)//Delete Ryu way and exit
close ()//Close numbers
```

B. Simulation of traffic flows:

The kind of organization traffic we use to group ping, DNS, Telnet, and standard voice. Ping traffic can be produced from a command line interface from one host to the other. In order to generate the remaining traffic flows D-ITG is utilized to make different flows of the Internet. Web circulation is classified "a website that can successfully make ipv4 and IPv6 by copying the usefulness of existing Internet applications." D-ITG can create traffic utilizing inaccurate model of park size (PS) and inter departure time (IDT) that emulates conduct levels. D-ITG is equipped for creating traffic insights for various exercises, for example, telnet and voip. Further work and limits depict the selection of boundaries to mimicked gadgets and the difficulties of utilizing D-ITG.

Steps to create traffic through conveyed Internet dispersion:

Stage 1: **Telnet** traffic comprises of telnet capacities. It will work with different setup convention control conventions overlooked.

Order utilized: `cd/Desktop/D-ITG-2.8.1-r1023/bash telnet`

Stage 2: **DNS** traffic comprises of DNS capacities that work with the datagram client convention and the control convention.

Order utilized: `cd/Desktop/D-ITG-2.8.1-r1023/bash dns.sh`

Stage 3: **VOICE** generate traffic which consists of voice characteristics this will work with user datagram protocol and different settings were ignored.

Order utilized: `cd/Desktop/D-ITG-2.8.1-r1023/bash voice`

C. IMPLEMENTATION OF TRAINING MODELS: Sequence of command involved for collecting the dataset

Order utilized: `Sudo python3 traffic_classifier_python3.py`

Implementing Network traffic classification using Logistic Regression:

Calculated relapse is the principal control of the AI calculations utilized in research. The result indicator is a parallel worth, yet assuming there are multiple objectives, pick an objective that has whatever number prospects as could be expected under the circumstances. In the program, I utilized a calculated relapse that functioned admirably with an exactness of ~ close to 99%. This is a design with the initial two standards whose fundamental limits are genuine.

Formula for accuracy = $2((\text{precision} * \text{recall}) / (\text{precision} + \text{recall}))$.

It is somewhere in the range of 0 and 1. Over the worth, this is a genuine model.

In the immediate test, the example rate was 0.90.

```
[23] print('Accuracy: %.2f%%' % (accuracy_score(predictions,y_test)*100))  
Accuracy: 98.89%
```

We see the model has a **98.89%** accuracy

Fig: 3 Accuracy obtained by logistic regression algorithm. Benefiting as much as possible from the reason for the disarray framework is the right decision. A deception framework is a grid marked as accepted characters on the y-hub and valid on the x-pivot. Any model that endeavors to foresee street segments for another example will be explained utilizing Figure 3, which outlines the above mentioned. To accurately decide the target a confusion matrix is a perfect choice.

Confusion matrix is a matrix with predicted labels on the y-axis and true labels on the x-axis. Any model which is trying to predict certain traffic classes on another class is more of then it will clearly evident using confusion Matrix figure 3 explain the above.

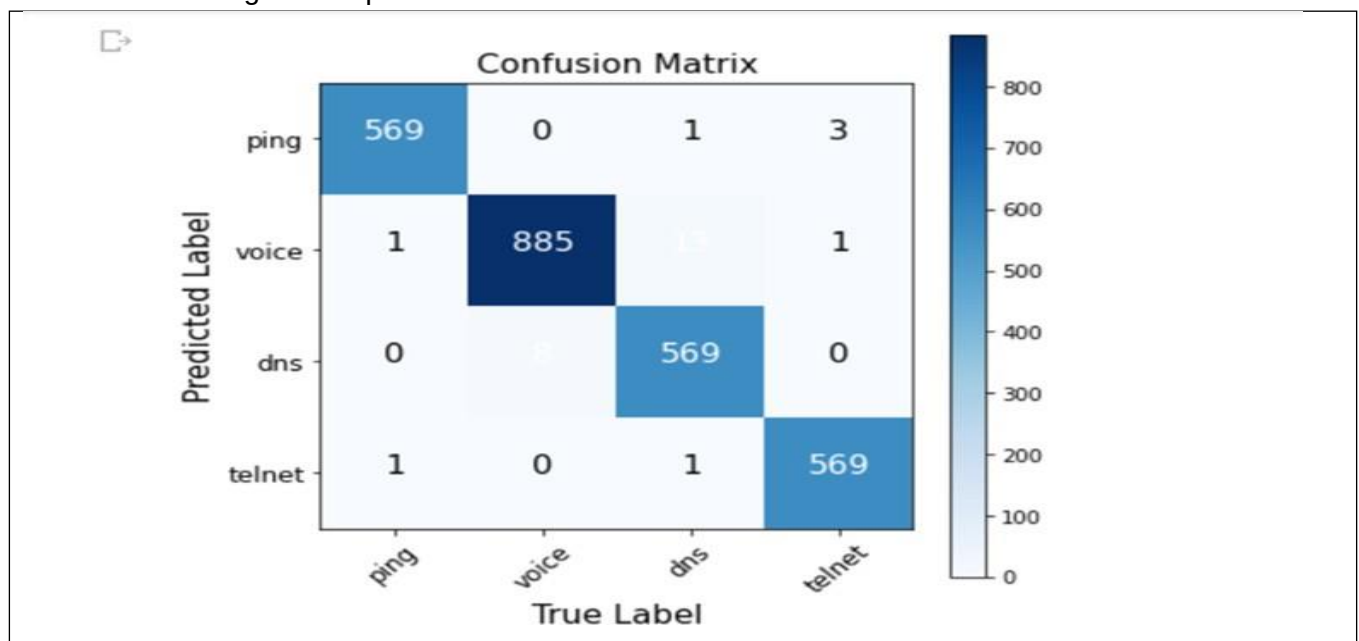


Fig: 4 To accurately decide the target the above confusion matrix is a used.

Implementation of Network traffic classification using K-means clustering:

K-means clustering is an unsupervised machine learning algorithm that groups data points in more than two- dimensional space together to form clusters and labels each data point in the cluster. Four clusters were used as an initial input parameter to the model for the four traffic types figure- 4 will explain to visualise each centroid of the four clusters. Where each square represents the dimension. If the dimension colour is dark then the higher the value this will display to a 12-dimensional vector space where the cluster centroids lie.

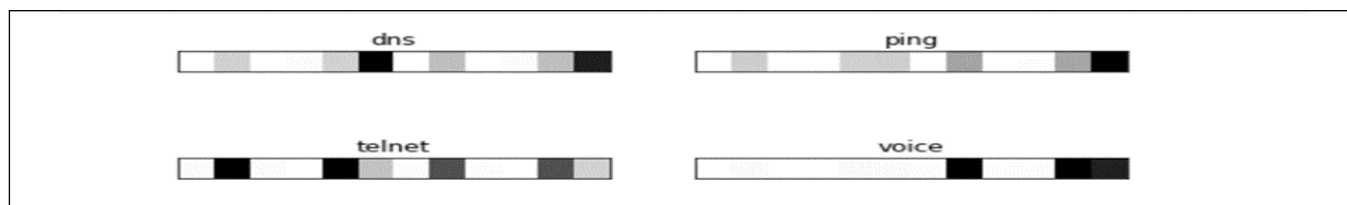


Fig: 5 Four clusters were used as an initial input parameter to the model.

For better visualization of model performance, I used principle component analysis where the reduction of number of dimensions from 12 to 2 and then plot in two- dimensional space Because of the non-auricularly groups of data the performance of the model was only around ~46% which is very poor .

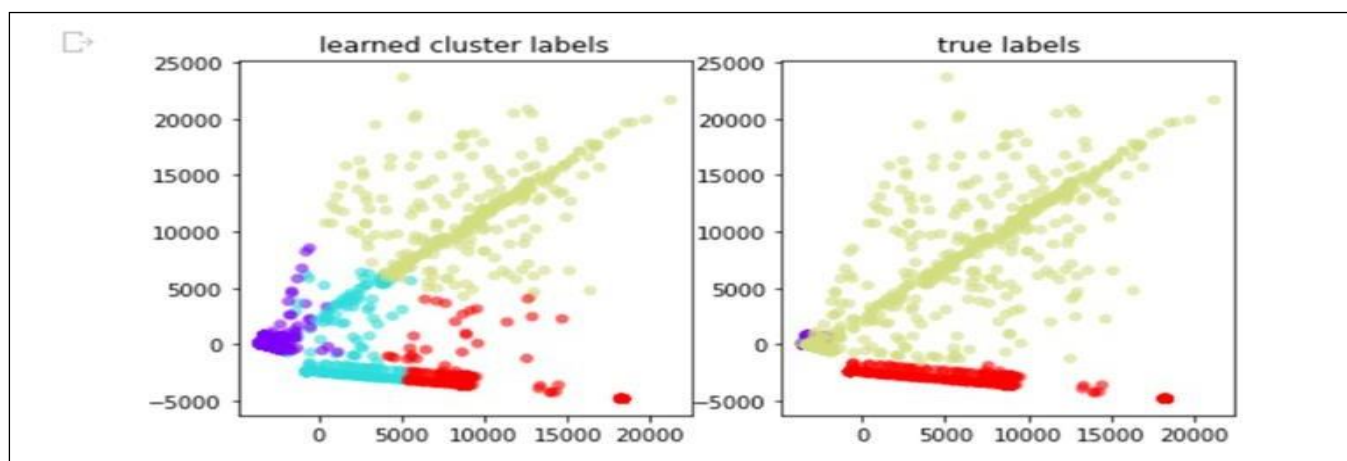


Fig: 6 explains about learned cluster labels from the training data. Right side graph explains about true labels from the testing data.

The above can be confirmed by visualizing confusion Matrix which explains a very disbalanced matrix

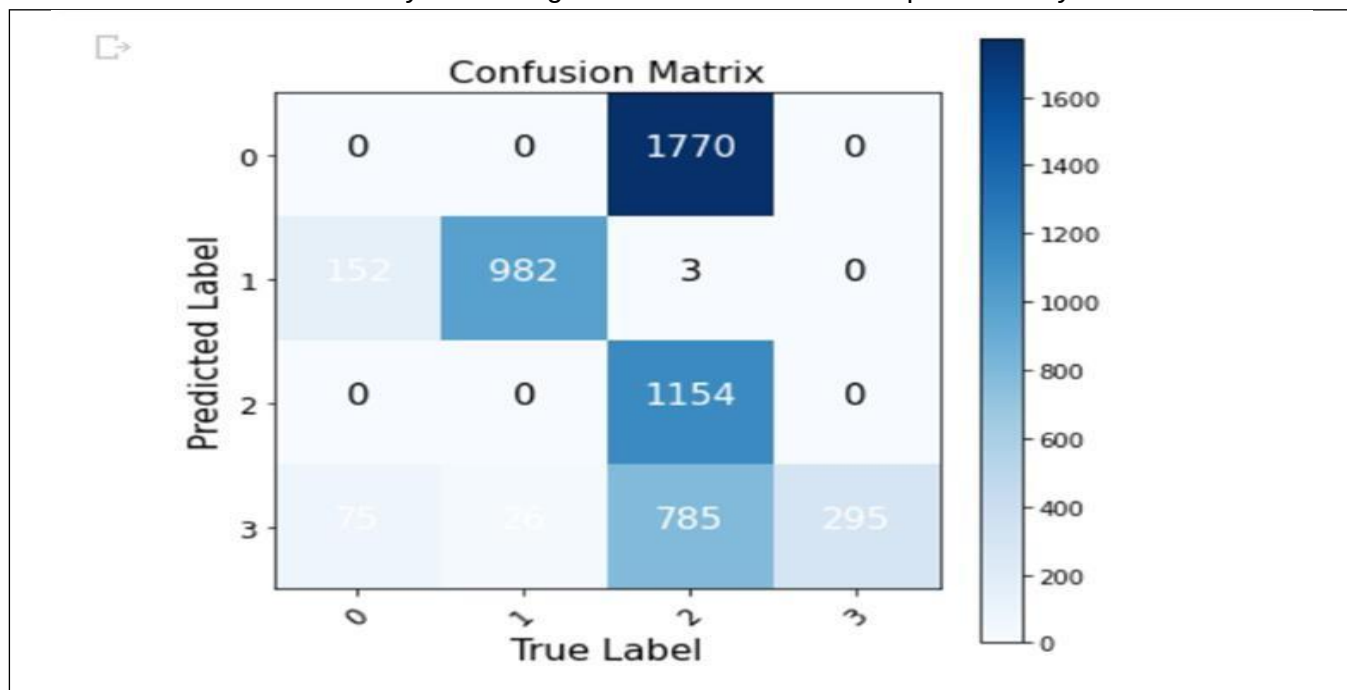


Fig: 7 Explaining about the disbalanced clustering of K-means algorithm using confusion matrix.

RESULTS

Input: Collection of training data:

The novelty of proposed work is creating a graphical user interface (GUI) for displaying the traffic type and secondly increasing the accuracy by logistic regression algorithm. The procedure to correct training data for the machine learning models are primarily to simulate the traffic flow of the specified traffic between two hosts using D-ITG tool. The secondary thing is to start network_traffic_classifier.py script. The training data generated from simple_monitor_ak.py script is collected and transfer to update the ascribe the traffic flows objects. The ascribe of the traffic flow objects are then regularly outputted to a csv file. After the completion of collection of generated traffic flow ascribes in the csv file, they all are combined in a Pandas data frame will be taken for further process of model training and testing.

COMPARING THE ACCURACY OF THE ALGORITHMS USED.

Algorithm	Training Accuracy	Test Accuracy
Logistic regression	98.89%	84.16%
K-Means Clustering	45.98%	39.32%

The results obtained from the proposed work are displayed below and they are discussed to get a clear picture to the users and validators. Below figures explain about the live classification of network traffic. [Fig 7] classify the traffic flow as ping traffic and display the status of the traffic as 'active'. After stopping the flow between the hosts, it will update to 'Inactive'. [Fig 8] classify the traffic flow as telnet traffic and display the status of the traffic as 'active'. After stopping the flow between the hosts, it will update to 'Inactive'. [fig 9] classify the traffic flow as voice traffic and display the status of the traffic as 'active'. After stopping the flow between the hosts, it will update to 'Inactive'.

The below figures 8,9,10,11 are the proof of classification

Flow ID	Src MAC	Dest MAC	Traffic Type	Forward Status	Reverse Status
8803204805936083895	4e:ea:50:20:10:c9	46:e9:41:a7:f0:56	ping	ACTIVE	INACTIVE

Fig:8 classification of network traffic as **ping** and updating the status of the traffic flow as Active/Inactive

Flow ID	Src MAC	Dest MAC	Traffic Type	Forward Status	Reverse Status
650268477836915217	aa:1e:c8:db:7f:fe	92:08:04:f0:d0:2a	telnet	ACTIVE	INACTIVE

Fig: 8 classification of network traffic as **telnet** and updating the status of the traffic flow as Active/Inactive state

Flow ID	Src MAC	Dest MAC	Traffic Type	Forward Status	Reverse Status
6406912990951498049	aa:1e:c8:db:7f:fe	92:08:04:f0:d0:2a	voip	ACTIVE	INACTIVE

Fig: 9 classification of network traffic as **VoIP** (voice traffic) and updating the status of the traffic flow as Active/Inactive

Flow ID	Src MAC	Dest MAC	Traffic Type	Forward Status	Reverse Status
6406912990951498049	aa:1e:c8:db:7f:fe	92:08:04:f0:d0:2a	dns	ACTIVE	INACTIVE

Fig: 10 classification of network traffic as **dns** and updating the status of the traffic flow as Active/Inactive

The attributes used in the classification are

- i. Flow Id: For each new flow generated a unique "flow id" is generated such that the same id is calculated for both directions of flow.
- ii. Src Mac: Source media access control address is the point where the traffic starts flowing (host1).
- iii. Dest Mac: Destination media access control address is the final point for traffic flow (host2).
- iv. Traffic Type: Classified Network traffic that is flowing between two hosts.
- v. Forward Status: Active indicates that particular network traffic is classified at the Source mac address. Inactive indicates classified network is not reached at the Source mac address.
- vi. Reverse Status: Active indicates that particular network traffic is classified at the Destination mac address. Inactive indicates classified network is not reached at the Destination mac address.

CONCLUSION

The proof-of-Concept and design of the problem is working well but it is limited in the below cases. This is unable to classify multiply flows between the same source and designation nodes. For instance:

- Enter ping from have A to have B and afterward add Telnet. Traffic from getting A to getting B accept ping and Telnet 1 as a private course and carry out it as ping. Subsequent to beginning and halting this way and that, the calculation doesn't erase the past one, yet the requesting segments will be refreshed with the most recent ones, in light of the normal park size and the general number of bytes. This will do the above mentioned.

Assuming traffic is light during this season of night, these at least two gridlocks will be diminished and the street sign will never again exist.

D-ITG tool was unable to generate traffic flows for video or game play because these are the frequent internet traffic this will helpful for real world classification situations. As we gone through K means clustering performance was too poor this model needs to be improved by further investigation using of different organisms will also help.

REFERENCES

- [1] Kwon, Jungmin, Daeun Jung, and Hyunggon Park. "Traffic Data Classification using Machine Learning Algorithms in SDN Networks." In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1031-1033. IEEE, 2020.
- [2] Jiang, Wei, Mathias Strufe, and Hans D. Schotten. "Experimental results for artificial intelligence-based self-organized 5G networks." In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1-6. IEEE, 2017.
- [3] Sun, Songlin, Liang Gong, Bo Rong, and Kejie Lu. "An intelligent SDN framework for 5G heterogeneous networks." *IEEE Communications Magazine* 53, no. 11 (2015): 142-147.
- [4] Owusu, Ampratwum Isaac, and Amiya Nayak. "An Intelligent Traffic Classification in SDN-IoT: A Machine Learning Approach." In *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1-6. IEEE, 2020.
- [5] He, Ying, F. Richard Yu, Nan Zhao, Victor CM Leung, and Hongxi Yin. "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach." *IEEE Communications Magazine* 55, no. 12 (2017): 31-37.
- [6] Amaral, Pedro, Paulo F. Pinto, Luis Bernardo, and Amineh Mazandarani. "Application aware SDN architecture using semi-supervised traffic classification." In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1-6. IEEE, 2018.
- [7] Wang, Pan, Feng Ye, Xuejiao Chen, and Yi Qian. "Datanet: Deep learning based encrypted network traffic classification in sdn home gateway." *IEEE Access* 6 (2018): 55380-55391.
- [8] Hayes, Matthew, Bryan Ng, Adrian Pekar, and Winston KG Seah. "Scalable architecture for SDN traffic classification." *IEEE Systems Journal* 12, no. 4 (2017): 3203-3214.
- [9] Amaral, Pedro, Joao Dinis, Paulo Pinto, Luis Bernardo, Joao Tavares, and Henrique S. Mamede. "Machine learning in software defined networks: Data collection and traffic classification." In *2016 IEEE 24th International conference on network protocols (ICNP)*, pp. 1-5. IEEE, 2016. [10] Shu, Fei, ShuTing Chen, Feng Li, JianYe Zhang, and Jia Chen. "Research and implementation of network attack and defense countermeasure technology based on artificial intelligence technology." In *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 475-478. IEEE, 2020.
- [11] Moore, Andrew W., and Denis Zuev. "Internet traffic classification using bayesian analysis techniques." In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 50-60. 2005.
- [12] Xiao, Peng, Wenyu Qu, Heng Qi, Yujie Xu, and Zhiyang Li. "An efficient elephant flow detection with costsensitive in SDN." In *2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom)*, pp. 24-28. IEEE, 2015.
- [13] Amaral, Pedro, Joao Dinis, Paulo Pinto, Luis Bernardo, Joao Tavares, and Henrique S. Mamede. "Machine learning in software defined networks: Data collection and traffic classification." In *2016 IEEE 24th International conference on network protocols (ICNP)*, pp. 1-5. IEEE, 2016. [14] Li, Yunchun, and Jingxuan Li. "MultiClassifier: A combination of DPI and ML for application-layer classification in SDN." In *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)*, pp. 682-686. IEEE, 2014.
- [15] Rossi, Dario, and Silvio Valenti. "Fine-grained traffic classification with netflow data." In *Proceedings of the 6th international wireless communications and mobile computing conference*, pp. 479-483. 2010. [16] Chen, Joy long Zong, and P. Hengjinda. "Early prediction of coronary artery disease (cad) by machine learning method-a comparative study." *Journal of Artificial Intelligence* 3, no. 01 (2021): 17-33.
- [17] Balasubramaniam, Vivekanadam. "Artificial intelligence algorithm with SVM classification using dermoscopic images for melanoma diagnosis." *Journal of Artificial Intelligence and Capsule Networks* 3, no. 1 (2021): 34-42.

