



**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.

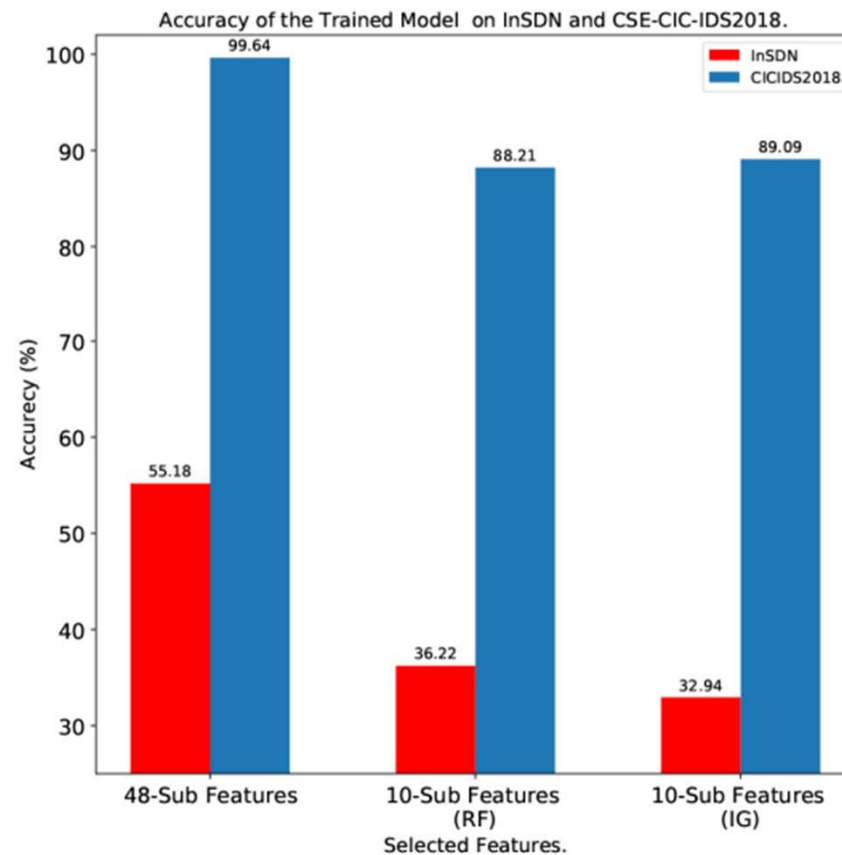
# LITERATURE SURVEY

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.

The screenshot shows the RStudio interface. The 'Environment' pane displays the 'flow\_dataset' data frame with 2667523 observations and 22 variables. The 'Data' pane shows a preview of the data with the following columns and their first few values:

Variable	Value 1	Value 2	Value 3	Value 4	Value 5
\$ timestamp	1.59e+09	1.59e+09	1.59e+09	1.59e+09	1.59e+09...
\$ datapath_id	1	1	1	2	2
\$ flow_id	"10.0.0.1505010.0.0.3542466"	"10.0.0.3542461..."			
\$ ip_src	"10.0.0.1"	"10.0.0.3"	"10.0.0.3"	"10.0.0.5"	...
\$ tp_src	5050	54246	54246	54246	0
\$ ip_dst	"10.0.0.3"	"10.0.0.1"	"10.0.0.5"	"10.0.0.3"	...
\$ tp_dst	54246	5050	5050	5050	0
\$ ip_proto	6	6	1	1	1
\$ icmp_code	-1	-1	0	0	0
\$ icmp_type	-1	-1	8	0	8
\$ flow_duration_sec	4	4	4	4	14
\$ flow_duration_nsec	480000000	486000000	484000000	415000000	4230...
\$ idle_timeout	20	20	20	20	20
\$ hard_timeout	100	100	100	100	100
\$ flags	0	0	0	0	0
\$ packet_count	50776	209360	3	3	3

# 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

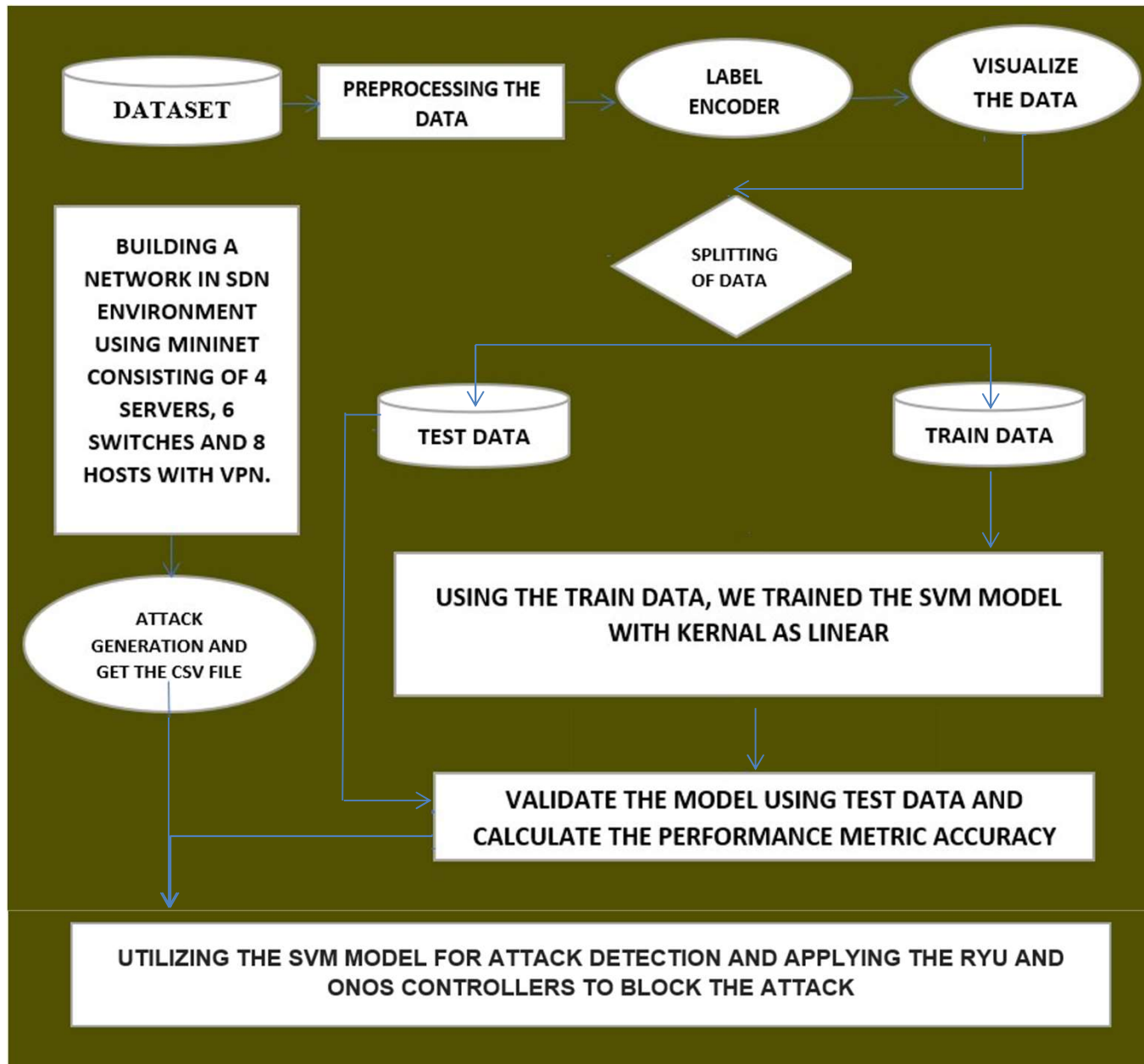
## **Software Interfaces:**

- 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  $SVM_p$  ( $w_{p1}, \dots, w_{pp}$ );

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

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

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

$Rank_p \leftarrow min.rank.criteria$ ;

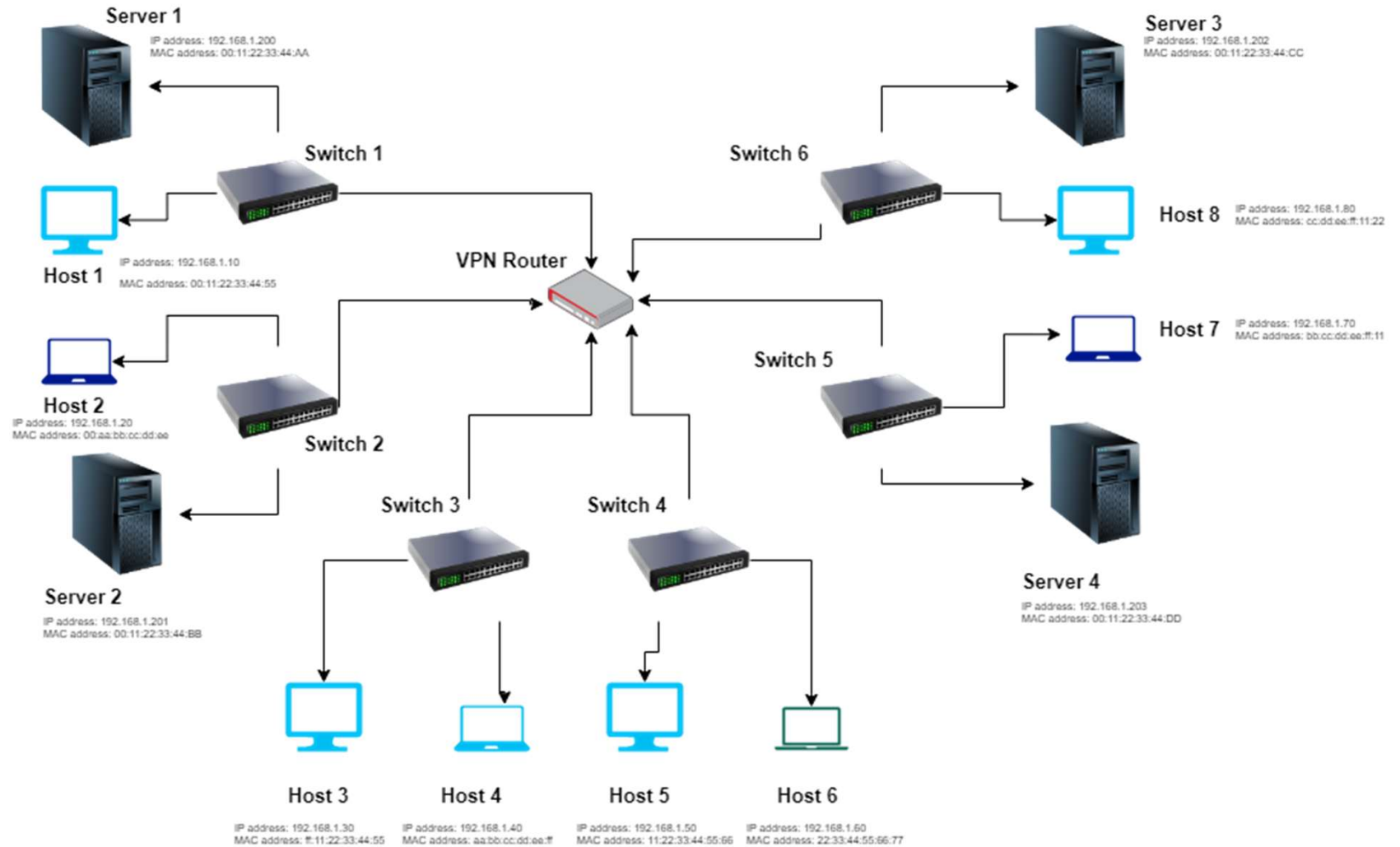
$p \leftarrow p - 1$  ;

**end**

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

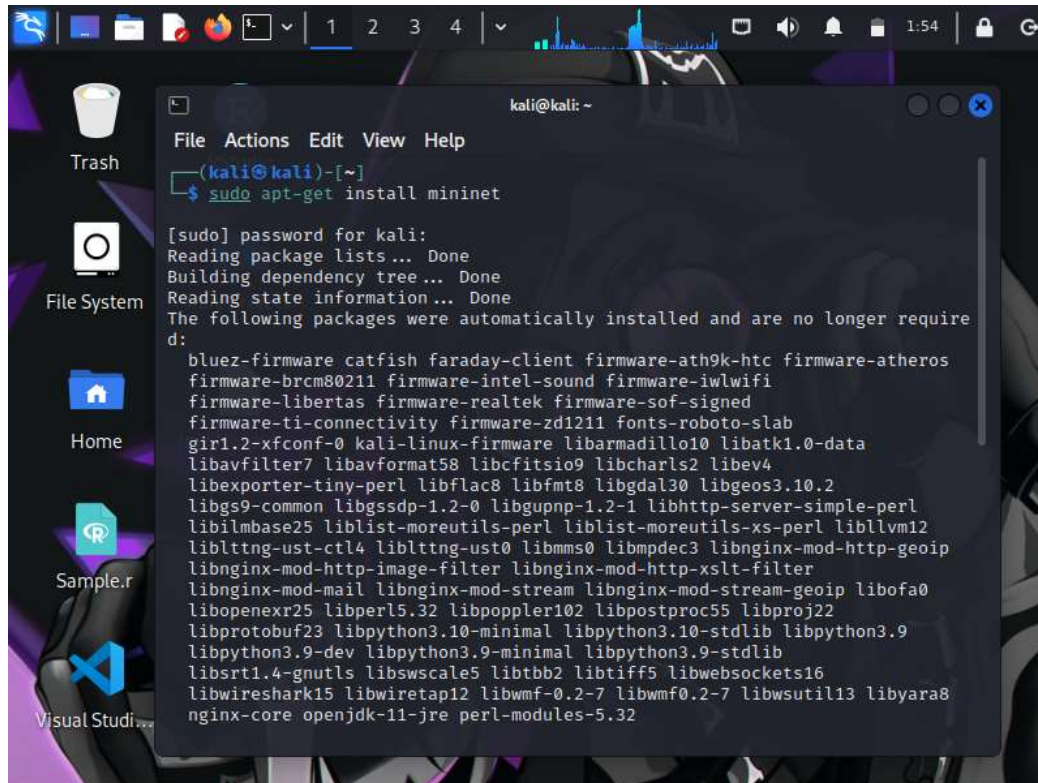
**return** ( $Rank_1, \dots, Rank_{p^*}$ )

# NETWORK TOPOLOGY



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

# NETWORK USING MININET



```
kali@kali: ~  
File Actions Edit View Help  
$ sudo apt-get install mininet  
[sudo] password for kali:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following packages were automatically installed and are no longer require  
d:  
bluez-firmware catfish faraday-client firmware-ath9k-htc firmware-atheros  
firmware-brcm80211 firmware-intel-sound firmware-iwlwifi  
firmware-libertas firmware-realtek firmware-sof-signed  
firmware-ti-connectivity firmware-zd1211 fonts-roboto-slab  
gir1.2-xfconf-0 kali-linux-firmware libarmadillo10 libatk1.0-data  
libavfilter7 libavformat58 libbcfitsio9 libcharls2 libev4  
libexptortiny-perl libflac8 libfmt8 libgdal30 libgeos3.10.2  
libgs9-common libgssdp-1.2-0 libgupnp-1.2-1 libhttp-server-simple-perl  
libilmbase25 liblist-moreutils-perl liblist-moreutils-xs-perl libllvm12  
liblttng-ust-ctl4 liblttng-ust0 libmms0 libmpdec3 libnginx-mod-http-geoip  
libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter  
libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip libofa0  
libopenexr25 libperl5.32 libpoppler102 libpostproc55 libproj22  
libprotobuf23 libpython3.10-minimal libpython3.10-stdlib libpython3.9  
libpython3.9-dev libpython3.9-minimal libpython3.9-stdlib  
libsrt1.4-gnutls libswscale5 libtbb2 libtiff5 libwebsockets16  
libwireshark15 libwiretap12 libwmf-0.2-7 libwmf0.2-7 libwsutil13 libyara8  
nginx-core openjdk-11-jre perl-modules-5.32
```

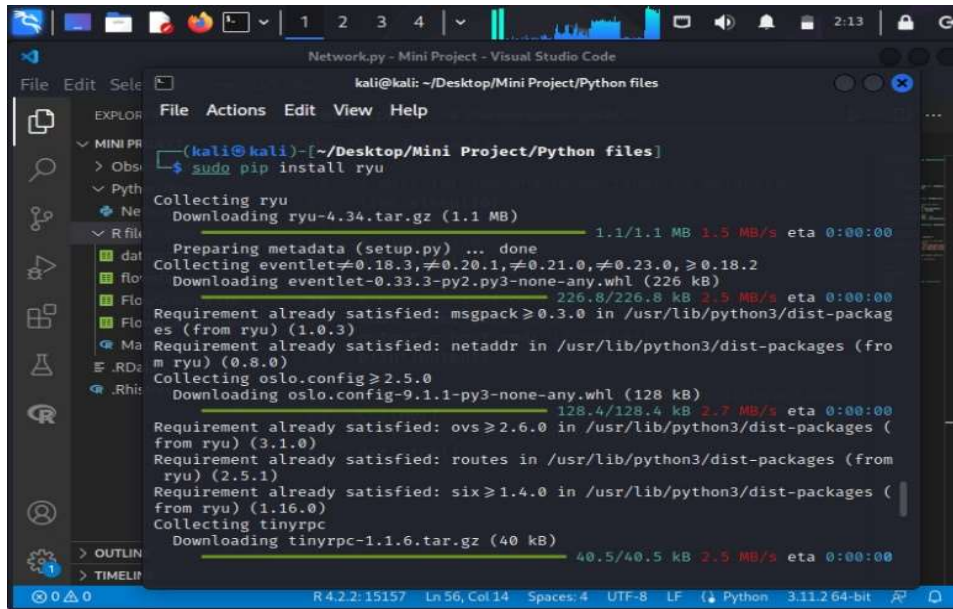
# Create the Mininet network with the custom Ryu controller  
`net = Mininet(topo=topology, link=TCLink,  
controller=ryu_controller)`

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

```
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()  
        *
```

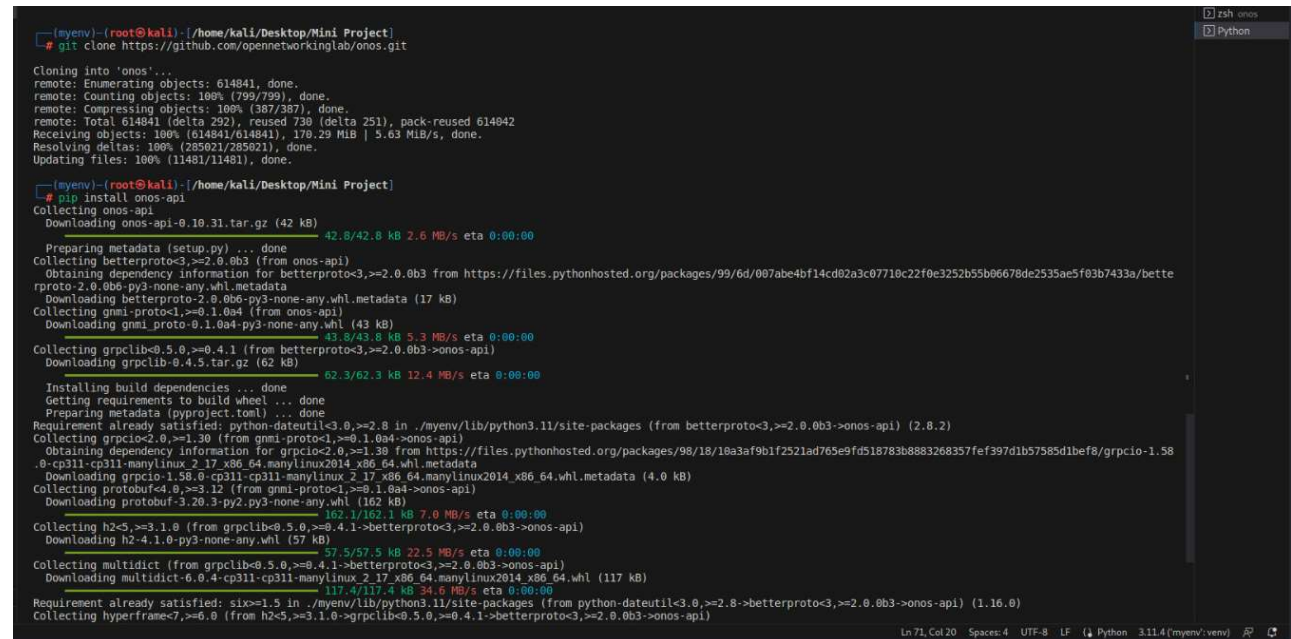


# Controllers



```
kali@kali: ~/Desktop/Mini Project/Python files
$ sudo pip install ryu

Collecting ryu
  Downloading ryu-4.34.tar.gz (1.1 MB)
    1.1/1.1 MB 1.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting eventlet<0.18.3,>=0.20.1,!=0.21.0,!=0.23.0,>=0.18.2
  Downloading eventlet-0.33.3-py2.py3-none-any.whl (226 kB)
    226.8/226.8 kB 2.5 MB/s eta 0:00:00
Requirement already satisfied: msgpack<0.3.0 in /usr/lib/python3/dist-packages (from ryu) (1.0.3)
Requirement already satisfied: netaddr in /usr/lib/python3/dist-packages (from ryu) (0.8.0)
Collecting oslo.config<2.5.0
  Downloading oslo.config-9.1.1-py3-none-any.whl (128 kB)
    128.4/128.4 kB 2.7 MB/s eta 0:00:00
Requirement already satisfied: ovs<2.6.0 in /usr/lib/python3/dist-packages (from ryu) (3.1.0)
Requirement already satisfied: routes in /usr/lib/python3/dist-packages (from ryu) (2.5.1)
Requirement already satisfied: six<1.4.0 in /usr/lib/python3/dist-packages (from ryu) (1.16.0)
Collecting tinyrpc
  Downloading tinyrpc-1.1.6.tar.gz (40 kB)
    40.5/40.5 kB 2.5 MB/s eta 0:00:00
```

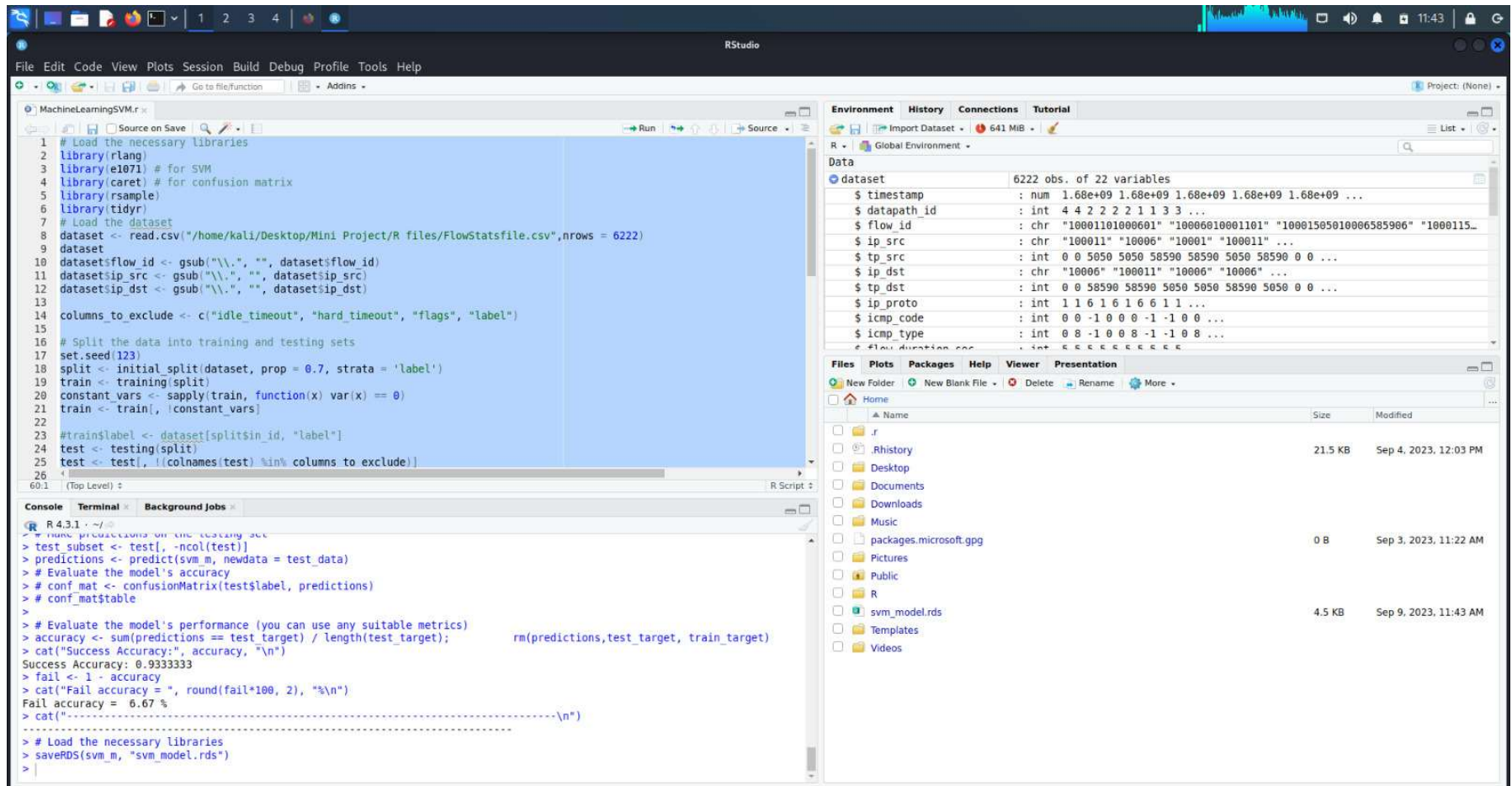


```
[myenv]-(root@kali)-[/home/kali/Desktop/Mini Project]
# git clone https://github.com/opennetworkinglab/onos.git
Cloning into 'onos'...
remote: Enumerating objects: 614841, done.
remote: Counting objects: 100% (799/799), done.
remote: Compressing objects: 100% (387/387), done.
remote: Total 614841 (delta 292), reused 730 (delta 251), pack-reused 614042
Receiving objects: 100% (614841/614841), 110.29 MiB | 5.63 MiB/s, done.
Resolving deltas: 100% (285021/285021), done.
Updating files: 100% (11481/11481), done.

[myenv]-(root@kali)-[/home/kali/Desktop/Mini Project]
# pip install onos-api
Collecting onos-api
  Downloading onos-api-0.10.31.tar.gz (42 kB)
    42.8/42.8 kB 2.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting betterproto<3,>=2.0.0b3 (from onos-api)
  Obtaining dependency information for betterproto<3,>=2.0.0b3 from https://files.pythonhosted.org/packages/99/6d/007abe4bf14cd02a3c07710c22f0e3252b55b06678de2535ae5f03b7433a/bette
rproto-2.0.0b6-py3-none-any.whl.metadata
  Downloading betterproto-2.0.0b6-py3-none-any.whl.metadata (17 kB)
Collecting gnm1-protoc<1,>=0.1.0a4 (from onos-api)
  Downloading gnm1-protoc-0.1.0a4-py3-none-any.whl (43 kB)
    43.8/43.8 kB 5.3 MB/s eta 0:00:00
Collecting grpcio<0.5.0,>=0.4.1 (from betterproto<3,>=2.0.0b3->onos-api)
  Downloading grpcio-0.4.5.tar.gz (62 kB)
    62.3/62.3 kB 12.4 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: python-dateutil<3.0,>=2.8 in /myenv/lib/python3.11/site-packages (from betterproto<3,>=2.0.0b3->onos-api) (2.8.2)
Collecting grpcio-2.0.0,>=1.30 (from gnm1-protoc<1,>=0.1.0a4->onos-api)
  Obtaining dependency information for grpcio-2.0.0,>=1.30 from https://files.pythonhosted.org/packages/98/18/10a3af9b1f2521ad765e9fd518783b8883268357fef397d1b57585d1bef8/grpcio-1.58
.0-cp311-cp311-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata
  Downloading grpcio-1.58.0-cp311-cp311-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (4.0 kB)
Collecting protobuf<4.0,>=3.12 (from gnm1-protoc<1,>=0.1.0a4->onos-api)
  Downloading protobuf-3.20.3-py2.py3-none-any.whl (162 kB)
    162.1/162.1 kB 7.0 MB/s eta 0:00:00
Collecting h2<5,>=3.1.0 (from grpcio<0.5.0,>=0.4.1->betterproto<3,>=2.0.0b3->onos-api)
  Downloading h2-4.1.0-py3-none-any.whl (57 kB)
    57.5/57.5 kB 22.5 MB/s eta 0:00:00
Collecting multidict (from grpcio<0.5.0,>=0.4.1->betterproto<3,>=2.0.0b3->onos-api)
  Downloading multidict-6.0.4-cp311-cp311-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (117 kB)
    117.4/117.4 kB 34.6 MB/s eta 0:00:00
Requirement already satisfied: six<1.5 in /myenv/lib/python3.11/site-packages (from python-dateutil<3.0,>=2.8->betterproto<3,>=2.0.0b3->onos-api) (1.16.0)
Collecting hyperframe<7,>=6.0 (from h2<5,>=3.1.0->grpcio<0.5.0,>=0.4.1->betterproto<3,>=2.0.0b3->onos-api)
```

# RESULTS

## Linear SVM model Prediction Accuracy (R Studio)



The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for loading libraries, reading a dataset, splitting it into training and testing sets, training an SVM model, and evaluating its performance.
- Environment:** Shows the loaded dataset with 6222 observations and 22 variables.
- Files:** Lists the files in the current project, including `svm_model.rds`.
- Console:** Displays the output of the R script, showing the success and failure accuracies of the SVM model.

```
1 # Load the necessary libraries
2 library(rlang)
3 library(e1071) # for SVM
4 library(caret) # for confusion matrix
5 library(rsample)
6 library(tidy)
7 # Load the dataset
8 dataset <- read.csv("/home/kali/Desktop/Mini Project/R files/FlowStatsfile.csv", nrow = 6222)
9 dataset
10 dataset$flow_id <- gsub("\\.", "", dataset$flow_id)
11 dataset$ip_src <- gsub("\\.", "", dataset$ip_src)
12 dataset$ip_dst <- gsub("\\.", "", dataset$ip_dst)
13
14 columns_to_exclude <- c("idle_timeout", "hard_timeout", "flags", "label")
15
16 # Split the data into training and testing sets
17 set.seed(123)
18 split <- initial.split(dataset, prop = 0.7, strata = 'label')
19 train <- training(split)
20 constant_vars <- sapply(train, function(x) var(x) == 0)
21 train <- train[, !constant_vars]
22
23 #train$label <- dataset[split$in_id, "label"]
24 test <- testing(split)
25 test <- test[, !(colnames(test) %in% columns_to_exclude)]
26
60.1 (Top Level) ↓
```

**Environment:** Global Environment

**Data:** dataset (6222 obs. of 22 variables)

\$ timestamp	:	num	1.68e+09	1.68e+09	1.68e+09	1.68e+09	1.68e+09	...														
\$ datapath_id	:	int	4	4	2	2	2	1	1	3	3	...										
\$ flow_id	:	chr	"10001101000601"	"10000010001101"	"10001505010006585906"	"1000115...																
\$ ip_src	:	chr	"100011"	"10006"	"10001"	"100011"	...															
\$ ip_dst	:	chr	"10006"	"100011"	"10006"	"10006"	...															
\$ tp_src	:	int	0	0	5050	5050	58590	58590	5050	58590	0	0	...									
\$ tp_dst	:	int	0	0	58590	58590	5050	5050	58590	5050	0	0	...									
\$ ip_proto	:	int	1	1	6	1	6	1	6	6	1	1	...									
\$ icmp_code	:	int	0	0	-1	0	0	-1	-1	0	0	...										
\$ icmp_type	:	int	0	8	-1	0	8	-1	-1	0	8	...										
\$ flow_duration...	:	int	5	5	5	5	5	5	5	5	5	...										

**Files:** New Folder, New Blank File, Delete, Rename, More

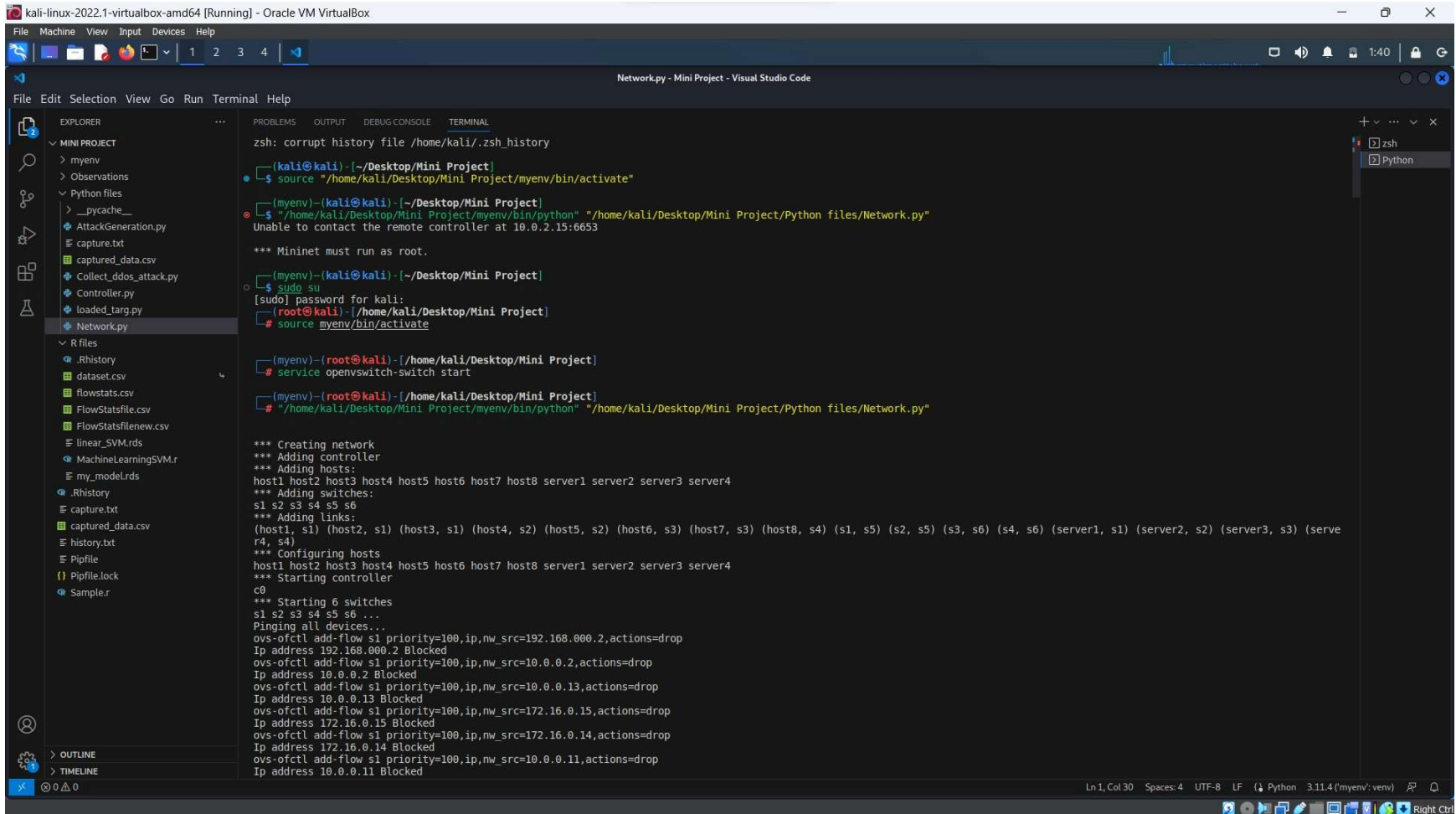
Name	Size	Modified
.r	21.5 KB	Sep 4, 2023, 12:03 PM
.Rhistory		
Desktop		
Documents		
Downloads		
Music		
packages.microsoft.gpg	0 B	Sep 3, 2023, 11:22 AM
Pictures		
Public		
R		
svm_model.rds	4.5 KB	Sep 9, 2023, 11:43 AM
Templates		
Videos		

**Console:** R 4.3.1 ~ /

```
> # Make predictions on the testing set
> test_subset <- test[, -ncol(test)]
> predictions <- predict(svm_m, newdata = test_data)
> # Evaluate the model's accuracy
> # conf_mat <- confusionMatrix(tests$label, predictions)
> # conf_mat$table
>
> # Evaluate the model's performance (you can use any suitable metrics)
> accuracy <- sum(predictions == test_target) / length(test_target); rm(predictions, test_target, train_target)
> cat("Success Accuracy:", accuracy, "\n")
Success Accuracy: 0.9333333
> fail <- 1 - accuracy
> cat("Fail accuracy = ", round(fail*100, 2), "%\n")
Fail accuracy = 6.67 %
> cat("-----\n")
> # Load the necessary libraries
> saveRDS(svm_m, "svm_model.rds")
>
```

# RESULTS

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



The screenshot shows a Visual Studio Code editor window titled "Network.py - Mini Project - Visual Studio Code". The left sidebar displays the Explorer view with a file tree for a project named "MINI PROJECT". The file tree includes folders for "myenv", "Observations", and "Python files", along with various CSV and text files. The main editor area shows a terminal window with the following content:

```
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali) - [~/Desktop/Mini Project]
$ source "/home/kali/Desktop/Mini Project/myenv/bin/activate"
(myenv)-(kali@kali) - [~/Desktop/Mini Project]
$ "/home/kali/Desktop/Mini Project/myenv/bin/python" "/home/kali/Desktop/Mini Project/Python files/Network.py"
Unable to contact the remote controller at 10.0.2.15:6653

*** Mininet must run as root.

(myenv)-(kali@kali) - [~/Desktop/Mini Project]
$ sudo su
[sudo] password for kali:
(root@kali) - [~/home/kali/Desktop/Mini Project]
# source myenv/bin/activate

(myenv)-(root@kali) - [~/home/kali/Desktop/Mini Project]
# service openvswitch-switch start

(myenv)-(root@kali) - [~/home/kali/Desktop/Mini Project]
# "/home/kali/Desktop/Mini Project/myenv/bin/python" "/home/kali/Desktop/Mini Project/Python files/Network.py"

*** Creating network
*** Adding controller
*** Adding hosts:
host1 host2 host3 host4 host5 host6 host7 host8 server1 server2 server3 server4
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(host1, s1) (host2, s1) (host3, s1) (host4, s2) (host5, s2) (host6, s3) (host7, s3) (host8, s4) (s1, s5) (s2, s5) (s3, s6) (s4, s6) (server1, s1) (server2, s2) (server3, s3) (serve
r4, s4)
*** Configuring hosts
host1 host2 host3 host4 host5 host6 host7 host8 server1 server2 server3 server4
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
Pinging all devices...
ovs-ofctl add-flow s1 priority=100,ip,nw_src=192.168.0.2,actions=drop
Ip address 192.168.0.2 Blocked
ovs-ofctl add-flow s1 priority=100,ip,nw_src=10.0.0.2,actions=drop
Ip address 10.0.0.2 Blocked
ovs-ofctl add-flow s1 priority=100,ip,nw_src=10.0.0.13,actions=drop
Ip address 10.0.0.13 Blocked
ovs-ofctl add-flow s1 priority=100,ip,nw_src=172.16.0.15,actions=drop
Ip address 172.16.0.15 Blocked
ovs-ofctl add-flow s1 priority=100,ip,nw_src=172.16.0.14,actions=drop
Ip address 172.16.0.14 Blocked
ovs-ofctl add-flow s1 priority=100,ip,nw_src=10.0.0.11,actions=drop
Ip address 10.0.0.11 Blocked
```

The terminal output shows the successful execution of the network configuration script, including the creation of the network, addition of hosts and switches, and the configuration of flow rules to block specific IP addresses. The status of the network is displayed at the bottom of the terminal window.