TÉCNICAS DE PERCEPÇÃO DE REDES

Project Presentation

Nuno Ferreira 121758
Patricia Cardoso 103243

OVERVIEW

Contextualization

Our Scenario

Data Sources

Data Processing

Metrics Extraction

Features Extraction

Features Analysis Over Time

Decision by Statistical Patterns

Training and Test features

Anomaly Detection Algorithms

Conclusions

APPLICATION LAYER DDOS ATTACK

It is designed to target the “top” layer in the OSI model where HTTP requests occur.

Overwhelms specific features of a website or app to disable them, preventing the application from delivering content to users.

They often target specific functions, such as login that require more processing power than static pages.

Challenge of L7 Attack for Traditional Solutions

01

Mimic legitimate traffic, making detection difficult

Layer 7 attacks, are designed to replicate normal user behavior

02

Low Traffic Volume

Unlike volumetric attacks that generate high traffic, Layer 7 attacks can be executed with a small number of requests

03

Limited Detection Capabilities

Many firewalls, IPS focus on network layer threats and are not equipped to analyze the content and context of application layer traffic

OUR SCENARIO

- Creation of a Website
- The client interacts with the Website
- Apache2 Web Server Logs are saved

DATA SOURCES

Logs were collected from two entities:

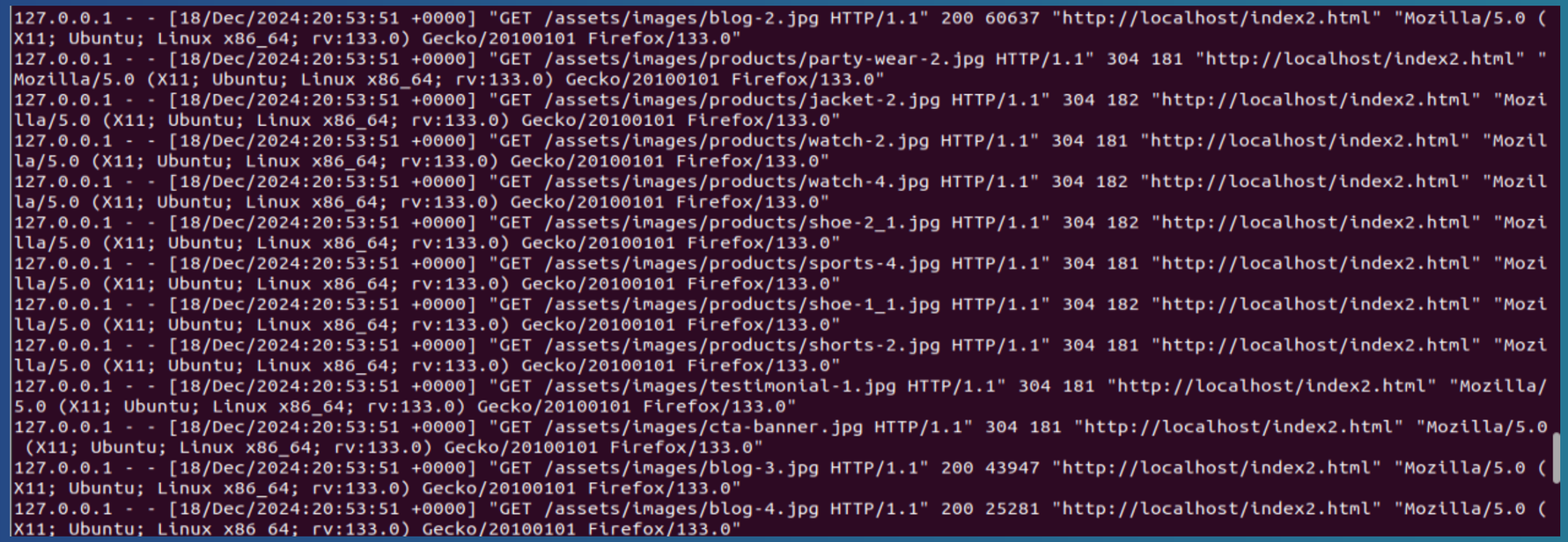
- **Normal behaviour**
- **Malicious behaviour**: Simulated by bots executing DDoS attacks, including repetitive requests to the homepage and its associated resources (CSS, JavaScript, and images). Using “Python Requests” and “BeautifulSoup”.
  - Creation of 8 bot levels.

Each normal user and each bot interacted one hour with the Website

DATA PROCESSING

Goal: Identify unusual events or patterns in the data by comparing malicious and normal user behavior

Data Filtering: Apache2 Web Server Logs fields.



DATA PROCESSING

Time Windowing:

- Main Window: 5-minute windows.Each 5-minute window has a sliding value of 1 minute
- Sub-Window: 30-second sub-windows within each main window.
- Sliding Window: Each 5-minute window advances in 1-minute increments.

Data sampling/aggregation:

- Metric Counting: Metrics are counted in each 30-second sub-window.
- Feature Creation: Features are generated for each 5-minute window based on aggregated metrics from the sub-windows

METRICS EXTRACTION

For each 30-second sub-windows we extract:

- numRequests
- tamanhoResposta
- totalImageSize
- RequestsJS
- RequestsHTML
- RequestsCSS
- totalSilenceTime
- silenceCount

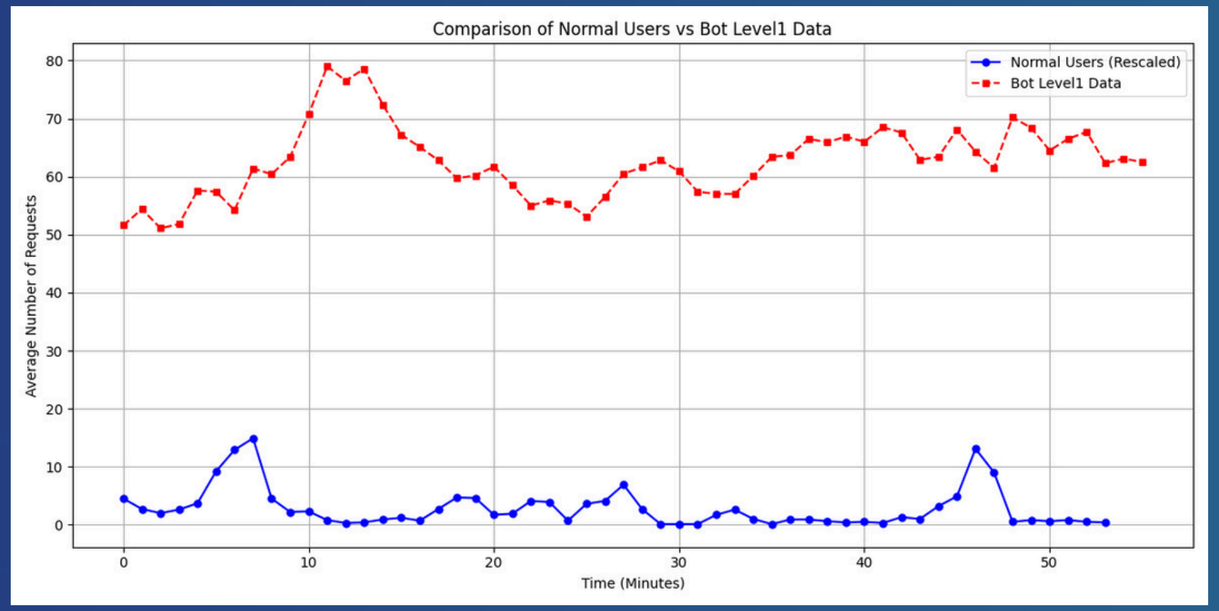
FEATURE EXTRACTION

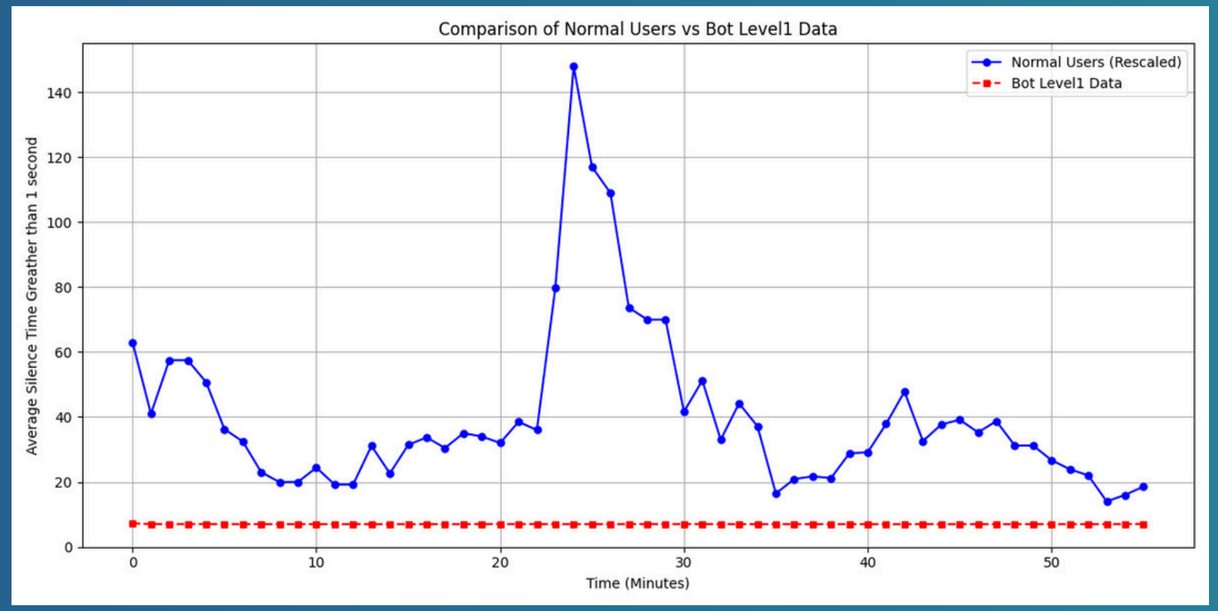
For each 5-minute window we extract:

- avgNumRequests
- avgTamanhoResposta
- avgTotalImageSize
- avgRequestsJS
- avgRequestsHTML
- avgRequestsCSS
- avgSilenceTime

FEATURES ANALYSIS OVER TIME

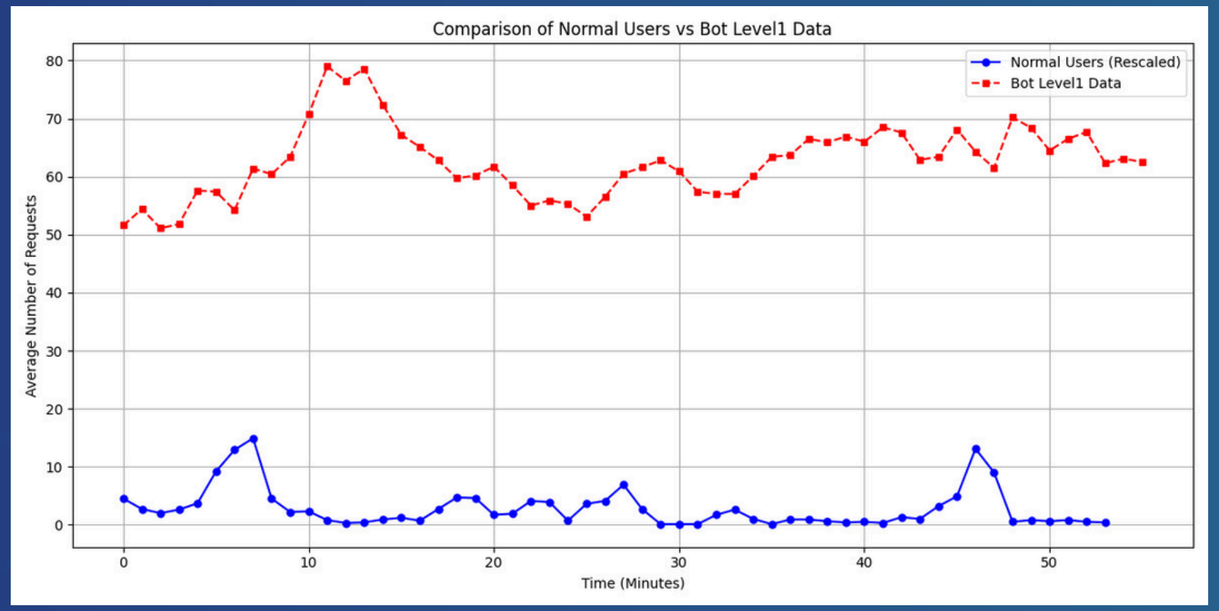
- Lvl 1: Repeatedly request the same page every "7" seconds

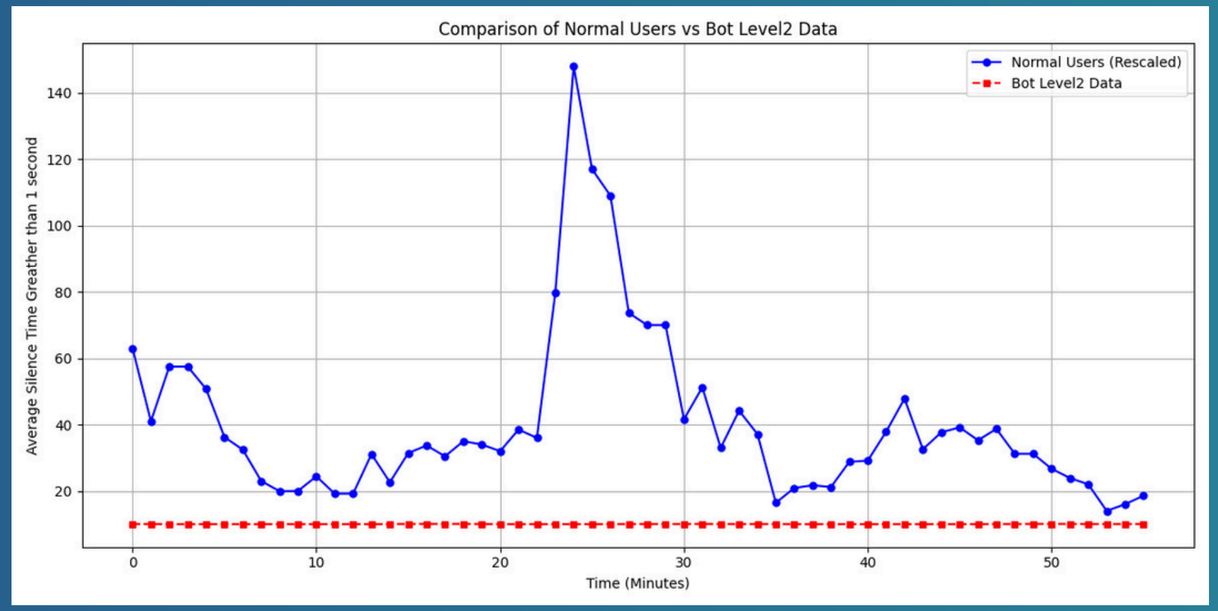




FEATURES ANALYSIS OVER TIME

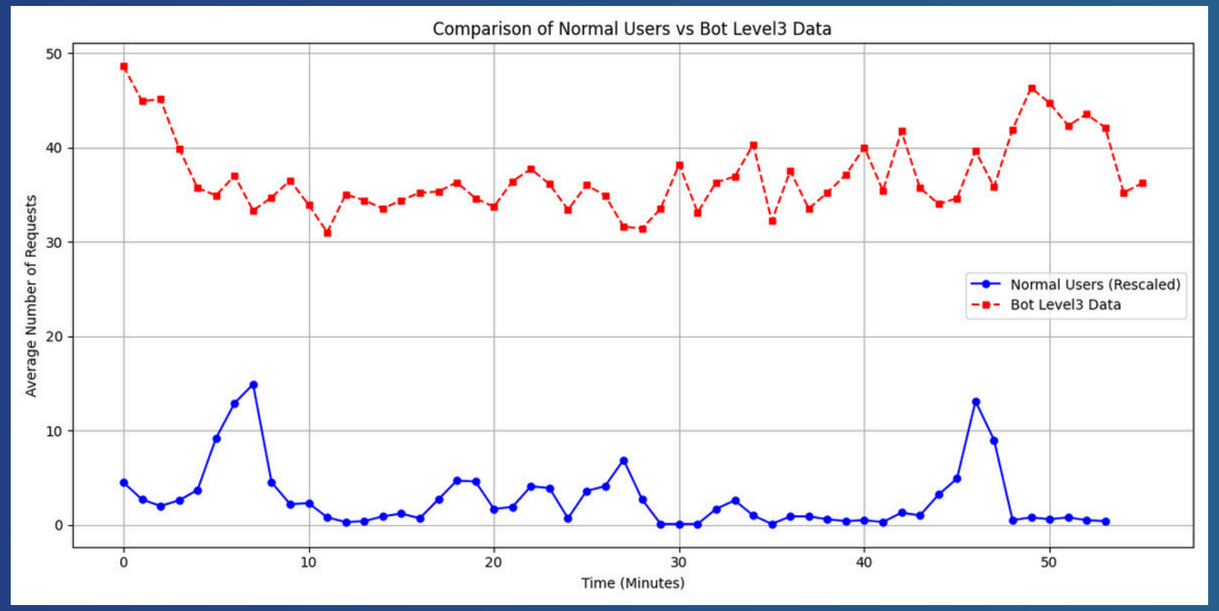
- Lvl 2: Repeatedly request 4 fixed pages randomly every "10" seconds

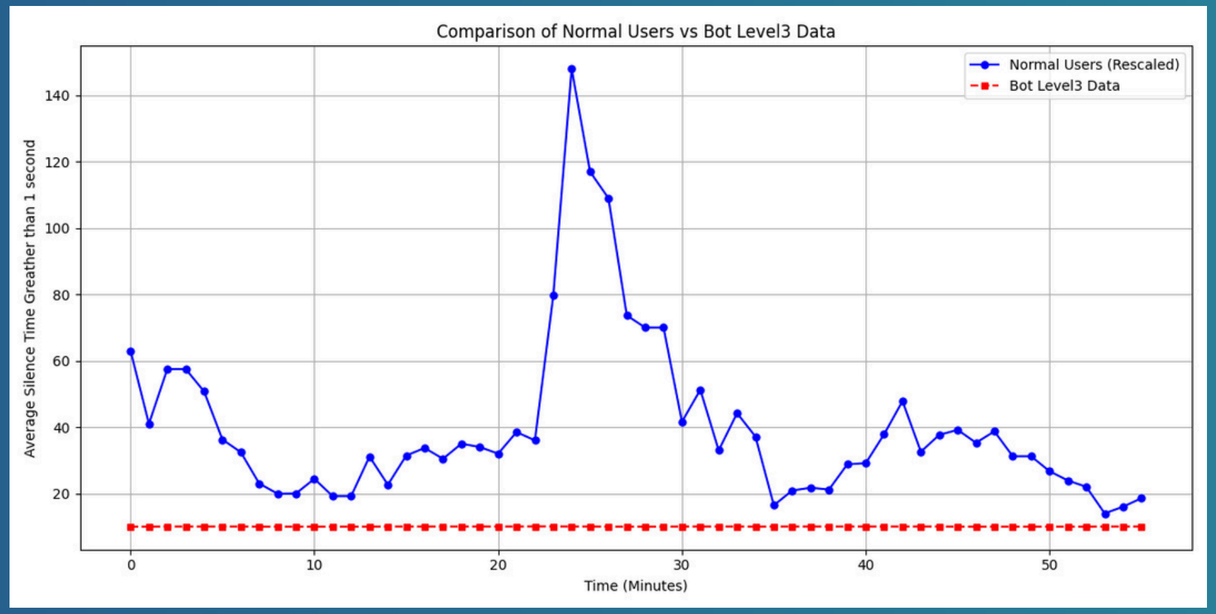




FEATURES ANALYSIS OVER TIME

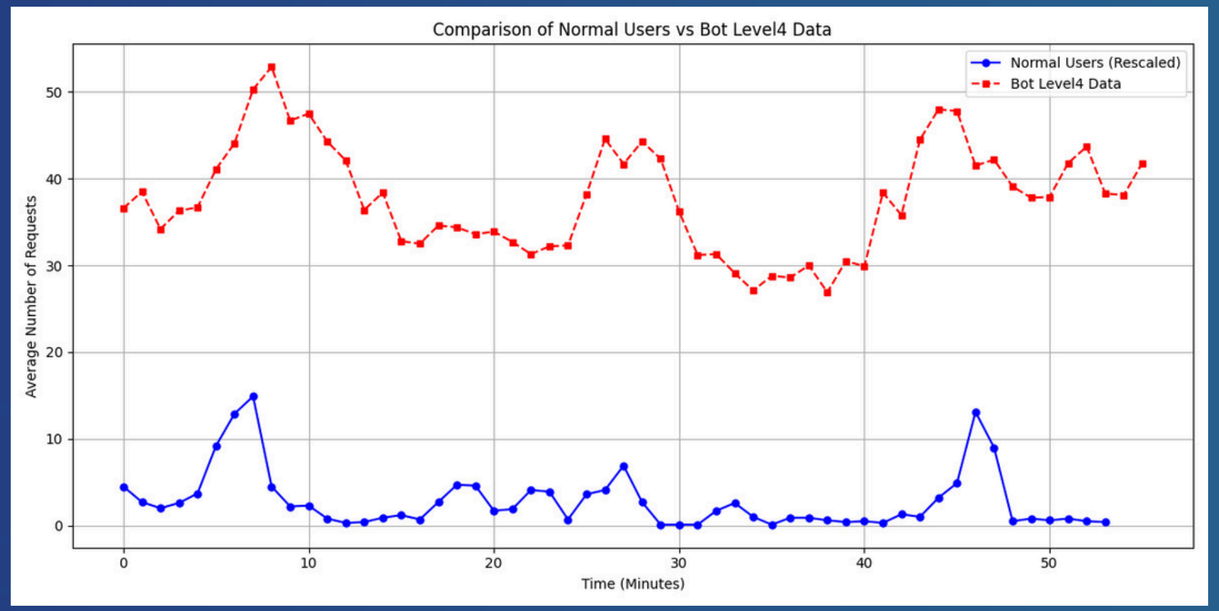
- Lvl 3: Repeatedly request a random page every "10" seconds

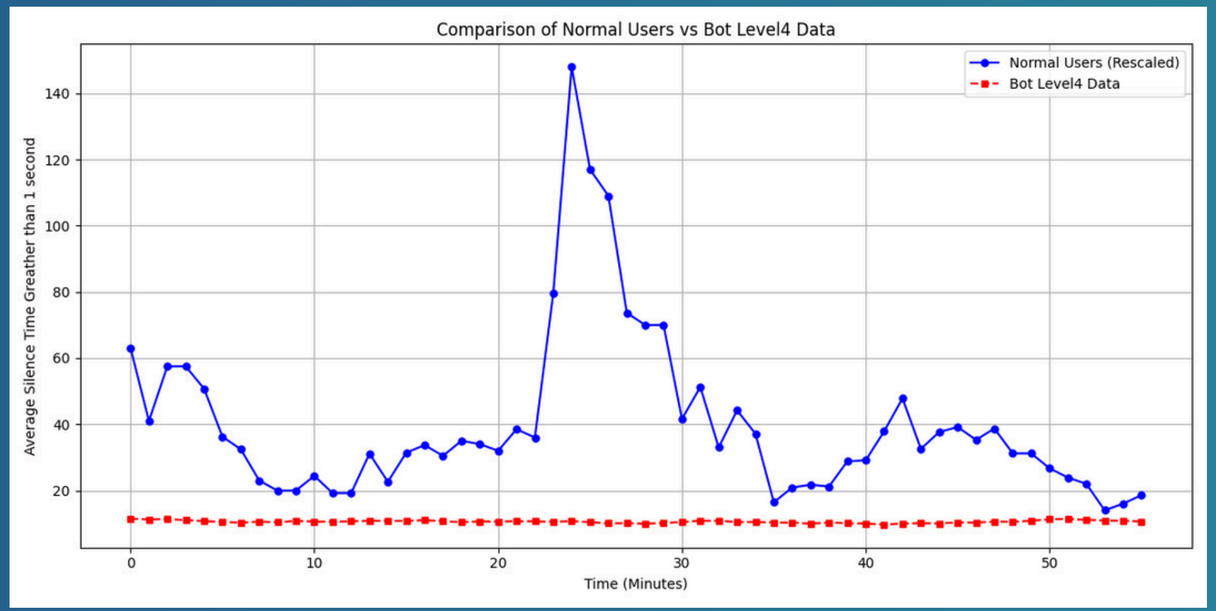




FEATURES ANALYSIS OVER TIME

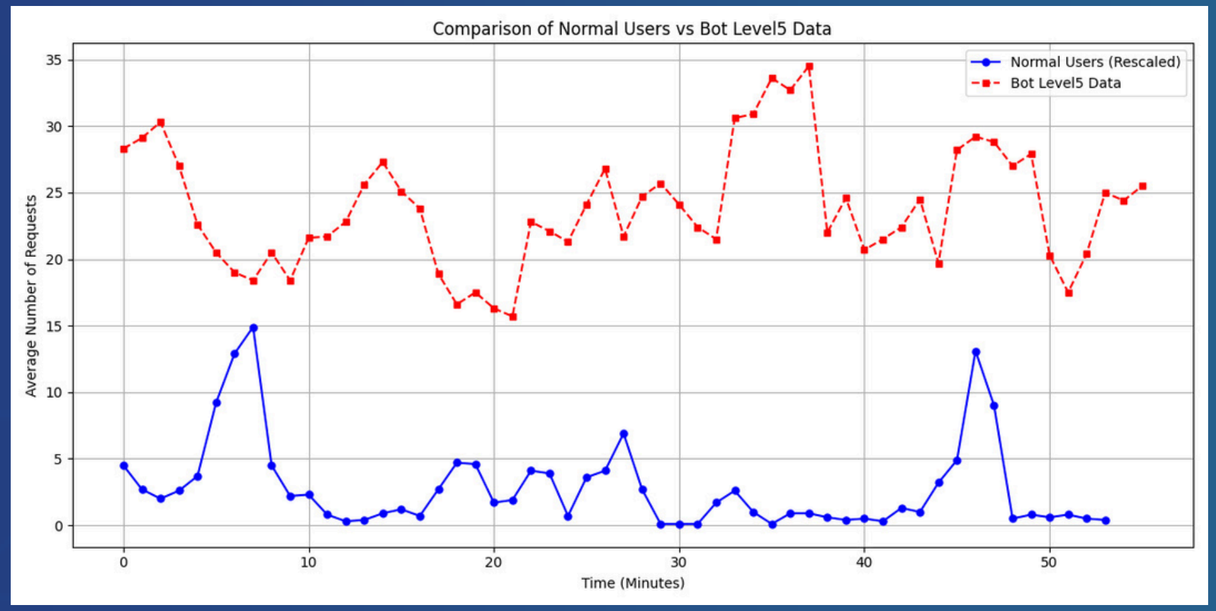
- Lvl 4: Repeatedly request a random page every random seconds

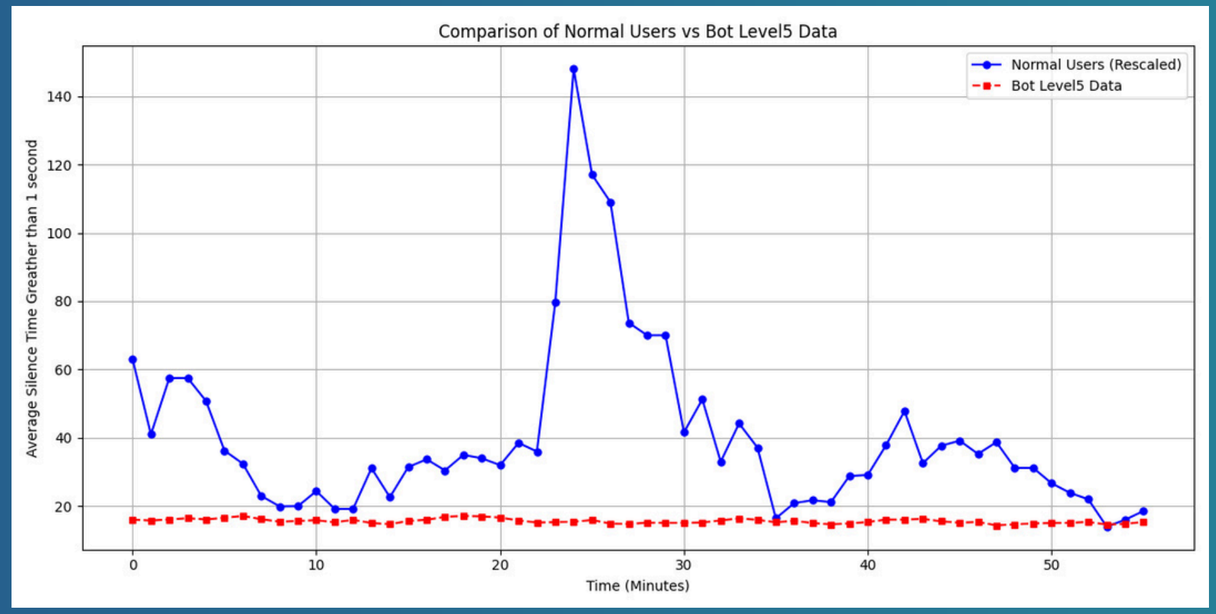




FEATURES ANALYSIS OVER TIME

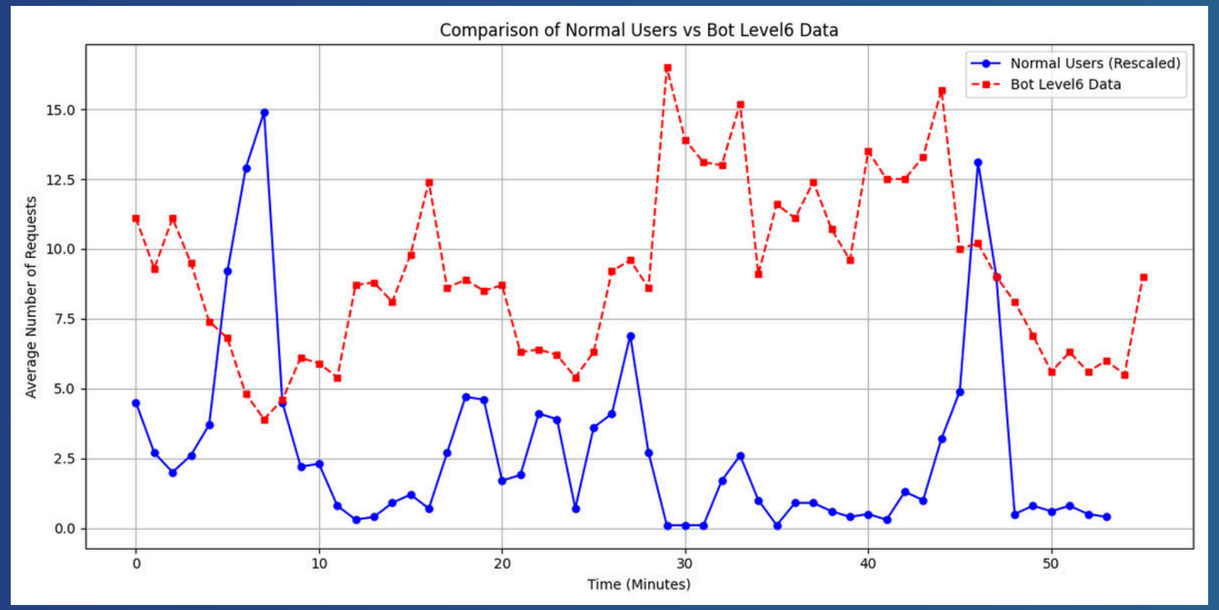
- Lvl 5: Repeatedly request a random page every random seconds based on a Gaussian distribution (average interval: 16s, std: 4)

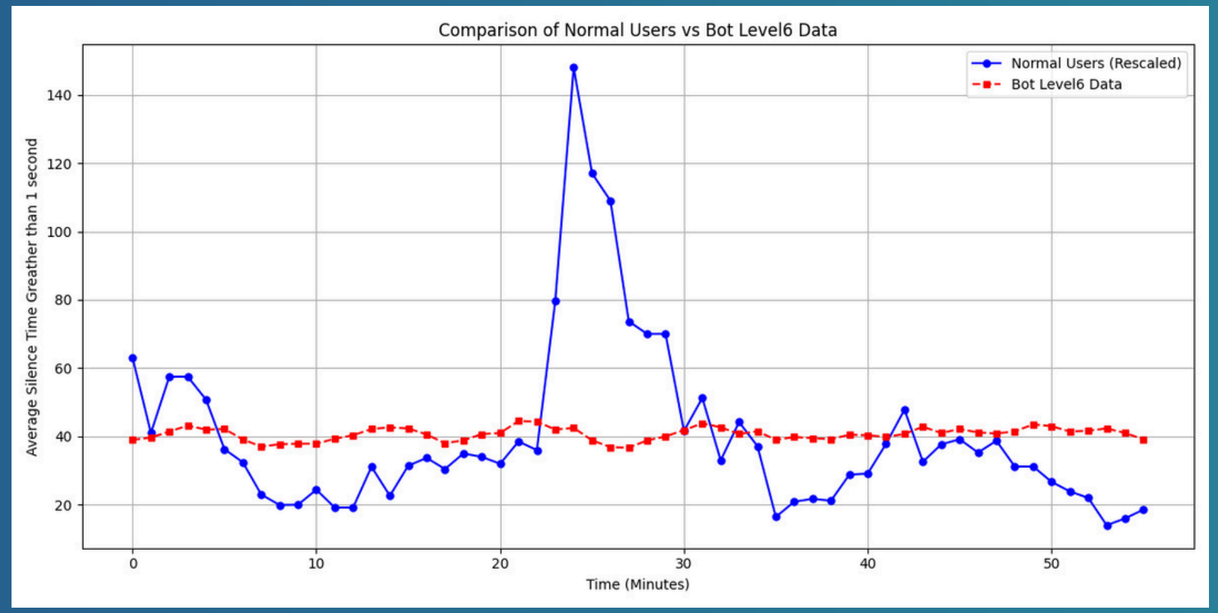




FEATURES ANALYSIS OVER TIME

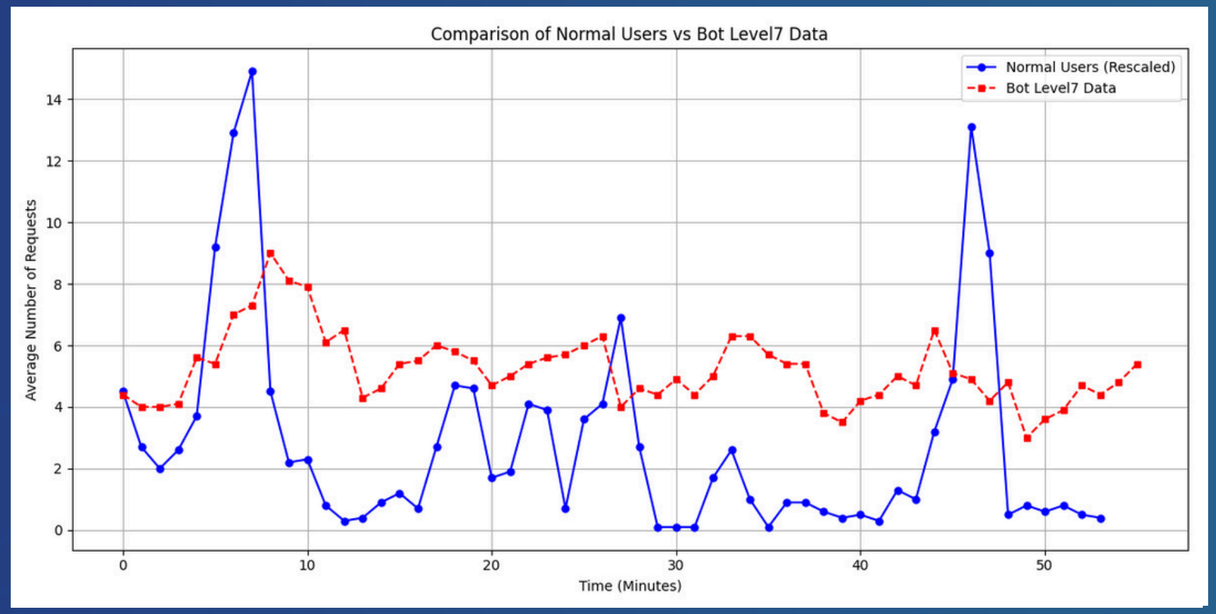
- Lvl 6: Repeatedly requests random pages at intervals based on a Gaussian distribution with a 40 second mean and 5 second standard deviation.

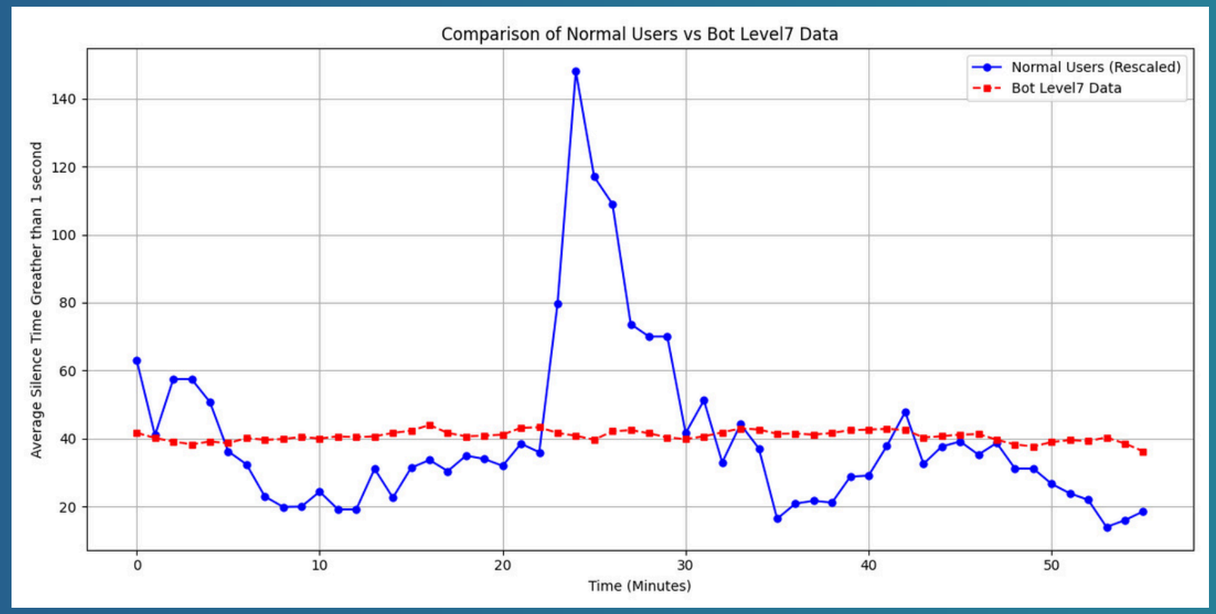




FEATURES ANALYSIS OVER TIME

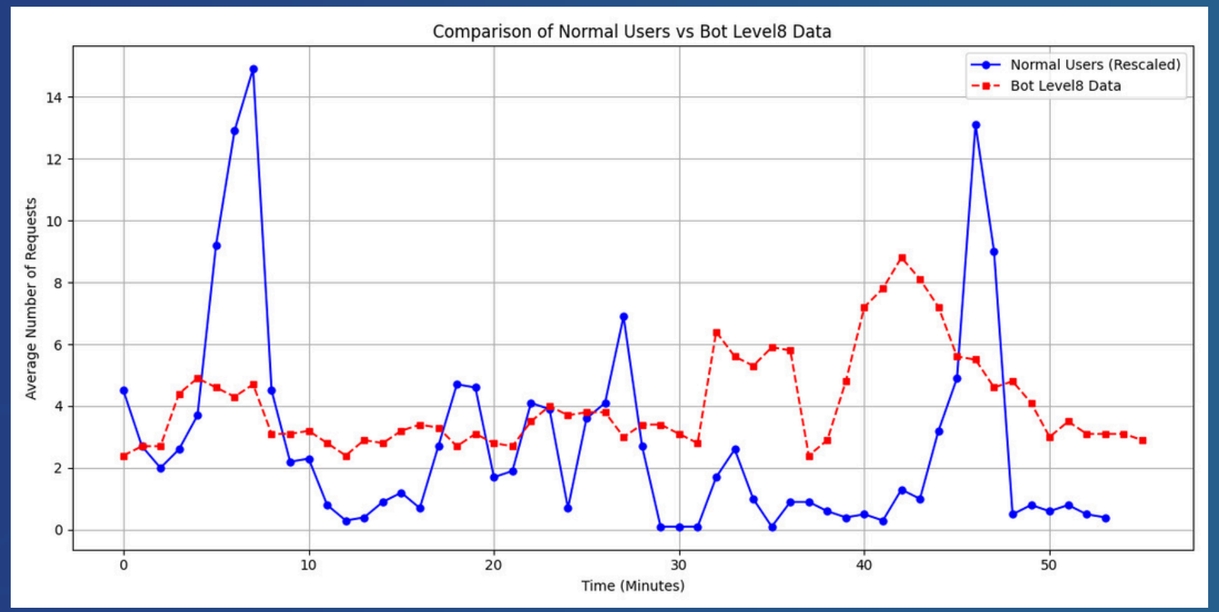
- Lvl 7: Repeatedly requests random pages at random intervals, using a Gaussian distribution based on the mean from the totalSilenceTime in the access logs

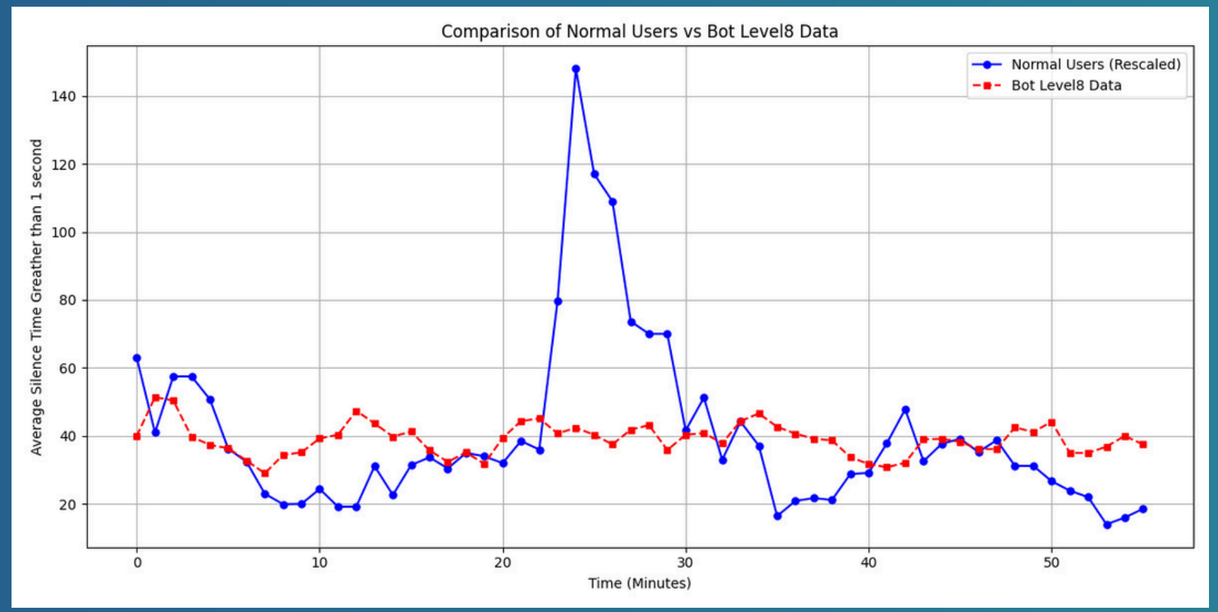




FEATURES ANALYSIS OVER TIME

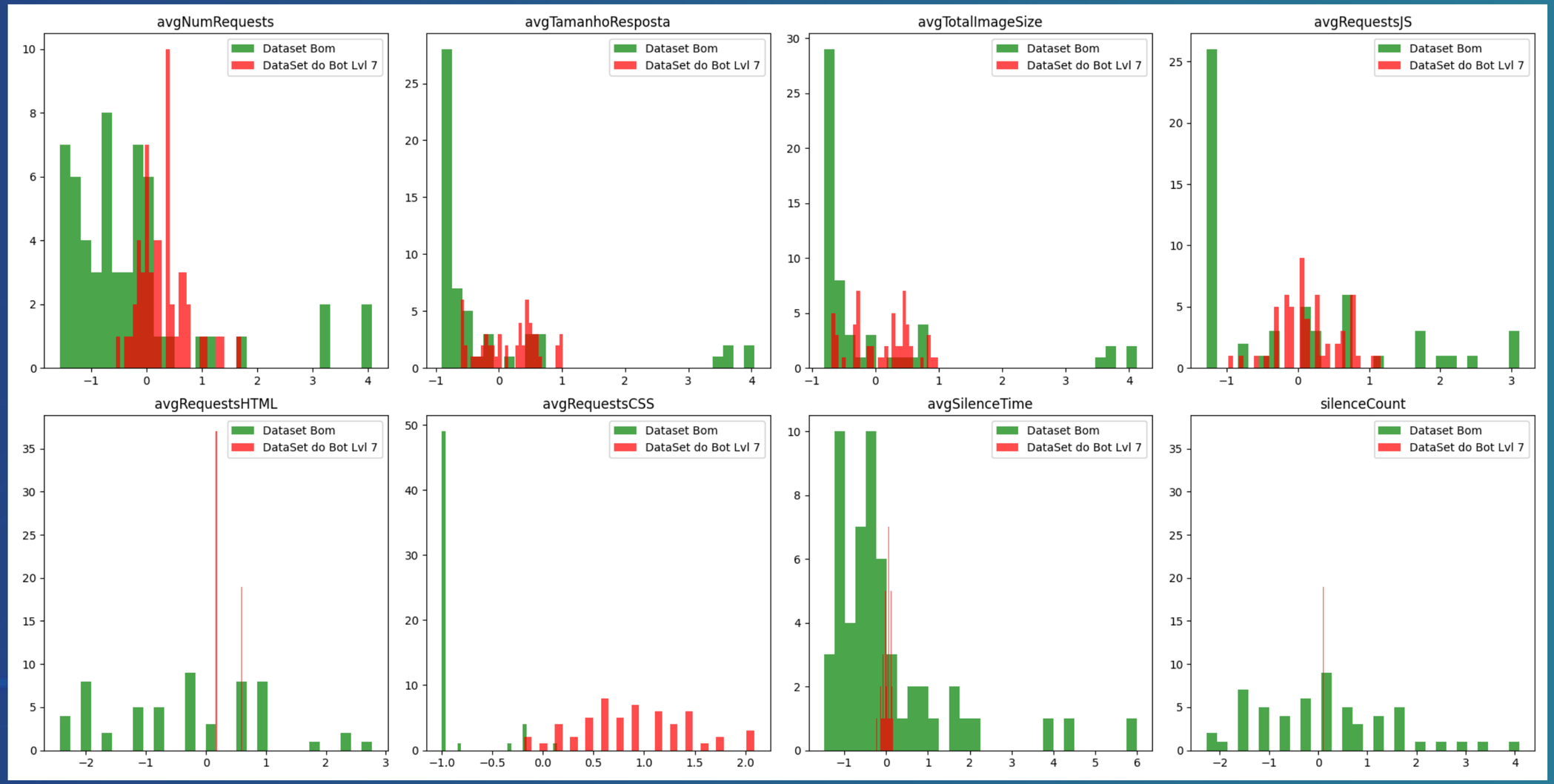
- Lvl 8: Repeatedly requests random pages at intervals based on a Gaussian distribution with a 40 second mean and 25 second standard deviation, but half of the requests every time.





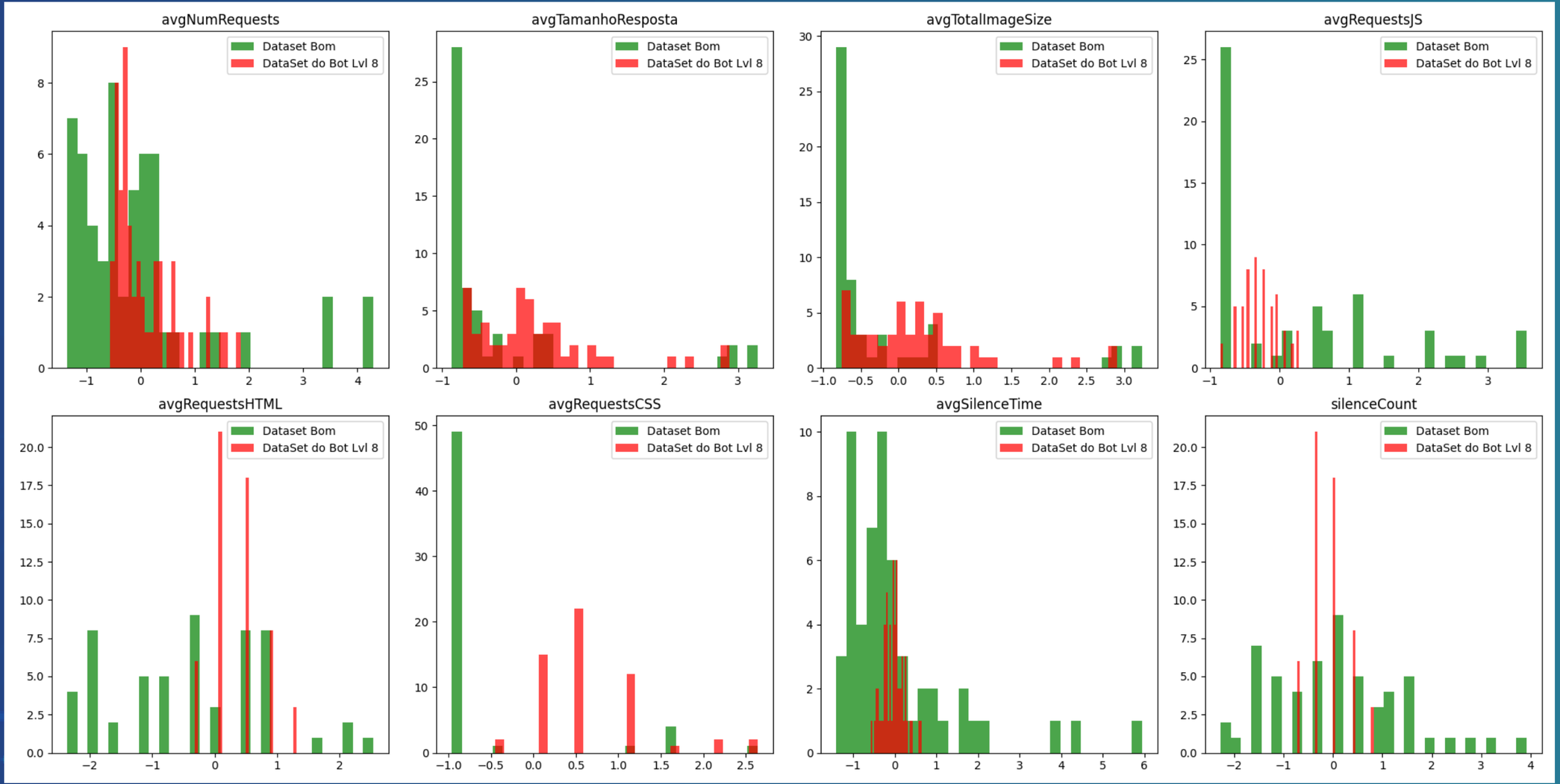
DECISION BY STATISTICAL PATTERNS

Z-Score: how many standard deviations a data point is from the mean (in our case normal distribution)



DECISION BY STATISTICAL PATTERNS

For bot lvl 8



DECISION BY STATISTICAL PATTERNS

Using Centroid Based Classification for Anomaly Detection

- **Data Normalization/Scaling:**

  By min/max, scaling each feature to a given range: [0, 1]

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- **Calculate The Centroid:**

  Calculate the mean value of each chosen feature across all training data points.

- **Calculate The Radius :**

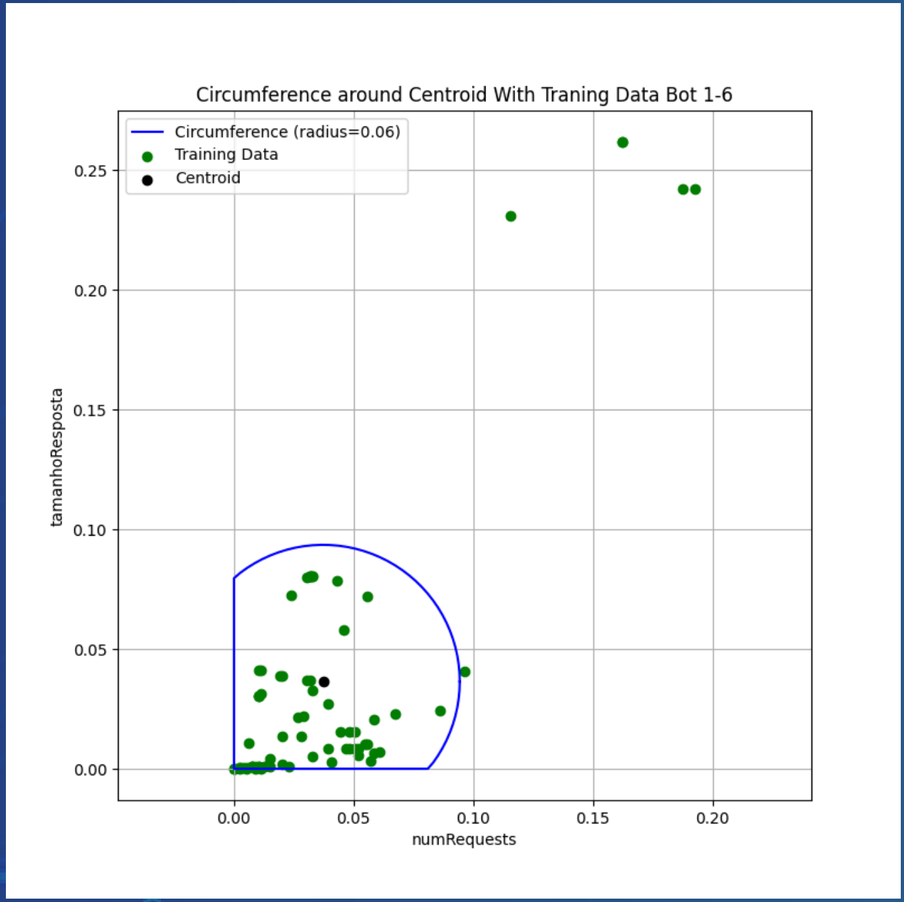
  Calculate the Euclidean distance from each point to the centroid and find the largest distance

  Calculate the Interquartile Range of the Euclidean distances (IQR), (Q1 and Q3 were the best results)

Radius = Median of distances + k× IQR IQR=Q3−Q1 k = 1.2

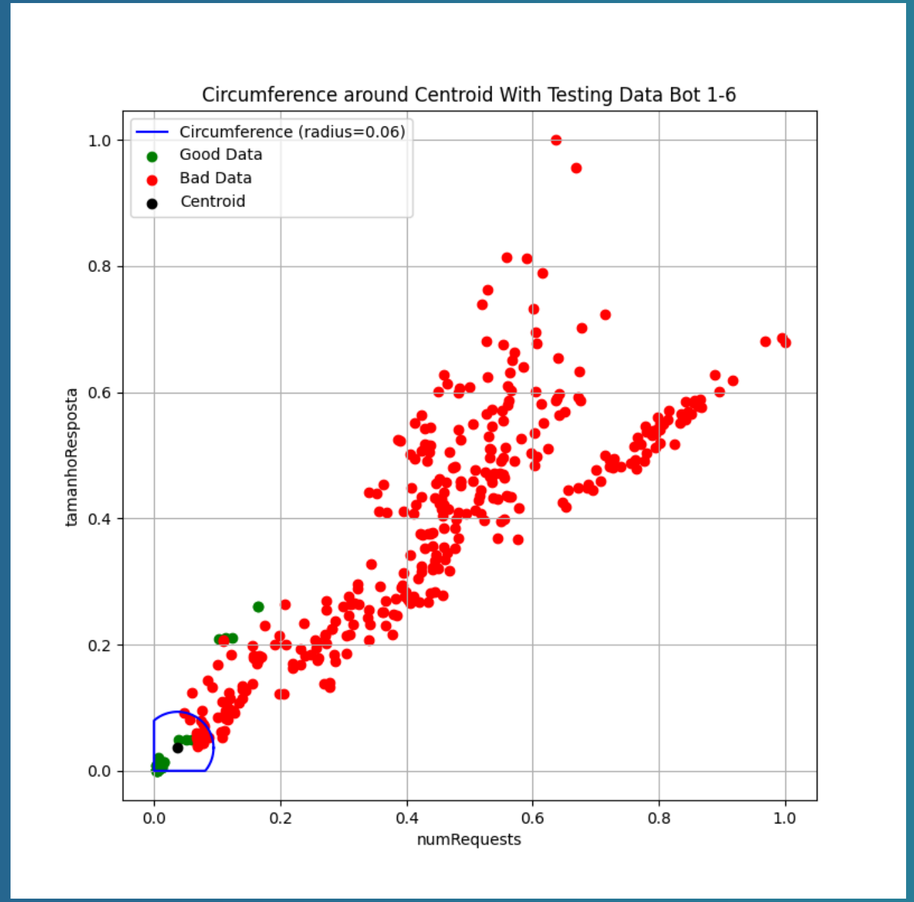
ANALYSIS OF THE RESULTS

Features: numRequests, tamanhoResposta



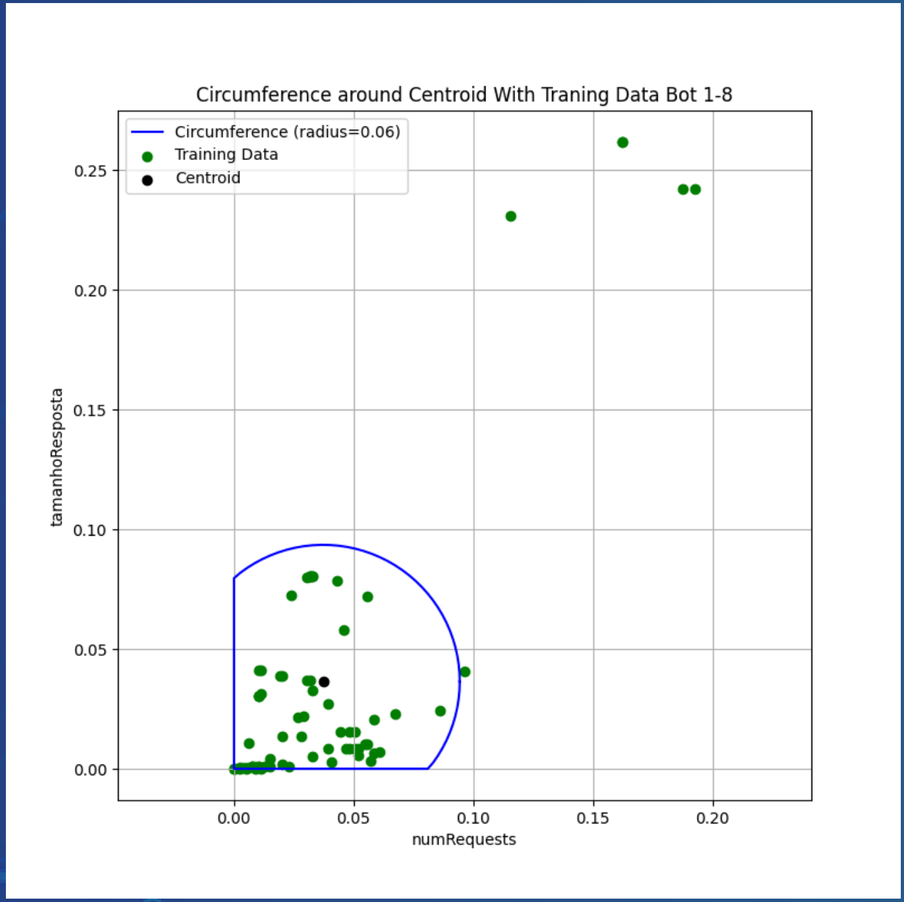
On bot lvl 1 to 7

Number of bot data inside the radius: 15



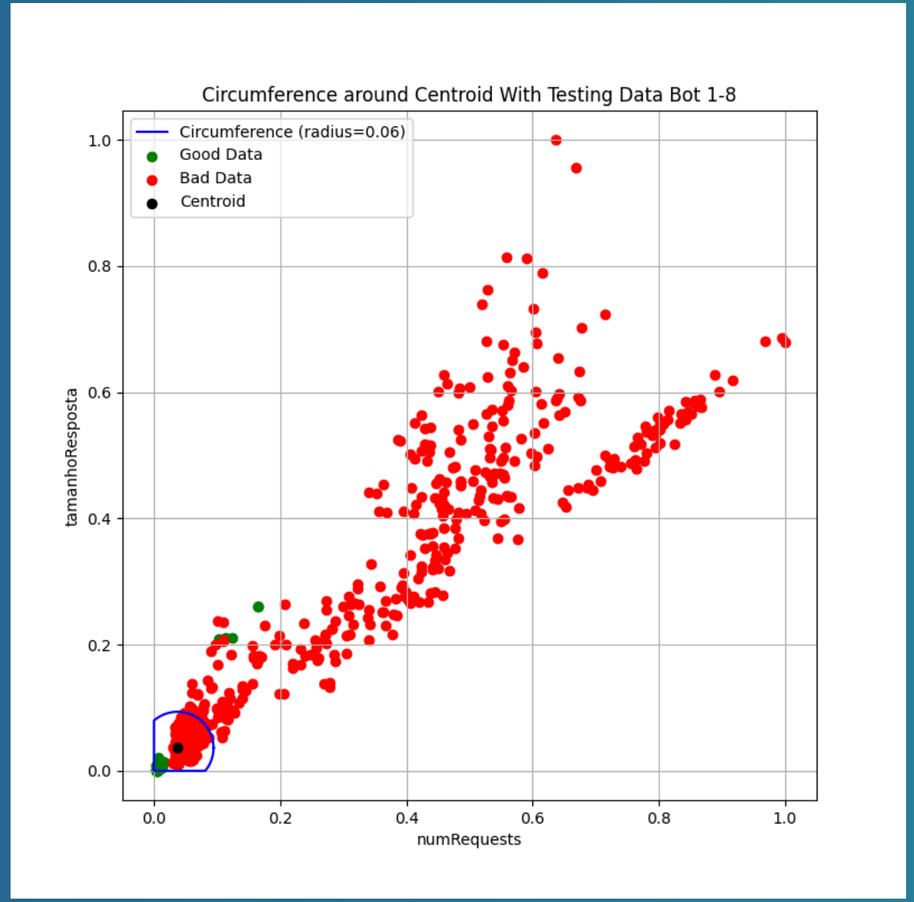
ANALYSIS OF THE RESULTS

Features: numRequests, tamanhoResposta



On bot lvl 1 to 8

Number of bot data inside the radius: 102



MACHINE LEARNING

- **Binary SVM**:

  1- When we have two distinct classes

  2- When we want to classify data into exactly two categories

70% of the data is used for training

30% of the data is used for testing

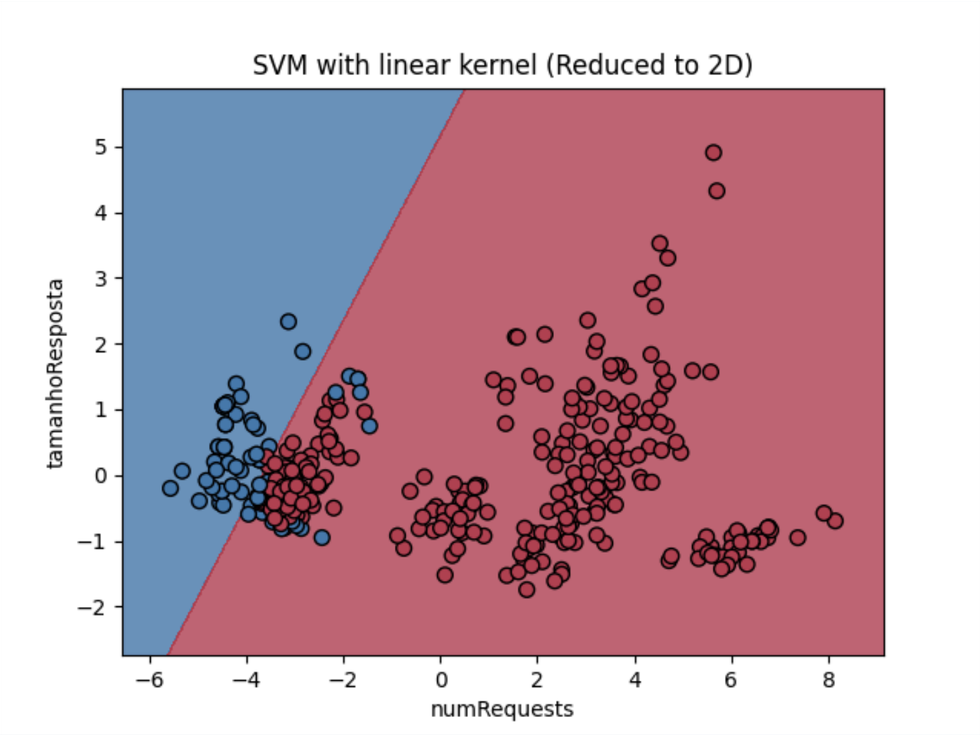
- **Random Forest:**

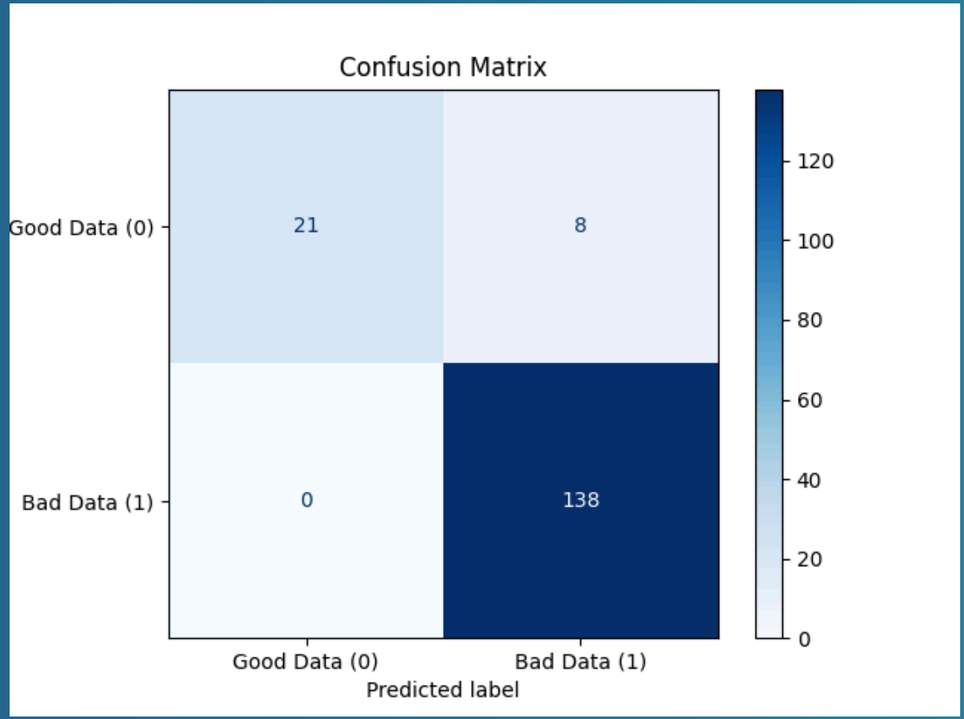
  1- Can handle imbalanced datasets well

  2- Adjust the class weights to prioritize the minority class

BINARY SVM

Using Linear Kernel



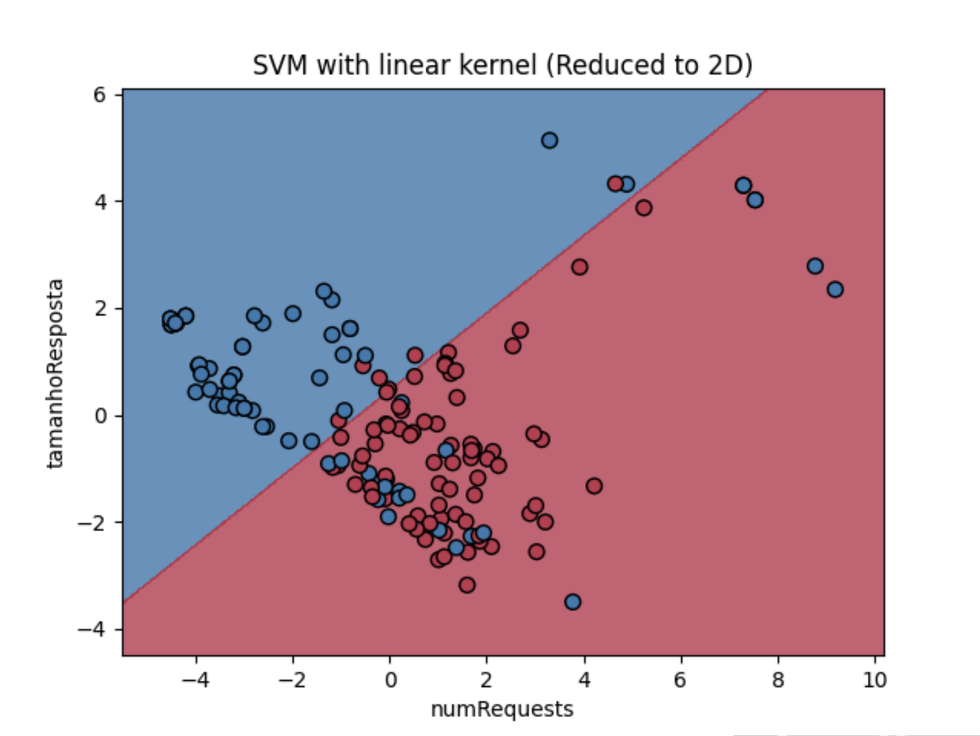


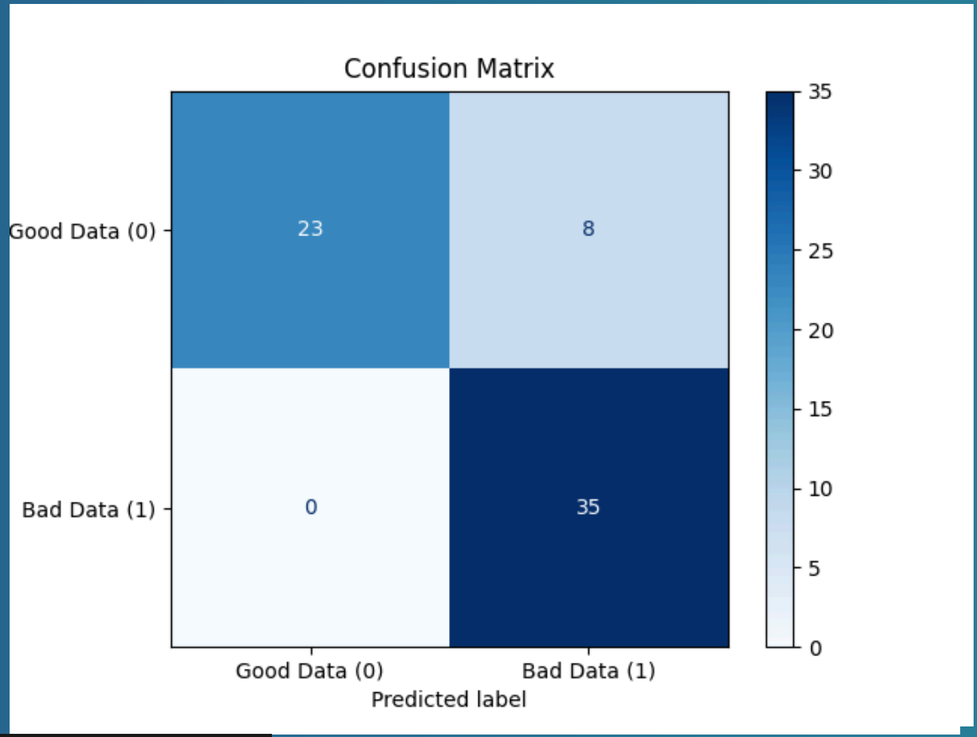
Overall Accuracy: 0.95

	precision	recall	f1-score	support
0	1.00	0.72	0.84	29
1	0.95	1.00	0.97	138
accuracy			0.95	167
macro avg	0.97	0.86	0.91	167
weighted avg	0.95	0.95	0.95	167

BINARY SVM

Using Linear Kernel with only lvl 7 and lvl 8 bot



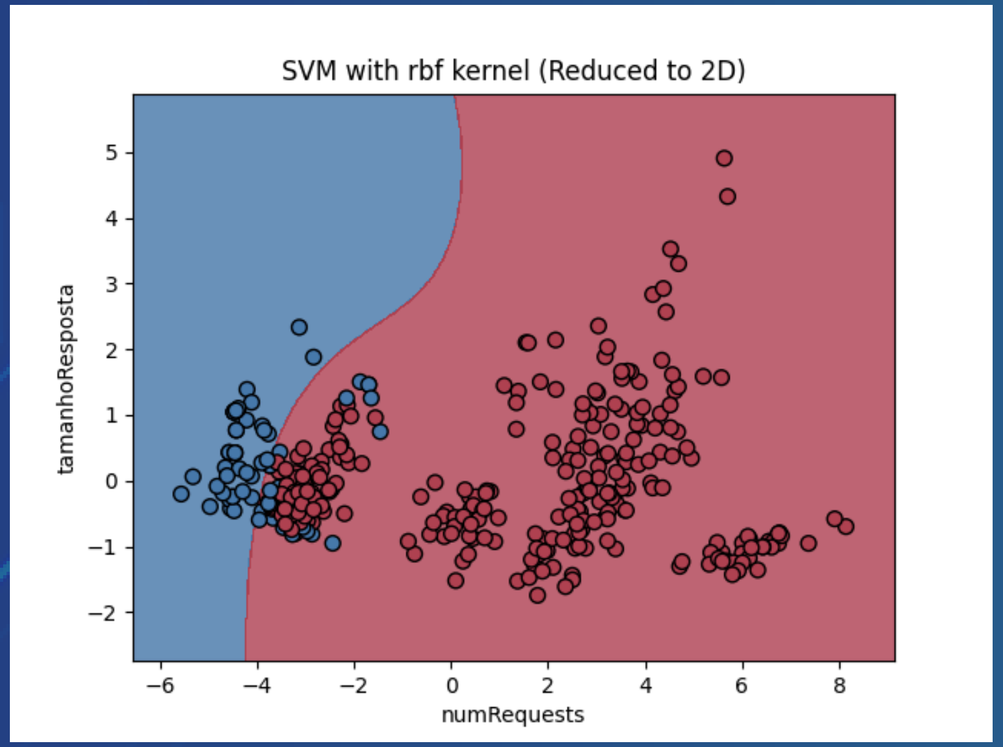


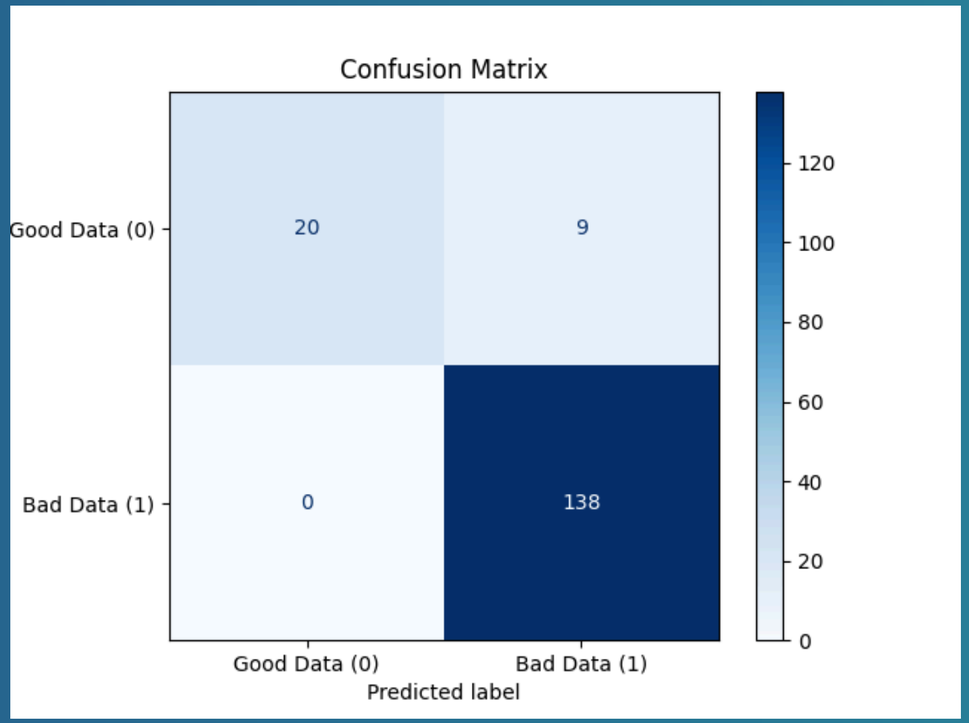
Overall Accuracy: 0.88

	precision	recall	f1-score	support
0	1.00	0.74	0.85	31
1	0.81	1.00	0.90	35
accuracy			0.88	66
macro avg	0.91	0.87	0.87	66
weighted avg	0.90	0.88	0.88	66

BINARY SVM

Using rbf Kernel



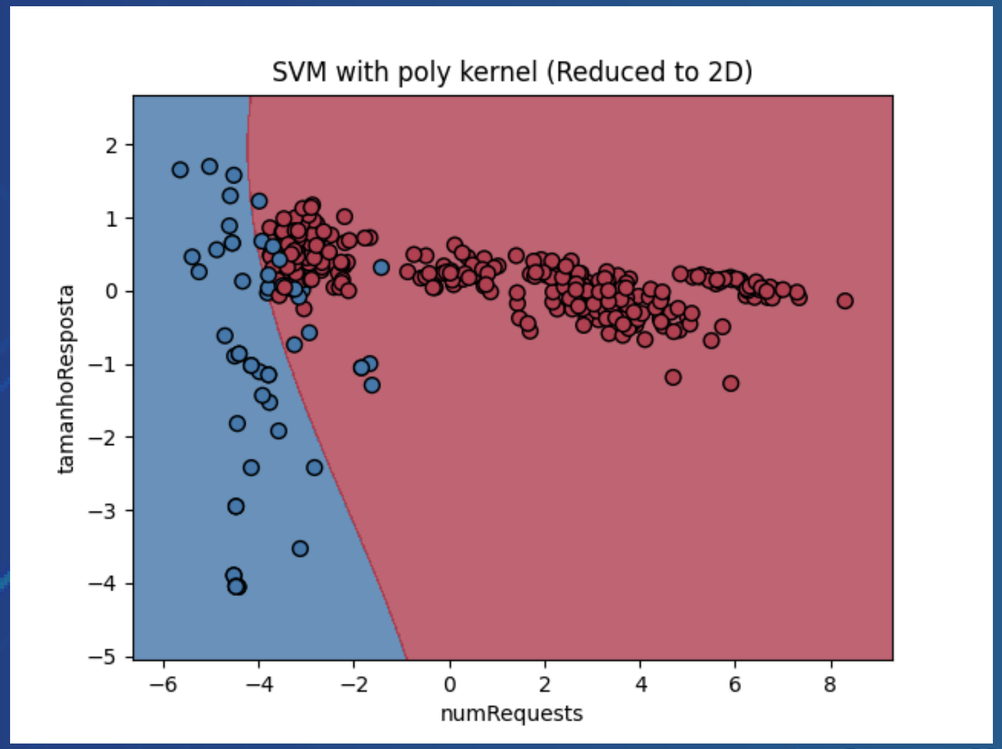


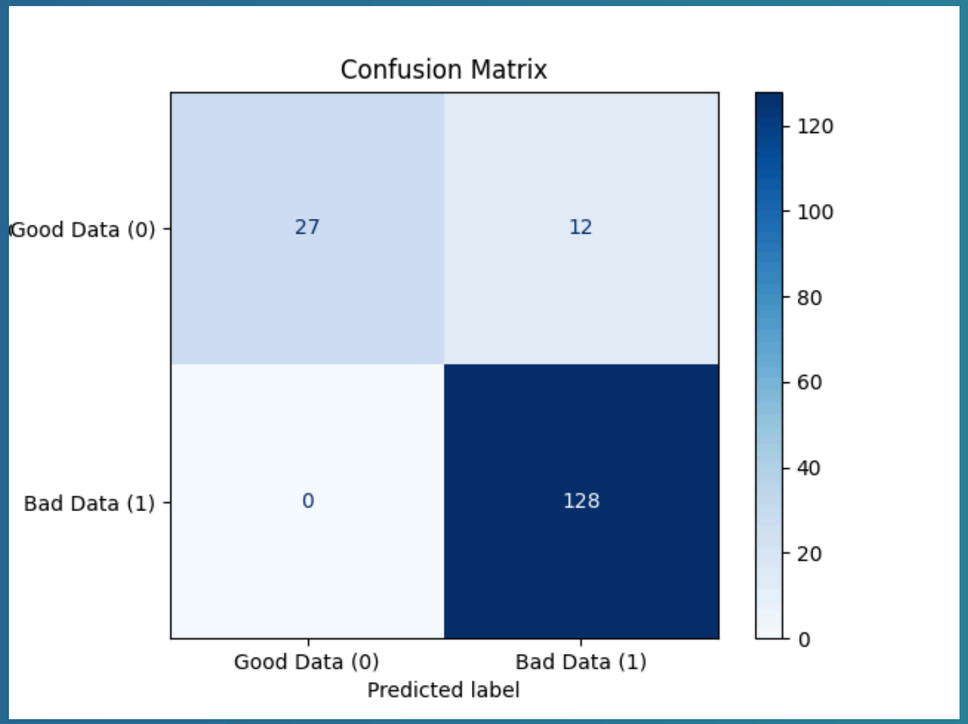
Overall Accuracy: 0.95

	precision	recall	f1-score	support
0	1.00	0.69	0.82	29
1	0.94	1.00	0.97	138
accuracy			0.95	167
macro avg	0.97	0.84	0.89	167
weighted avg	0.95	0.95	0.94	167

BINARY SVM

Using poly Kernel (with degree=3)



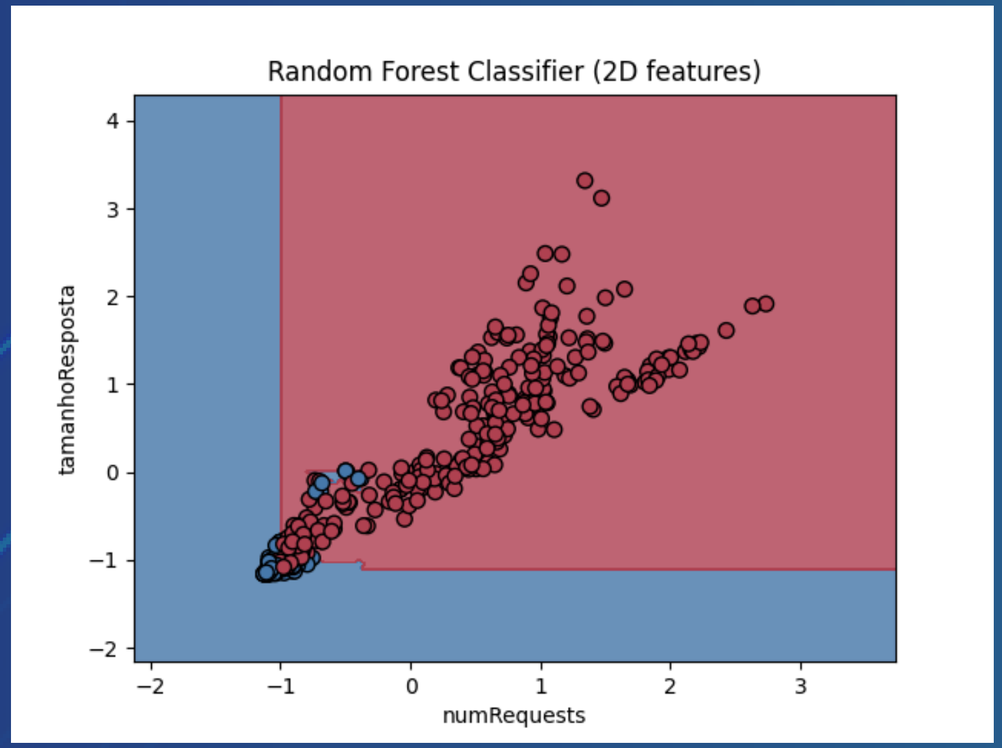


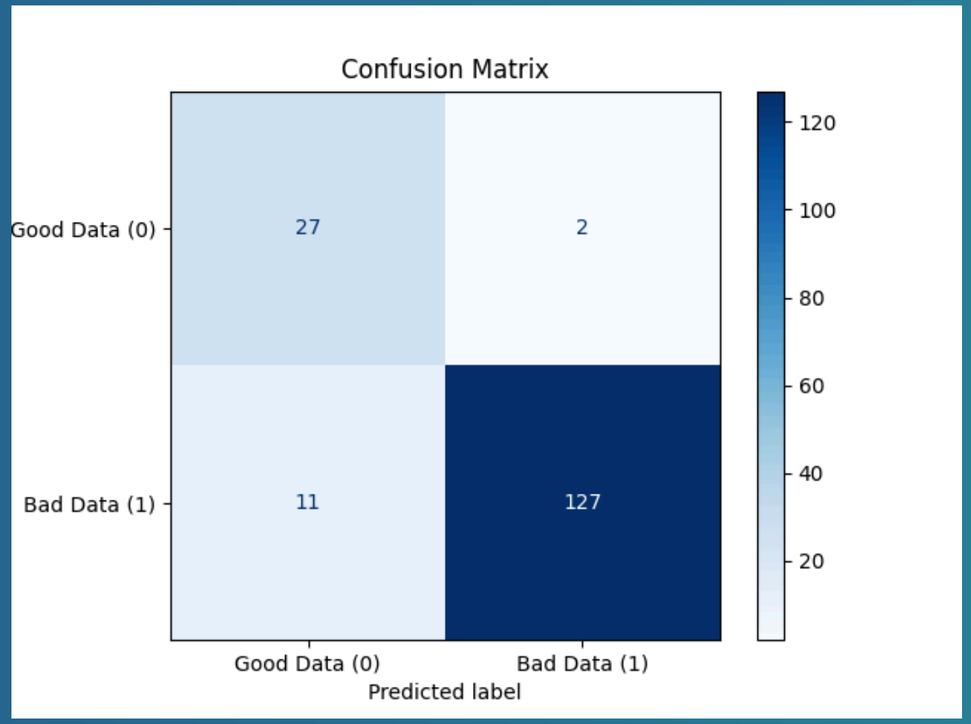
Overall Accuracy: 0.93

	precision	recall	f1-score	support
0	1.00	0.69	0.82	39
1	0.91	1.00	0.96	128
accuracy			0.93	167
macro avg	0.96	0.85	0.89	167
weighted avg	0.93	0.93	0.92	167

RANDOM FOREST

class_weight='balanced with all the bots'

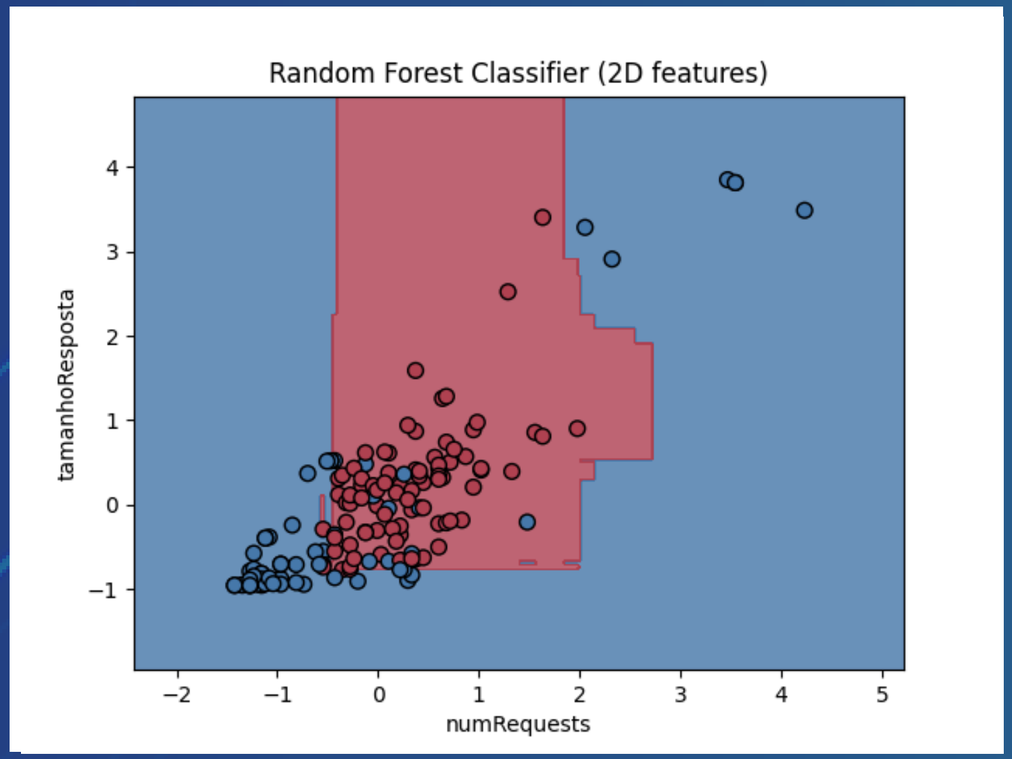


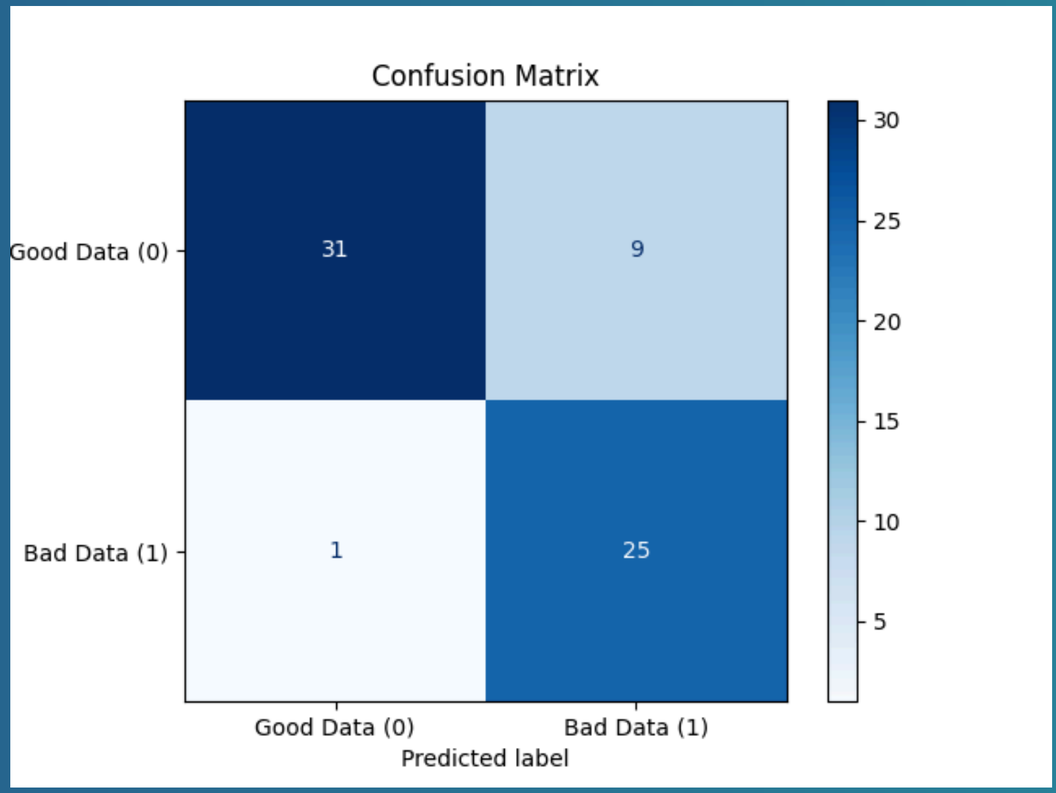


	precision	recall	f1-score	support
Overall Accuracy: 0.92				
0	0.71	0.93	0.81	29
1	0.98	0.92	0.95	138
accuracy			0.92	167
macro avg	0.85	0.93	0.88	167
weighted avg	0.94	0.92	0.93	167

RANDOM FOREST

class_weight='balanced' with only bot lvl 7 and 8





Overall Accuracy: 0.85

	precision	recall	f1-score	support
0	0.97	0.78	0.86	40
1	0.74	0.96	0.83	26
accuracy			0.85	66
macro avg	0.85	0.87	0.85	66
weighted avg	0.88	0.85	0.85	66

CONCLUSIONS

- Detection of bots up to level 6: Based on graphs showing feature evolution over time, we were able to detect bots up to level 6.

- Detection of bots at levels 7 and 8: For detecting bots at level 7 and 8, we switched to Z-Score and the Distances to Central Point(s) method.

- Final confirmation using machine learning: To ensure accuracy, we applied machine learning techniques to detect all bots with 95% precision in our best model.

THANK YOU!