UNIVERSIDADE DE AVEIRO

universidade de aveiro

# *DNS Tunneling*

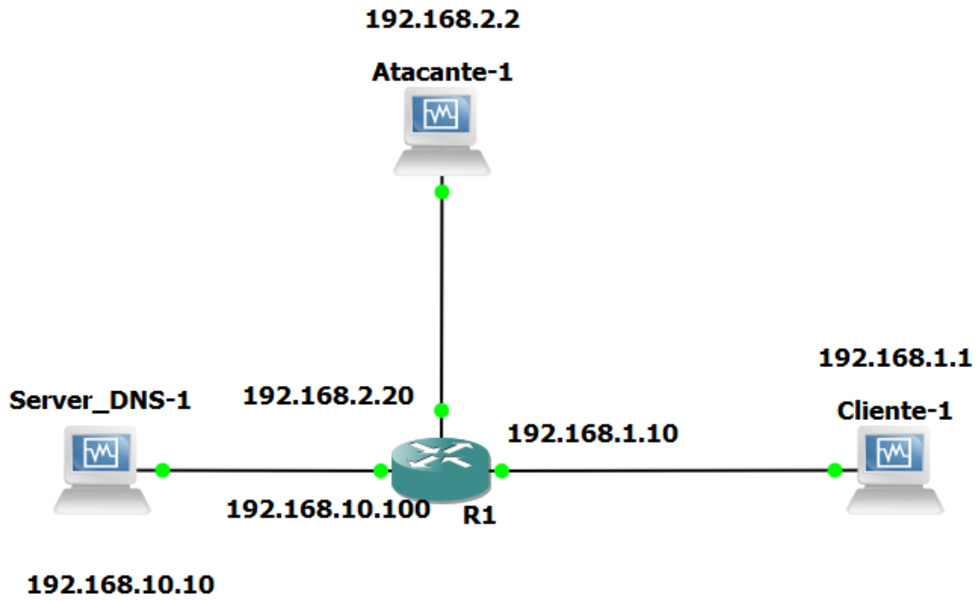
## Técnicas de Percepção de Redes

MARTA OLIVEIRA 97613
BRUNO SILVA 97931

# INDEX

# Our Scenario

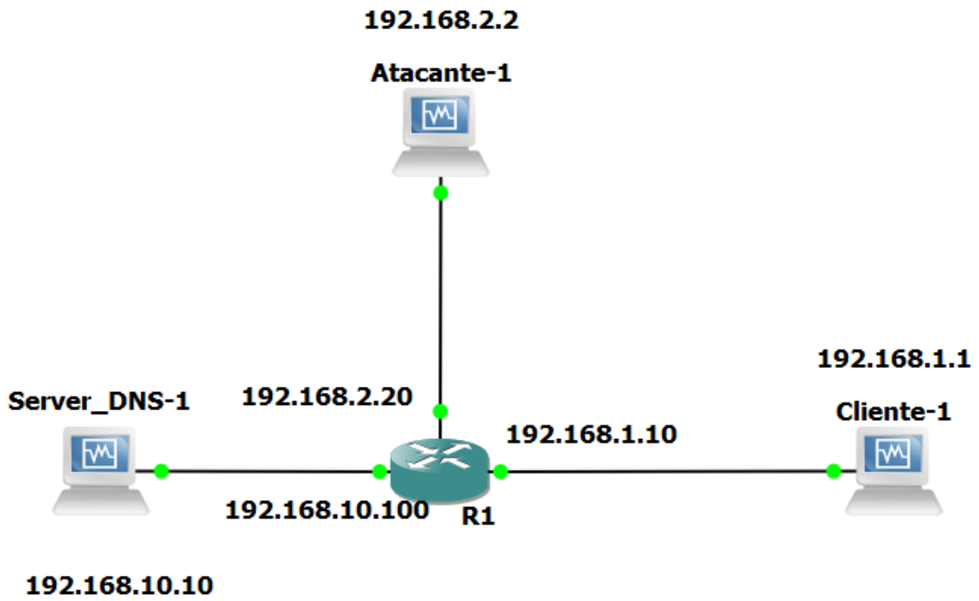
- Creation of a Virtual Machine that will act as a client
- Connect the virtual machine to a virtual router or switch in your simulated network.
- Set up a DNS Server
- The client is connected to the DNS Server that will redirect the traffic to the attacker’s machine.
- Iodine tunnel between attacker and client



# Our Scenario

- The client side component initiates a DNS request.

- Perfom operations through the established tunnel
  - Files transfer(extract data) through SSH session
  - Commands execution



# Data Sources

**Normal behaviour**

Capture traffic (DNS packets) between a computer and a DNS server in order to define the normal network behavior.

**Malicious Behaviour**

Capturing packets on the client computer where a DNS tunnel is established between the client and server (attacker).

# Data Sources

Collected DNS traffic data from 4 sessions, each lasting around 1 hour and 30 minutes.

- Monitored the DNS activities of two distinct client entities.
- 2 DNS traffic captures originated from an attacker, with 2 distinct behaviors:
  - "dumb" script.
  - "smart" script for varied attack techniques.

# Dumb Script

The attacker got access to the client machine.

- Begins to transfer files (scp command) with random sizes (1 KB to 100 MB) through an *SSH* session.  He waits a random amount of time (1 to 30 seconds) between each file transfer. **(based on a script)**

- Executes commands randomly between transfers **(human behaviour)**

# Smart Script

- The attacker logs into the client machine

- Begins to copy files
  - Splits the file into smaller chunks (of variable sizes)
  - Copies the chunks by a random order
  - There is a random gaussian delay between the copy of each file
  - Deletes the chunk of files afterwards
  - Repeat process until file is transfered

- Execution of a random number of commands
  - with gaussian delays in between

- Gaussian delay between events (copy files and execution of commands)

# Gaussian Delays

- We used several random gaussian delays between events
- Introduce randomness and variability into the timing between actions, making the behavior less predictable and more human-like.
- Time between each copy:
  - Mean 90 and deviation 60
- Time between commands:
  - Mean 10 and deviation 5
- Time between events (between the exfiltraction and execution of commands)
  - Mean 90 and deviation 60

# Data Processing

- **Anomaly Detection:** our goal is to identify unusual events or patterns in the data
  - Comparing the malicious dataset with the normal behaviour dataset

- **Data filtering:** DNS packets (traffic UDP) on port 53.

- **Data sampling/aggregation:**
  - Volume of DNS traffic (packet count)
    - per IP address
  - Time interval between communications(Periods of Silence)

- **Observation process:**
  - Sliding Window
    - Sliding value: 1 minute
    - Width: 5 minutes

# Metrics Extracted

For each capture:

- **Packet Count**
  - Number of Uploaded Packets
  - Number of Downloaded Packets

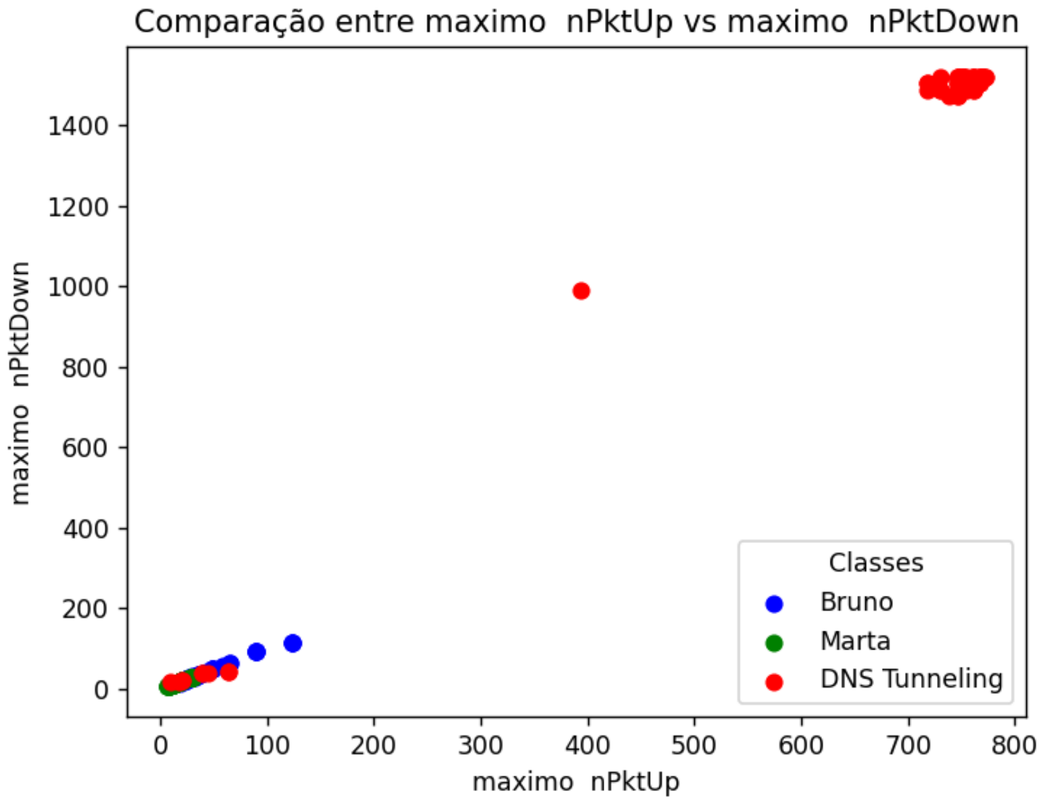
- **Number of Bytes**
  - Number of Uploaded Bytes
  - Number of Downloaded Bytes

# Features

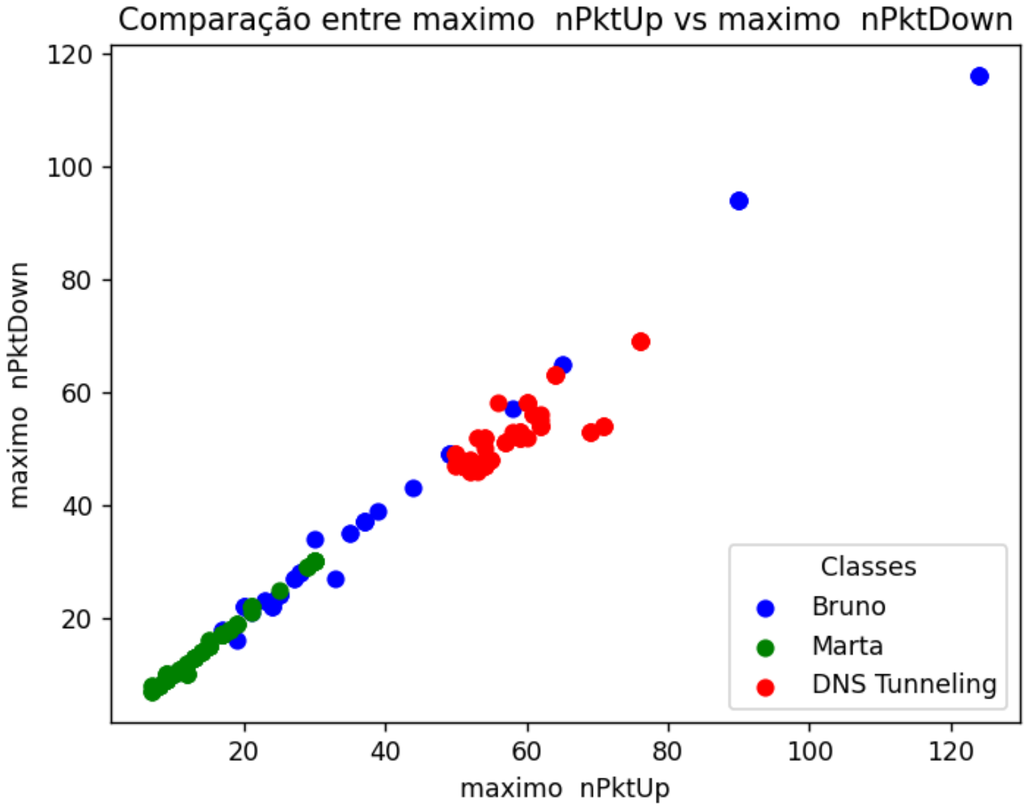
- **Volume of DNS traffic per IP address**
  - Maximum
  - Mean, median, standard deviation
  - Percentiles (98%, 95%)

- **Time interval between communications(periods of silence)**
  - Mean, median, standard deviation

# Plot Features

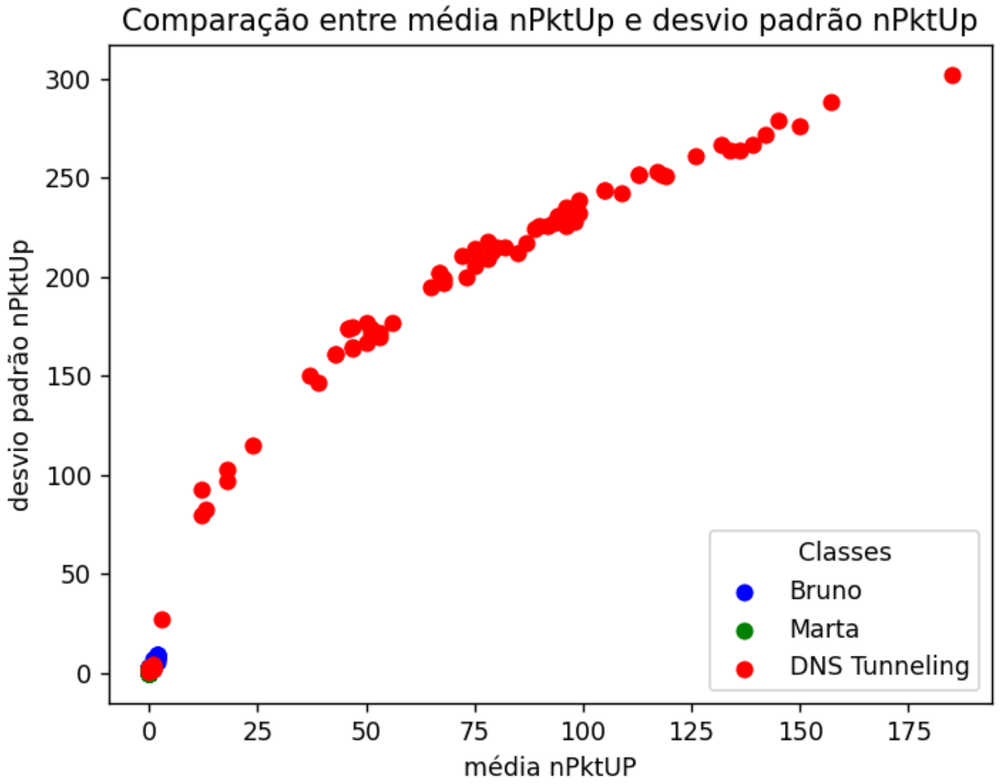



*Dumb script*

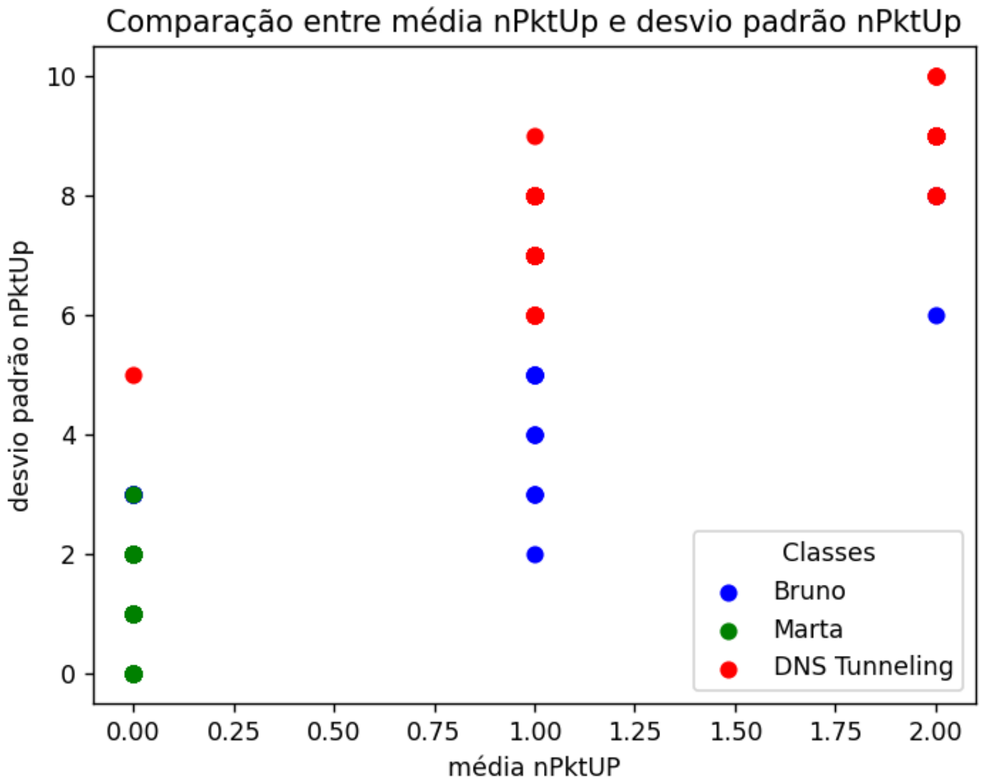



*Smart script*

# Plot Features

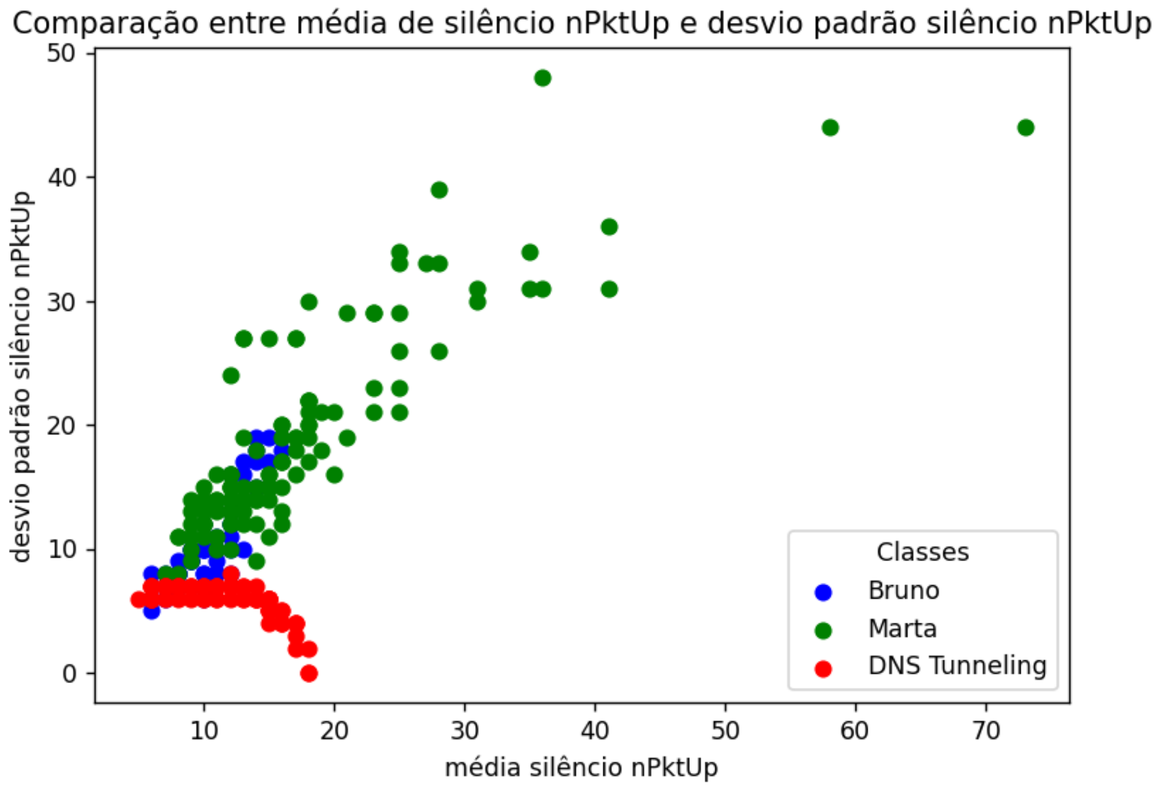



*Dumb script*

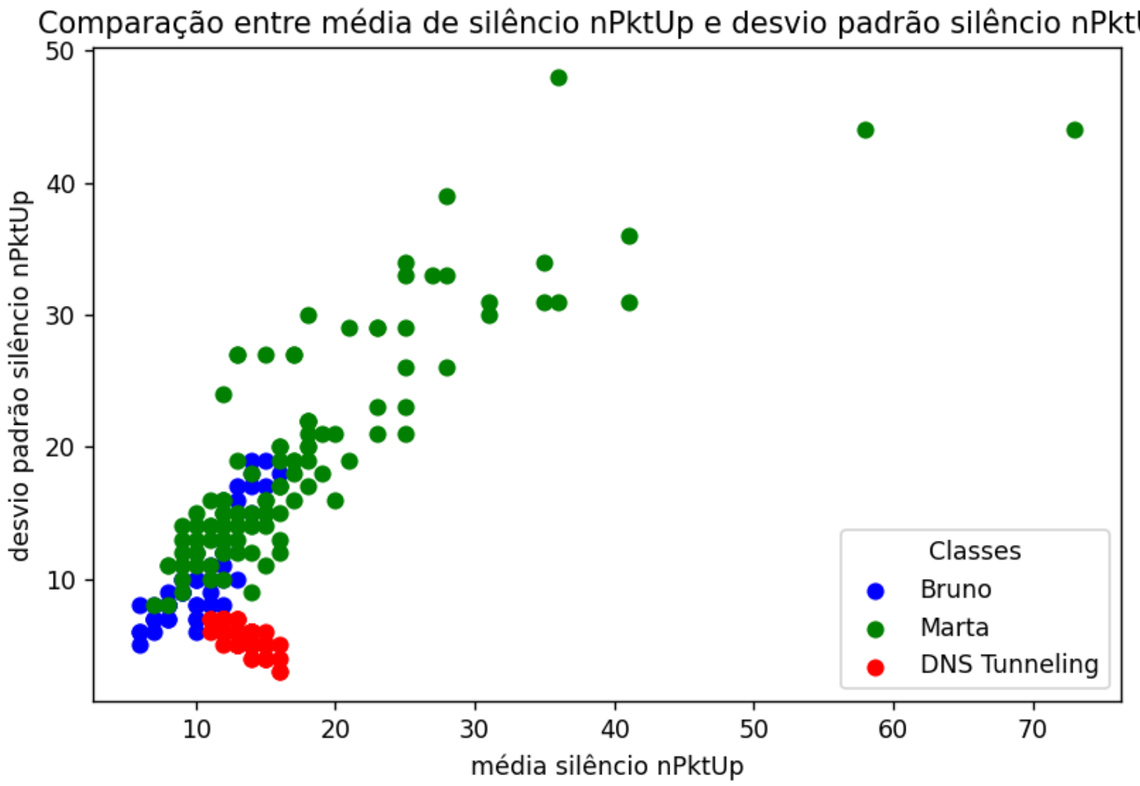



*Smart script*

# Plot Features

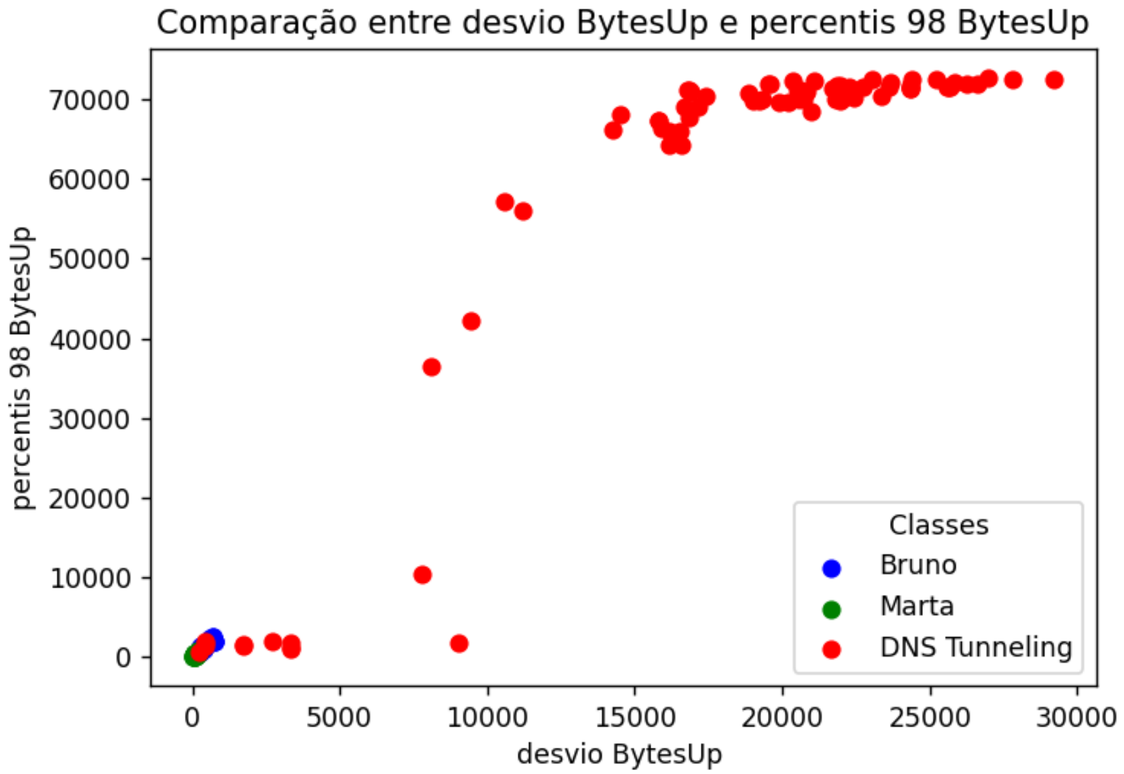



*Dumb script*

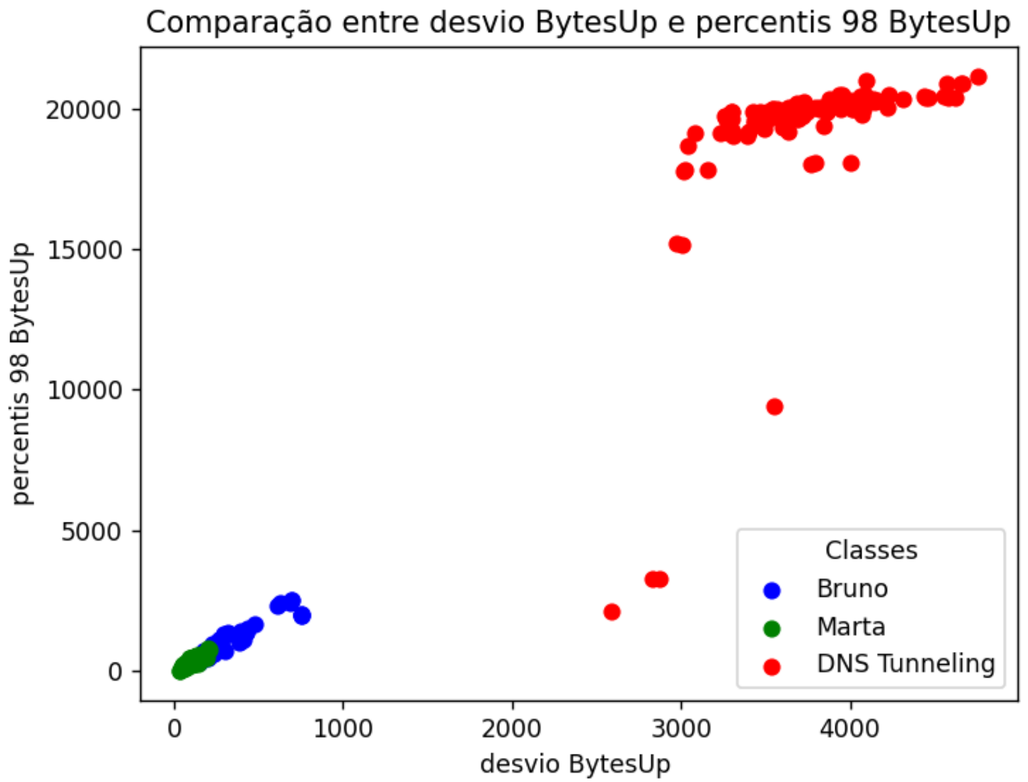



*Smart script*

# Plot Features




*Dumb script*




*Smart script*

# Training & Test Features

The following approach was followed:

- One train set with 50% of the features for each client and one train set with 50% of the features for each of the attacker’s scripts.
- One test set with the rest of the attacker features (‘dumb’ or ‘smart’ script)
- One test set with the rest of the features for each of the clients.

# Anomaly Detection

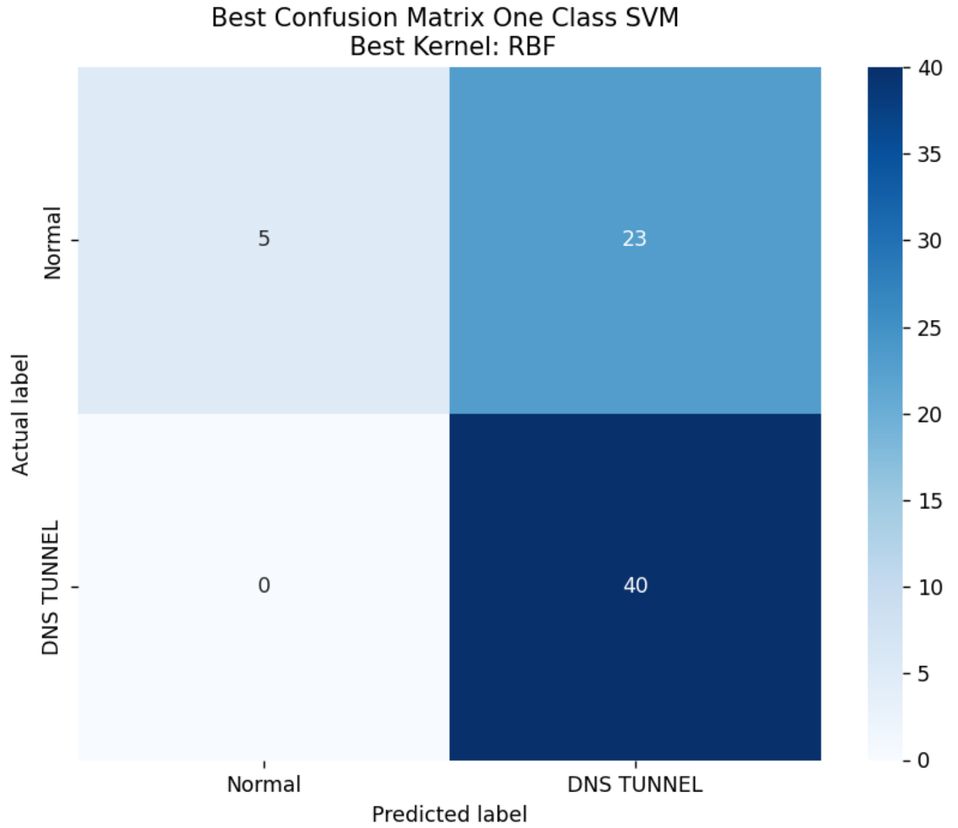
**The following algorithms were used:**

- One Class Support Vector Machines without PCA features
  - Linear, RBF and Poly Kernels
- One Class Support Vector Machines with PCA features
  - Linear, RBF and Poly Kernels
- Support Vector Machines without PCA features
  - Linear, RBF and Poly Kernels
- Support Vector Machines with PCA features
  - Linear, RBF and Poly Kernels
- Neural Networks without PCA
- Neural Networks with PCA

# Results: One Class SVM without PCA

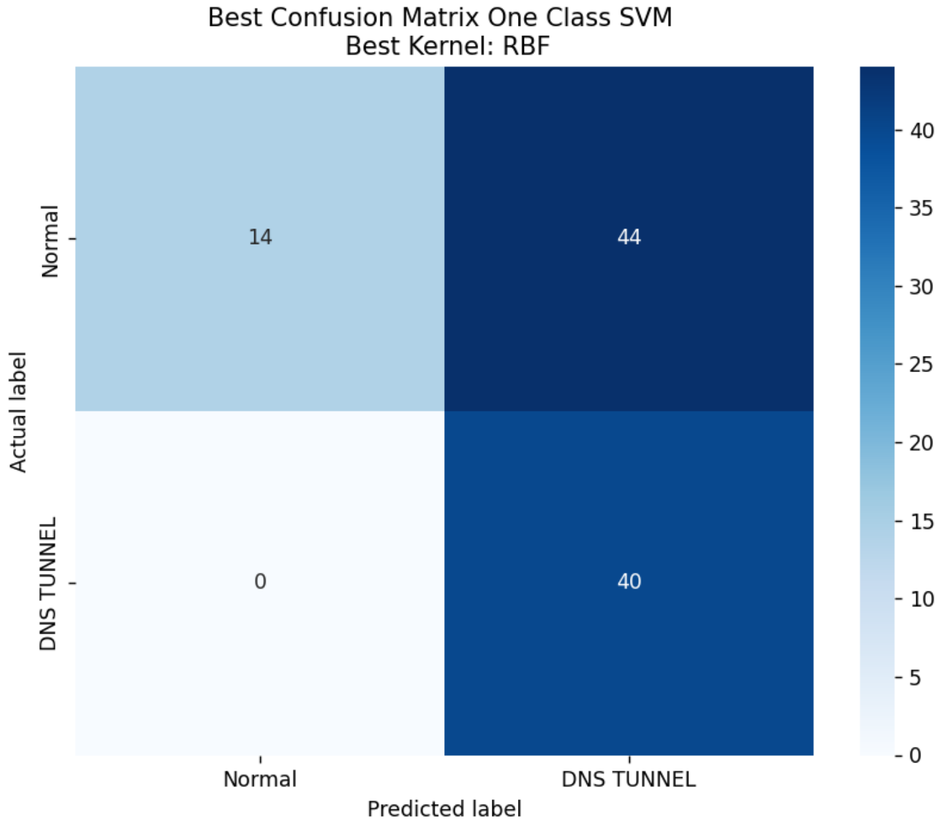
## Dumb Script

F1 Score: 77.7




Bruno's Model

F1 Score: 64.52

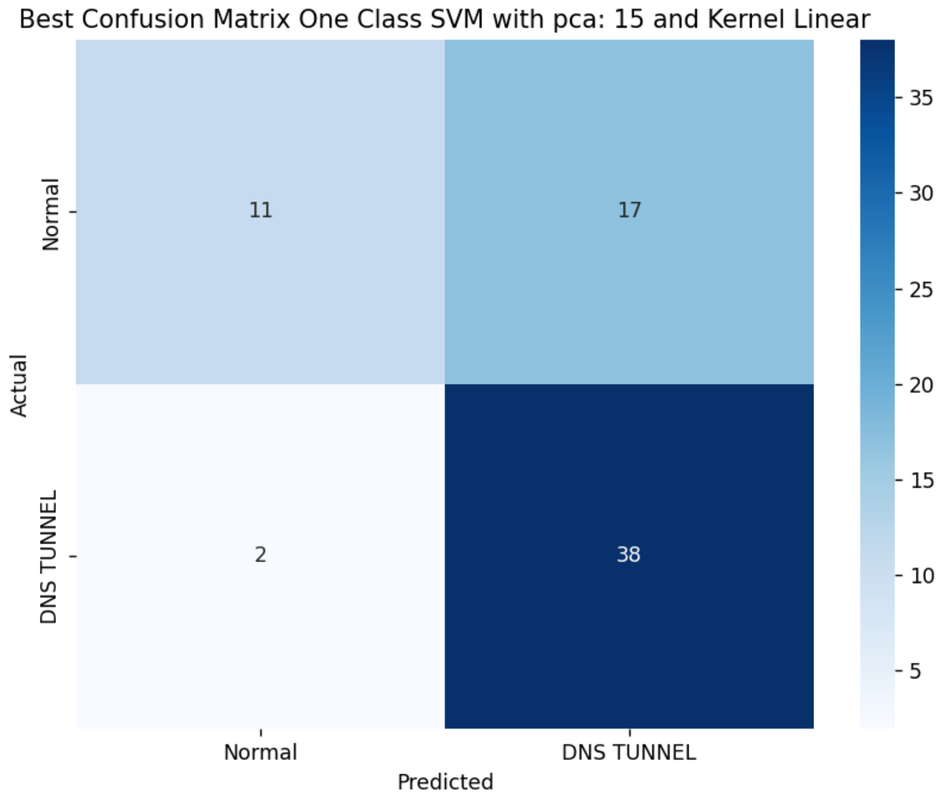



Marta's Model

# Results: One Class SVM with PCA

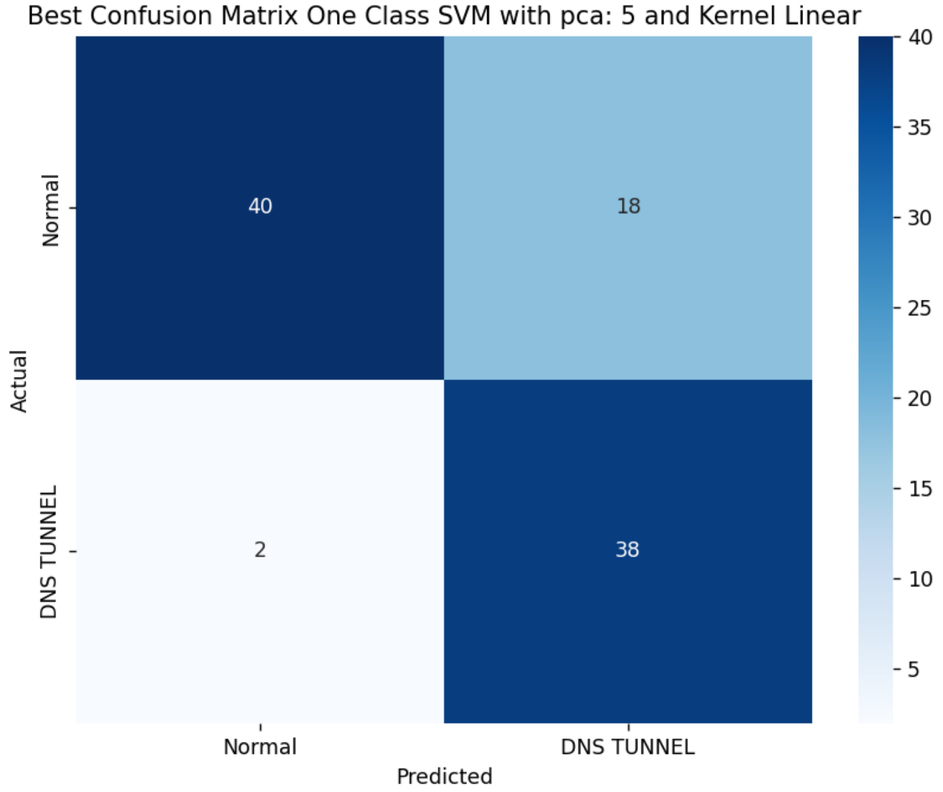
## Dumb Script

F1 Score: 80




Bruno's Model

F1 Score: 79.17

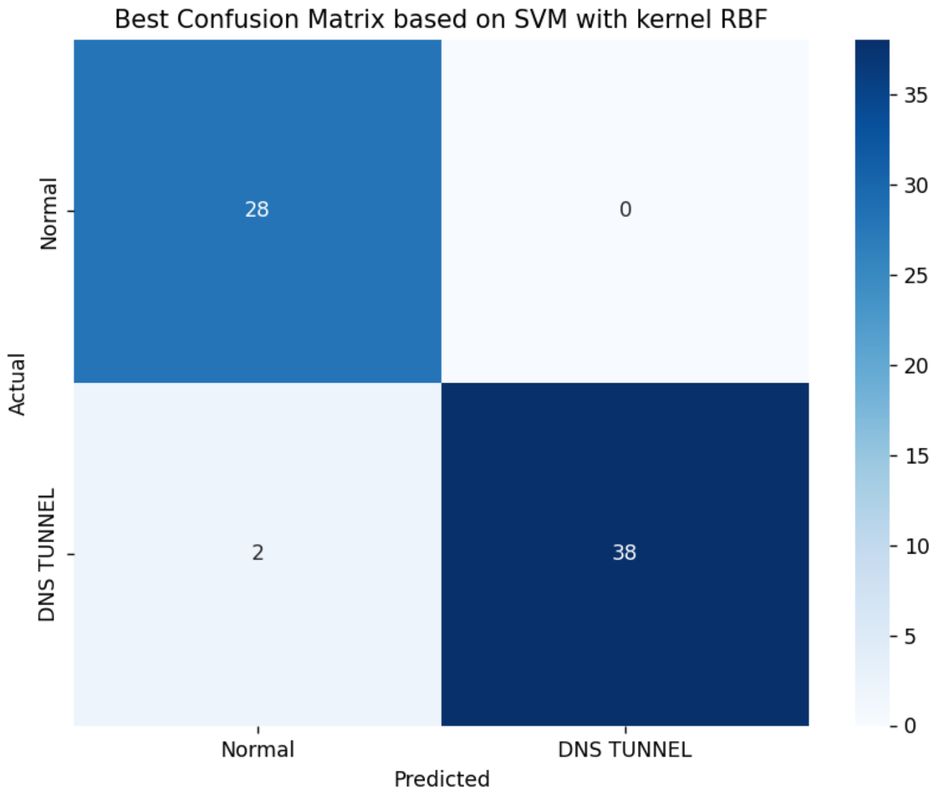



Marta's Model

# Results: SVM without PCA

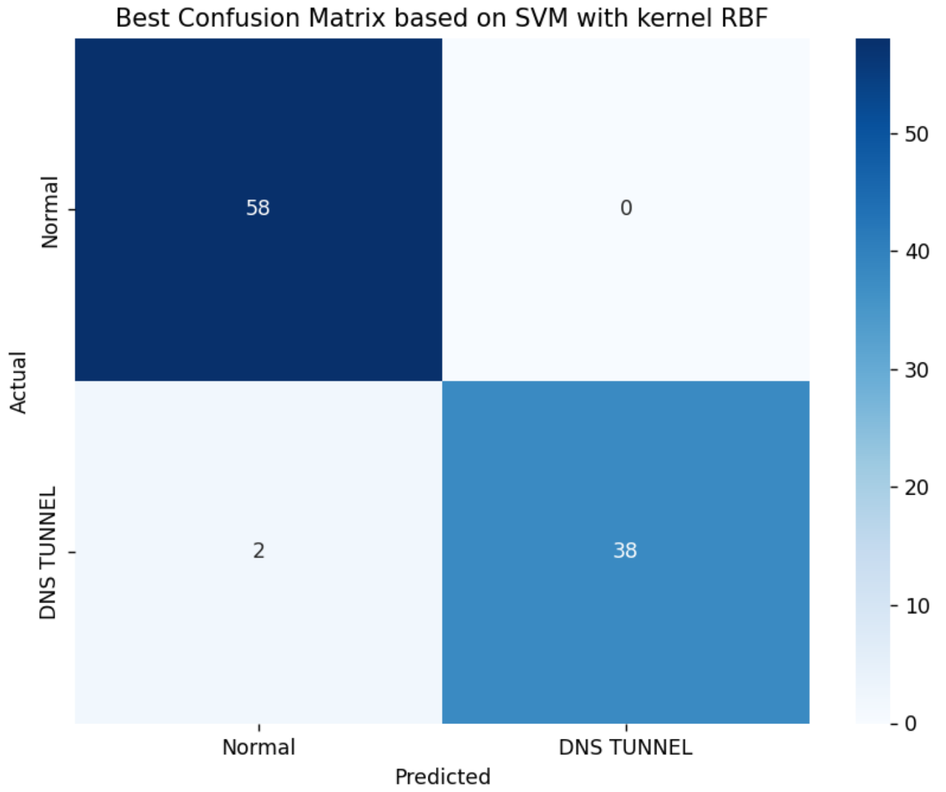
## Dumb Script

F1 Score: 97.4




Bruno's Model

F1 Score: 97.4

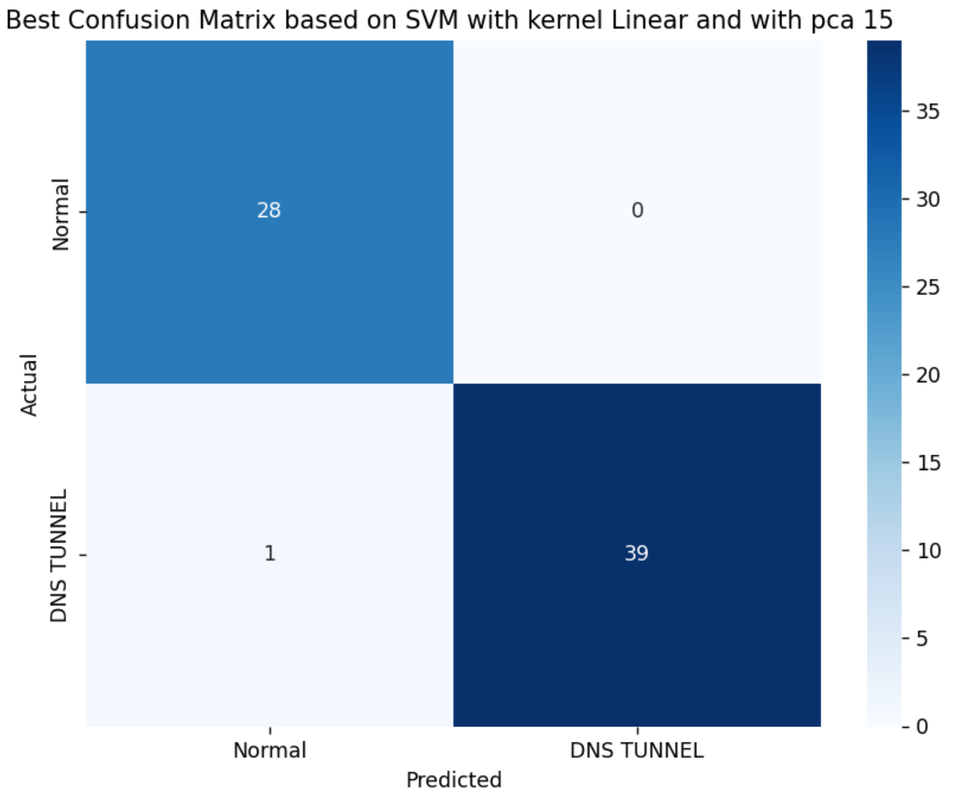



Marta's Model

# Results: SVM with PCA

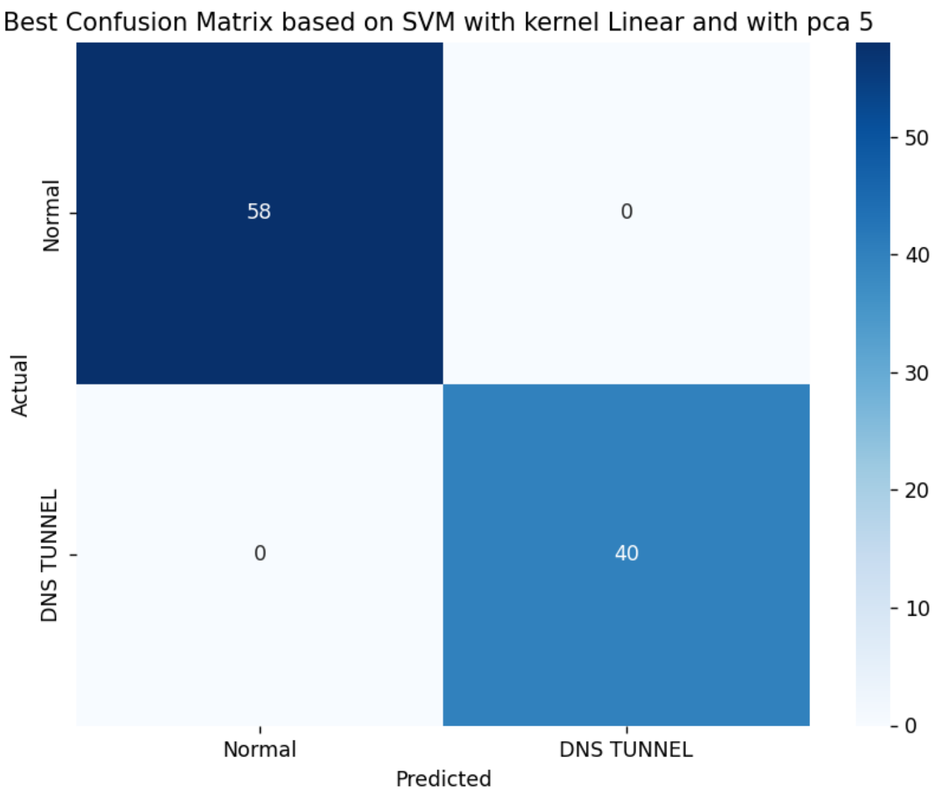
## Dumb Script

F1 Score: 97.4




Bruno’s Model

F1 Score: 100

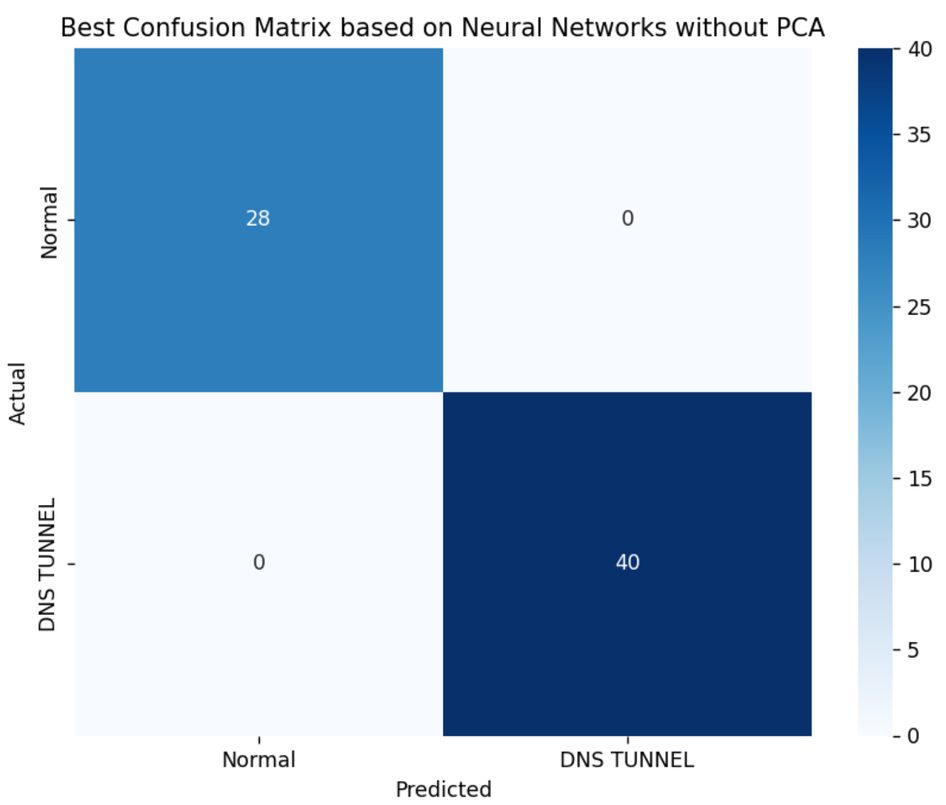



Marta’s Model

# Results: Neural Network without PCA

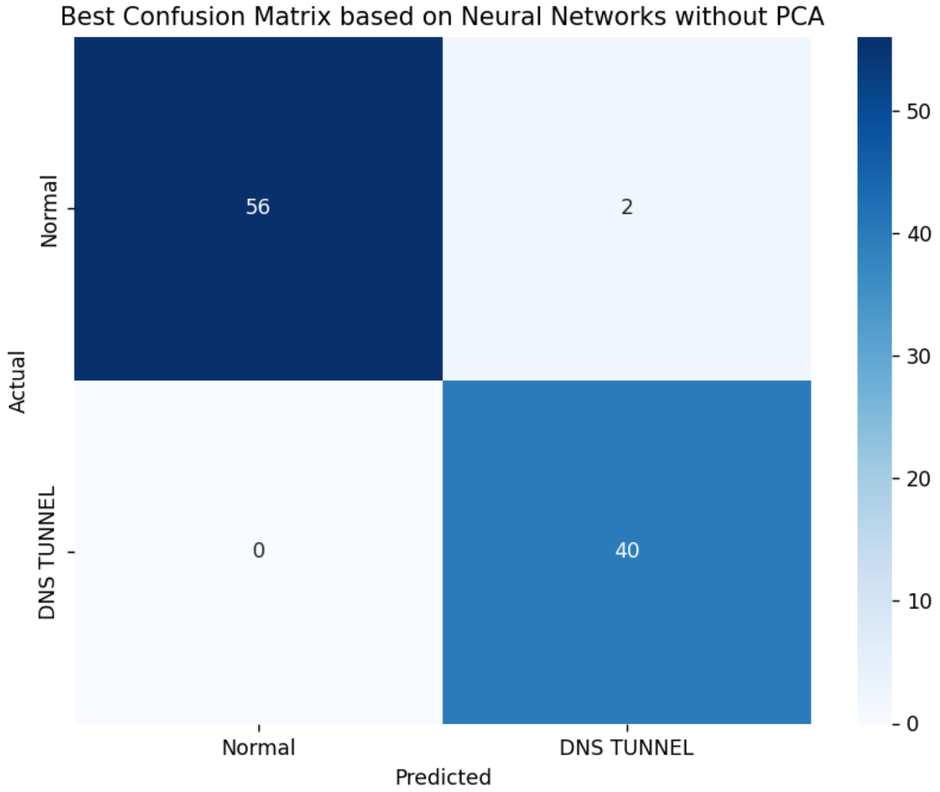
## Dumb Script

F1 Score: 100




Bruno's Model

F1 Score: 97.6

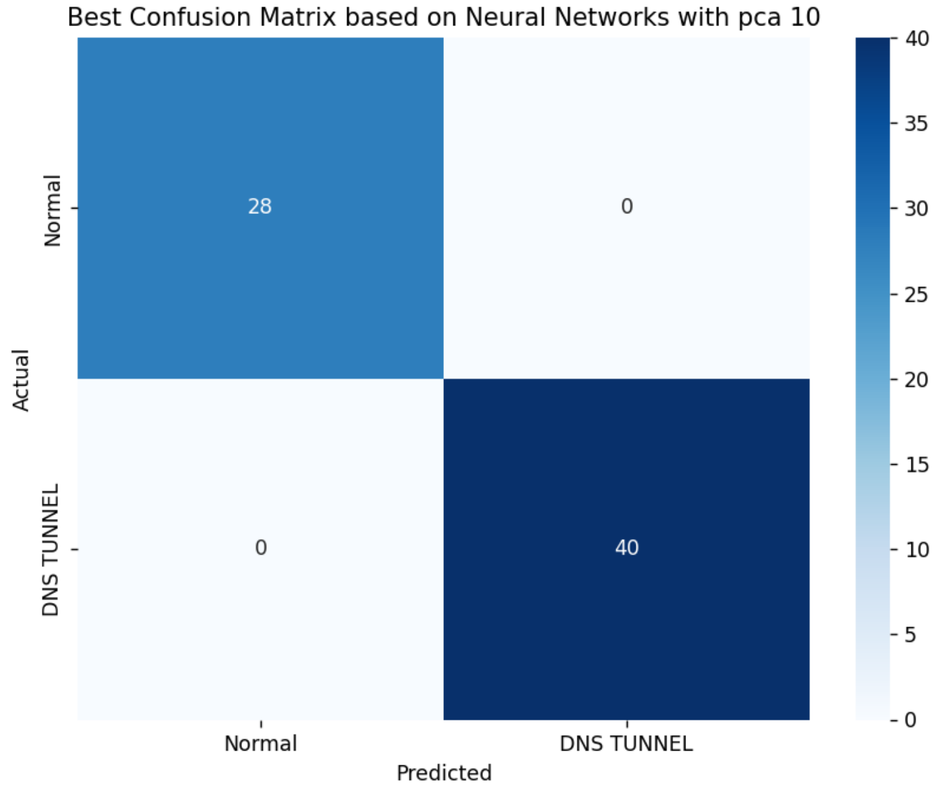



Marta's Model

# Results: Neural Network with PCA

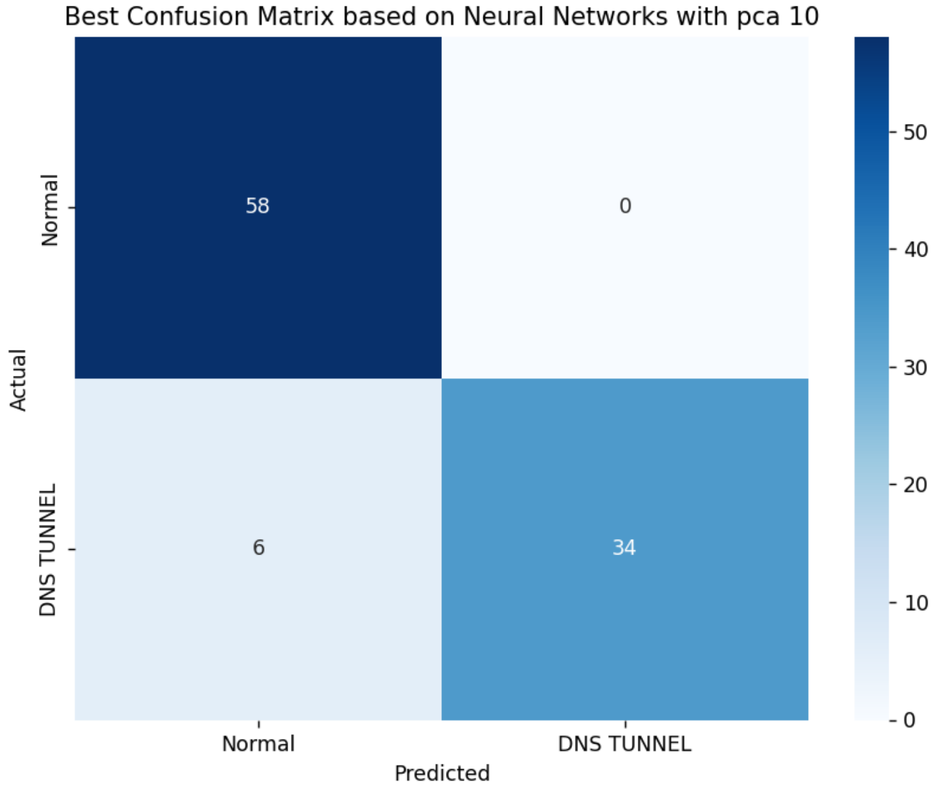
## Dumb Script

F1 Score: 100




Bruno's Model

F1 Score: 100

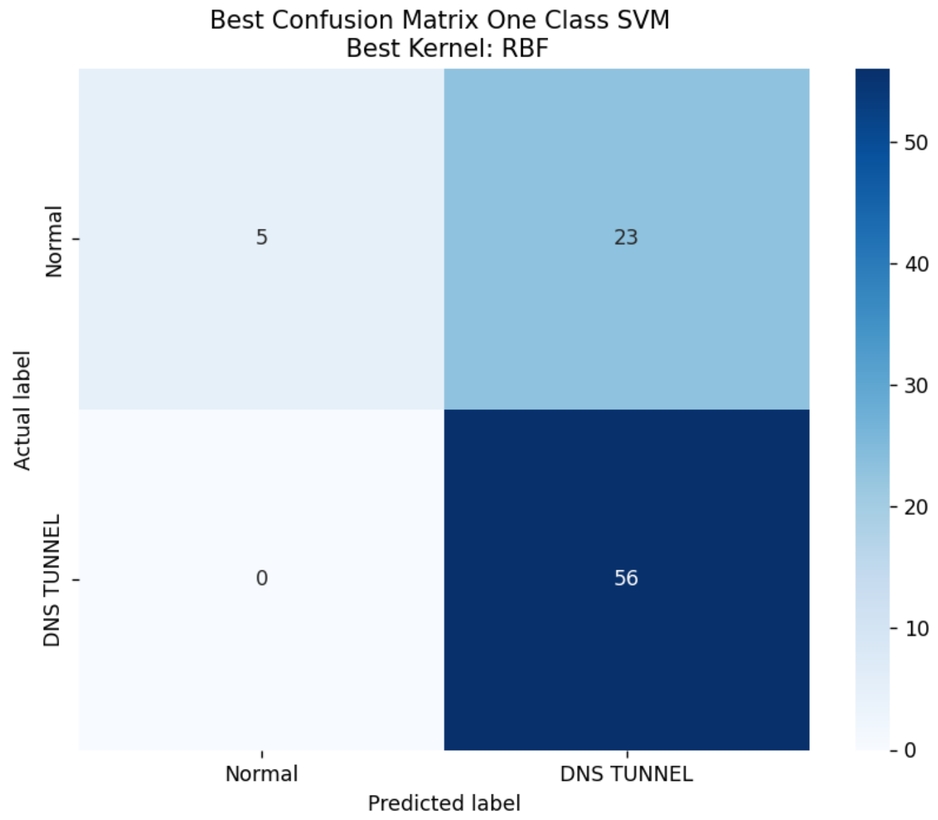



Marta's Model

# Results: One Class SVM without PCA

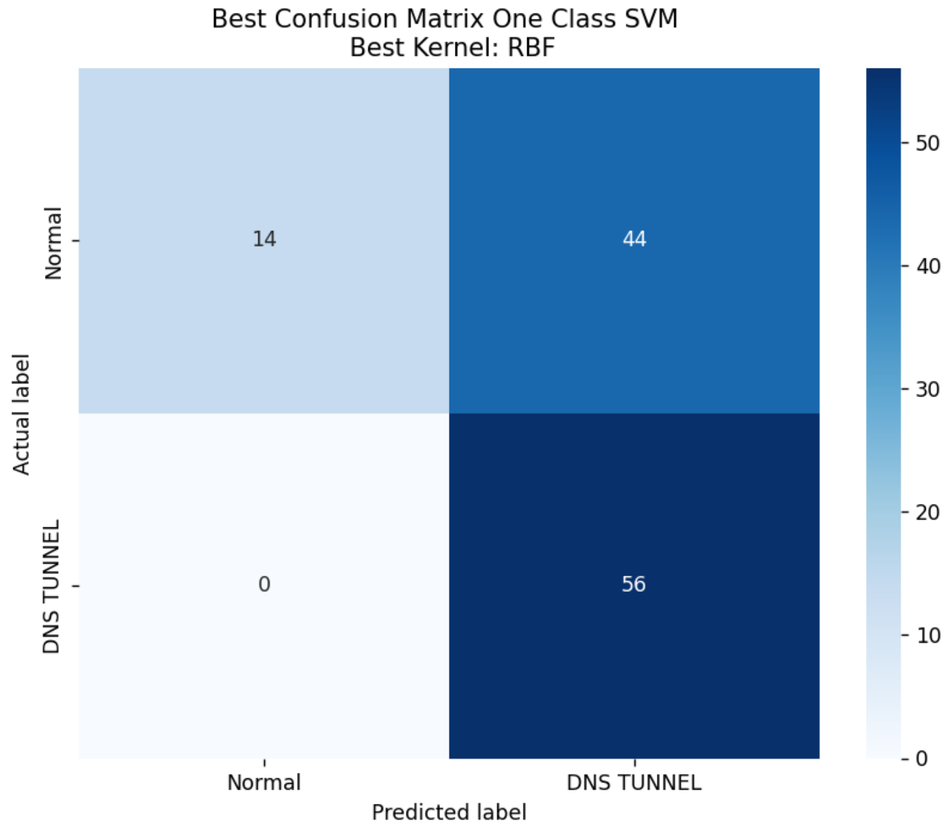
## Smart Script

F1 Score: 83




Bruno’s Model

F1 Score: 71.7

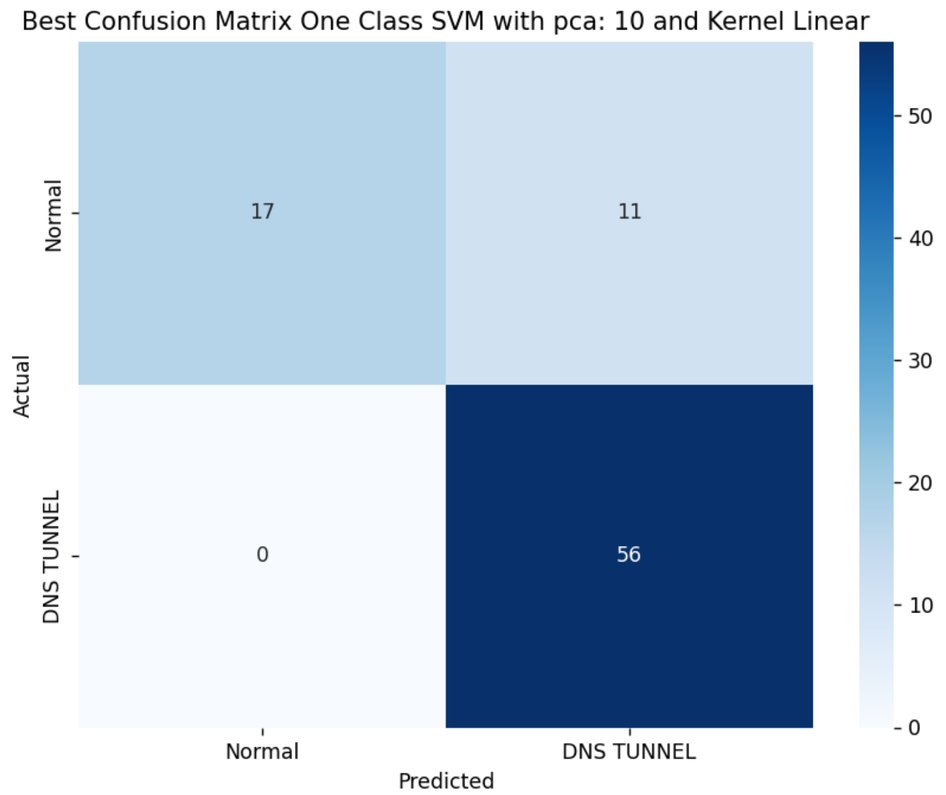



Marta’s Model

# Results: One Class SVM with PCA

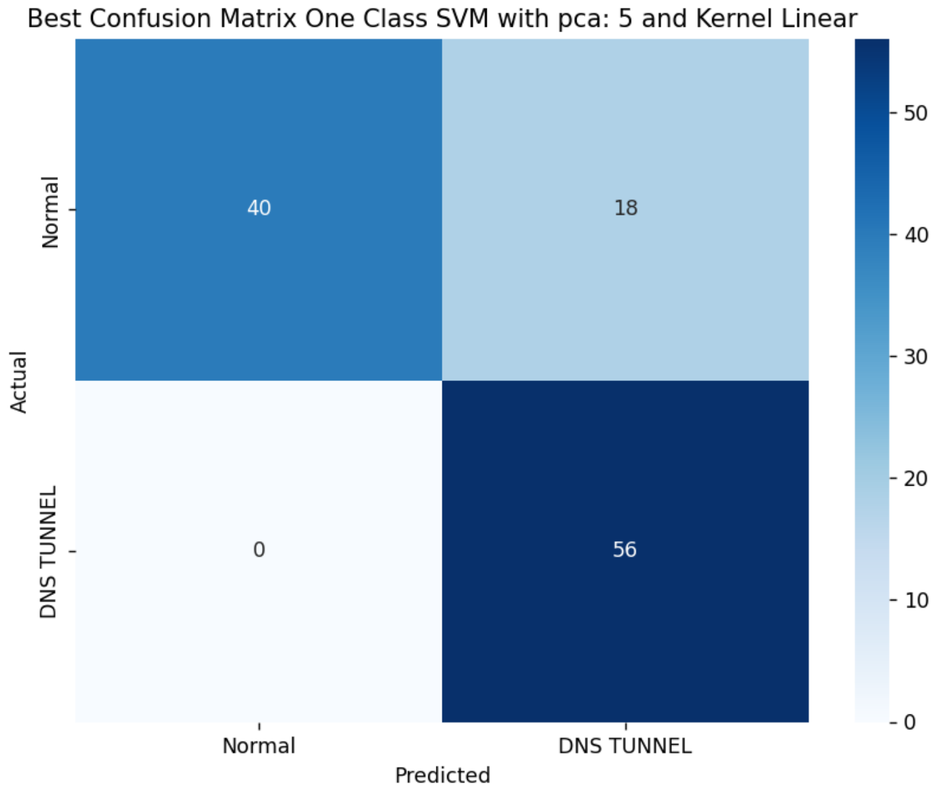
## Smart Script

F1 Score: 91.1




Bruno's Model

F1 Score: 86.1

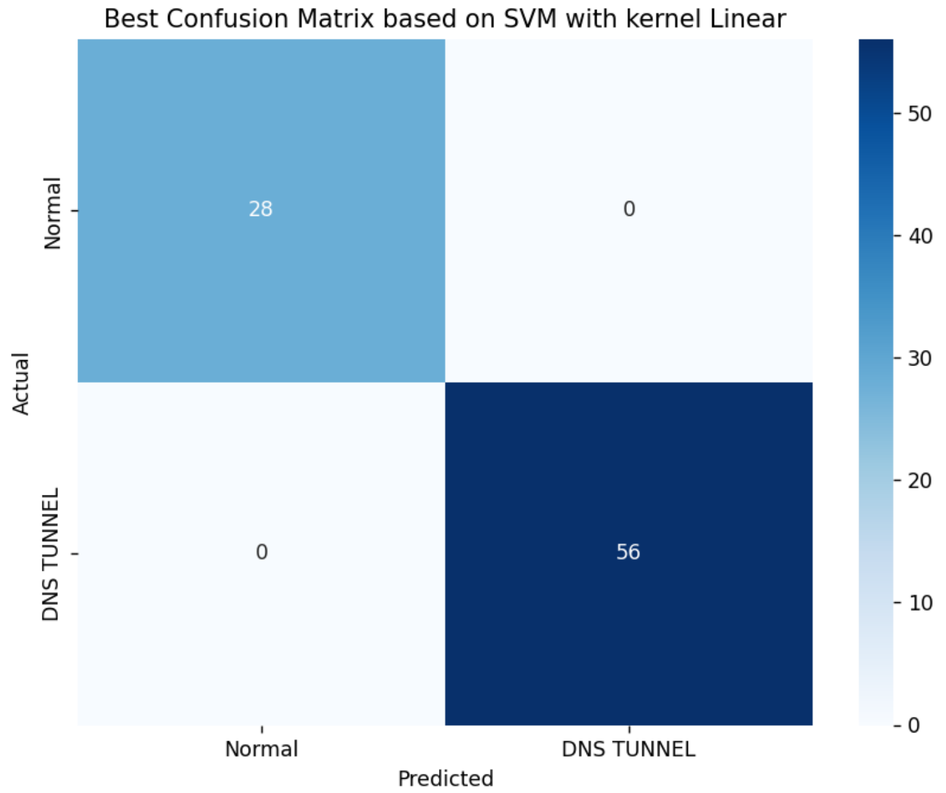



Marta's Model

# Results: SVM without PCA

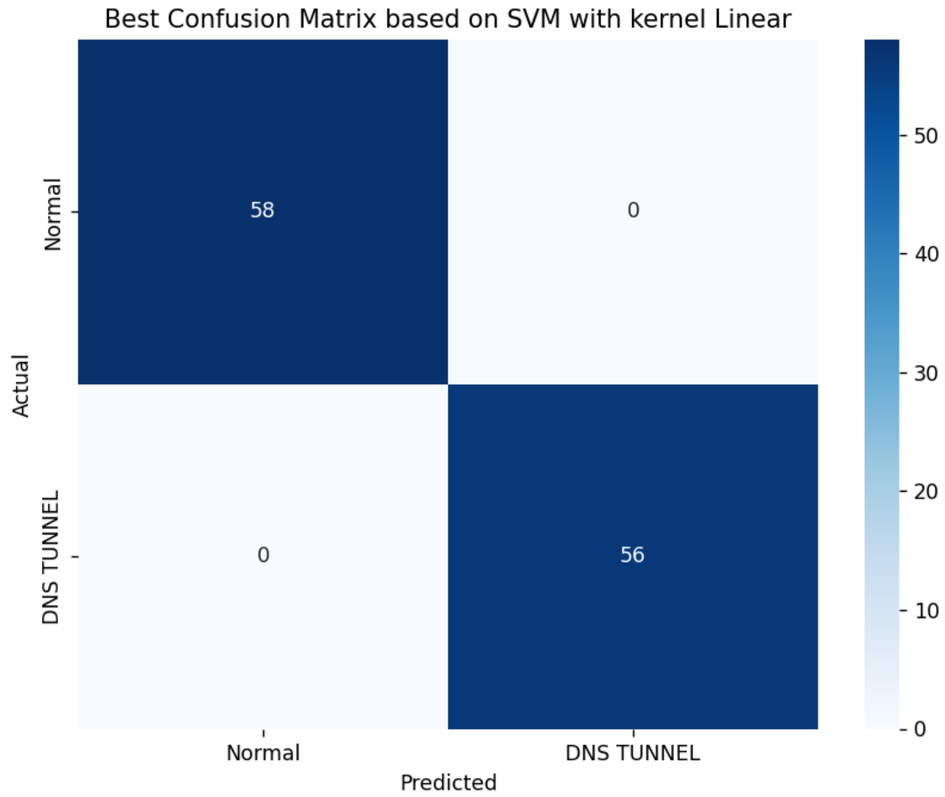
## Smart Script

F1 Score: 100




Bruno's Model

F1 Score: 100

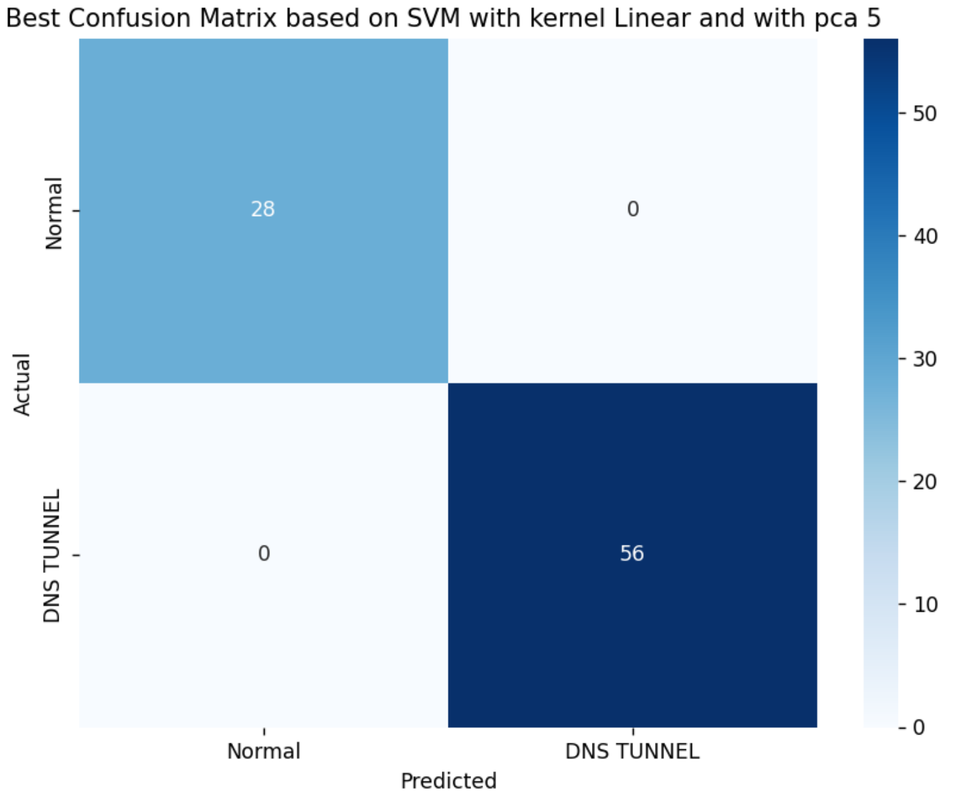



Marta's Model

# Results: SVM with PCA

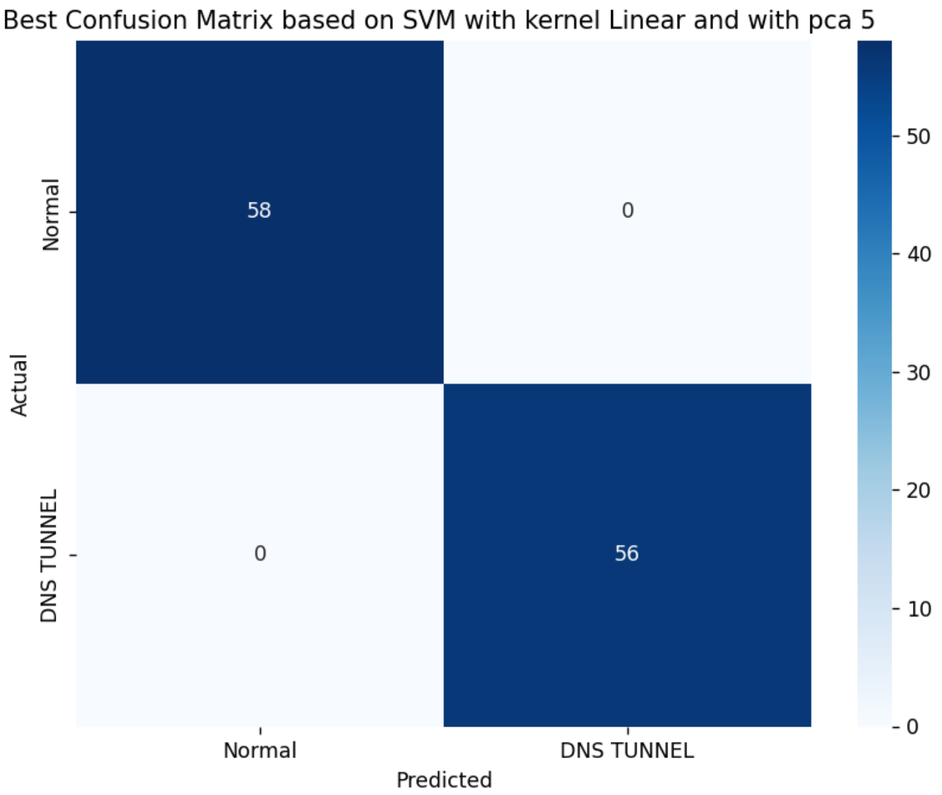
## Smart Script

F1 Score: 100




Bruno’s Model

F1 Score: 100

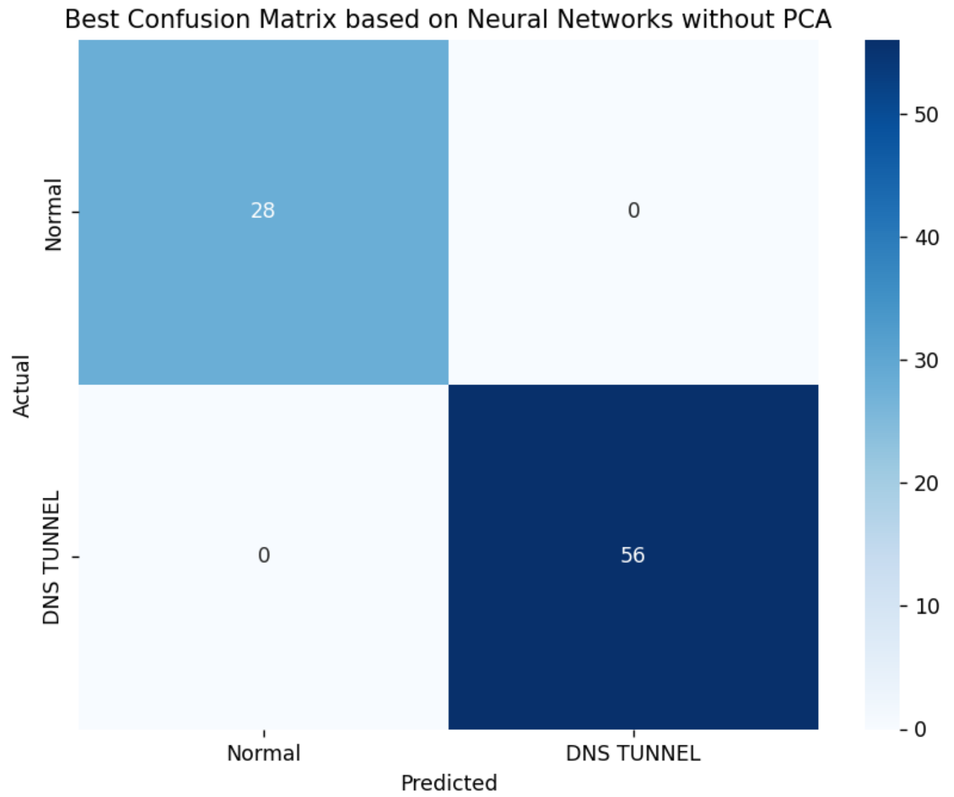



Marta’s Model

# Results: Neural Network without PCA

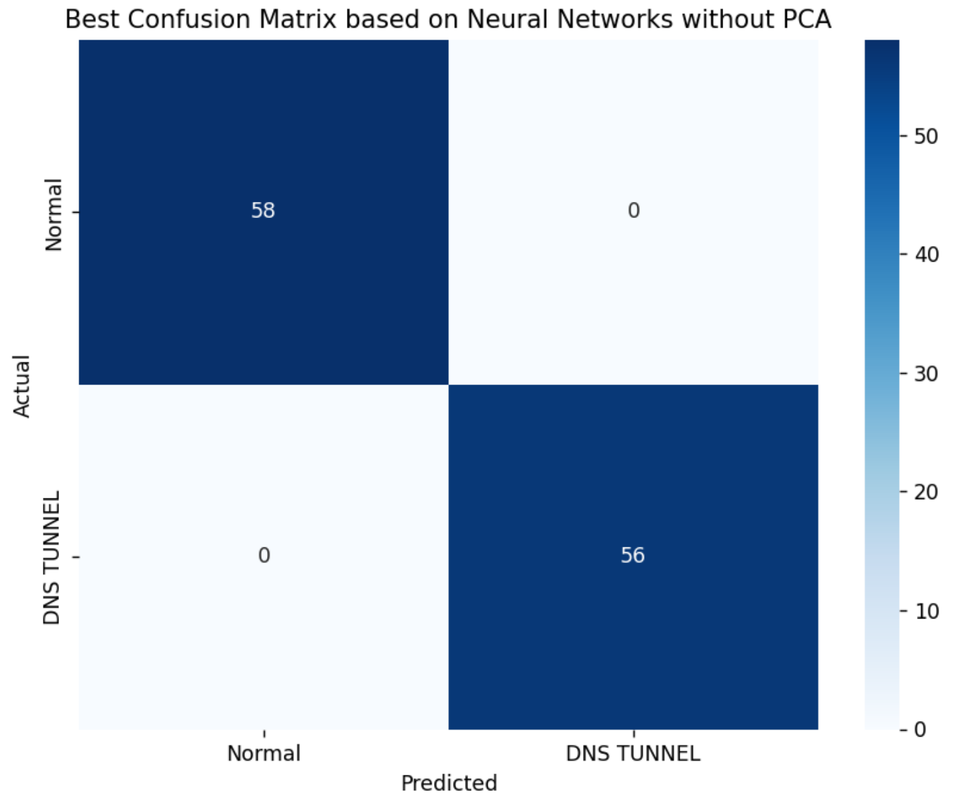
## Smart Script

F1 Score: 100




Bruno’s Model

F1 Score: 100

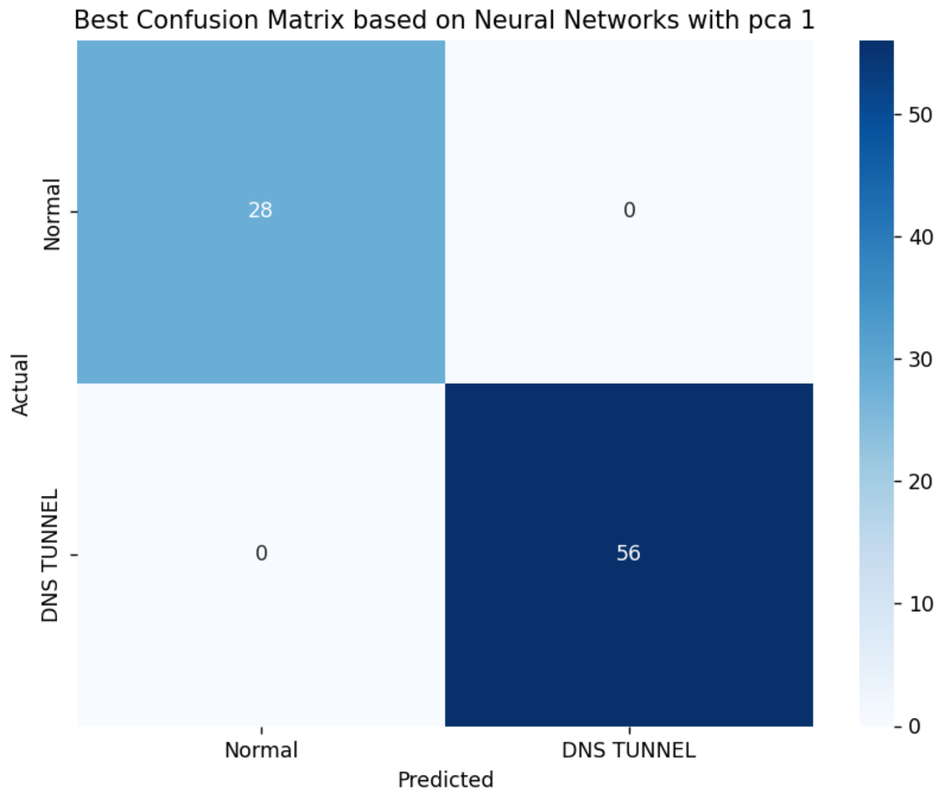



Marta’s Model

# Results: Neural Network with PCA

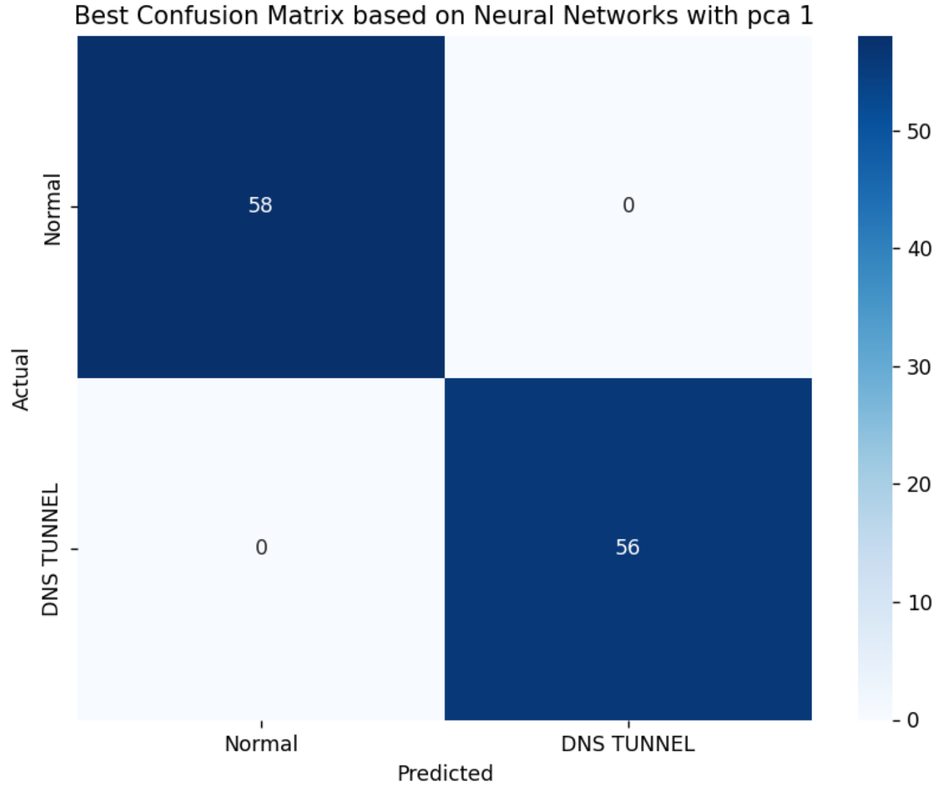
## Smart Script

F1 Score: 100




Bruno’s Model

F1 Score: 100




Marta’s Model

# Ensemble

- Bagging Method
  - Final decision based on the results given by each methodology used with equal weight

Dumb script:

```
Mean F1 Score: 91.31893387047866
The mean F1 Score is above 50%. Possible attack detected!
```

Smart script:

```
Mean F1 Score: 94.65224459574743
The mean F1 Score is above 50%. Possible attack detected!
```

# Code

- https://github.com/martaaoliveira/TPR_PROJETO