Department of Information Technology
V R Siddhartha Engineering College

Preventing Network Attacks using Support Vector Machine and Software Defined Networking (SDN) Integration

Network Security, Cyber Security
and Information Security

B.Tech in Information Technology
Mini Project 2 Review Presentation

Presented by

Mr. Dinakar Laxmi Viswanath (208W1A1201)
Mr. Sotsava Skandhaa (208W1A1202)

Under the guidance of
S. Kranthi , Assistant professor

AGENDA

- Problem Statement
- Objectives of the Project
- Outcomes of the Project
- Literature Review and Summary
- Dataset and Requirements
- Architecture diagram
- Implementation Steps
- Algorithm
- Results

PROBLEM DEFINITION

The increasing number of cyber-attacks on networks has become a major concern for organizations and individuals. Traditional security solutions, such as firewalls and intrusion detection systems, are becoming less effective in defending against these attacks. This project aims to address this problem by using a deep learning model-based solution for detecting and preventing DDOS network attacks in a software-defined networking (SDN) environment. The solution uses a trained linear Support Vector Machine (SVM) algorithm model to control the behavior of one or more SDN controllers to prevent attacks.

OBJECTIVES

- To develop an efficient machine learning model that can classify the given network traffic dataset to various attacks with maximum accuracy.

- Using SDN controllers to stop traffic from a host based on its IP address.

OUTCOMES

- Demonstrated the system's ability to detect and respond to network attacks in a controlled environment.
- Reduced the false positive rate to 6% while maintaining a true positive rate of 93%.
- The integration of machine learning with SDN controllers can lead to more adaptive and efficient network defenses.

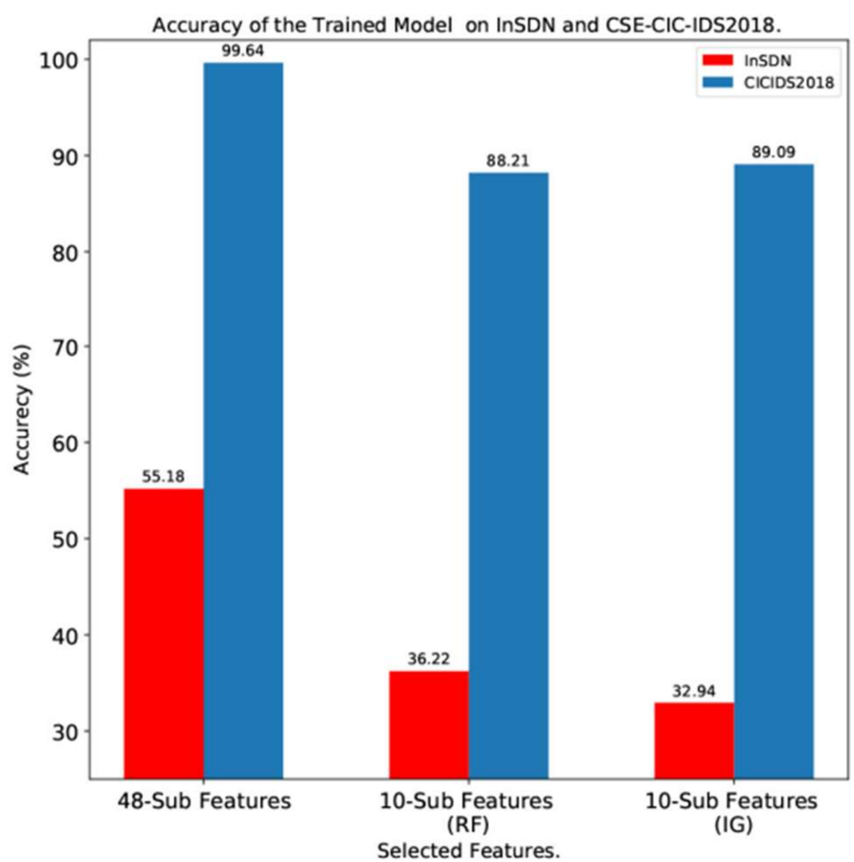
LITERATURE SURVEY

Title of the Project	Authors	Methodology	Source	Summary
A Flow-Based Anomaly Detection Approach With Feature Selection Method Against DDoS Attacks in SDNs	Mahmoud Said El Sayed Nhien-An Le-Khac , Marianne A. Azer and Anca D. Jurcut (2020)	LSTM-Autoencoder	IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING	Our approach provides a high detection rate and presents a more efficient better time to build the model. We further tested the trained model on the performance of the SDN controllers to evaluate how the used dataset can impact on the performance of the SDN controller. The results showed that the proposed approach does not deteriorate the network performance
Deep Neural Networks for Intrusion Detection in Software-Defined Networking	Wang et al. (2019)	Random Forests	IEEE (Institute of Electrical and Electronics Engineers)	The authors evaluate the performance of their proposed solution using both simulated and real-world network security datasets and show that deep neural networks can significantly improve the accuracy of intrusion detection in SDN

Title of the Project	Authors	Methodology	Source	Summary
End-to-end intrusion detection in software-defined networks using deep reinforcement learning	Qin et al. (2019)	A K-mean clustering and random forest based multilevel model .	IEEE (Transactions on Network and Service Management)	The proposed solution can effectively detect various types of network attacks in real-time and provide a flexible and scalable solution for SDN security.
Anomaly-based Intrusion Detection in Software-Defined Networks: A Deep Learning Approach	Zhang et al. (2019)	Naïve Bayes classifier	IEEE Access	The proposed method uses an autoencoder to learn the normal behavior of the network and identify anomalies, which are then classified as either benign or malicious using a deep neural network.
Distributed abnormal behavior detection approach	Naila Marir HuiqiaWa Guangshe Bingyang	DBN and SVM	IEEE, 2018	DBN is used to extract SVM ensemble characteristics and to anticipate the voting process.

REVIEW PAPER : Mahmoud Said El Sayed Nhien-An Le-Khac , Marianne A. Azer and Anca D. Jurcut. "A Flow-Based Anomaly Detection Approach With Feature Selection Method Against DDoS Attacks in SDNs." In 2022 IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING.

SUMMARY :



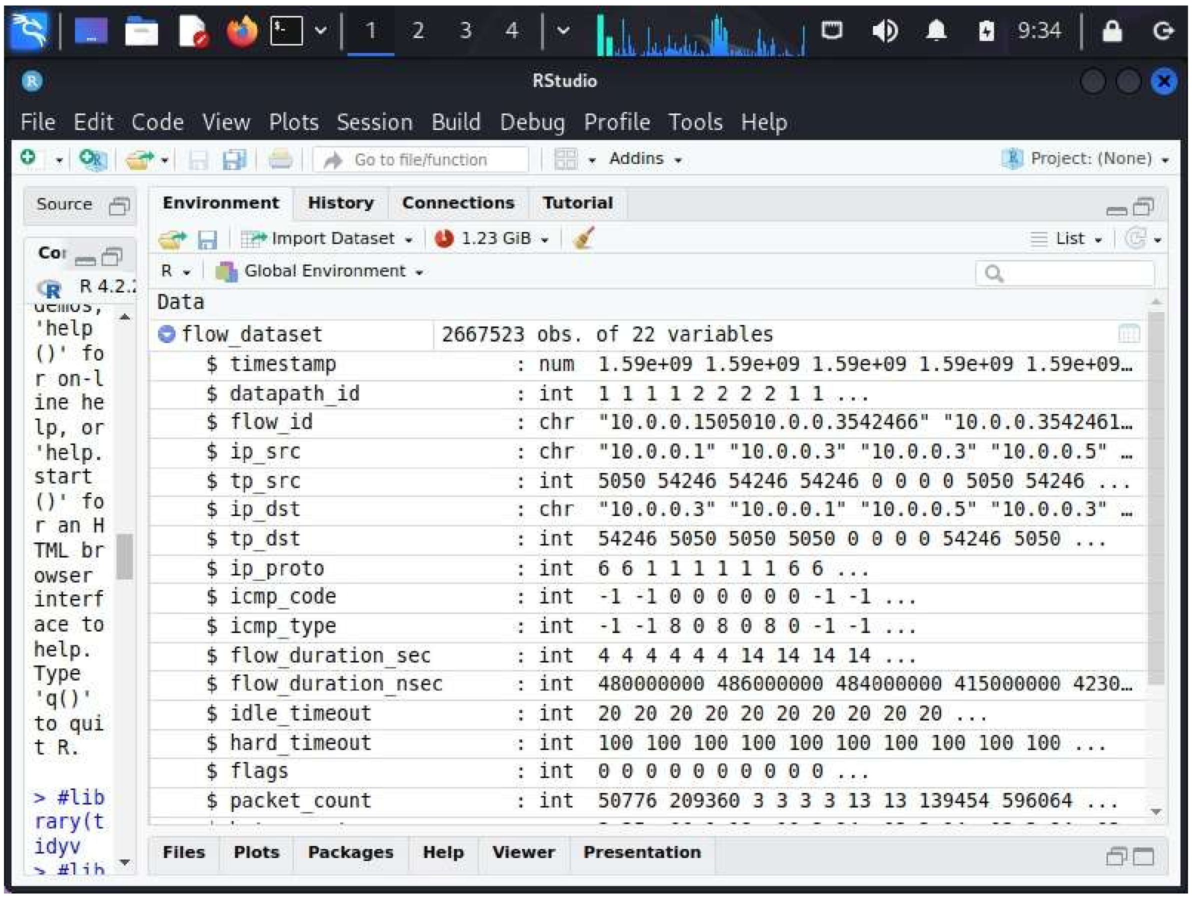
REVIEW PAPER : Mahmoud Said El Sayed Nhien-An Le-Khac , Marianne A. Azer and Anca D. Jurcut. "A Flow-Based Anomaly Detection Approach With Feature Selection Method Against DDoS Attacks in SDNs." In 2022 IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING.

SUMMARY :

The aim of this work is to reduce the redundant or irrelevant features without any significant impact on the classification accuracy. We have selected 10 features out of available 48 features using two common feature selection methods IG and RF. The approach provides a high detection rate and presents a more efficient better time to build the model. We further tested the trained model on the performance of **More Than One SDN controller** to evaluate how the used dataset can impact on the performance of the SDN controller. The results showed that the proposed approach does not deteriorate the network performance.

DATASET DESCRIPTION

- We have a huge amount of data entries (2667523 Observations)
- This is a snapshot of the sample data with column names.



REQUIREMENTS

User Interface:

- This system's user interface is the Linux os, which is a user-friendly interface.

Hardware Interfaces:

- Oracle Virtual Machine, Kali Linux, and Kaggle

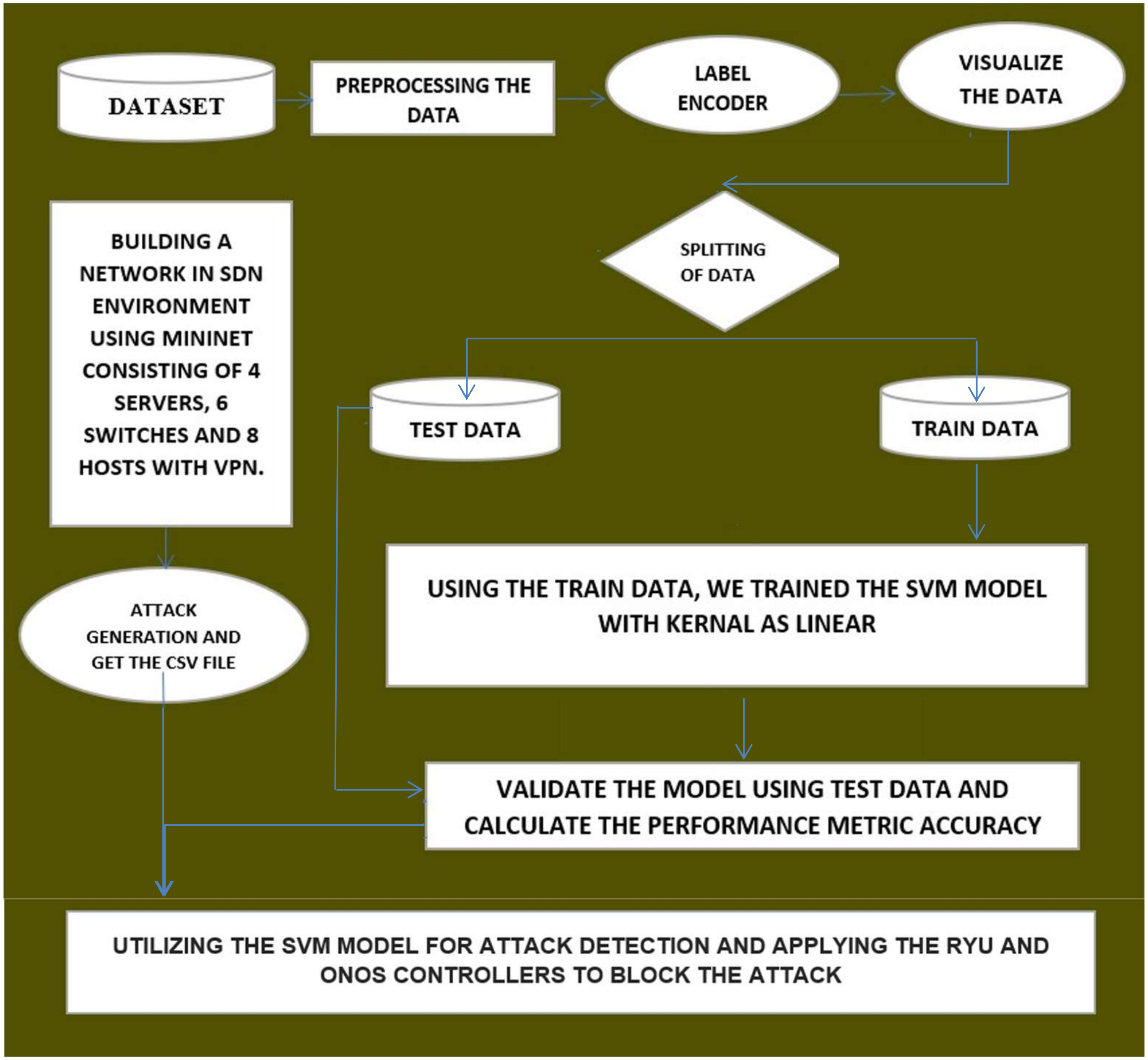
Software Interfaces:

- Required modules (tidyverse, e1071, caret, graphics, ggplot2, class, KNN, SVM)

Hardware Requirements:

1. Processor – Pentium-IV
2. RAM – 4GB (Minimum)
3. HDD/SSD – 256GB (Minimum)

Architecture Diagram



IMPLEMENTATION STEPS

- Machine Learning Model Development
- Network Topology Setup
- SDN Controller Setup
- Integration of ML Model and SDN
- Attack Generation
- Decision and Action Logic
- Results and Analysis

Algorithms

- Algorithm Linear Support Vector Machine (SVM) :
- Input: 02152020-threats-02-15-2020.csv Dataset
- Output: Success Accuracy of SVM trained Model on test Data

Find the optimal values for the tuning parameters of the SVM model;

Train the SVM model;

$p \leftarrow p^*$;

while $p \geq 2$ **do**

  $SVM_p \leftarrow$ SVM with the optimized tuning parameters for the p variables and observations in **Data**;

  $w_p \leftarrow$ calculate weight vector of the SVM_p $(w_{p1}, \ldots, w_{pp})$;

  $rank.criteria \leftarrow (w_{p1}^2, \ldots, w_{pp}^2)$;

  $min.rank.criteria \leftarrow$ variable with lowest value in $rank.criteria$ vector;

  Remove $min.rank.criteria$ from **Data**;

  $Rank_p \leftarrow min.rank.criteria$;

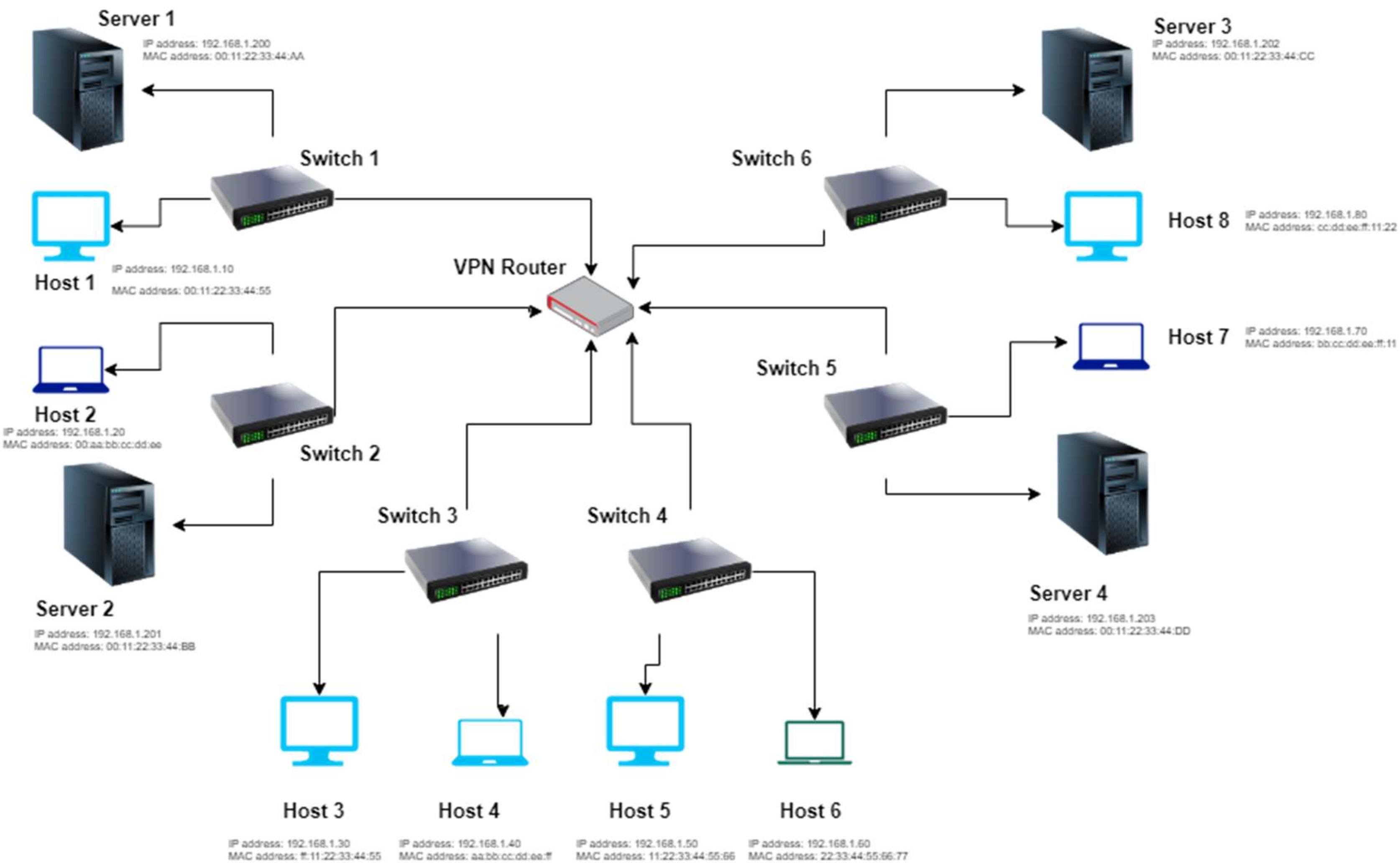
  p ← p - 1 ;

end

$Rank_1 \leftarrow$ variable in **Data** $\notin (Rank_2, \ldots, Rank_{p^*})$;

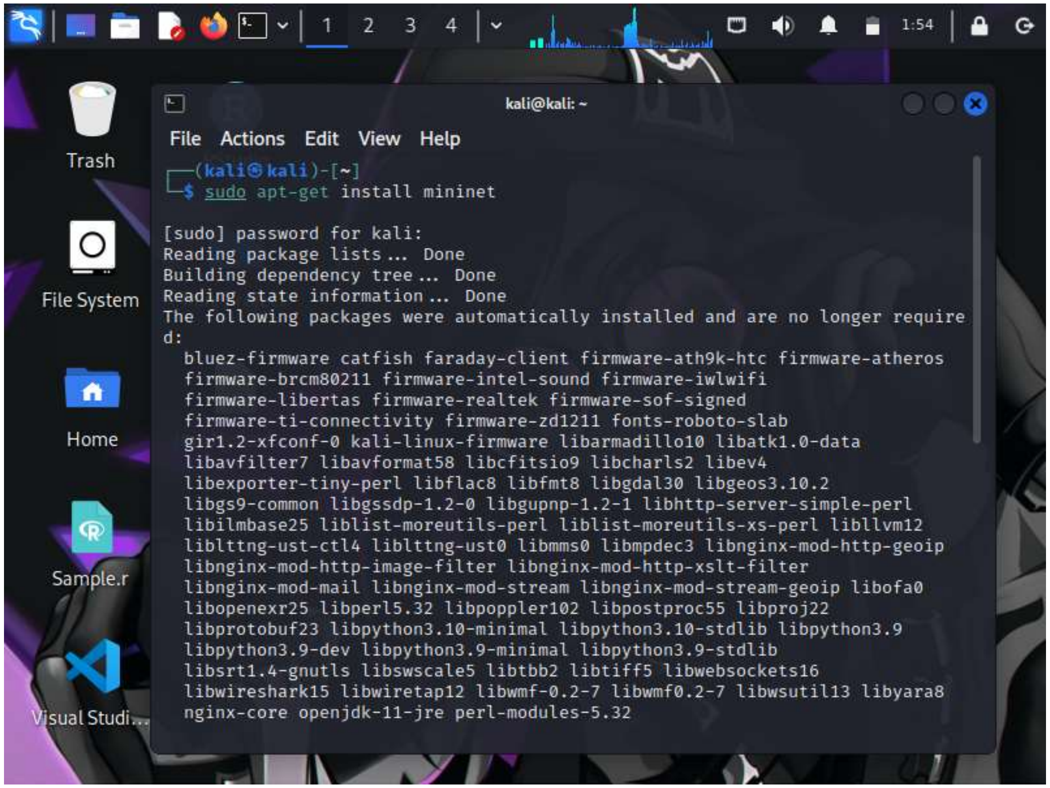
return $(Rank_1, \ldots, Rank_{p^*})$

NETWORK TOPOLOGY



The above network is built using the **Mininet** to provide SDN environment

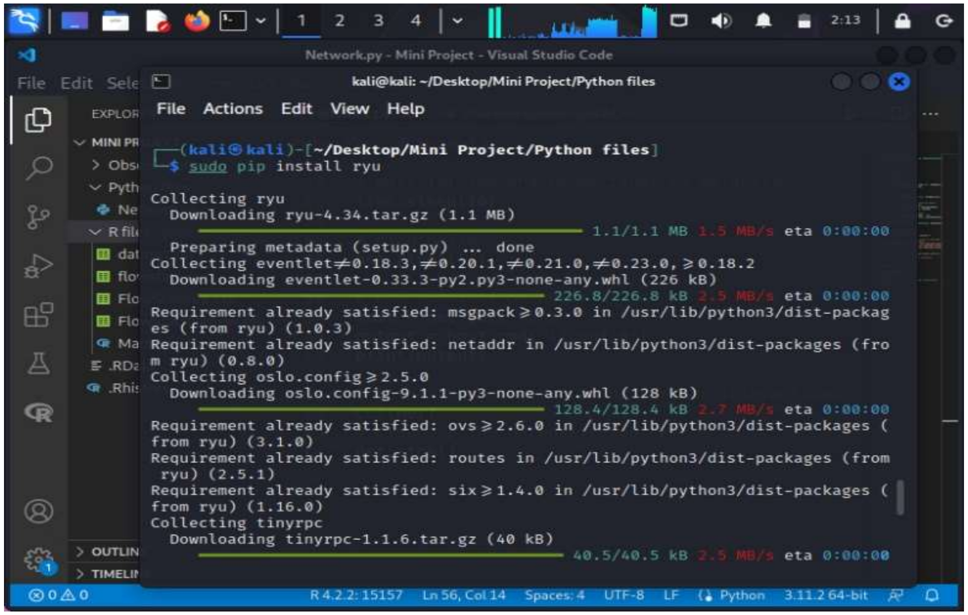
NETWORK USING MININET

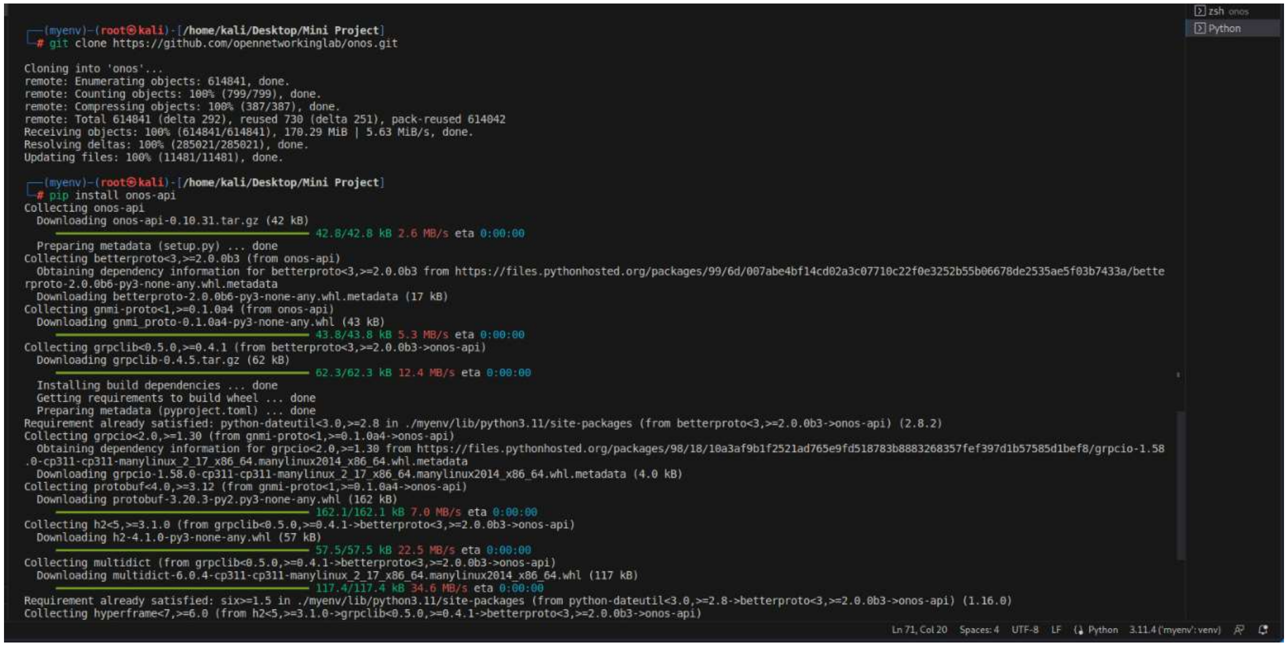



```python
from mininet.net import Mininet
from mininet.topo import Topo
from mininet.node import OVSSwitch, Controller
from mininet.cli import CLI
from mininet.link import TCLink

class MyTopology(Topo):
    def __init__(self):
        Topo.__init__(self)

        # Add switches
        switches = []
        for i in range(6):
            switch = self.addSwitch('s{}'.format(i+1), cls=OVSSwitch)
            switches.append(switch)

        # Add hosts
        hosts = []
        for i in range(8):
            host = self.addHost('h{}'.format(i+1))
            hosts.append(host)

        # Add servers
        servers = []
        for i in range(4):
            server = self.addHost('server{}'.format(i+1))
            servers.append(server)

        # Add links between switches and hosts
        for i in range(len(hosts)):
            switch = switches[i % len(switches)]
            self.addLink(hosts[i], switch)

        # Add links between switches and servers
        for i in range(len(servers)):
            switch = switches[i % len(switches)]
            self.addLink(servers[i], switch)

        # Add links between switches
        for i in range(len(switches) - 1):
            self.addLink(switches[i], switches[i+1])
topology = MyTopology()
```

```python
# Create the Mininet network with the custom Ryu controller
net = Mininet(topo=topology, link=TCLink,
controller=ryu_controller)

# Start the Mininet network
net.start()
```

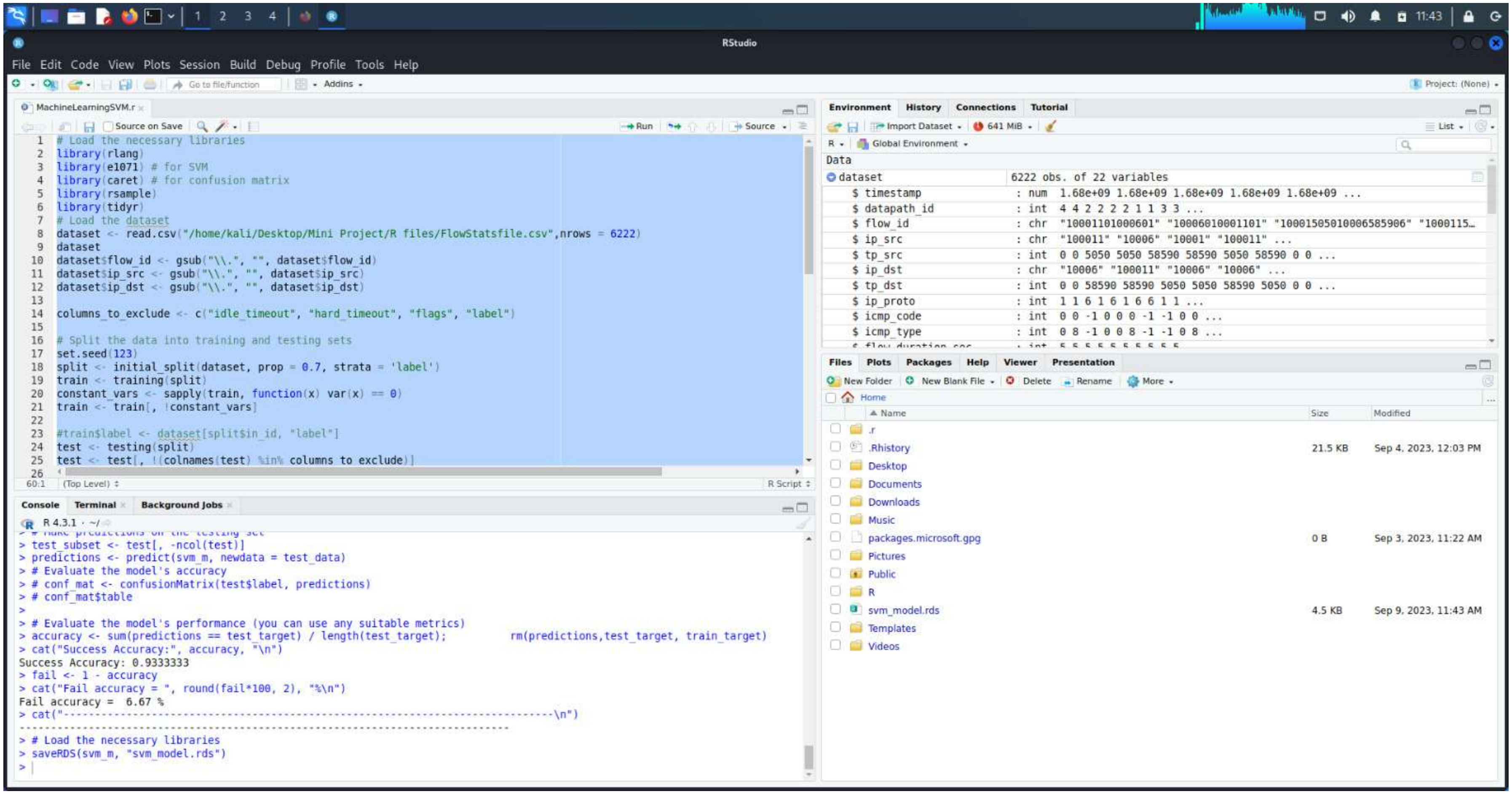
Controllers





RESULTS

Linear SVM model Prediction Accuracy (R Studio)



RESULTS

Ryu Controller and ONOS Controller Blocking the attacks (VS Code)