black hat®
ARSENAL 2023

Webhawk/Catch

Open-Source AI based web attack traces detection

Walid DABOUBI

```
$ whoami
  Walid DABOUBI, Engineer
$ pwd
  Heading the Group Security Data Analytics at Richemont Geneva/Switzerland
$ history
  R&D Cloud Software Engineer at Dassault Systemes Paris/France
  Software engineering intern at Apple Cupertino/CA and at Siemens Princeton/NJ
  Speaker at:
    ISF Dublin (Information Security Forum) 2019 – ML applications in Cyber
    AMLD Lausanne (Applied Machine Learning Days) 2022 – Tornado HITL open-source
  CTF player:
    2 SANS HolidayHack – Super Honorable Mentions
```

Agenda

1 About Webhawk/Catch

2 The motivation behind

3 A brief history

4 How does it work

5 Webhawk/Catch in action (Demo)

6 Integration within a SOC ecosystem

7 What’s next ?

About

Webhawk/Catch helps **quickly** finding web attack traces in logs.

No **signature-based/pre-set** detection rules are required

Uses **unsupervised machine learning** to group log lines into clusters

Detects the outlier log lines or the ones that belong to minority clusters

Can be further **fine-tuned** according to the user level/experience

Generates an **easy-to-read** detection reports

Motivation

A lot of AI/ML based detection tools already exist; however, they present some limitations:

1. They mostly focus of network traffic data (Application logs are generally not covered)

The first lines of code of Webhawk were developed on a train

Motivation

A lot of AI/ML based detection tools already exist; however, they present some limitations:

1. They mostly focus of network traffic data (Application logs are generally not covered)
2. Most of the time these tools are black boxes, and we don't really know how they make detection

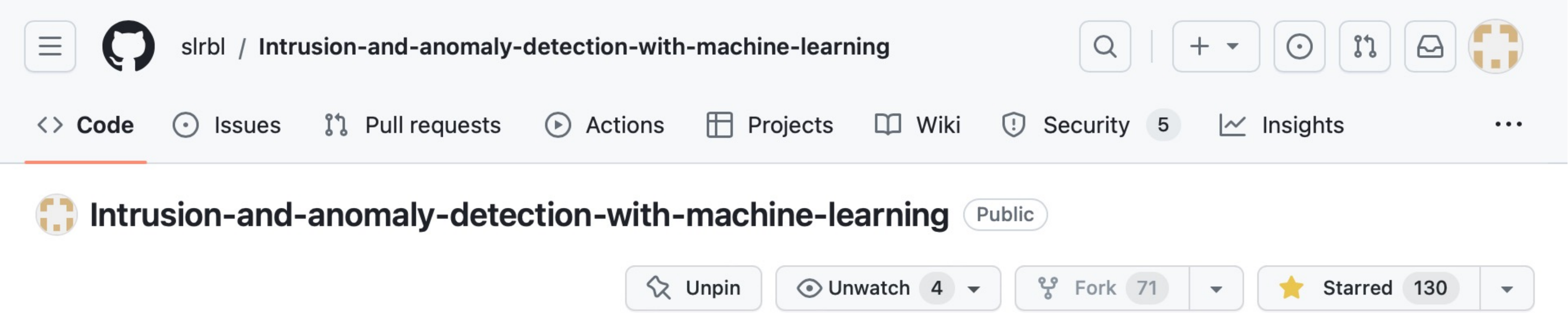
The first lines of code of Webhawk were developed on a train

A brief history

Webhawk was started as a supervised learning detection tool in **2017**

Needs labelled data to train a detection model, which is not always easy to find

`https://github.com/slrbl/Intrusion-and-anomaly-detection-with-machine-learning`



slrbl / Intrusion-and-anomaly-detection-with-machine-learning
Code
Issues
Pull requests
Actions
Projects
Wiki
Security 5
Insights
Intrusion-and-anomaly-detection-with-machine-learning
Public
Unpin
Unwatch 4
Fork 71
Starred 130

The unsupervised version (**Webhawk/Catch**) has been added as a separate project in 2022

How does it work?

Input

Raw HTTP log file

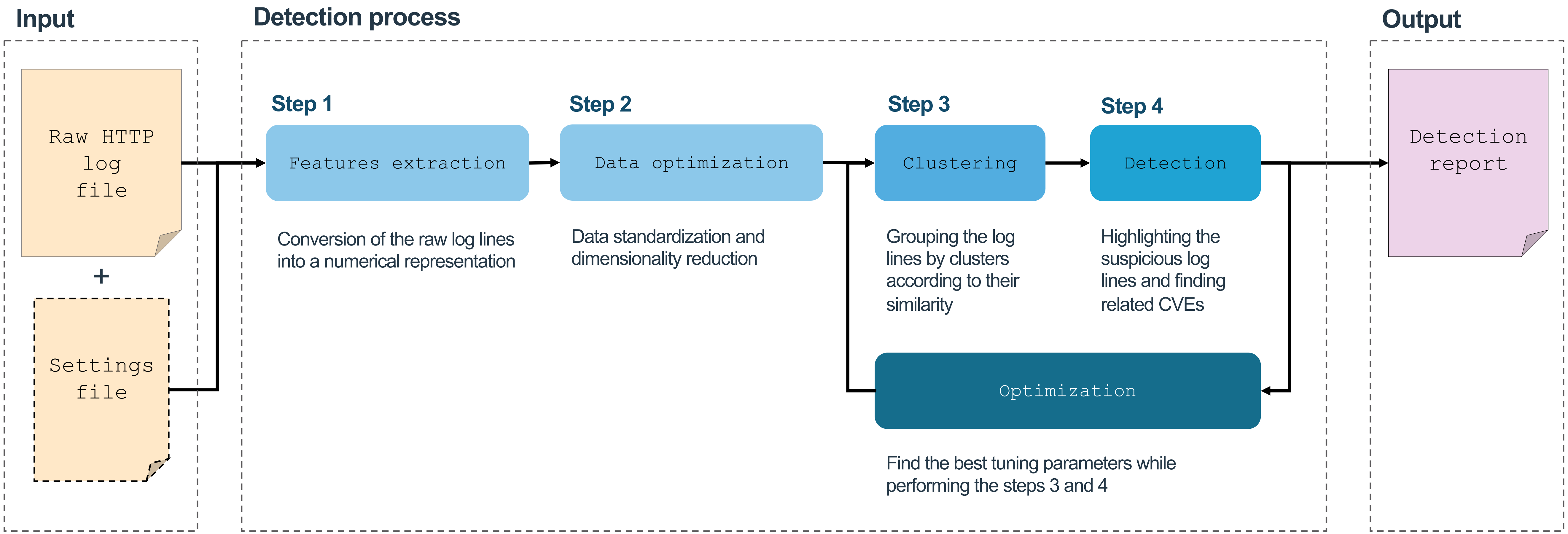
+

Settings file

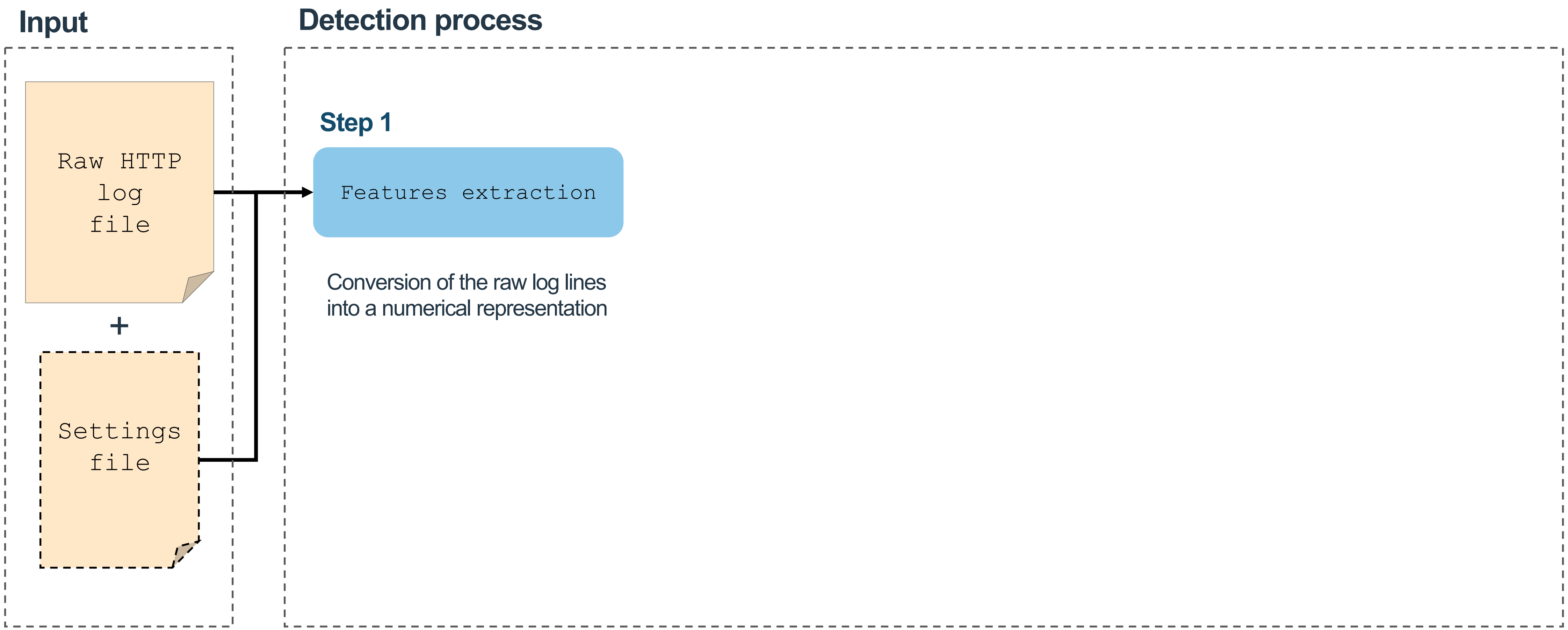
Output

Detection report

How does it work?


Input
Raw HTTP log file
+
Settings file
Detection process
Step 1
Features extraction
Conversion of the raw log lines into a numerical representation
Step 2
Data optimization
Data standardization and dimensionality reduction
Step 3
Clustering
Grouping the log lines by clusters according to their similarity
Step 4
Detection
Highlighting the suspicious log lines and finding related CVEs
Optimization
Find the best tuning parameters while performing the steps 3 and 4
Output
Detection report

Step 1 Features extraction



Conversion of the raw log lines into a numerical representation

Step 1 Features extraction

Numerical Features	Categorical Features
Number of parameters	User agent
URL length	HTTP query
Number of upper-case characters	IP Address
Number of lower-case characters	Return Code
URL depth	

The numerical values are taken as they are

Label encoding technique is used to convert into numerical value

Step 1 Features extraction

Numerical Features	Categorical Features
Number of parameters	User agent
URL length	HTTP query
Number of upper-case characters	IP Address
Number of lower-case characters	Return Code
URL depth	

HTTP query example

Raw Value	Label encoded value
GET	0
POST	1
PUT	2
DELETE	3
Etc..	4

The numerical values are taken as they are

Two option ca be used:

- Label encoding
- Fraction/Frequency encoding **(by default option)**

Step 1 Features extraction

Numerical Features	Categorical Features
Number of parameters	User agent
URL length	HTTP query
Number of upper-case characters	IP Address
Number of lower-case characters	Return Code
URL depth	

HTTP query example (m=1340)

Raw Value	Freq	Encoded value
GET	987	0.73
POST	311	0.23
PUT	34	0.02
DELETE	8	0.005

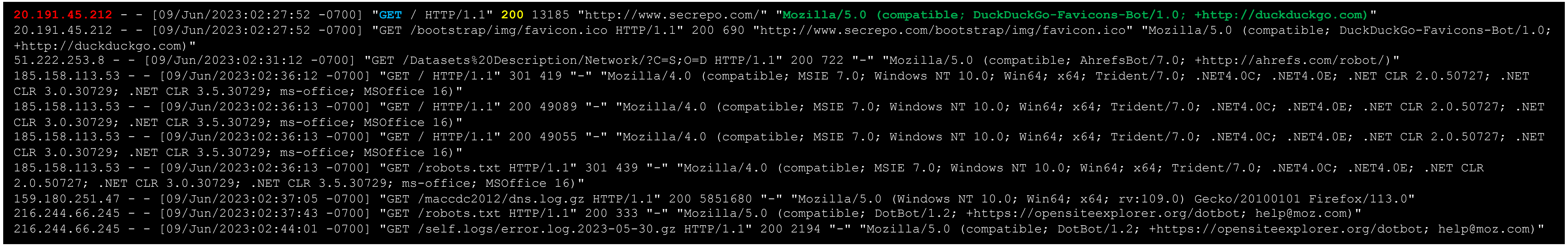
The numerical values are taken as they are

Two option ca be used:

- Label encoding
- Fraction/Frequency encoding **(by default option)**

Step 1 Features extraction

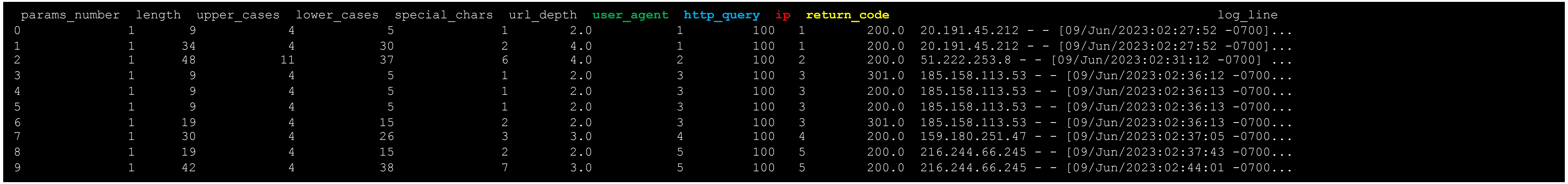
Raw HTTP logs



```
20.191.45.212 - - [09/Jun/2023:02:27:52 -0700] "GET / HTTP/1.1" 200 13185 "http://www.secrepo.com/" "Mozilla/5.0 (compatible; DuckDuckGo-Favicons-Bot/1.0; +http://duckduckgo.com)"
20.191.45.212 - - [09/Jun/2023:02:27:52 -0700] "GET /bootstrap/img/favicon.ico HTTP/1.1" 200 690 "http://www.secrepo.com/bootstrap/img/favicon.ico" "Mozilla/5.0 (compatible; DuckDuckGo-Favicons-Bot/1.0;
+http://duckduckgo.com)"
51.222.253.8 - - [09/Jun/2023:02:31:12 -0700] "GET /Datasets%20Description/Network/?C=S;O=D HTTP/1.1" 200 722 "-" "Mozilla/5.0 (compatible; AhrefsBot/7.0; +http://ahrefs.com/robot/)"
185.158.113.53 - - [09/Jun/2023:02:36:12 -0700] "GET / HTTP/1.1" 301 419 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET
CLR 3.0.30729; .NET CLR 3.5.30729; ms-office; MSOffice 16)"
185.158.113.53 - - [09/Jun/2023:02:36:13 -0700] "GET / HTTP/1.1" 200 49089 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET
CLR 3.0.30729; .NET CLR 3.5.30729; ms-office; MSOffice 16)"
185.158.113.53 - - [09/Jun/2023:02:36:13 -0700] "GET / HTTP/1.1" 200 49055 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET
CLR 3.0.30729; .NET CLR 3.5.30729; ms-office; MSOffice 16)"
185.158.113.53 - - [09/Jun/2023:02:36:13 -0700] "GET /robots.txt HTTP/1.1" 301 439 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR
2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; ms-office; MSOffice 16)"
159.180.251.47 - - [09/Jun/2023:02:37:05 -0700] "GET /maccdc2012/dns.log.gz HTTP/1.1" 200 5851680 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0"
216.244.66.245 - - [09/Jun/2023:02:37:43 -0700] "GET /robots.txt HTTP/1.1" 200 333 "-" "Mozilla/5.0 (compatible; DotBot/1.2; +https://opensiteexplorer.org/dotbot; help@moz.com)"
216.244.66.245 - - [09/Jun/2023:02:44:01 -0700] "GET /self.logs/error.log.2023-05-30.gz HTTP/1.1" 200 2194 "-" "Mozilla/5.0 (compatible; DotBot/1.2; +https://opensiteexplorer.org/dotbot; help@moz.com)"
```

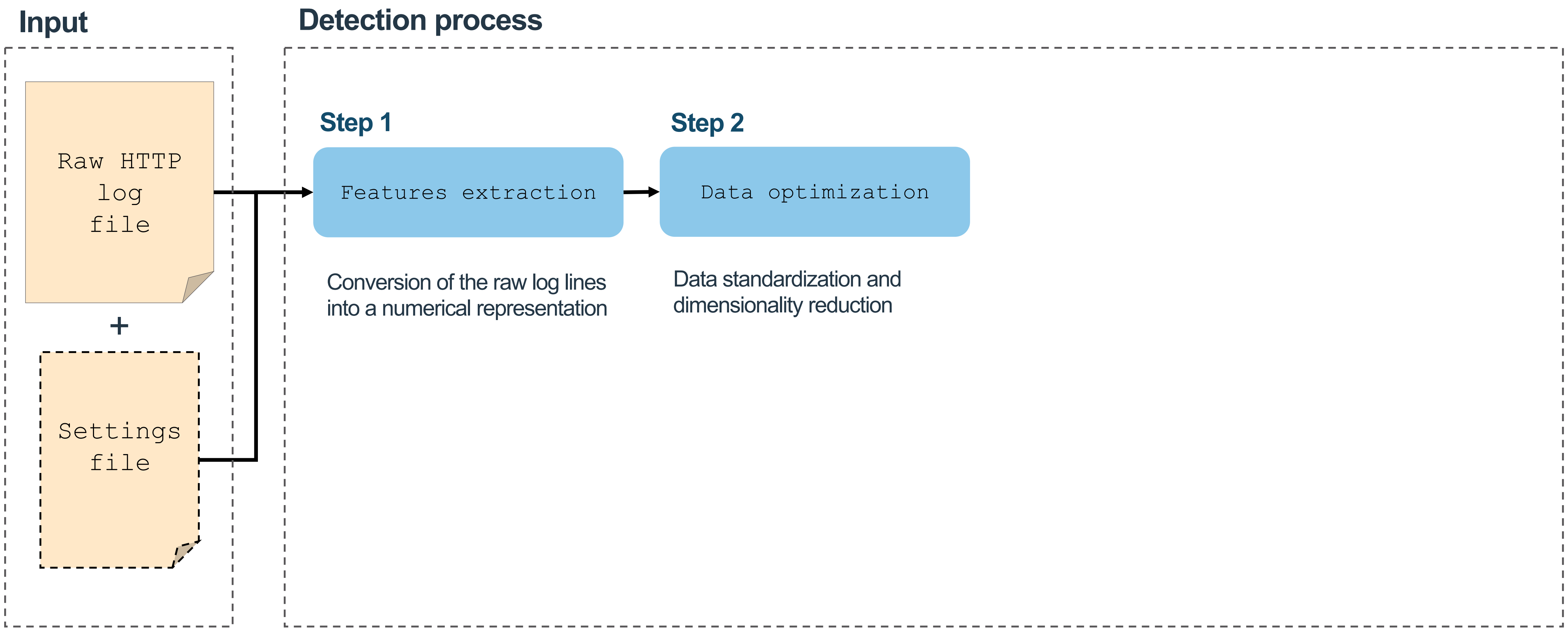
Convert raw HTTP logs to a numerical representation

Encoded HTTP logs



```
  params_number  length  upper_cases  lower_cases  special_chars  url_depth  user_agent  http_query  ip  return_code                        log_line
0              1       9            4            5              1        2.0           1         100   1        200.0  20.191.45.212 - - [09/Jun/2023:02:27:52 -0700]...
1              1      34            4           30              2        4.0           1         100   1        200.0  20.191.45.212 - - [09/Jun/2023:02:27:52 -0700]...
2              1      48           11           37              6        4.0           2         100   2        200.0  51.222.253.8 - - [09/Jun/2023:02:31:12 -0700] ...
3              1       9            4            5              1        2.0           3         100   3        301.0  185.158.113.53 - - [09/Jun/2023:02:36:12 -0700...
4              1       9            4            5              1        2.0           3         100   3        200.0  185.158.113.53 - - [09/Jun/2023:02:36:13 -0700...
5              1       9            4            5              1        2.0           3         100   3        200.0  185.158.113.53 - - [09/Jun/2023:02:36:13 -0700...
6              1      19            4           15              2        2.0           3         100   3        301.0  185.158.113.53 - - [09/Jun/2023:02:36:13 -0700...
7              1      30            4           26              3        3.0           4         100   4        200.0  159.180.251.47 - - [09/Jun/2023:02:37:05 -0700...
8              1      19            4           15              2        2.0           5         100   5        200.0  216.244.66.245 - - [09/Jun/2023:02:37:43 -0700...
9              1      42            4           38              7        3.0           5         100   5        200.0  216.244.66.245 - - [09/Jun/2023:02:44:01 -0700...
```

Step 2 Data optimization



Conversion of the raw log lines into a numerical representation

Data standardization and dimensionality reduction

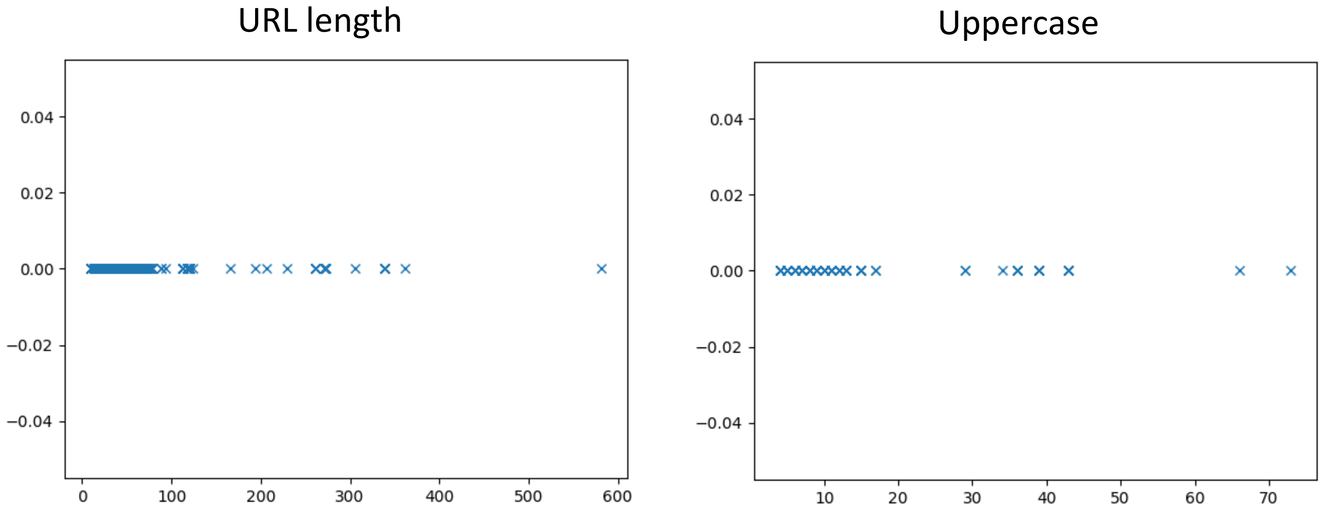
Step 2 Data optimization

Standardization

Why do we need data standardization

- Our features have different scales
- Clustering is based on measuring the Euclidean distance between points
- Distances can become meaningless when working on highly different scales – Risk of biased clustering

More important ? No.



How?

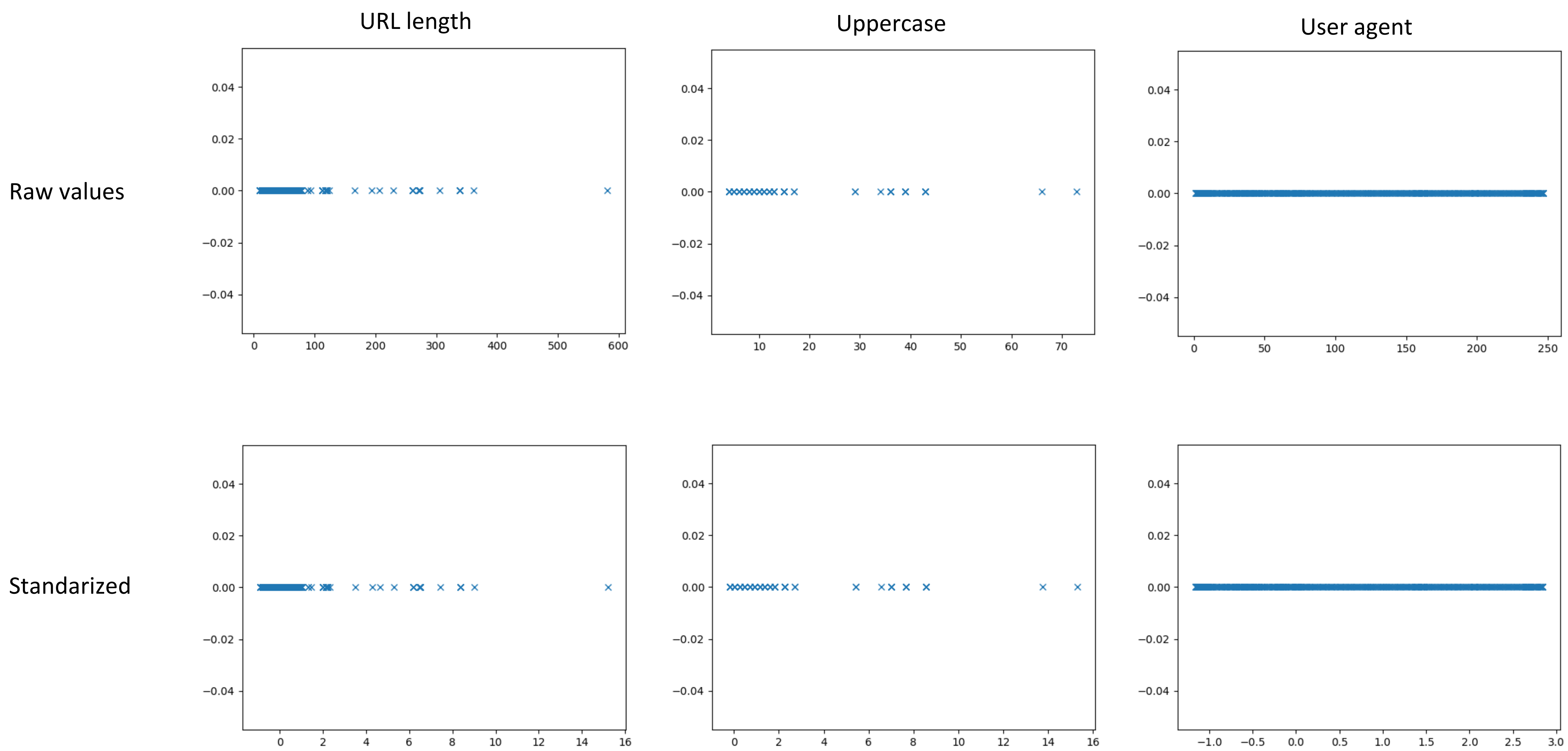
We transform each value of the different features using the following:

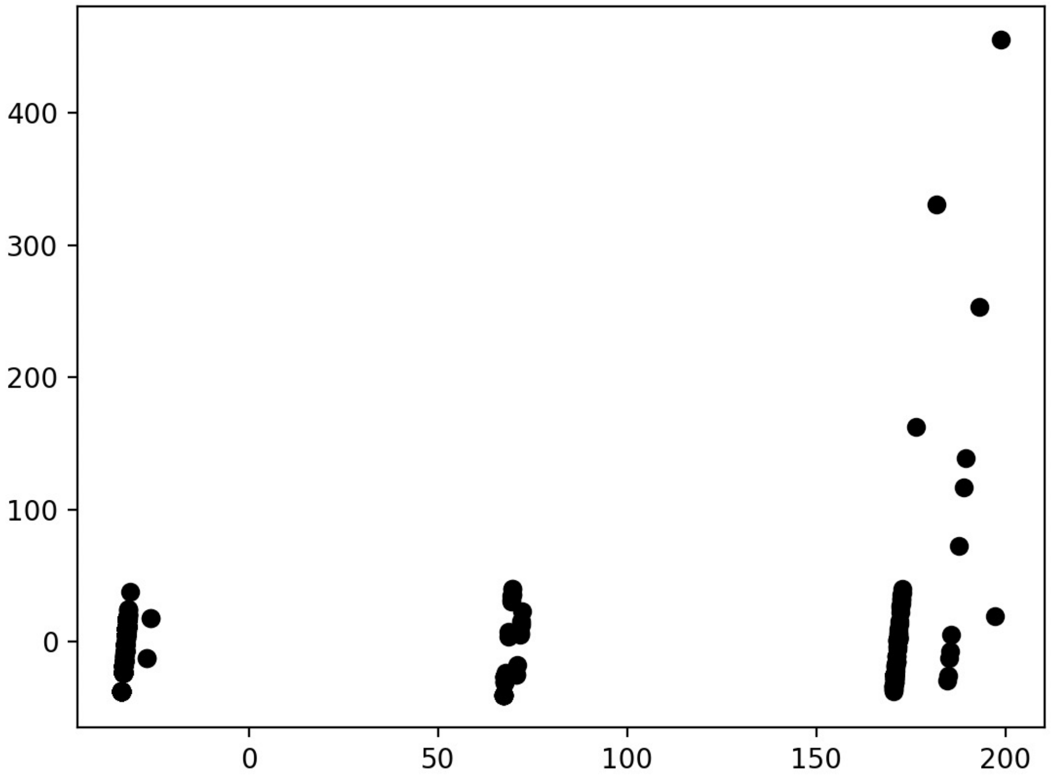
$$\boldsymbol{z} = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

with

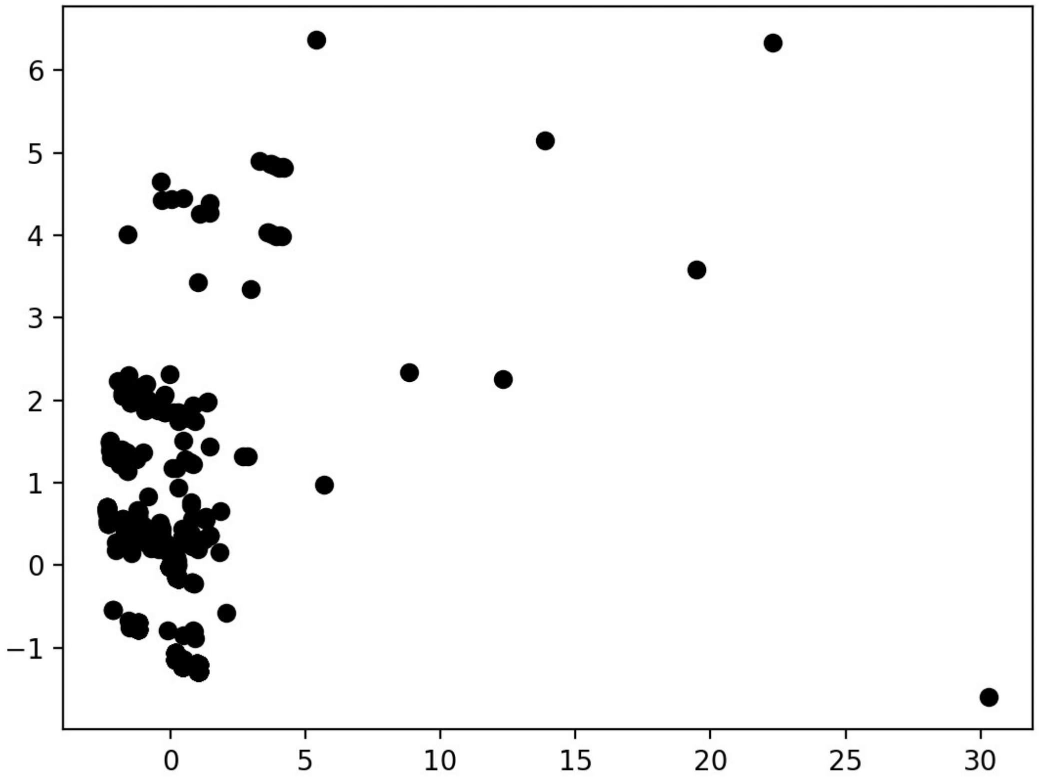
$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$ Mean

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$ Std deviation


URL length
Uppercase
User agent
Raw values
Standarized
0.04
0.02
0.00
−0.02
−0.04
0
100
200
300
400
500
600
10
20
30
40
50
60
70
0
50
100
150
200
250
0
2
4
6
8
10
12
14
16
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0



Without Data Standardization



With Data Standardization

Step 2 Data optimization

Dimensiality reduction

Why do we need to reduce data dimensions ?

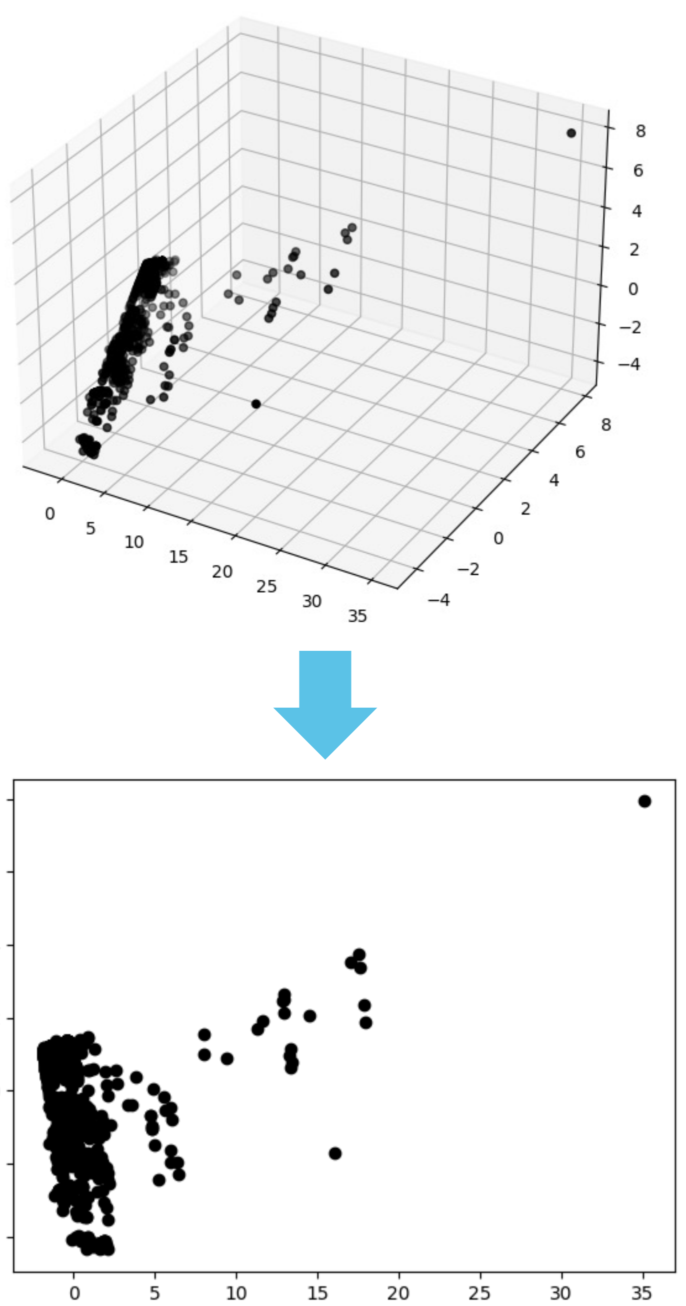
We reduce the dimensions (number of features) to 2 dimensions for tow reasons

- Clustering algorithms that are based on the measurement of distances between the data points. They performs better in lower dimension spaces
- A better visualization and explainability of the clustering process

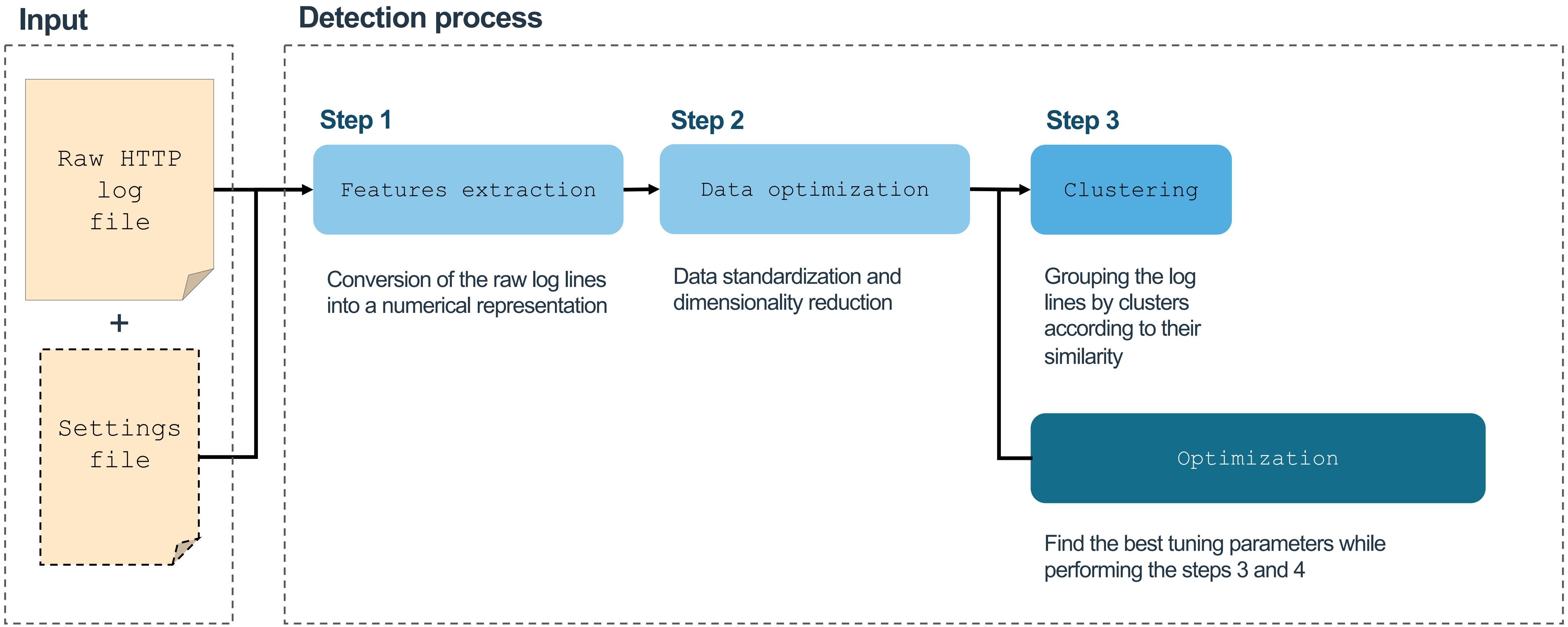
How ?

We apply PCA (Principal Component Analysis) algorithm

An informative visualisation of 3 selected features



Step 3 Clustering



Step 1 – Conversion of the raw log lines into a numerical representation

Step 2 – Data standardization and dimensionality reduction

Step 3 – Grouping the log lines by clusters according to their similarity

Optimization – Find the best tuning parameters while performing the steps 3 and 4

Step 3 Clustering

Divide the log lines into groups

- Separate the data into clusters using **DBSCAN** algorithm
- The points that don’t belong to any cluster are considered as outliers

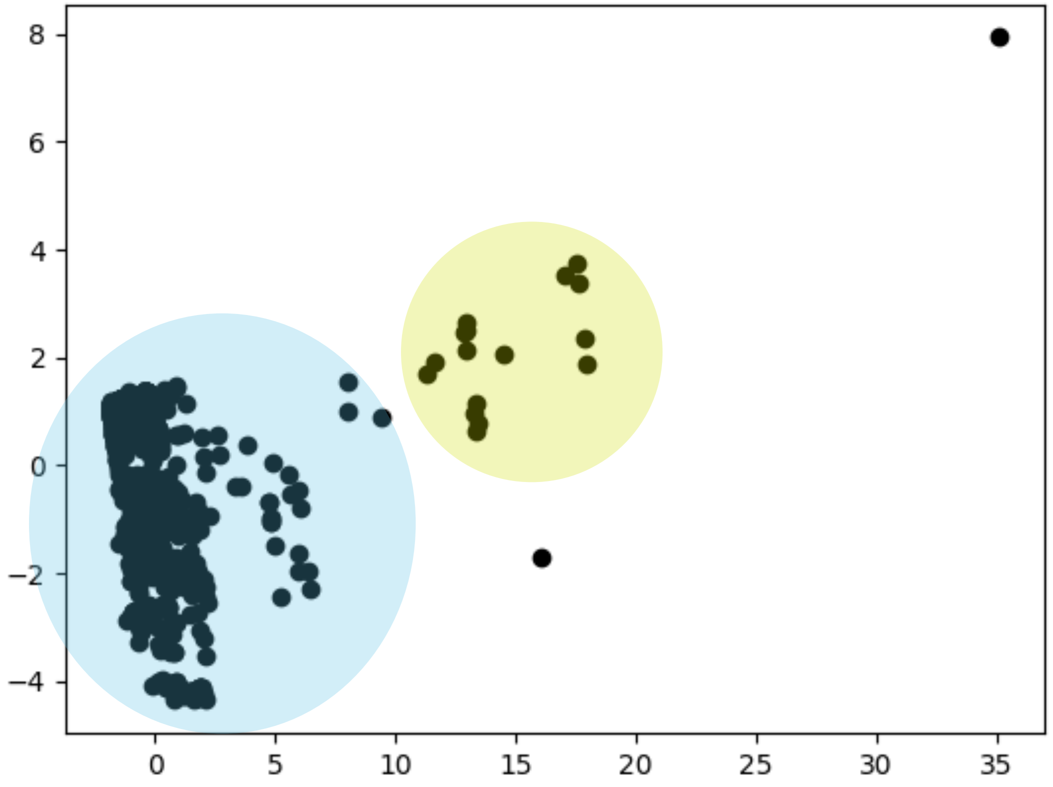
Optimization parameters

eps: Epsilon the maximum distance between tow point to be considered as belonging to the same cluster
min_samples: Minimum number of points within the same cluster

Optimization target

silouhette_score: to intra clusters distance mean (best value is 1)

We can imagine this:

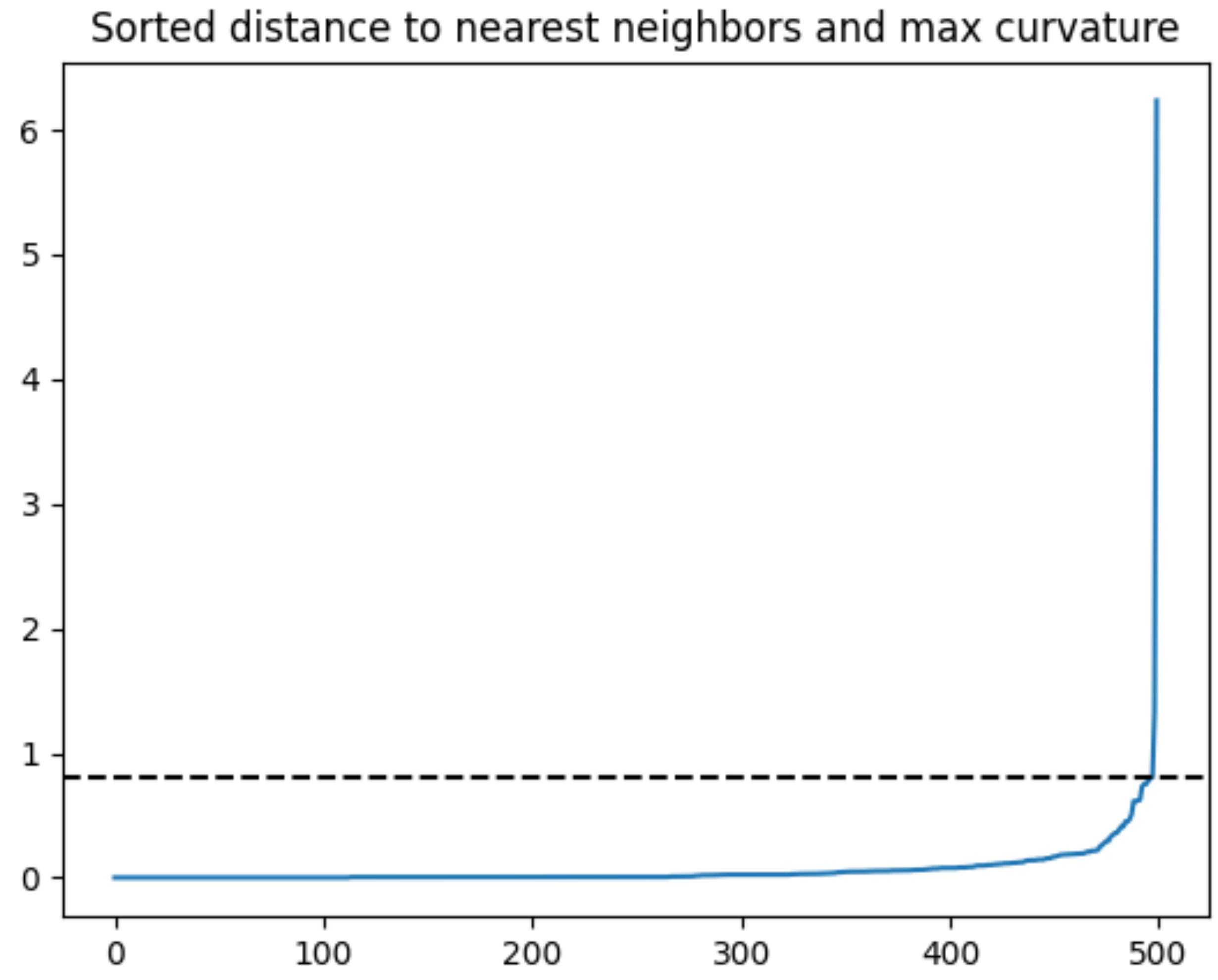


Optimization

Optimization possibilities

1. Manual - The user select a personalized Epsilon

Optimization

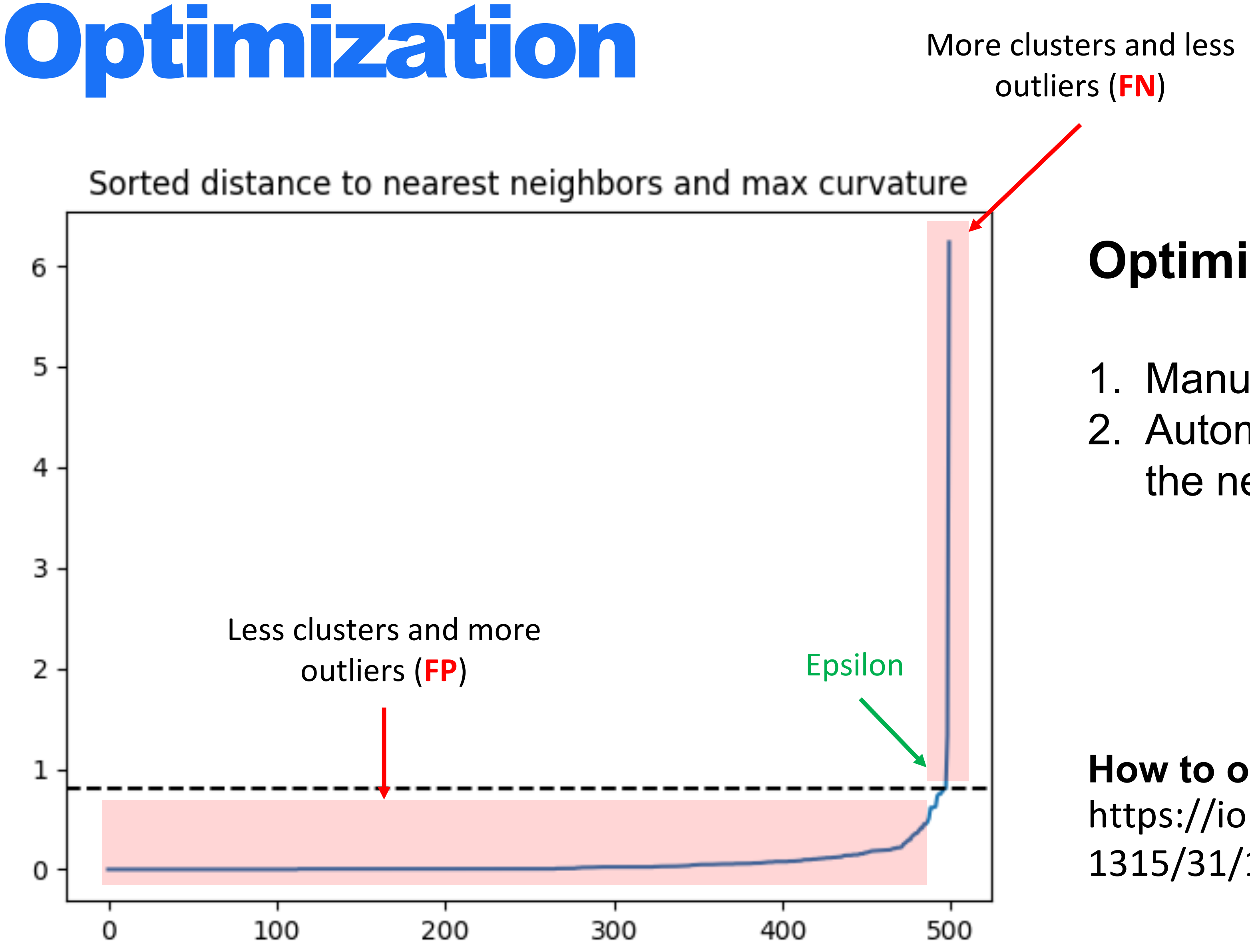


Optimization possibilities

1. Manual - The user select a personalized Epsilon
2. Automatic selection of Epsilon using the max curvature of the nearest neighbors

How to optimize Epsilon Value
https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf

Optimization

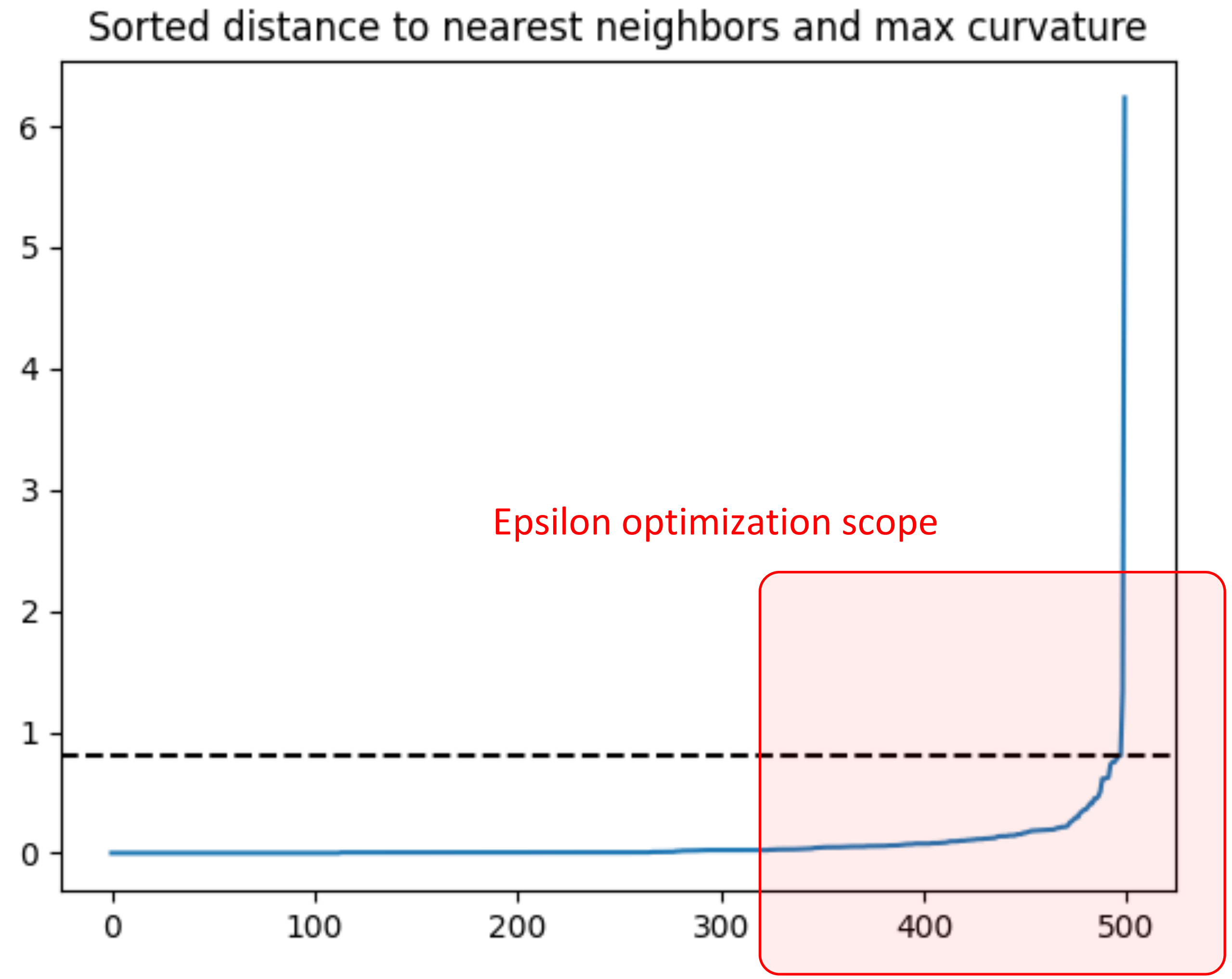


Optimization possibilities

1. Manual - The user select a personalized Epsilon
2. Automatic selection of Epsilon using the max curvature of the nearest neighbors

How to optimize DBSCAN Epsilon
https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf

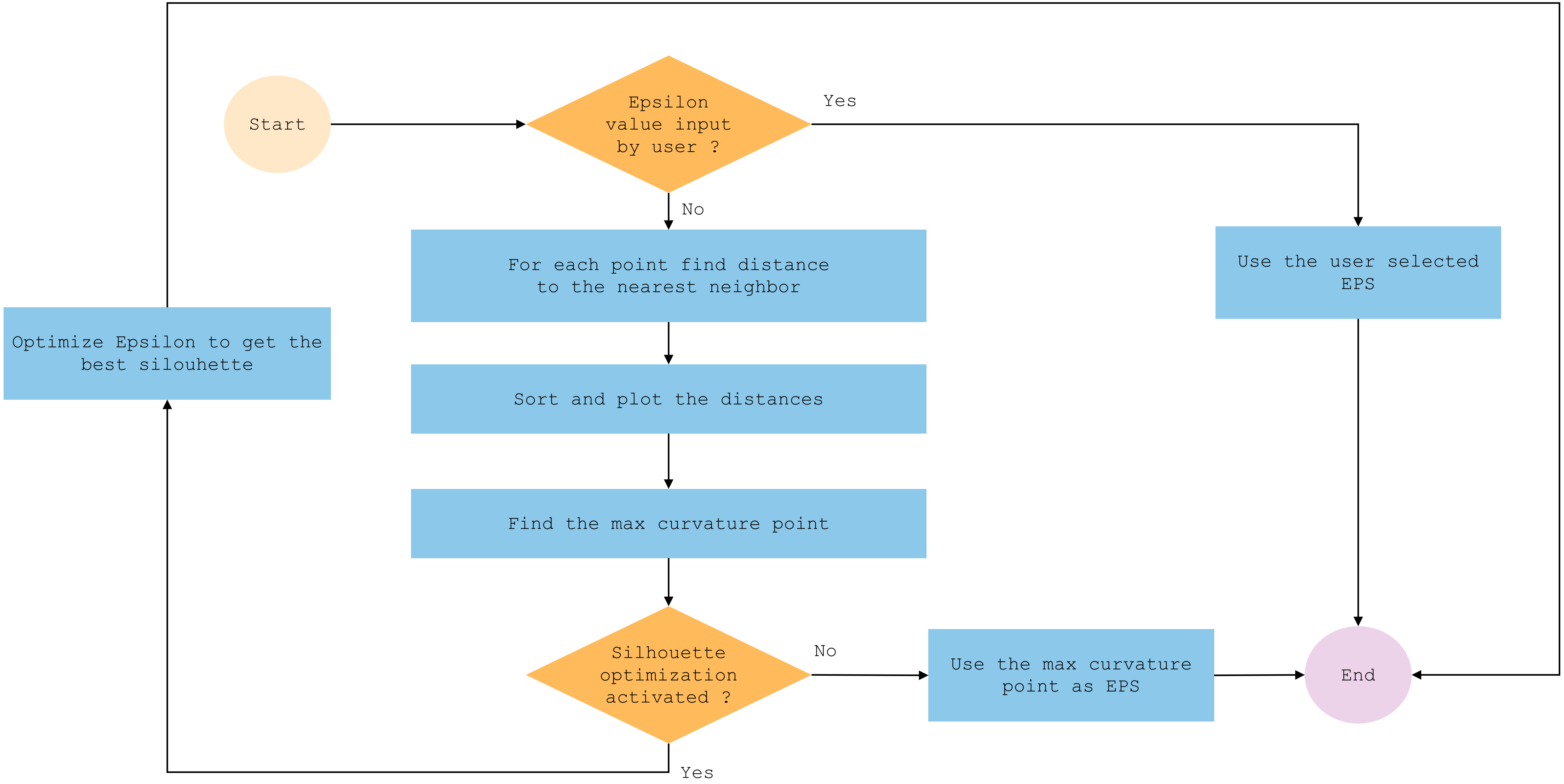
Optimization



Optimization possibilities

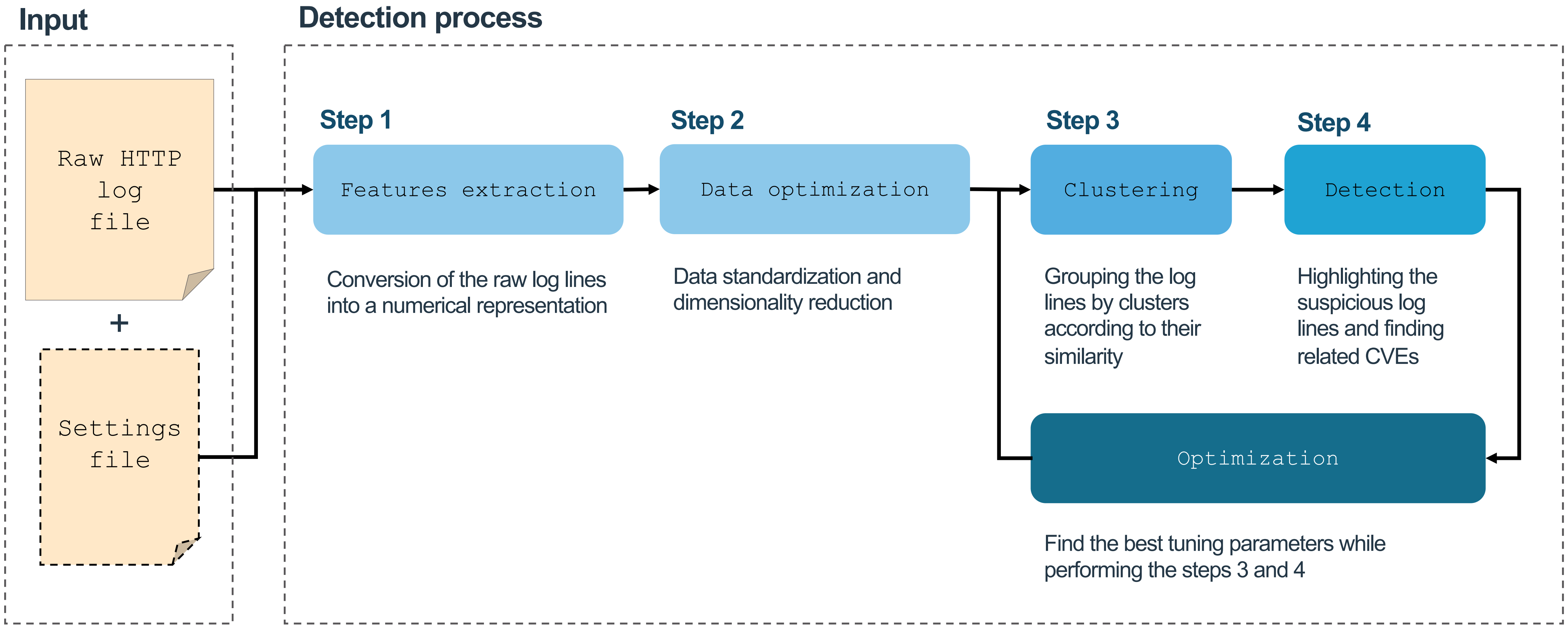
1. Manual - The user select a personalized Epsilon
2. Automatic selection of Epsilon using the max curvature of the nearest neighbors
3. Optimization of Epsilon to get the best DBSCAN Silhouette

How to optimize DBSCAN Epsilon
https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf



Start
Epsilon value input by user ?
Yes
No
For each point find distance to the nearest neighbor
Sort and plot the distances
Find the max curvature point
Silhouette optimization activated ?
No
Yes
Use the max curvature point as EPS
Use the user selected EPS
Optimize Epsilon to get the best silouhette
End

Step 5 Detection



Input

Raw HTTP log file

+

Settings file

Detection process

Step 1

Features extraction

Conversion of the raw log lines into a numerical representation

Step 2

Data optimization

Data standardization and dimensionality reduction

Step 3

Clustering

Grouping the log lines by clusters according to their similarity

Step 4

Detection

Highlighting the suspicious log lines and finding related CVEs

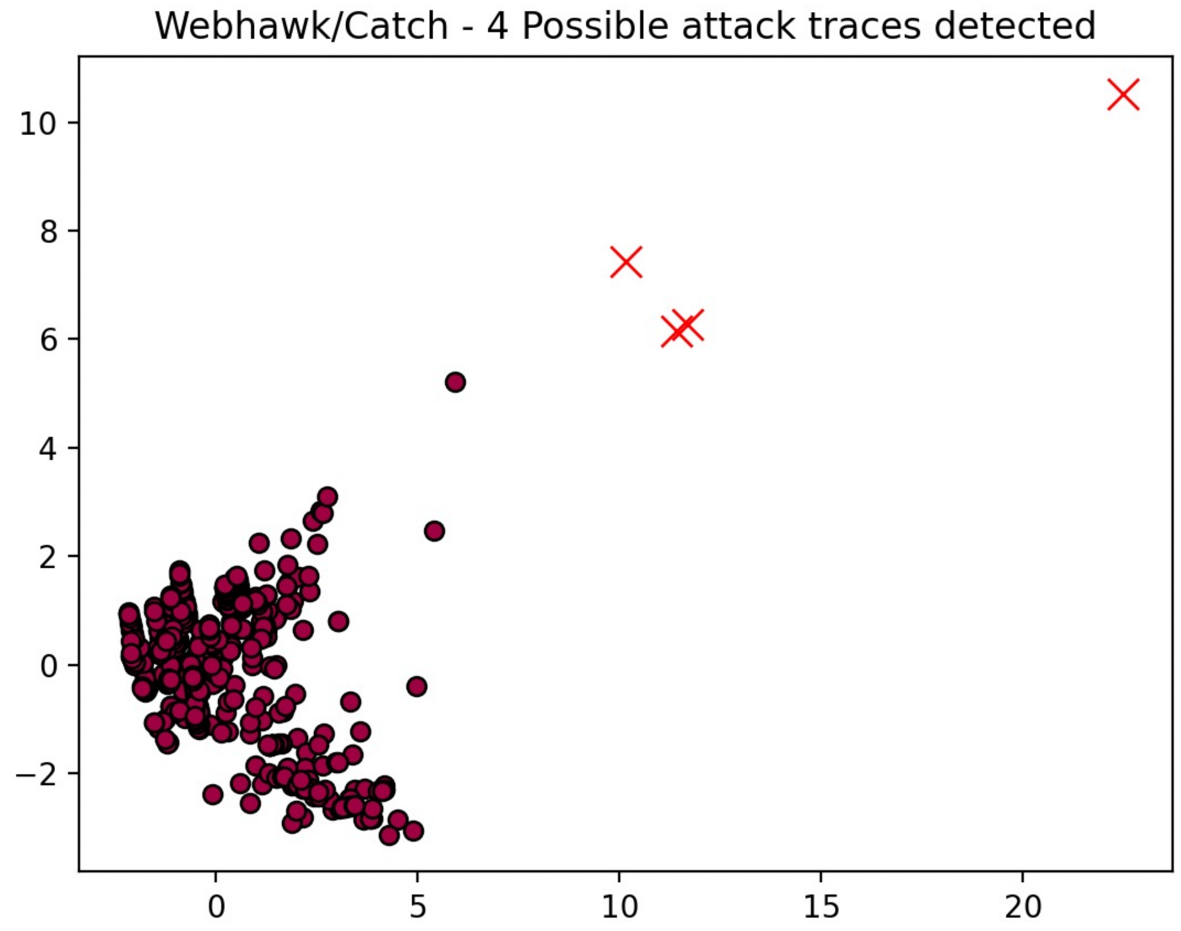
Optimization

Find the best tuning parameters while performing the steps 3 and 4

Step 5 Detection

Log lines are now grouped into clusters

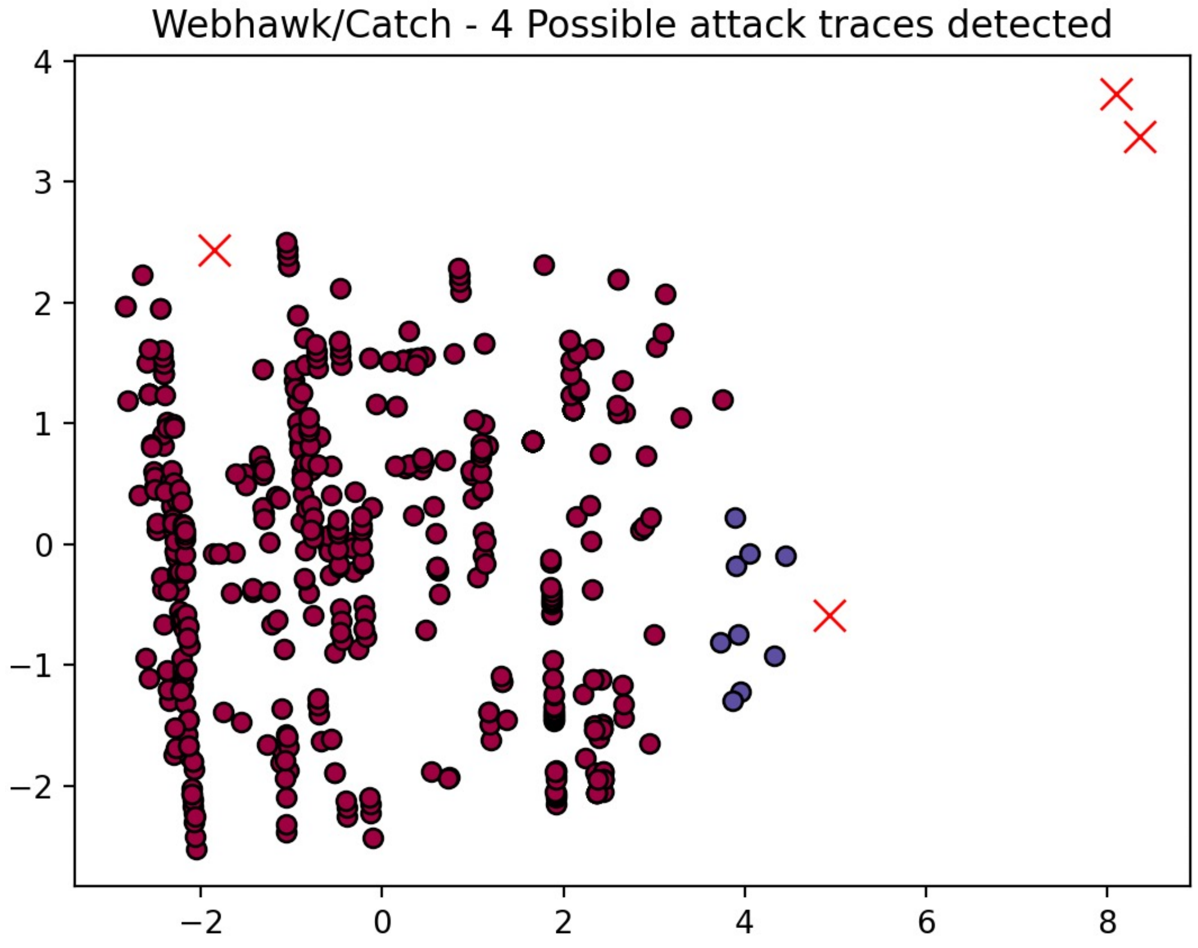
- The ones that don’t belong to any cluster are considered (outliers) as **High** severity detections
- The ones that belong to minority clusters are considered as **Medium** severity clusters



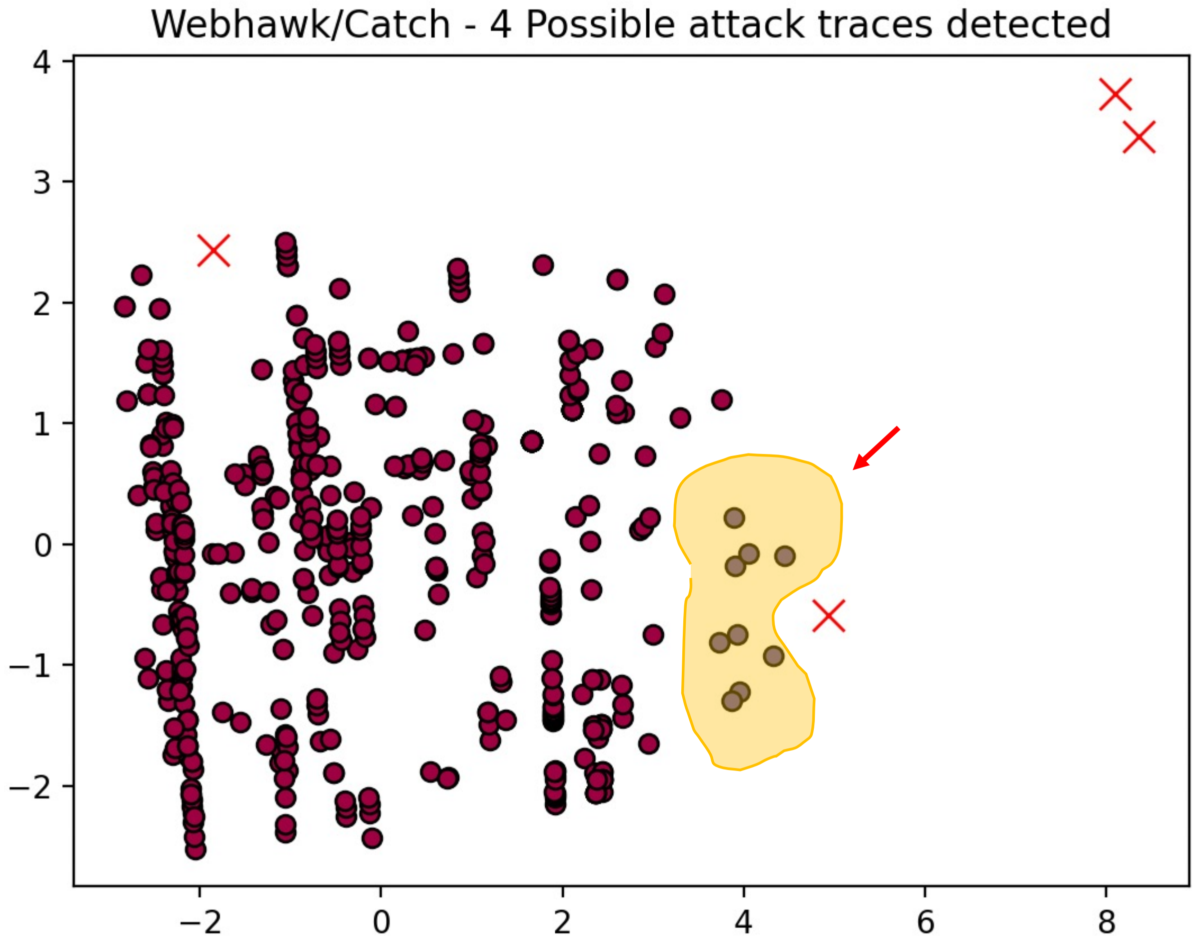
Step 5 Detection

Log lines are now grouped into clusters

- The ones that don't belong to any cluster are considered (outliers) as **High** severity detections
- The ones that belong to minority clusters are considered as **Medium** severity clusters



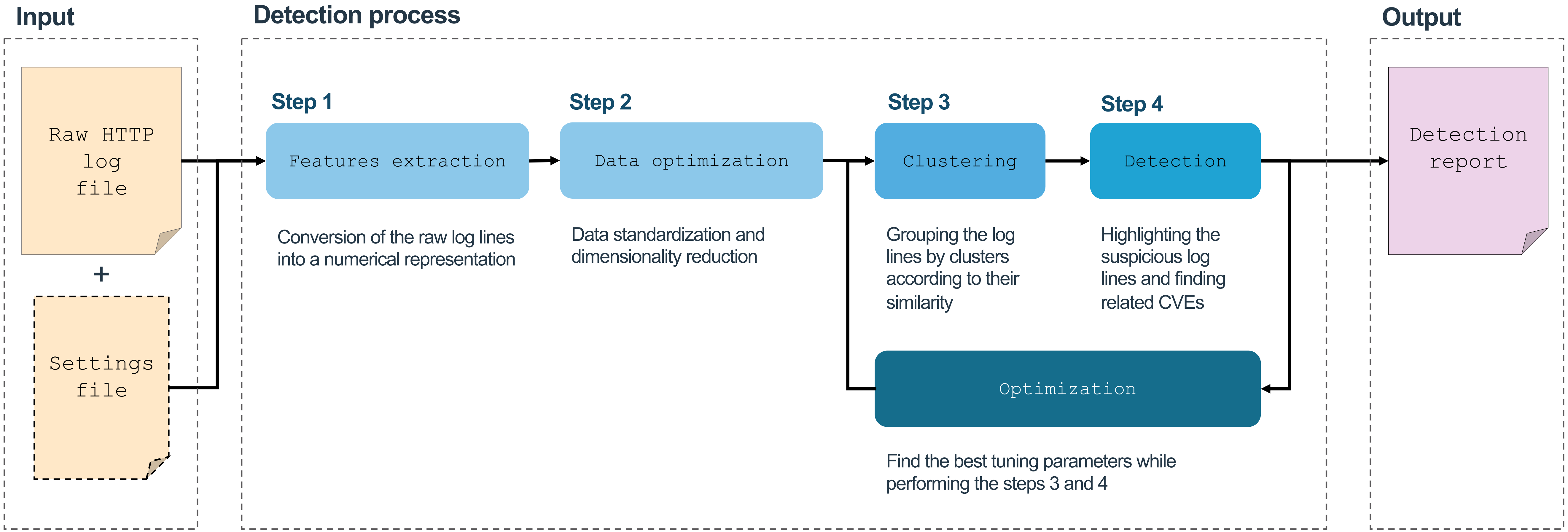
Optimization



Optimization parameters

minority_threshold: Maximum number of points to consider a cluster as a minority cluster – consider is as containing medium severity attack traces

Step 5 Detection



Input

Raw HTTP log file

+

Settings file

Detection process

Step 1

Features extraction

Conversion of the raw log lines into a numerical representation

Step 2

Data optimization

Data standardization and dimensionality reduction

Step 3

Clustering

Grouping the log lines by clusters according to their similarity

Step 4

Detection

Highlighting the suspicious log lines and finding related CVEs

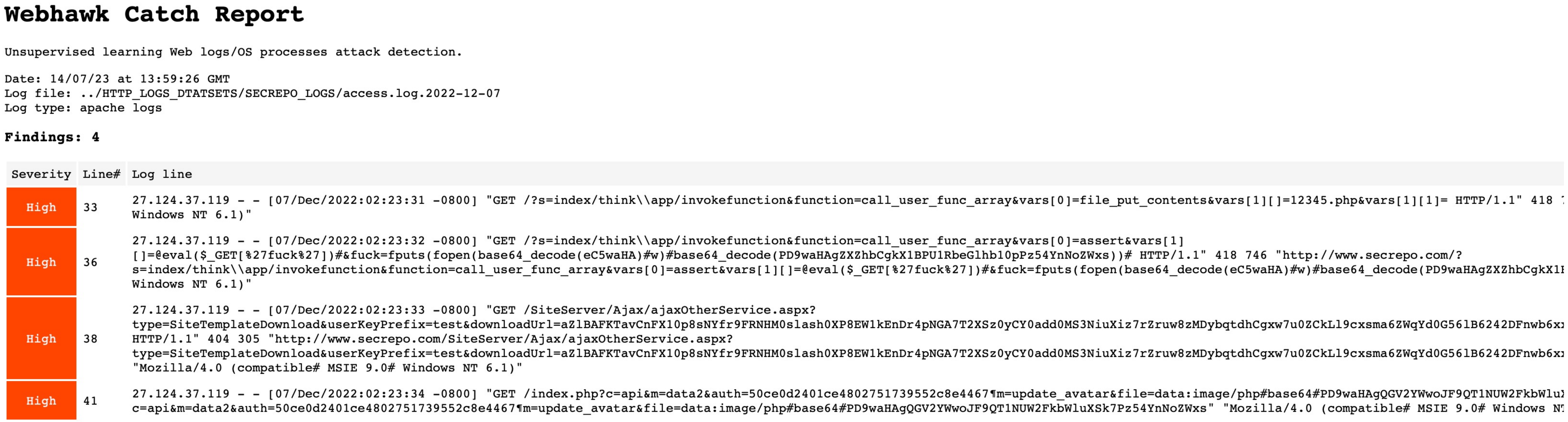
Optimization

Find the best tuning parameters while performing the steps 3 and 4

Output

Detection report

Step 5 Detection



Webhawk Catch Report

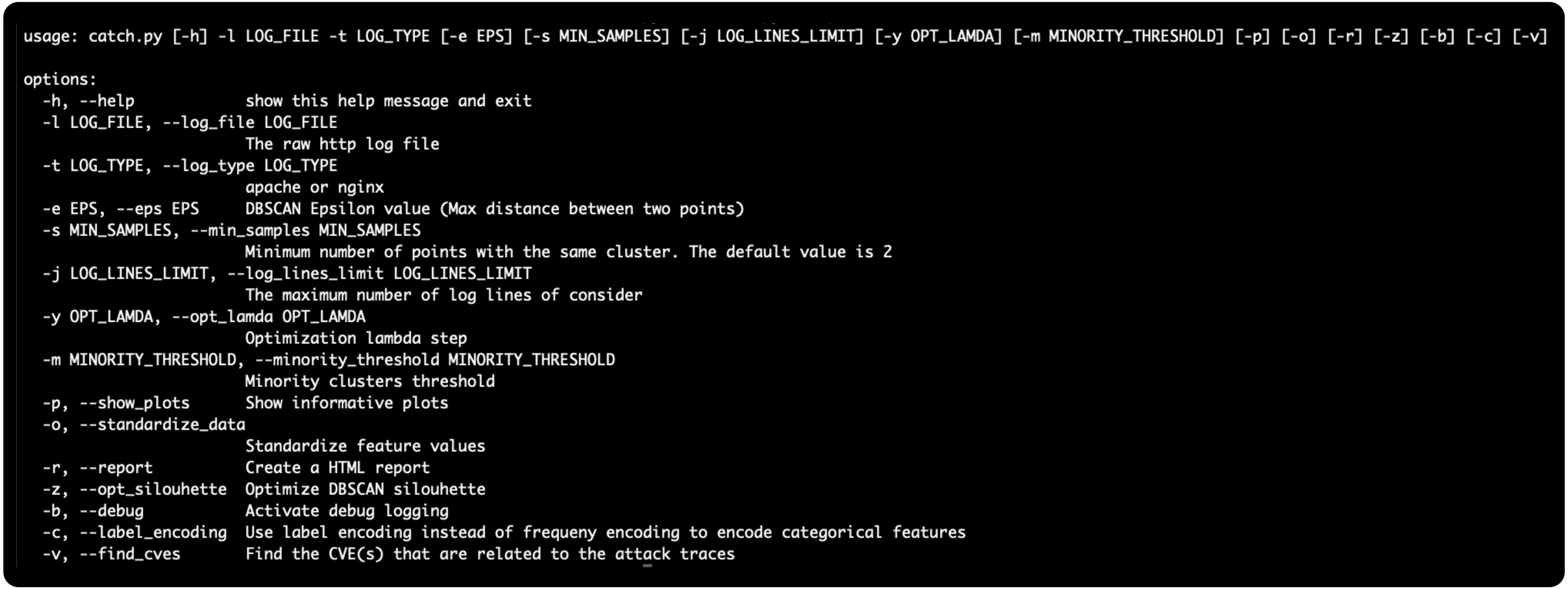
Unsupervised learning Web logs/OS processes attack detection.

Date: 14/07/23 at 13:59:26 GMT
Log file: ../HTTP_LOGS_DTATSETS/SECREPO_LOGS/access.log.2022-12-07
Log type: apache logs

Findings: 4

Severity	Line#	Log line
High	33	27.124.37.119 - - [07/Dec/2022:02:23:31 -0800] "GET /?s=index/think\\app/invokefunction&function=call_user_func_array&vars[0]=file_put_contents&vars[1][]=12345.php&vars[1][1]= HTTP/1.1" 418 Windows NT 6.1)"
High	36	27.124.37.119 - - [07/Dec/2022:02:23:32 -0800] "GET /?s=index/think\\app/invokefunction&function=call_user_func_array&vars[0]=assert&vars[1][]=@eval($_GET[%27fuck%27])#&fuck=fputs(fopen(base64_decode(eC5waHA)#w)#base64_decode(PD9waHAgZXZhbCgkX1BPU1RbeGlhb10pPz54YnNoZWxs))# HTTP/1.1" 418 746 "http://www.secrepo.com/?s=index/think\\app/invokefunction&function=call_user_func_array&vars[0]=assert&vars[1][]=@eval($_GET[%27fuck%27])#&fuck=fputs(fopen(base64_decode(eC5waHA)#w)#base64_decode(PD9waHAgZXZhbCgkX1 Windows NT 6.1)"
High	38	27.124.37.119 - - [07/Dec/2022:02:23:33 -0800] "GET /SiteServer/Ajax/ajaxOtherService.aspx?type=SiteTemplateDownload&userKeyPrefix=test&downloadUrl=aZlBAFKTavCnFX10p8sNYfr9FRNHM0slash0XP8EW1kEnDr4pNGA7T2XSz0yCY0add0MS3NiuXiz7rZruw8zMDybqtdhCgxw7u0ZCkLl9cxsma6ZWqYd0G561B6242DFnwb6x HTTP/1.1" 404 305 "http://www.secrepo.com/SiteServer/Ajax/ajaxOtherService.aspx?type=SiteTemplateDownload&userKeyPrefix=test&downloadUrl=aZlBAFKTavCnFX10p8sNYfr9FRNHM0slash0XP8EW1kEnDr4pNGA7T2XSz0yCY0add0MS3NiuXiz7rZruw8zMDybqtdhCgxw7u0ZCkLl9cxsma6ZWqYd0G561B6242DFnwb6x "Mozilla/4.0 (compatible# MSIE 9.0# Windows NT 6.1)"
High	41	27.124.37.119 - - [07/Dec/2022:02:23:34 -0800] "GET /index.php?c=api&m=data2&auth=50ce0d2401ce4802751739552c8e4467¶m=update_avatar&file=data:image/php#base64#PD9waHAgQGV2YWwoJF9QT1NUW2FkbWlu c=api&m=data2&auth=50ce0d2401ce4802751739552c8e4467¶m=update_avatar&file=data:image/php#base64#PD9waHAgQGV2YWwoJF9QT1NUW2FkbWluXSk7Pz54YnNoZWxs" "Mozilla/4.0 (compatible# MSIE 9.0# Windows N

Demo usage



```
usage: catch.py [-h] -l LOG_FILE -t LOG_TYPE [-e EPS] [-s MIN_SAMPLES] [-j LOG_LINES_LIMIT] [-y OPT_LAMDA] [-m MINORITY_THRESHOLD] [-p] [-o] [-r] [-z] [-b] [-c] [-v]

options:
  -h, --help            show this help message and exit
  -l LOG_FILE, --log_file LOG_FILE
                        The raw http log file
  -t LOG_TYPE, --log_type LOG_TYPE
                        apache or nginx
  -e EPS, --eps EPS     DBSCAN Epsilon value (Max distance between two points)
  -s MIN_SAMPLES, --min_samples MIN_SAMPLES
                        Minimum number of points with the same cluster. The default value is 2
  -j LOG_LINES_LIMIT, --log_lines_limit LOG_LINES_LIMIT
                        The maximum number of log lines of consider
  -y OPT_LAMDA, --opt_lamda OPT_LAMDA
                        Optimization lambda step
  -m MINORITY_THRESHOLD, --minority_threshold MINORITY_THRESHOLD
                        Minority clusters threshold
  -p, --show_plots      Show informative plots
  -o, --standardize_data
                        Standardize feature values
  -r, --report          Create a HTML report
  -z, --opt_silouhette  Optimize DBSCAN silouhette
  -b, --debug           Activate debug logging
  -c, --label_encoding  Use label encoding instead of frequeny encoding to encode categorical features
  -v, --find_cves       Find the CVE(s) that are related to the attack traces
```

Demo 1

```
Log_file=access.log.2023-02-18

python catch.py --log_file $log_file --log_type apache --report

python catch.py --log_file $log_file --log_type apache --standardize_data --report

python catch.py --log_file $log_file --log_type apache --standardize_data --find_cves --report



--opt_silouhette to reduce FP
```

Demo 2

```
Log_file=access.log.2023-06-09

python catch.py --log_file $log_file --log_type apache --report

python catch.py --log_file $log_file --log_type apache --standardize_data --report

python catch.py --log_file $log_file --log_type apache --standardize_data --find_cves --report



--show_plots
```

Demo 3

```
Log_file=access.log.2022-12-07

python catch.py --log_file $log_file --log_type apache --report

python catch.py --log_file $log_file --log_type apache --standardize_data --report

python catch.py --log_file $log_file --log_type apache --standardize_data --find_cves --report



--show_plots
```

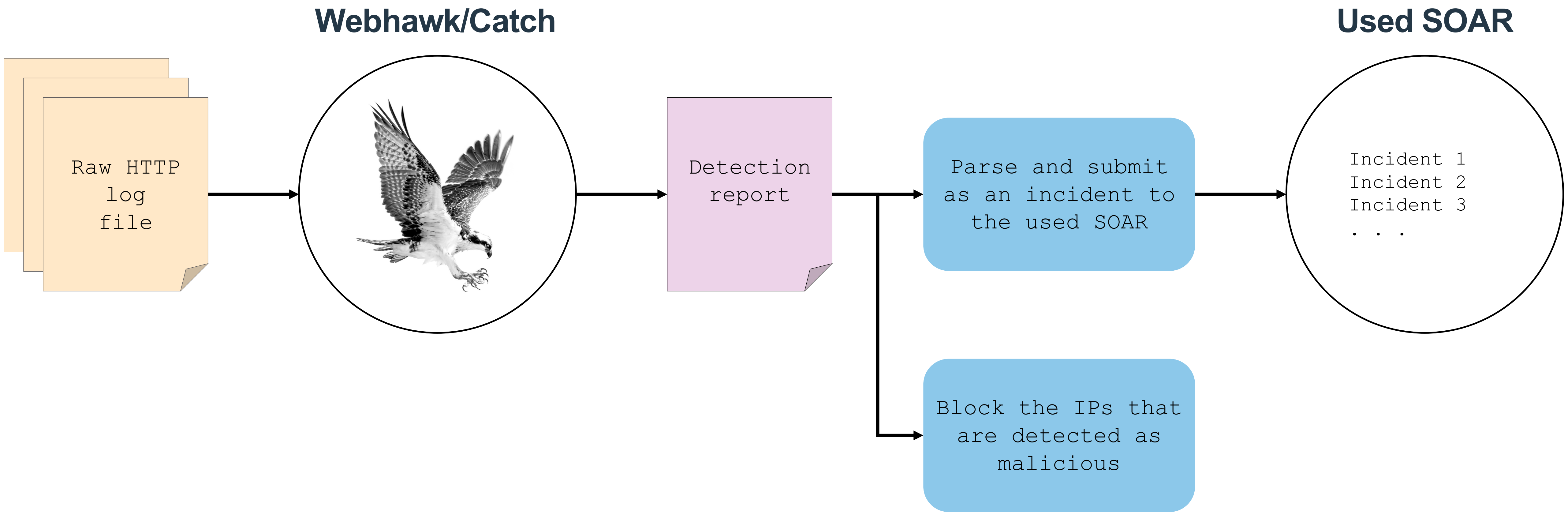
Demo 4

```
Log_file=access.log.2023-04-02

python catch.py --log_file $log_file --log_type apache --report

python catch.py --log_file $log_file --log_type apache --standardize_data --report

python catch.py --log_file $log_file --log_type apache --standardize_data --find_cves --report




--show_plots
```

Demo 5

```
Log_file=access.log.2023-02-04

python catch.py --log_file $log_file --log_type apache --report

python catch.py --log_file $log_file --log_type apache --standardize_data --report

python catch.py --log_file $log_file --log_type apache --standardize_data --find_cves --report



--show_plots
```

SOC integration



Webhawk/Catch

Used SOAR

Raw HTTP log file

Detection report

Parse and submit as an incident to the used SOAR

Block the IPs that are detected as malicious

Incident 1
Incident 2
Incident 3
. . .

What’s next

- To add more application log types
- Further optimizing the detection process
- Automatically find the related CVE and add them to the report (***part. done***)
- Add an API to simplify the integration with other tools

Thank you

Give Webhawk a ⭐ at Github!
Contribute? All you pull requests are welcome

https://github.com/slrbl/unsupervised-learning-attack-detection-webhawk-catch

https://github.com/slrbl/Intrusion-and-anomaly-detection-with-machine-learning