black hat ARSENAL

DECEMBER 6-7, 2023

EXCEL LONDON / UNITED KINGDOM

black hat ARSENAL

Packing Box

Breaking Detectors & Visualizing Packing

By Alexandre D'Hondt, Romain Jennes, Sébastien Martinez Balbuena, Charles-Henry Bertrand Van Ouytsel and Axel Legay

Outline

1. Introduction
2. Background
3. Framework
4. Breaking Detectors
5. Visualizing Packing
6. Conclusion

Outline

- Problem statement
- Objectives

Problem statement (1)

Packing =

- Set of transformations
- On binary file
- That preserves the original working at runtime

→ Large coverage in scientific literature

→ Static detection increasingly relying on Machine Learning

→ Often employed with malware

Problem statement (2)

- Experimental toolkit focused on executable packing
- Solves experiments repeatability

Packing Box: Playing with Executable Packing (BHEU22)

Static detection challenges (con't) :

- Exhaustive feature engineering
- Static features robustness to adversarial attacks
- Feature set inspection for quality & validation

- No focus on adversarial study yet
- Almost no coverage on unsupervised learning in related works

Objectives

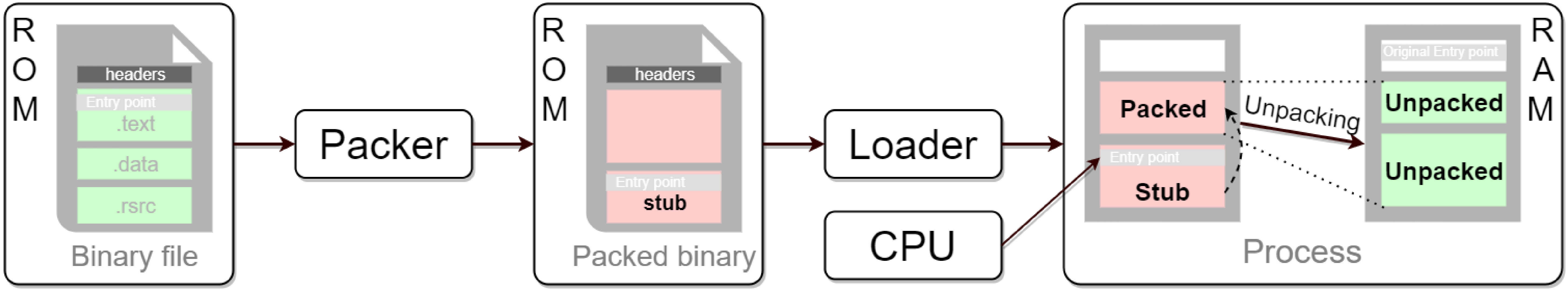
1. Extend Packing Box with adversarial and unsupervised learning capabilities
2. Build binary alterations and break detectors using them
3. Explore features through visualization and train models with unsupervised learning

Outline

1. Introduction
2. Background
3. Framework
4. Breaking Detectors
5. Visualizing Packing
6. Conclusion

- Packing / unpacking
- Static detection & features
- Experimental Toolkit
- Adversarial Learning
- Unsupervised Learning

Packing / unpacking



Transformations :

- Compression
- Encryption
- Protection
- Bundling
- Mutation
- Virtualization

Common usage :

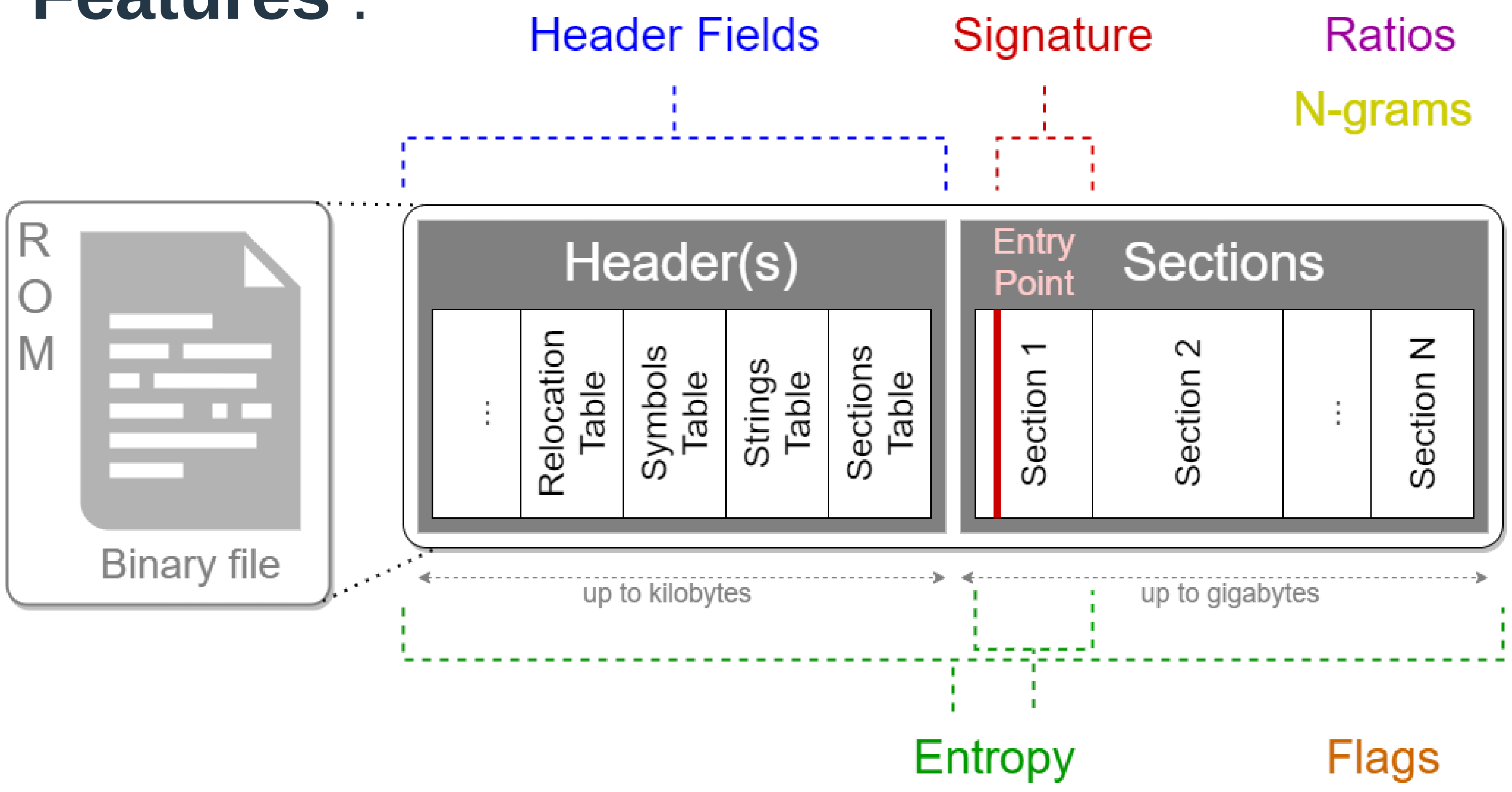
- Size reduction
- SW piracy prevention / License management
- Malware

Static detection & features

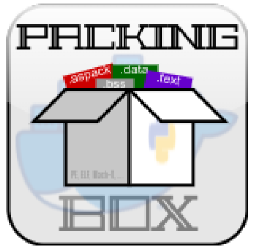
Static (no execution) :

- Entropy threshold
- Pattern matching
- Signatures
- Heuristics
- Disassembly
- Machine Learning
- …

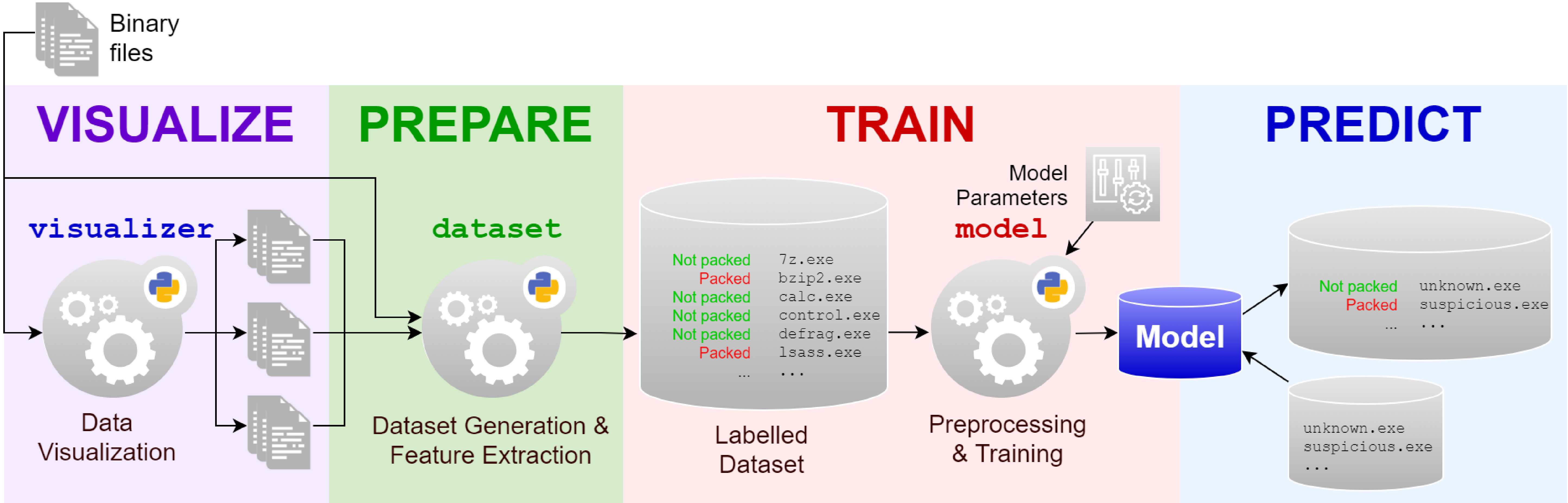
Features :



Experimental toolkit



Learning pipeline automation



Adversarial Learning (1)

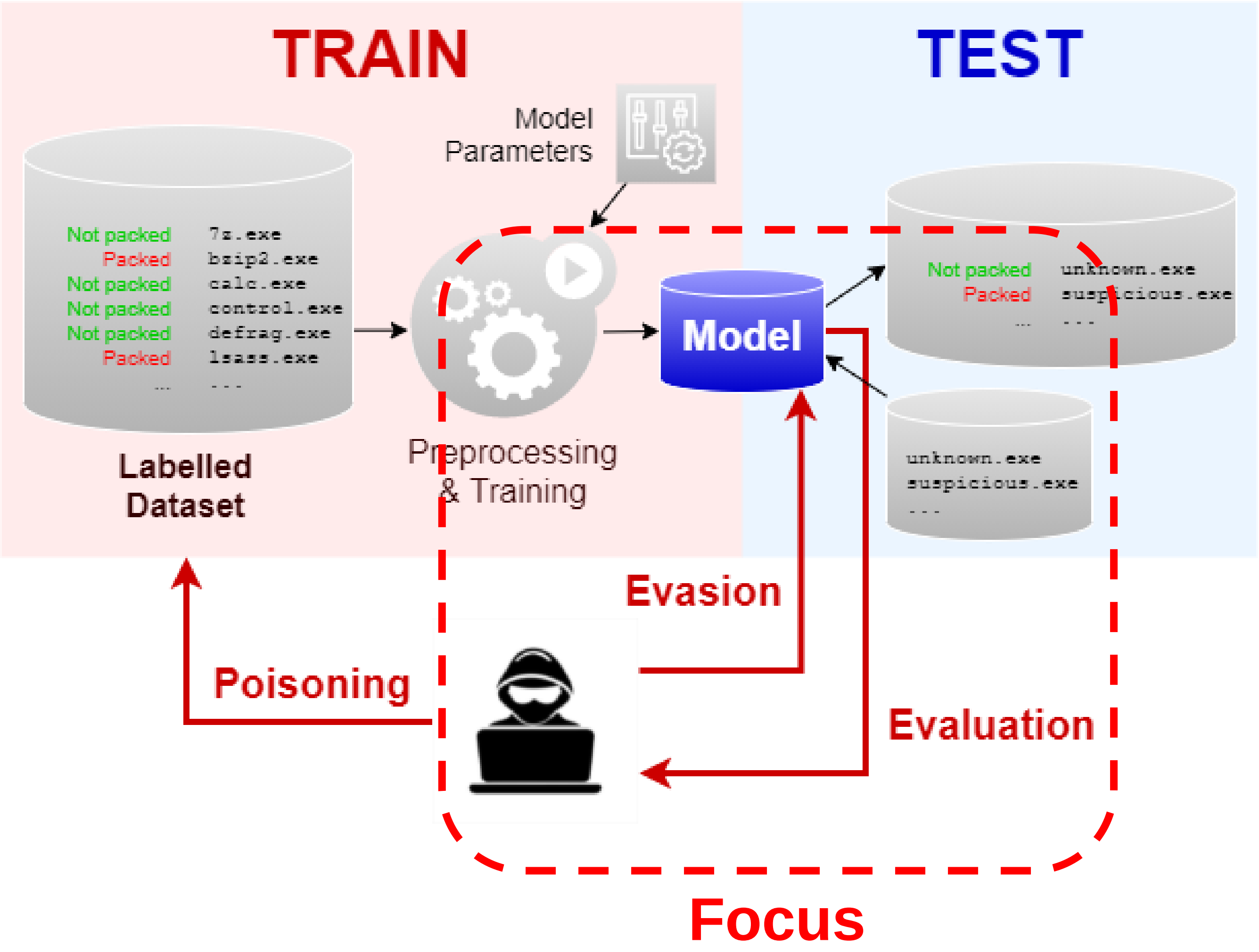
Threat model

- Attack Surface :

  Train (poisoning) VS Test (evasion) phase

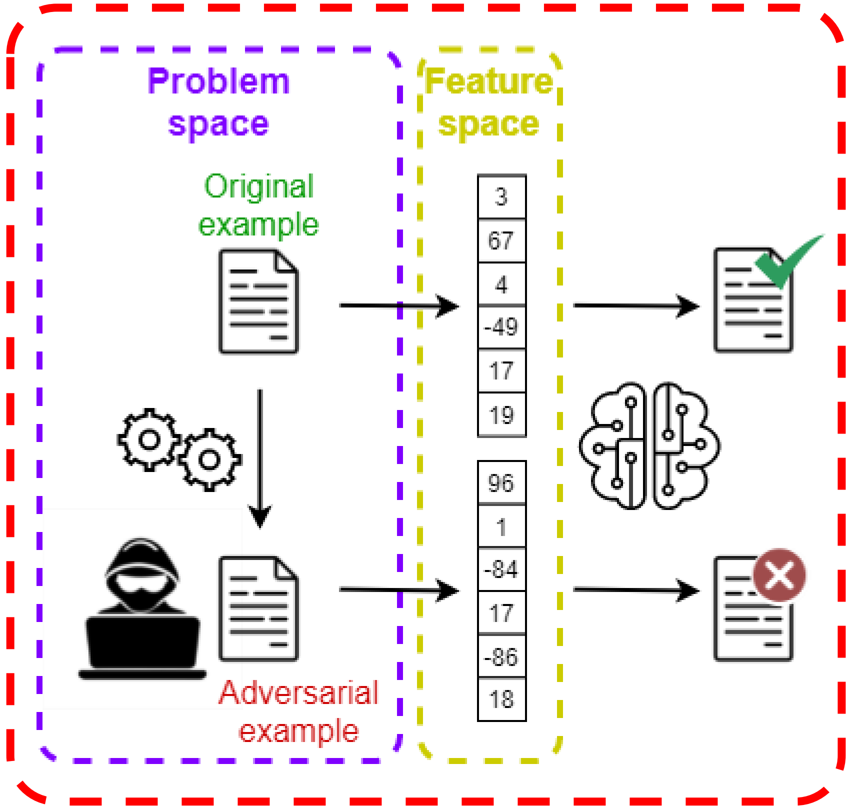
- Adversary :
  - Goal : Untargeted VS targeted
  - Capatibilities : ability to modify samples

    (tied to executable formats)
  - Knowledge : white-box VS black-box



Adversarial Learning (2)

Problem-space VS Feature-space attacks



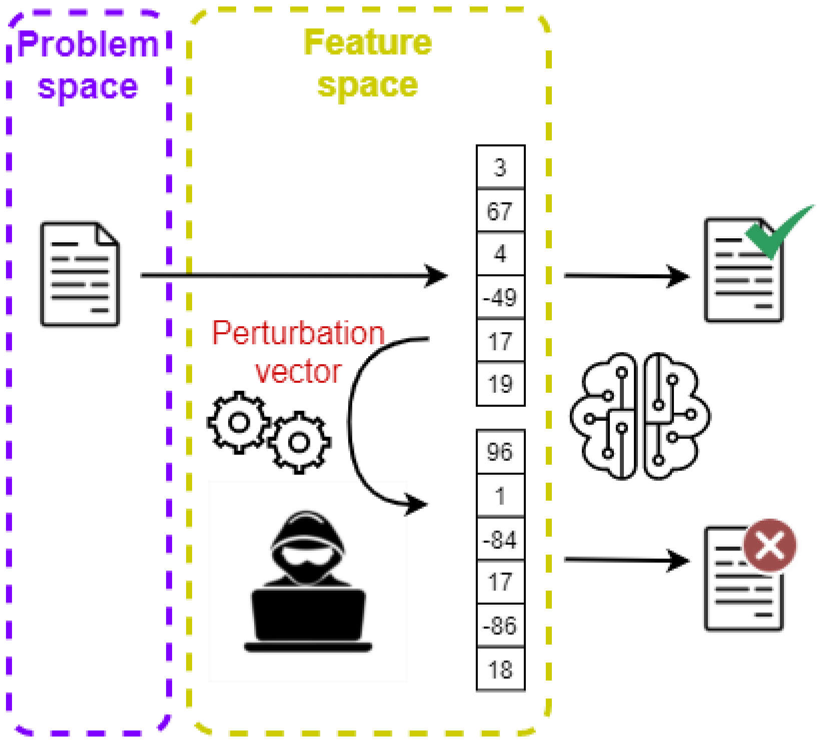
Focus

Problem-space : data transformation

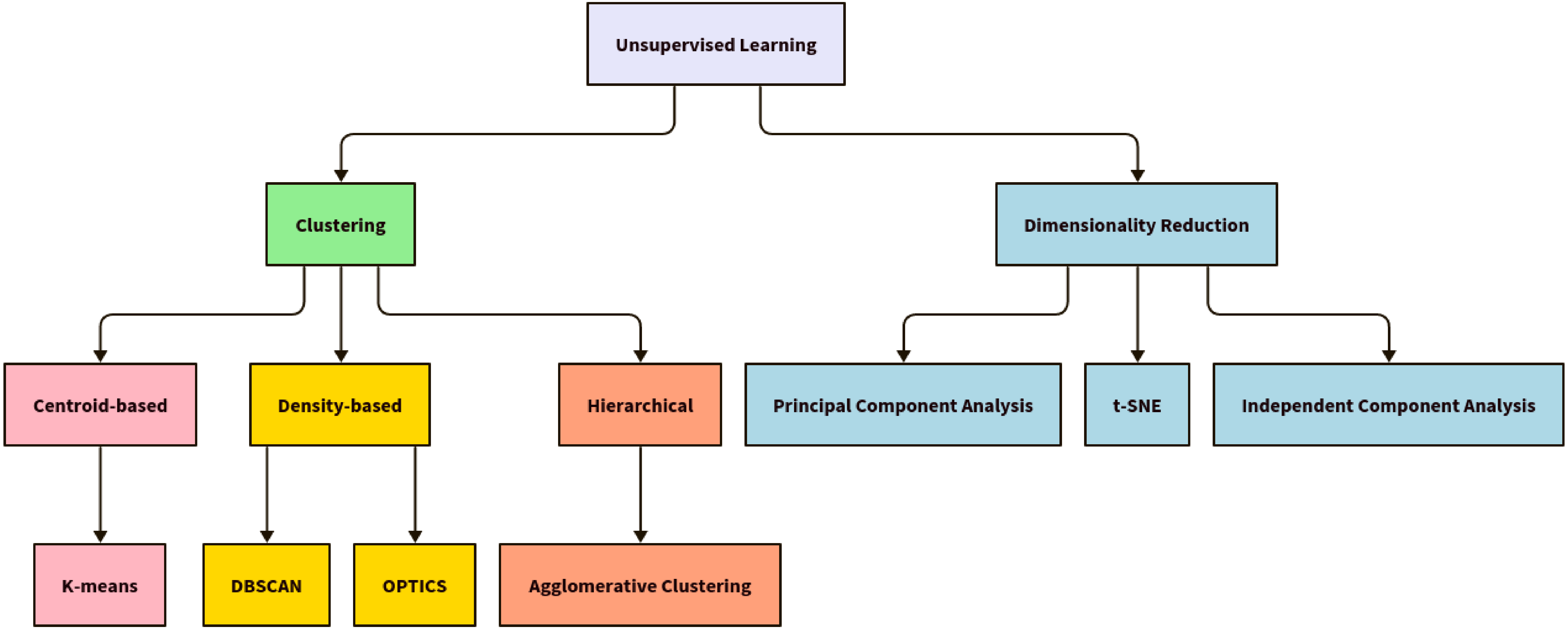
- Can check validity of data
- No direct control on features

Feature-space : features perturbation

- Requires to feed features to the model
- Feature-to-problem mapping
- Easier

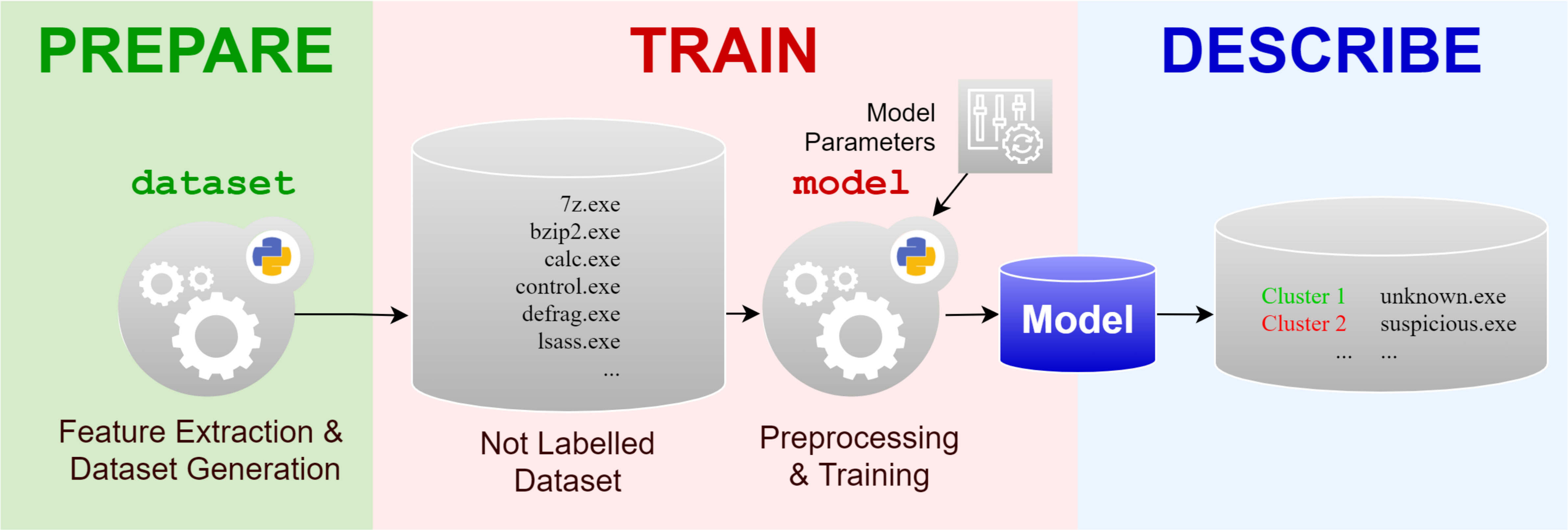


Unsupervised Learning (1)



Unsupervised Learning (2)

Learning pipeline



Outline

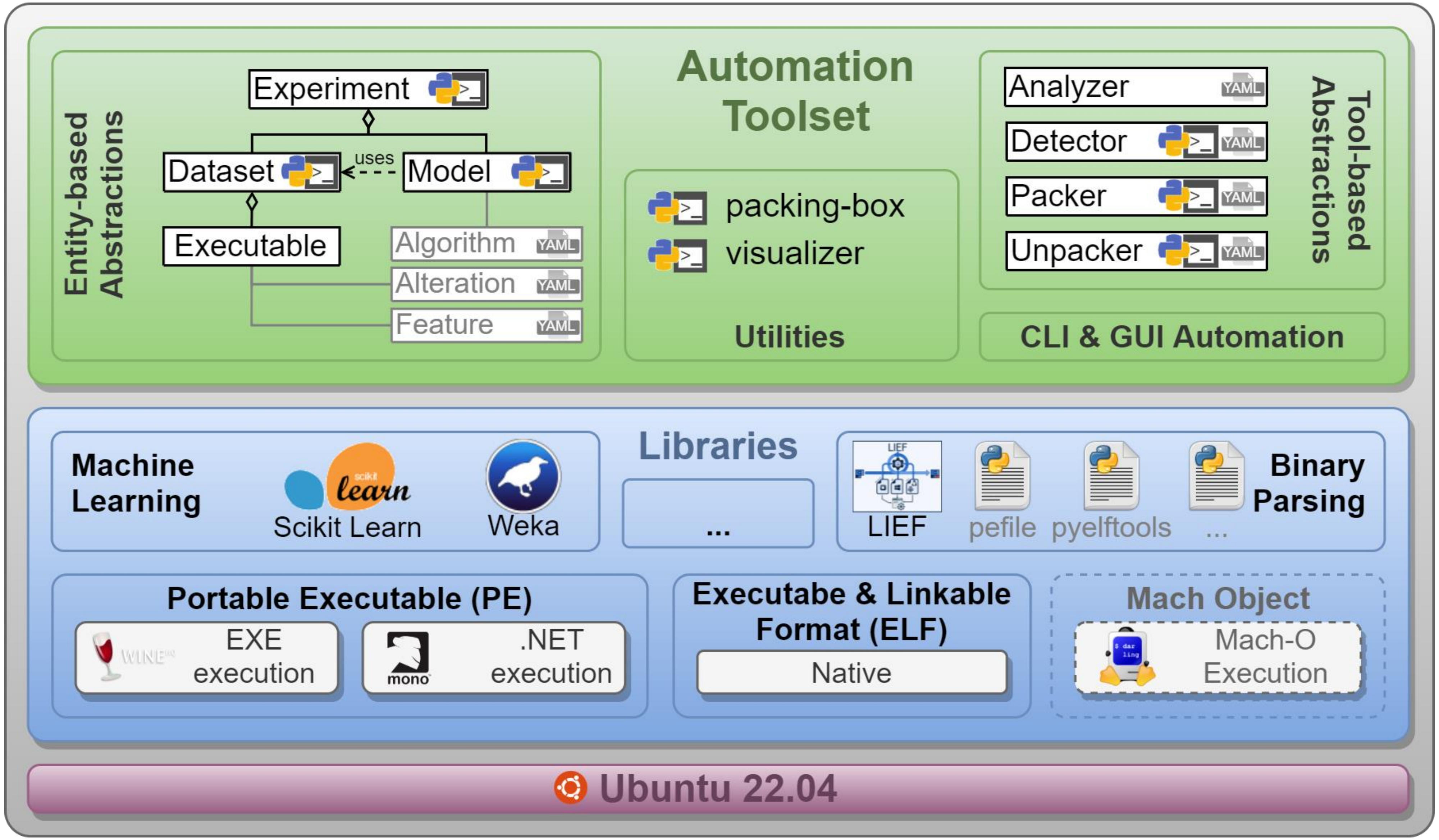
- New requirements
- Updated architecture
- Added capabilities
- Getting started

New requirements

A. Support for binary alterations
   I. Binary parsing & modification
   II. Alterations as combinations of modifications
   III. Plots for validating impact on features

B. Support for unsupervised learning
   I. Unlabelled samples
   II. Variety of new algorithms
   III. Plots for visualizing clusters & models

Updated architecture



Workspace

- Configurations
- Datasets & models
- Executable format-specific data
- (Figures of visualizations)
- (Scripts)

Experiment = dedicated workspace

Added capabilities

Read the Docs

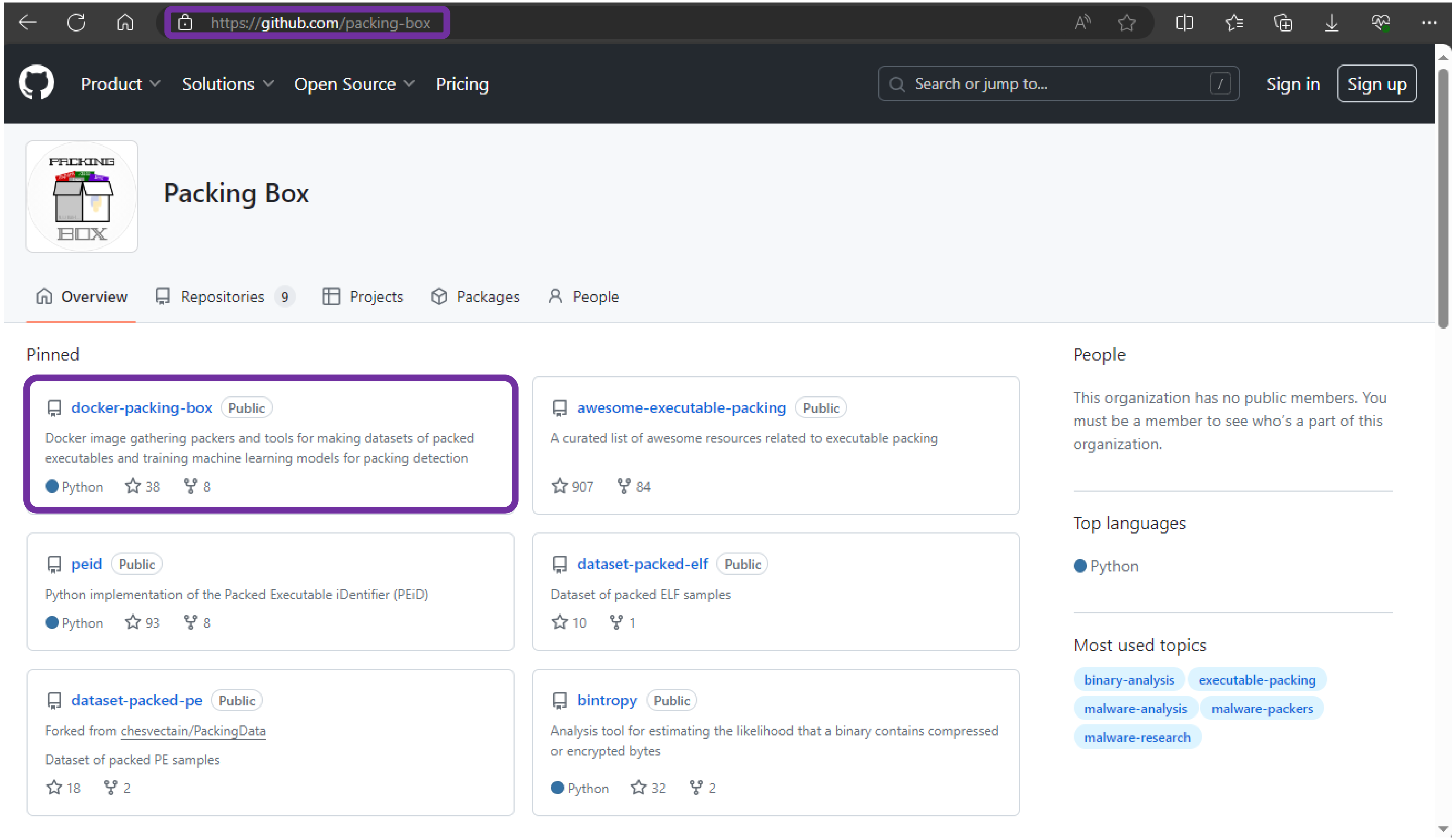
- Executable parsing layer (parser selection)
- Alterations relying on modifiers (similar to Features relying on extractors)
- Support for semi-supervised & unsupervised learning (dataset labelling)
- Visualization :
  - For comparing datasets' features
  - Of reduced data (i.e. clusters)

Getting started (1)

See: [Packing Box: Playing with Executable Packing (BHEU22)]

Starting point :

1. Open terminal
2. Clone the repo



Getting started (2)

See presentation of Black Hat Europe 2022 for basic demonstrations :

Basic

- Build & run Docker image
- Getting help
- Installing items
- Playing with datasets
- Playing with models

Advanced

- Model for PE packers
- Visualization of files & models
- Evaluation of detectors

Outline

1. Introduction
2. Background
3. Framework
4. Breaking Detectors
5. Visualizing Packing
6. Conclusion

- Scope
- Methodology
  1. Samples inspection
  2. Performance evaluation
  3. Alterations
  4. Re-evaluation

Scope

Targeted detectors

- Detect It Easy (DIE)
  Signature-based with ad hoc heuristic
- PEiD
  Signature-based
- Manalyze
  Simple pattern matching on section names

Dataset

Ready-to-use dataset of packed and not-packed PE files from the enriched version of this repository

- ASPack
- UPX
- Yoda Protector


```
$ experiment open breaking-detectors
$ for P in ASPack UPX Yoda-Protector; do dataset udpate $P --source dataset-packed-pe/packed/$P
   --labels dataset-packed-pe/labels.json; done
```

Methodology

Steps

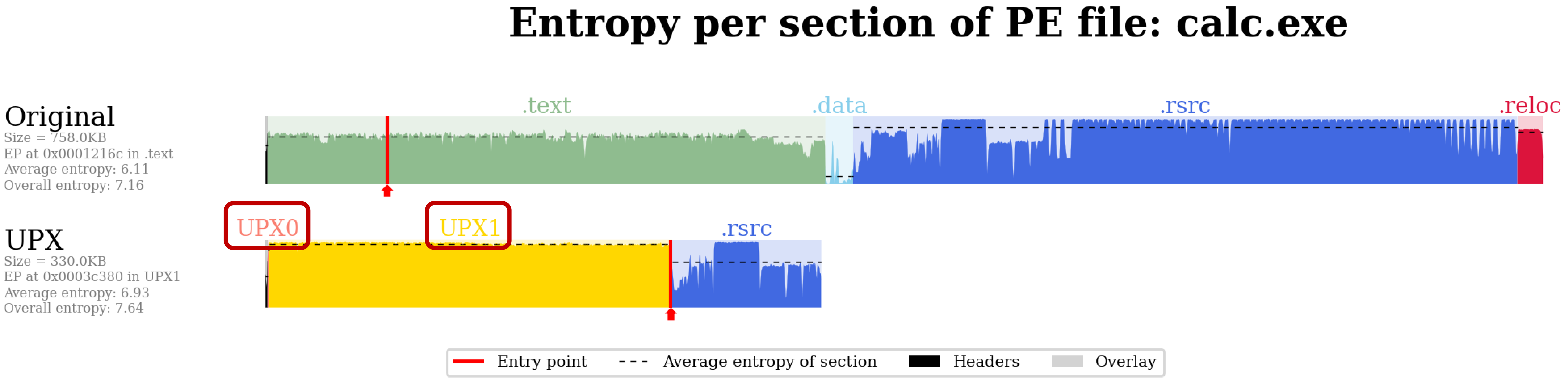
1. Inspect samples from [dataset-packed-pe]
   Visualising binary samples first to analyze packer's specificities (additionally remove outliers)

2. Evaluate the performance of targeted detectors
   Baseline the detection rate on the input dataset

3. Build alterations and apply them
   Based on packer's specificities, design combinations of modifications

4. Evalute the performance after alterations
   Once alterations are applied, refresh the detection rate and compare with the baseline

1. Samples inspection



Try It Now !

```
$ visualizer plot "calc.exe$" dataset-packed-pe --label not-packed --label UPX --scale
$ dataset plot samples UPX -n 10
```

Please report any issue at : https://github.com/packing-box/docker-packing-box/issues

2. Performance evaluation

Preliminary notes

- Only **binary classification** is considered (packed or not packed)
- Only **accuracy** is relevant here (small datasets of packed samples only)

	DIE	PEiD	Manalyze
ASPack	100,00%	100,00%	100,00%
UPX	100,00%	100,00%	100,00%
Yoda Protector	100,00%	100,00%	100,00%


```
$ for P in ASPack UPX Yoda-Protector; do \
    for $D in die peid reminder; do detector $P --detector $D; done; \
  done
```

3. Alterations (build)

[A1] Rename sections

- ASPack : .aspack → .code ; .adata → .data
- UPX : UPX0 → .data ; UPX1 → .text
- Yoda Protector : .yP → .code

[A2] Add low entropy code section

[A3] Move EP to new code section

Try It Now !

```
$ experiment edit alterations
```

```
rename_packer_sections:
  decription: Rename UPX0 and UPX1 sections to .data and .text
  apply: true
  result:
    PE:
      - rename_section("UPX0", ".data")
      - rename_section("UPX1", ".text")

add_low_entropy_text_section:
  description: Add a code section with low entropy and common name
  result:
    PE: add_section(".text",
                    section_type=pe['SECTION_TYPES']['TEXT’],
                    data=b"\x01"*(1<<16))

move_entrypoint_to_new_low_entropy_section:
  description: Move EP to a new low-entropy section with common name
  result:
    PE: move_entrypoint_to_new_section(".text",
                                       post_data=b'\x00'*64)
```

3. Alterations (apply)

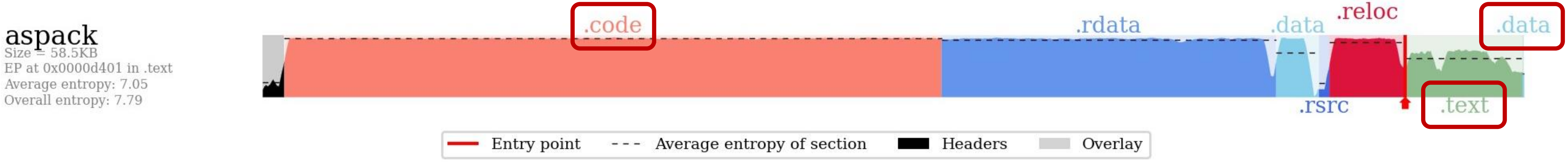
Configure & run

- Set `apply:true`
- For `rename_packer_sections`, adapt according to packer

```
rename_packer_sections:
  apply: true
  decription: Rename .aspack to .text and .adata to .data
  result:
    PE:
      - rename_section(".text", ".code", error=False)
      - rename_section(".aspack", ".text")
      - rename_section(".adata", ".data")
```

Try It Now !

```
$ for I in {1..3}; do dataset select ASPack aspack-a$I; done  # restart for UPX and Yoda-Protector
$ experiment edit alterations                                 # configure in-scope alteration
$ dataset alter aspack-a1                                     # restart from previous command for upx-aX
$ dataset plot samples aspack-a1 -n 5
```



4. Re-evaluation (rates)

	Detect It Easy (DIE)			PEiD			Manalyze		
	A1	A2	A3	A1	A2	A3	A1	A2	A3
ASPack	100,00%	100,00%	0,00%	100,00%	100,00%	0,00%	0,00%	100,00%	100,00%
UPX	96,69%	96,69%	0,83%	100,00%	100,00%	0,00%	3,31%	100,00%	100,00%
Yoda Protector	100,00%	100,00%	0,00%	100,00%	100,00%	0,00%	0,00%	100,00%	100,00%

A1: rename sections – **A2**: add low entropy code section – **A3**: move EP to new code section

Try It Now !

```
$ for P in aspack upx yp; do \
    for $D in die peid reminder; \
      do for I in {1..3}; do detector $P-a$I --detector $D; done; \
    done
  done
```

4. Re-evaluation (impact)

	Detect It Easy (DIE) Signatures + ad hoc heuristic			PEiD Signatures			Manalyze Pattern matching on section names		
ASPack	100,00%	100,00%	0,00%	100,00%	100,00%	0,00%	0,00%	100,00%	100,00%
UPX	96,69%	96,69%	0,83%	100,00%	100,00%	0,00%	3,31%	100,00%	100,00%
Yoda Protector	100,00%	100,00%	0,00%	100,00%	100,00%	0,00%	0,00%	100,00%	100,00%
	A1	A2	A3	A1	A2	A3	A1	A2	A3

Impact

- **DIE** is almost completely disturbed when adding a new code section with the EP
- **PEiD** matches signatures after the EP, hence detection is broken when moving EP
- **Manalyze**'s plugin for packer identification is only based on pattern matching against (very) common packer section names

Outline

- Setup
- Exploratory Data Analysis
- Clustering Models

5. Visualizing Packing

Scope

ML activities

- Exploratory data analysis
  Plotting reduced data regarding characteristics
- Unsupervised model training
  Building a model with a reduced dataset
- Dataset description
  Observing classes using the trained model

Dataset

Ready-to-use dataset of packed and not-packed PE files from the enriched version of this repository

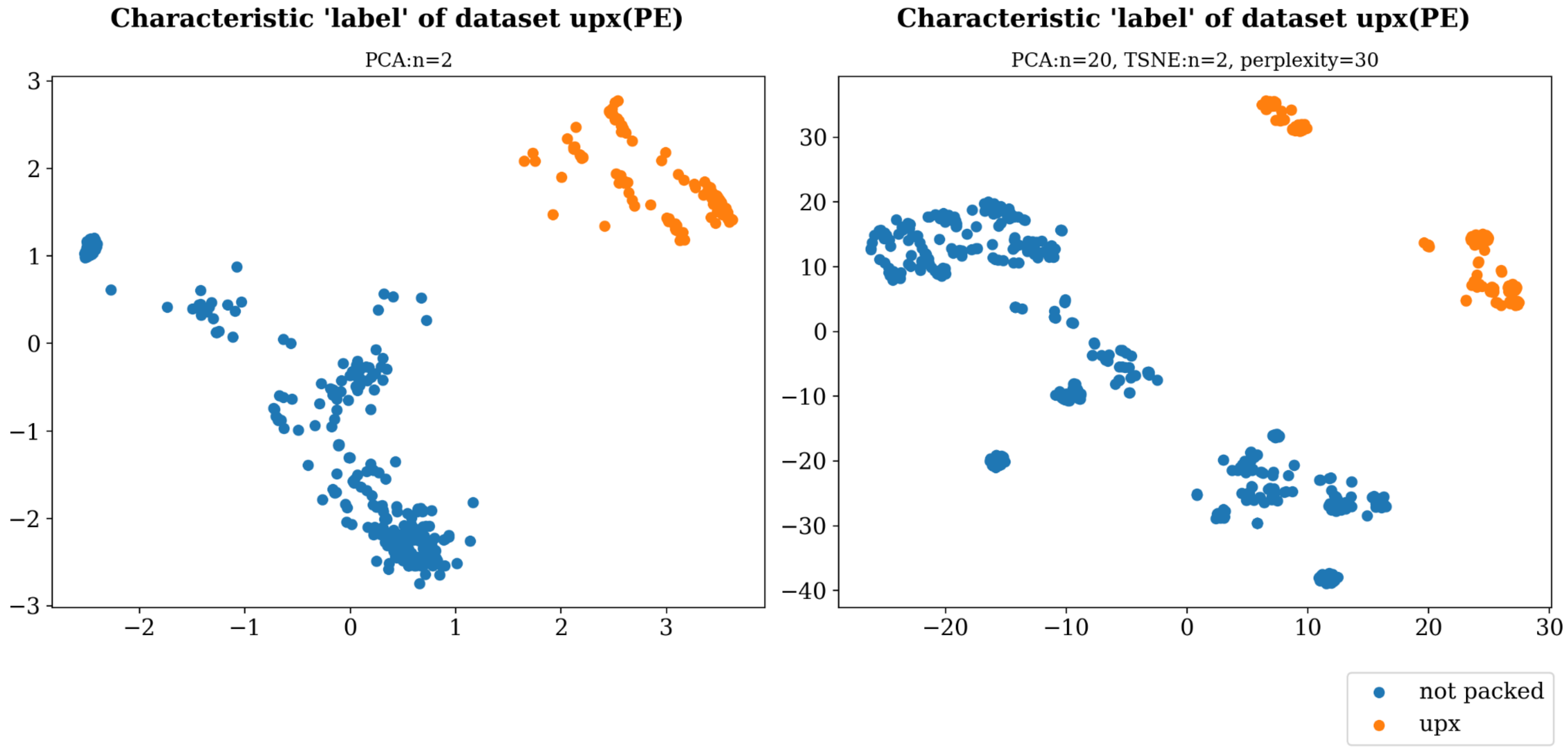
- ASPack
- JDPack
- NSPack
- PECompact
- UPX


```
$ experiment open visualizing-packing
$ dataset udpate upx --source dataset-packed-pe/packed/UPX --labels dataset-packed-pe/labels.json
$ for P in ASPack JDPack NSPack PECompact UPX; do dataset udpate main --source dataset-packed-
   pe/packed/$P --labels dataset-packed-pe/labels.json; done
```

Exploratory Data Analysis (1)

Observations

- UPX only (very simplistic case)
- Clear clusters per label
- Any unsupervised learning algorithm will achieve perfect classification if using 2 clusters
- TSNE does not improve results here ; even worse, it visually gives 4-5 clusters



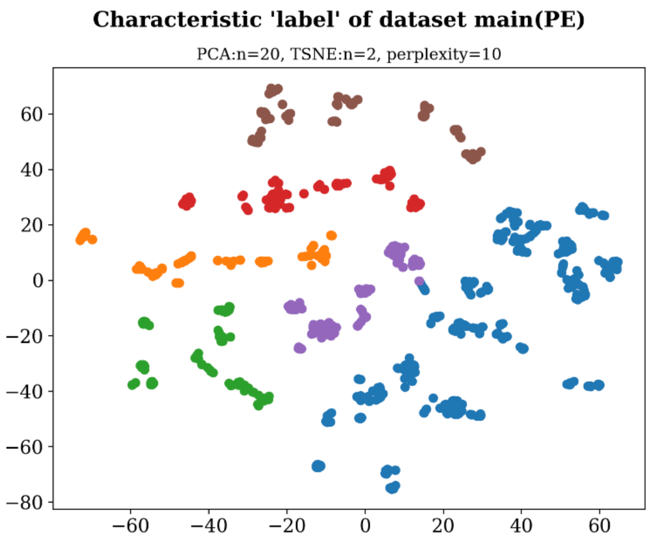
Try It Now !

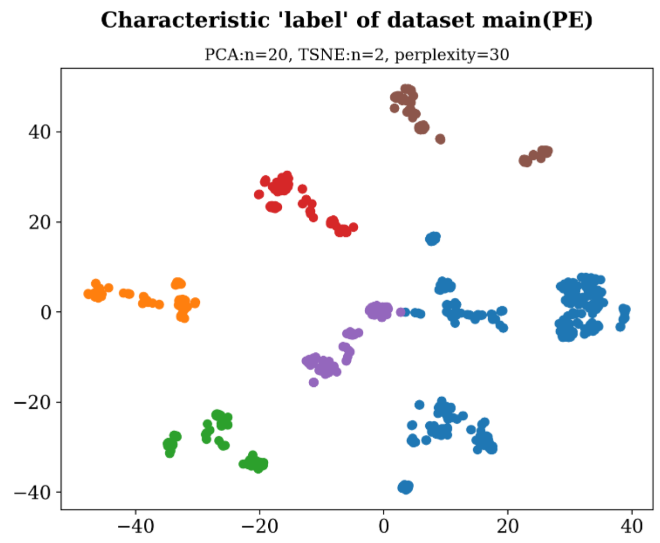
```
$ dataset convert upx                                  # precompute features
$ dataset plot characteristic upx label                # PCA with 20 components then TSNE 2D
$ dataset plot characteristic upx label -n 2           # PCA with 2 components (hence no TSNE)
```

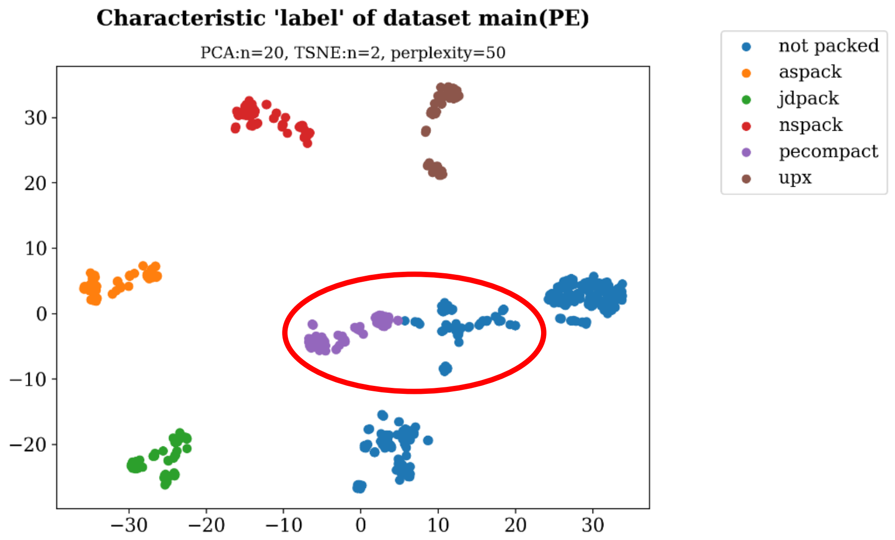
Exploratory Data Analysis (2)

Observations

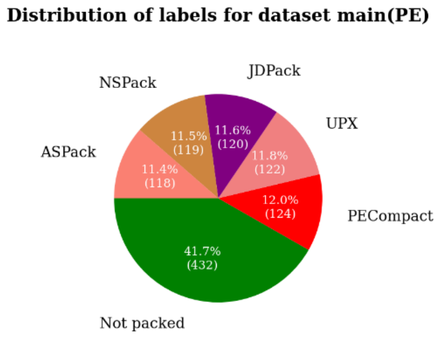
- Multiple compression packers
- Low perplexity yields unclear clusters ; increasing it reveals better clusters
- Some properties of PECompact make it difficult to distinguish from not packed samples
- At the end, we get possibly 6-7 clusters with model training ; maybe 2 for not packed samples








```
$ dataset convert main
$ dataset plot labels
$ for P in 10 30 50; do dataset plot characteristic
         main label --perplexity $P; done
```



Clustering Models (1)

Setting

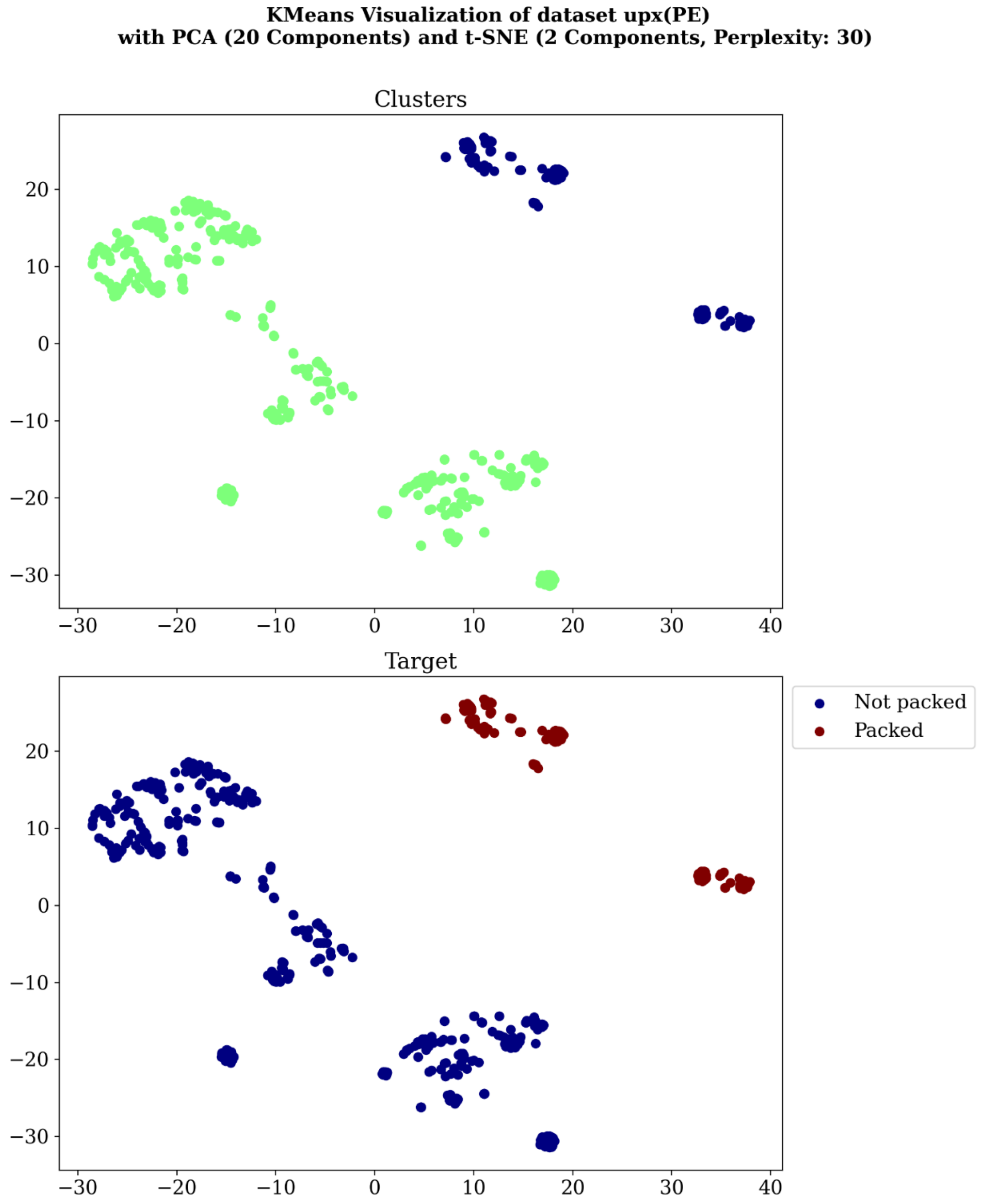
- UPX only (very simplistic case)
- Clustering algorithm : K-Means
- Hyperparameter $N_{clusters}$ set to 2 (result of EDA)

Observations

- 2 distinct clusters as expected
- Perfect classification
- Even setting $N_{clusters}$ to auto yields 2 clusters too

Try It Now !

```
$ experiment edit algorithms
$ model train upx -a kmeans
$ model visualize upx_pe_554_kmeans_f138 --export
        --plot-labels
```



Clustering Models (2)

Setting

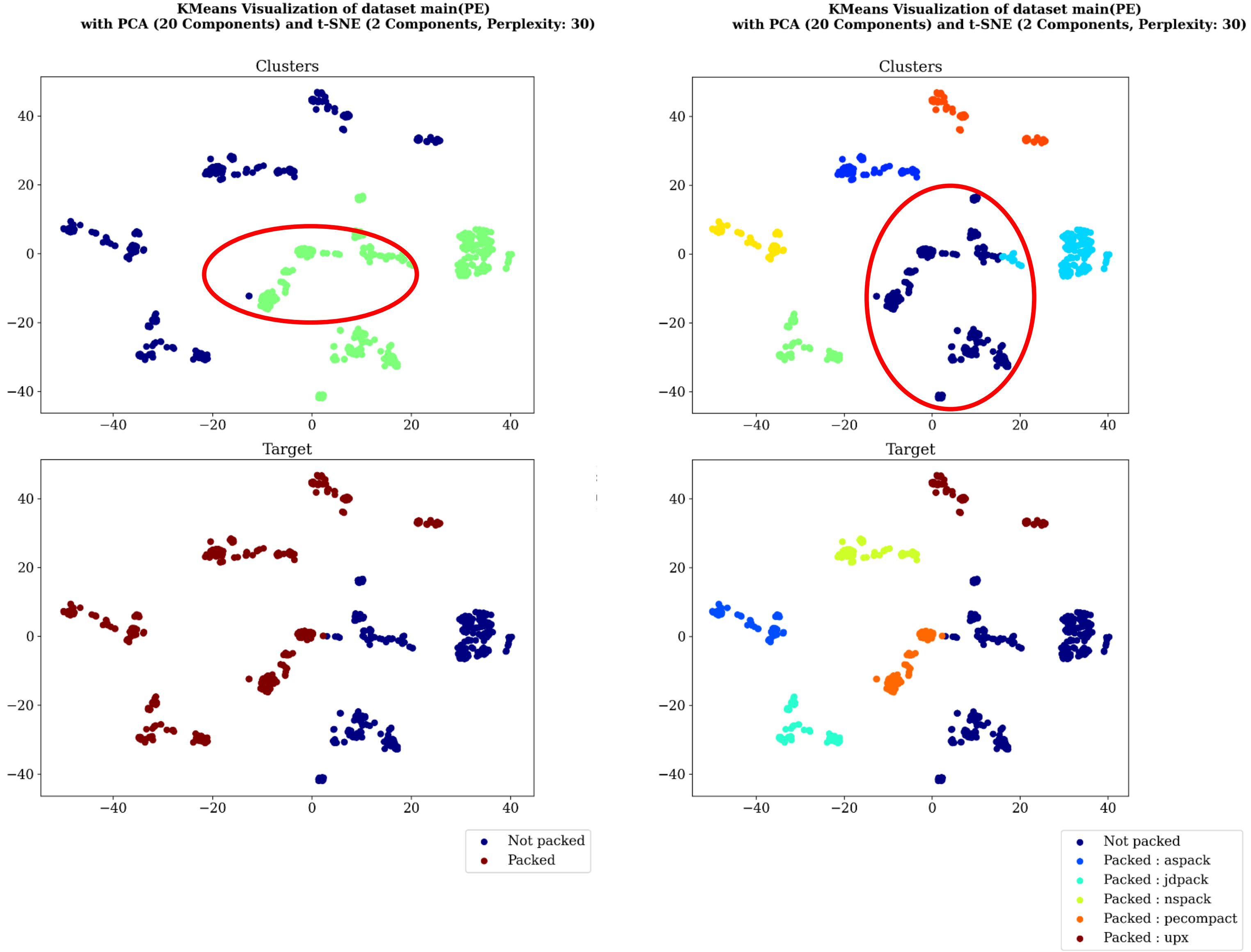
- Multiple compression packers
- Clustering algorithm : K-Means
- Hyperparameter $N_{clusters}$ set to 6 (result of EDA)

Observations

- Binary :
  - Cluster of not packed samples includes PECompact
  - Accuracy < 90%
- Multiclass :
  - Clear clusters for all packers but PECompact, cluster including not packed and PECompact
  - Accuracy < 80%

Try It Now !

```
$ experiment edit algorithms
$ model train main -a kmeans
$ model visualize main_pe_1035_kmeans_f142 --export
        --plot-labels --multiclass
```



Outline

1. Introduction
2. Background
3. Framework
4. Breaking Detectors
5. Visualizing Packing
6. Conclusion

- Contribution
- Future work

Contribution

Toolkit extensions for adversarial & unsupervised learning

- Support for altering binaries based on user-defined alterations
- Support for clustering algorithms
- Support for many more visualizations (altered binaries, impacted features, clustering, …)

Future work

- Combining alterations and optimizing effect on common detectors
- Optimization attack on learning models
- Weaponization of adversarial attacks
- Research of robust features
- More secure learning models

black hat ARSENAL

DECEMBER 6-7, 2023

EXCEL LONDON / UNITED KINGDOM

https://github.com/packing-box

PACKING BOX

awesome Executable Packing

Awesome list gathering our whole bibliography and many other references to documentation, tools, etc.

Dataset of Packed ELF

Ready-to-use dataset of packed and not-packed ELF files

Dataset of Packed PE

Ready-to-use dataset of packed and not-packed PE files from the enriched version of https://github.com/chesvectain/PackingData

bintropy

Entropy-based tool inspired from the study of Lyda et al. in 2007

REsponse tool for Malware INDication

Heuristic-based tool inspired from the study of Han et al. in 2009

PyPACKER DETECT

Operationalized fork of https://github.com/cylance/PyPackerDetect

pe peid

Python fork of the popular tool, PEiD

DSFF

Custom exchange format for datasets (supports conversion to ARFF, CSV, Packing-Box dataset)