ML for threat analysis

By Ashu Sharma

What is Machine Learning?

- “Machine learning is programming computers to optimize a performance criterion using example data or past experience.” Intro to Machine Learning, Alpaydin, 2010
- Examples:
  - Facial recognition
  - Digit recognition
  - Molecular classification

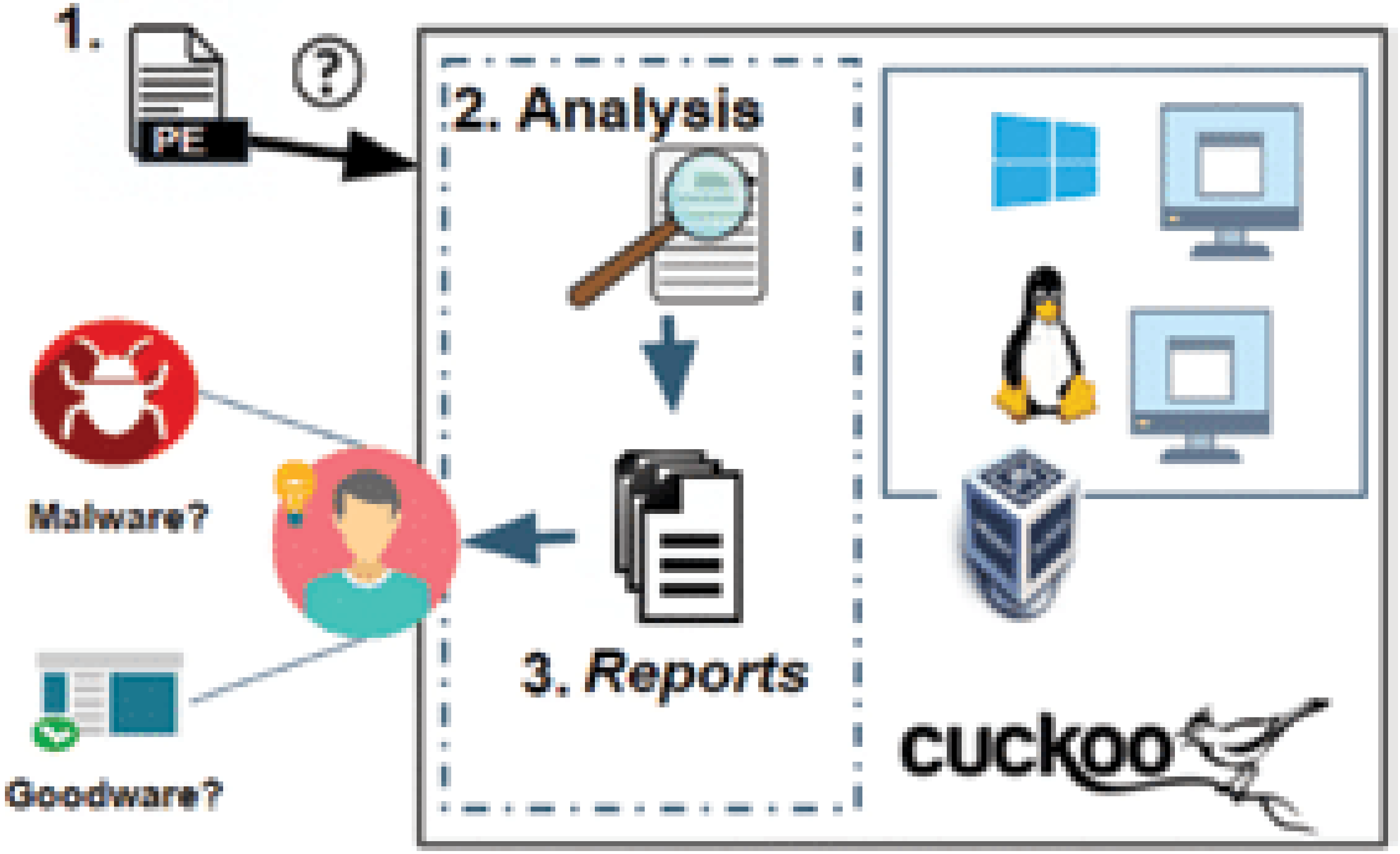
Basic approaches to malware detection

An efficient, robust and scalable malware recognition module is the key component of every cybersecurity product. Malware recognition modules decide if an object is a threat, based on the data they have collected on it. This data may be collected at different phases:

- **– Pre-execution phase** data is anything you can tell about a file without executing it. This may include executable file format descriptions, code descriptions, binary data statistics, text strings and information extracted via code emulation and other similar data.

- **– Post-execution phase** data conveys information about behavior or events caused by process activity in a system.

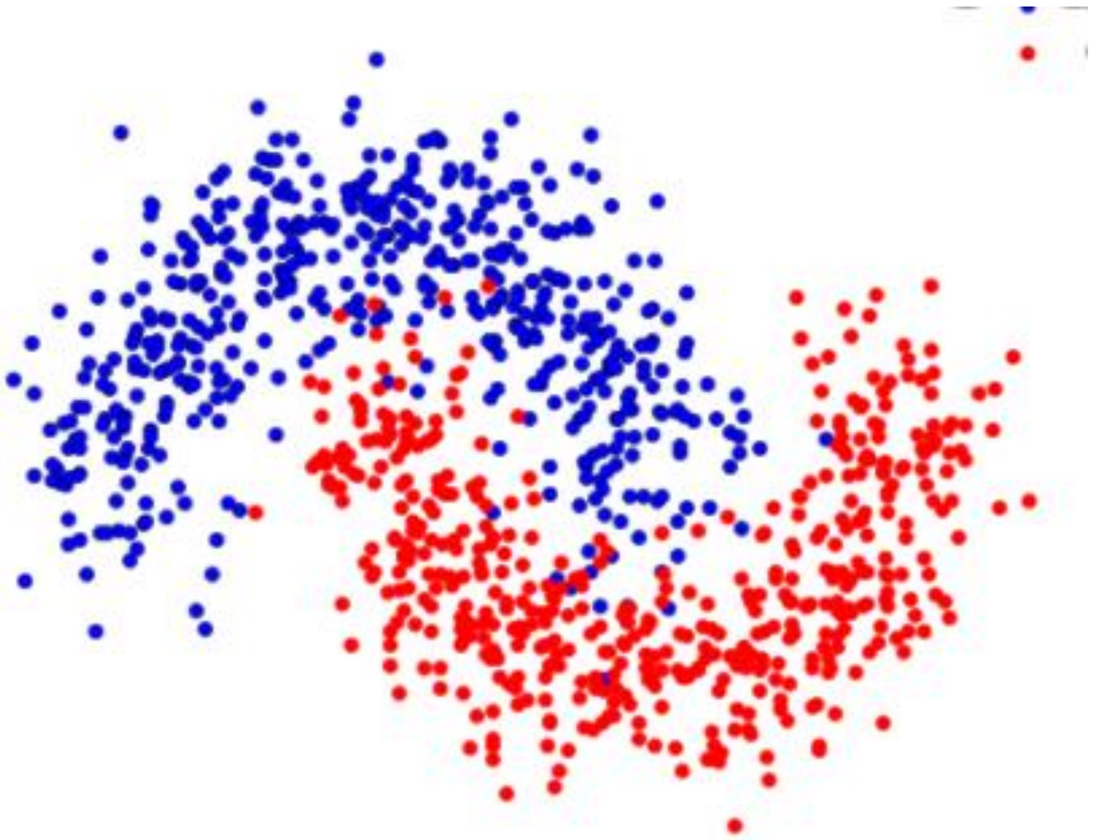
WHY ML IN MALWARE ANALYSIS?



Work Process

Similarity Based

- Unsupervised
- Supervised

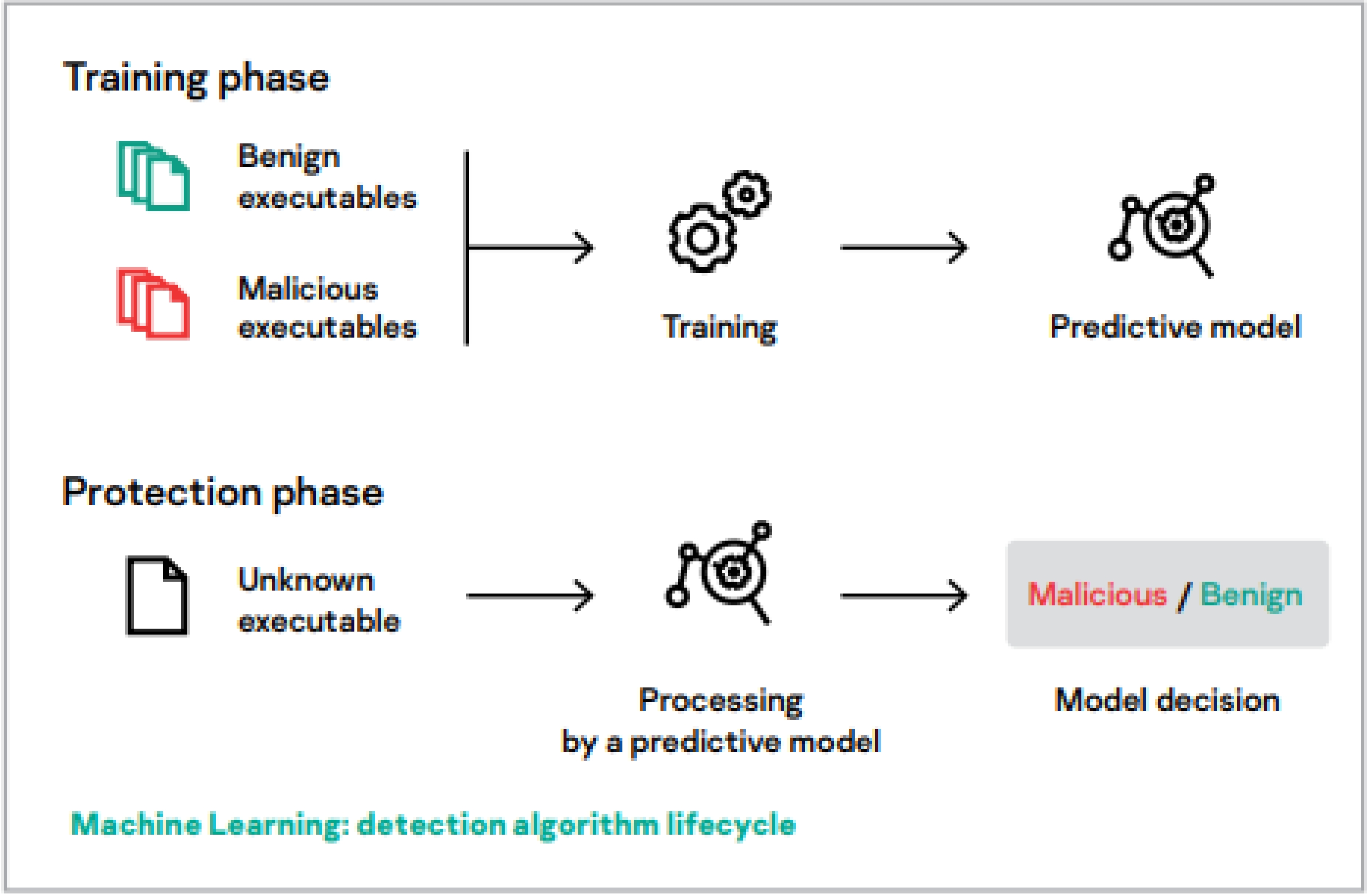


Threat Intel using ML

Example TO DETECT MALWARE BY ML

- Machine Learning stack
  - Data Extraction
  - Data Preprocessing
  - Data reduction/feature selection/Noise reduction
  - Training / generating Classification models
  - Validating / measuring performance of the created models
  - Testing the performance by external or Realtime data
  - Evaluating performance of the models

Non Signature Based: Malware Detection using ML



An example for creating the models for malware identification.

- Downloaded 11088 malware from the malacia-project.
- We collected 4006 benign programs (also verified from virustotal.com) from different Windows desktops.

Obfuscation Techniques

- Register usage exchange
- This method was used by the Win95/RegSwap virus, which was created by the virus writer Vecna and released in 1998.
- **Different generations of the virus will use the same code but with different registers**

a)

```
5A                  pop edx
BF04000000          mov edi,0004h
8BF5                mov esi,ebp
B80C000000          mov eax,000Ch
81C288000000        add edx,0088h
8B1A                mov ebx,[edx]
899C8618110000      mov [esi+eax*4+00001118],ebx
```

b)

```
58                  pop eax
BB04000000          mov ebx,0004h
8BD5                mov edx,ebp
BF0C000000          mov edi,000Ch
81C088000000        add eax,0088h
8B30                mov esi,[eax]
89B4BA18110000      mov [edx+edi*4+00001118],esi
```

Obfuscation Techniques

- Register usage exchange
- This method was used by the Win95/RegSwap virus, which was created by the virus writer Vecna and released in 1998.
- **Different generations of the virus will use the same code but with different registers**

a)

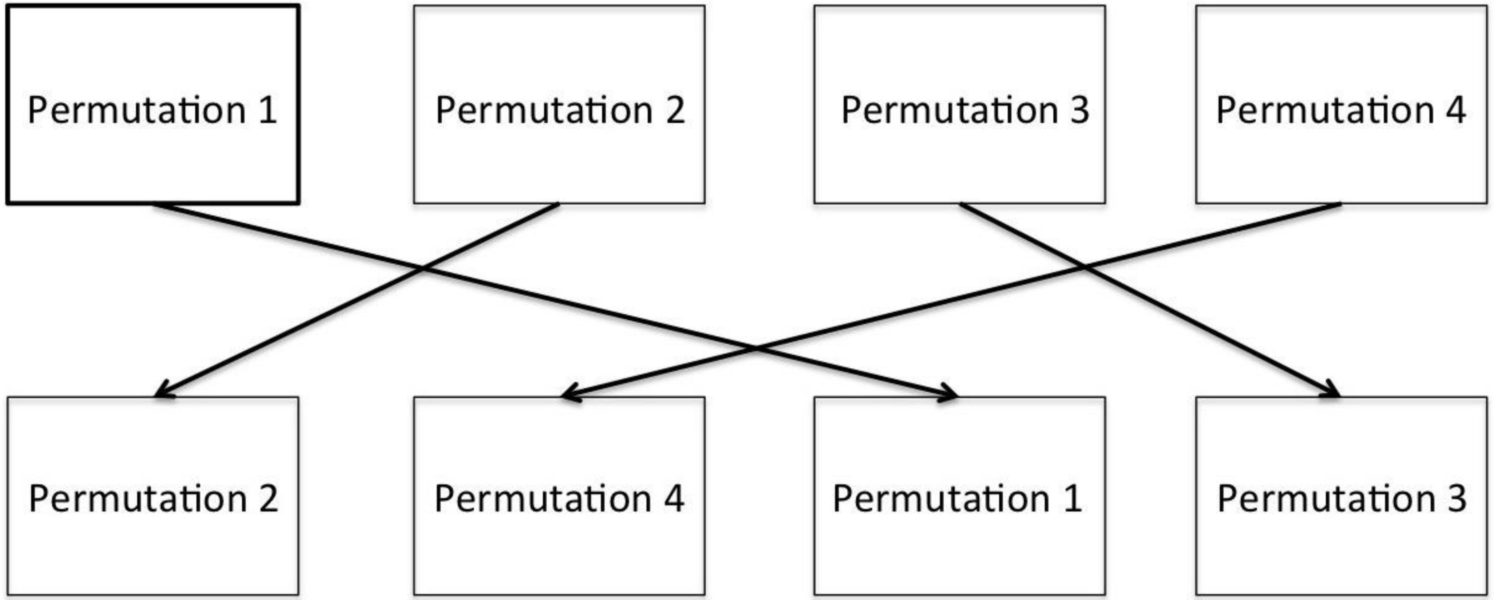
```
5A                pop edx
BF04000000        mov edi,0004h
8BF5              mov esi,ebp
B80C000000        mov eax,000Ch
81C288000000      add edx,0088h
8B1A              mov ebx,[edx]
899C8618110000    mov [esi+eax*4+00001118],ebx
```

b)

```
58                pop eax
BB04000000        mov ebx,0004h
8BD5              mov edx,ebp
BF0C000000        mov edi,000Ch
81C088000000      add eax,0088h
8B30              mov esi,[eax]
89B4BA18110000    mov [edx+edi*4+00001118],esi
```

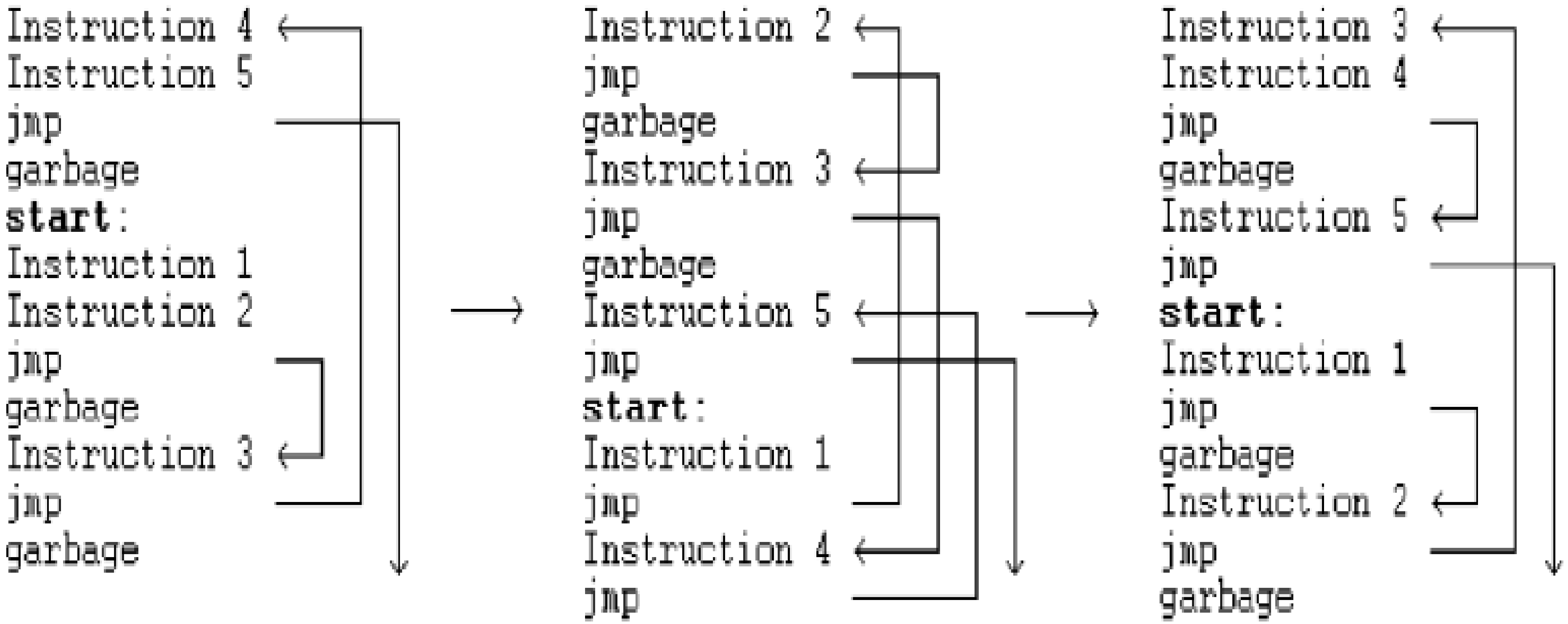
Obfuscation Techniques

- Permutation Techniques



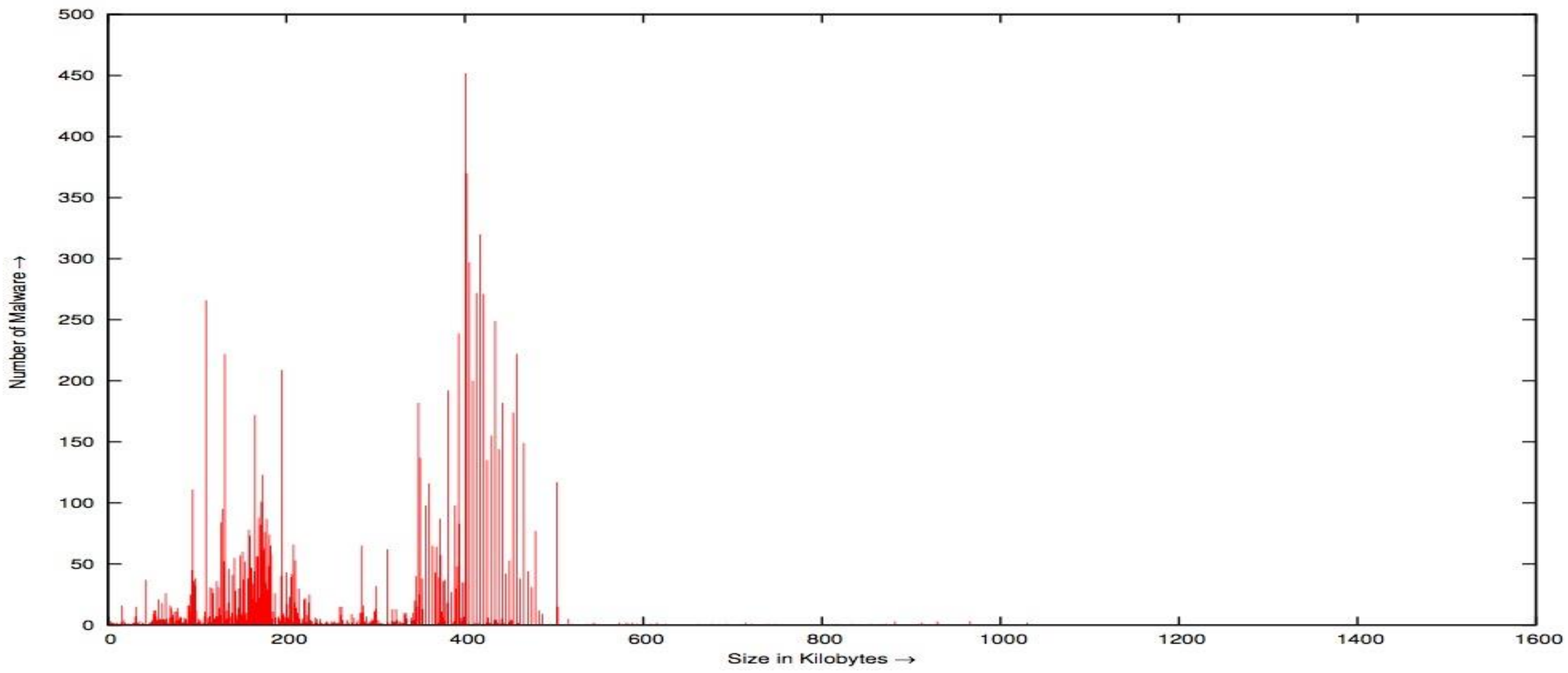
Obfuscation Techniques

- Insertion of Jump Instructions



Data Preprocessing:

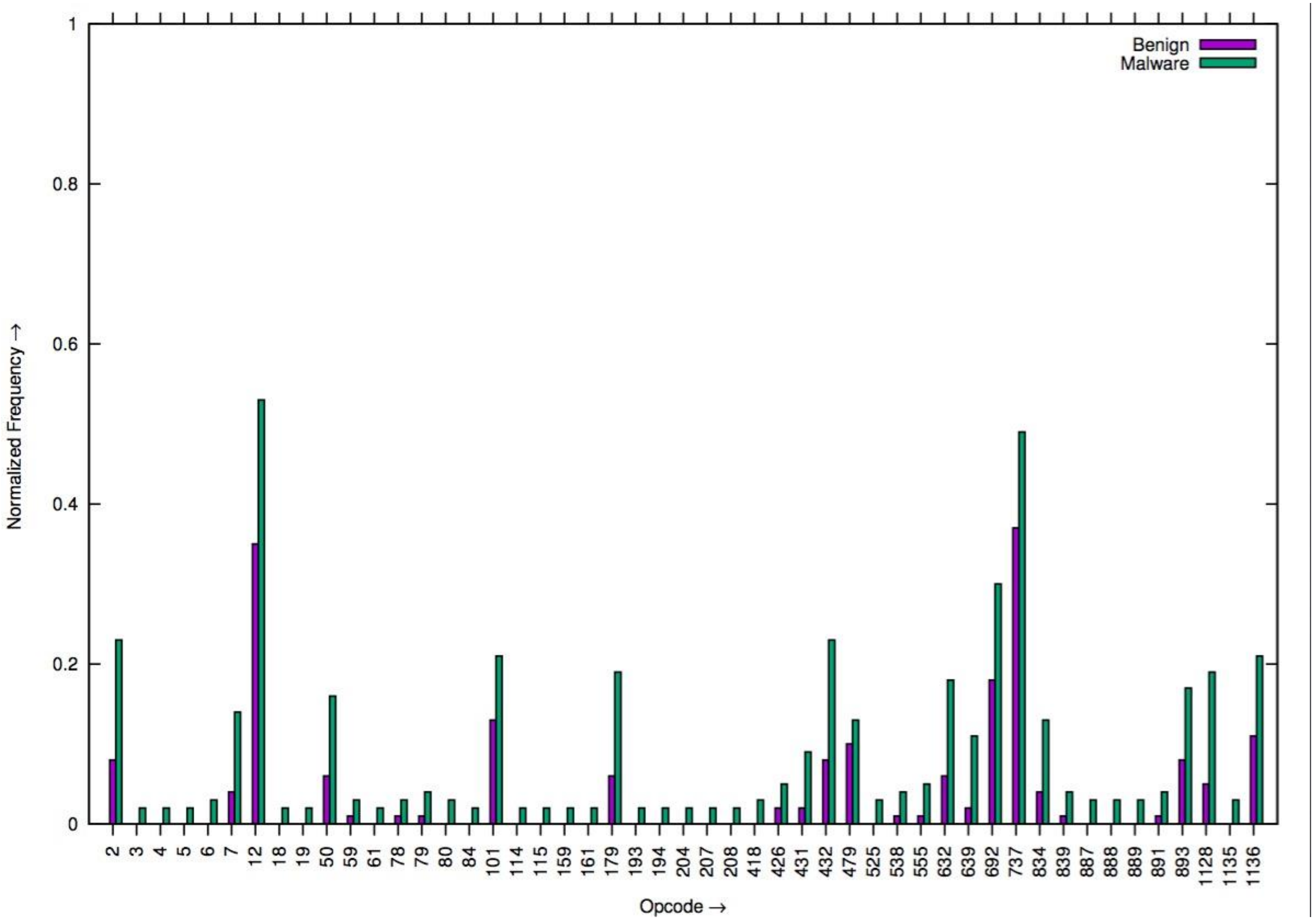
- In the Malicia malware dataset we observed that 97.18% malware are below 500 KB, hence for the analysis we took the data set (both malware and benign executables) which are below 500 KB.



Data Preprocessing

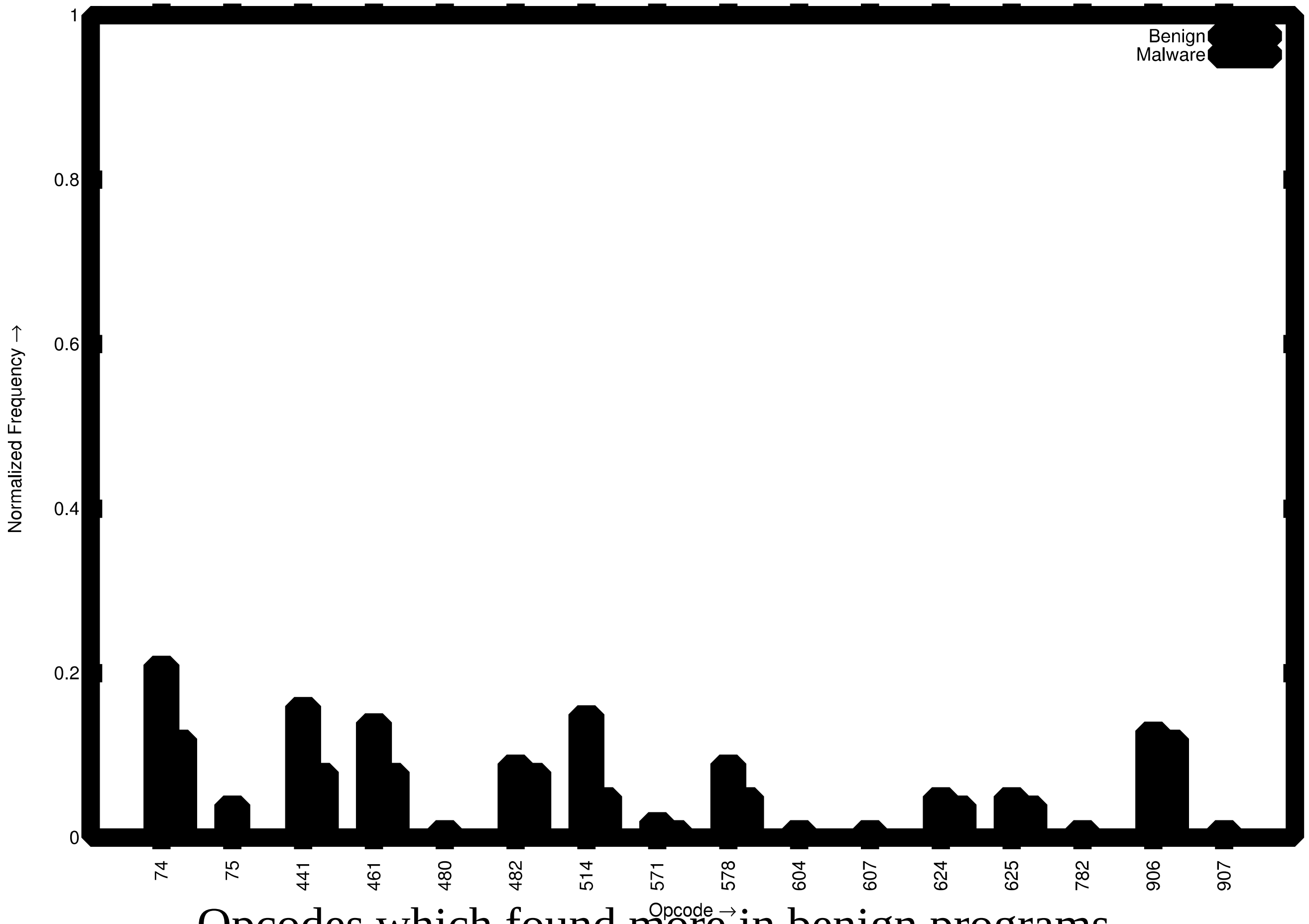
- Converted all the selected executables (10558 malware and 2454 benign) to their assembly codes by objdump utility available in the Linux system.
- Executables are based on 1147 unique opcodes.
- Opcodes are mapped with a fixed integer.

Data Preprocessing



Opcodes which found more in malware

Data Preprocessing



Opcodes which found more in benign programs

Feature Selection

INPUT: Pre-processed data
$\mathbf{N_b}$: Number of benign executables, $\mathbf{N_m}$: Number of malware executables, **n**: Number of features required
OUTPUT: List of features

BEGIN
for all benign data **do**
Add all frequency $\mathbf{f_i}$ of each opcode **o** and Normalize them with respect to $\mathbf{N_b}$

$$F_b(o_j) = (\sum f_i(o_j))/N_b$$

end for
for all malware data **do**
Add all frequency $\mathbf{f_i}$ of each opcode **o** and Normalize them with respect to $\mathbf{N_m}$

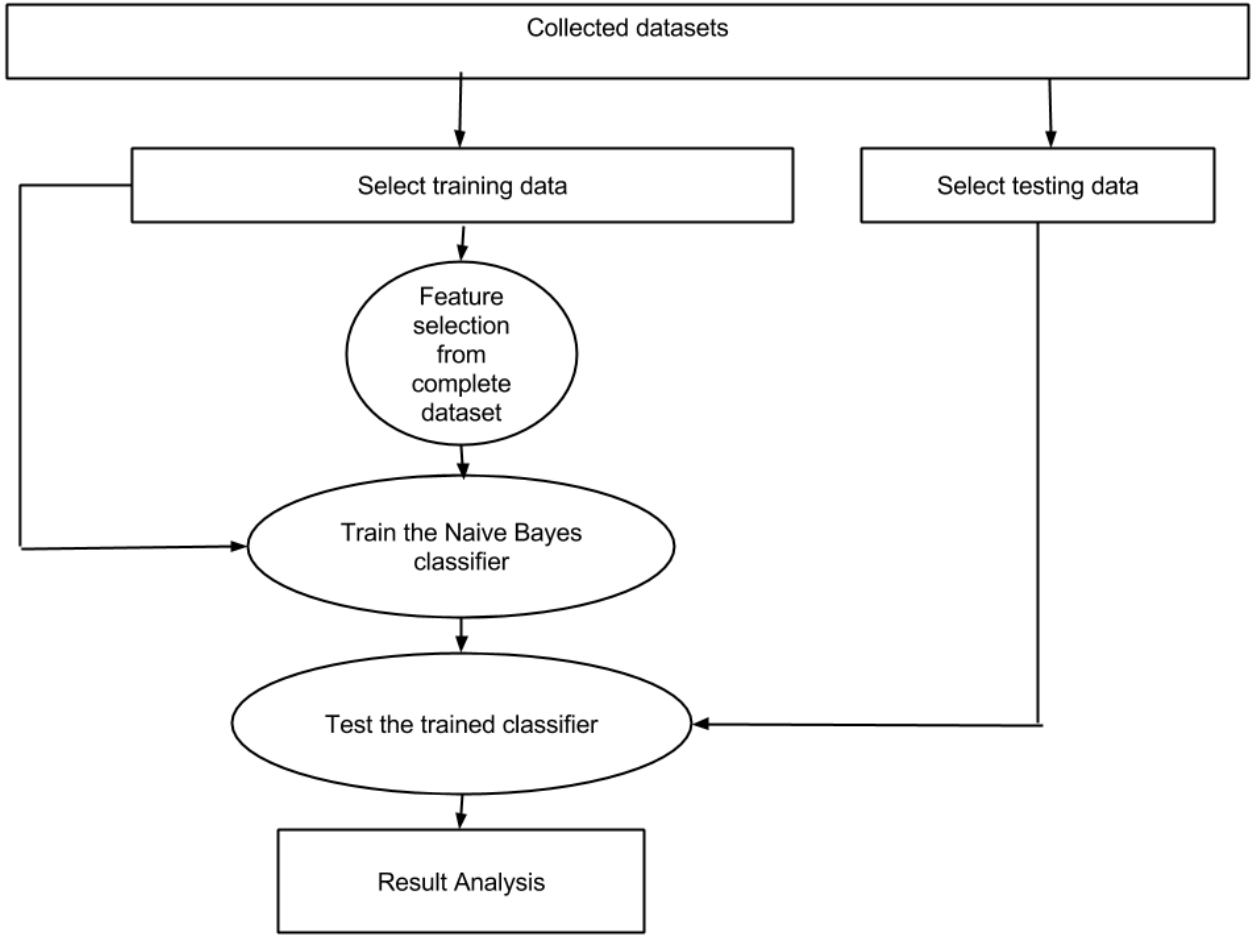
$$F_m(o_j) = (\sum f_i(o_j))/N_m$$

end for
for all opcode $\mathbf{o_j}$ **do**
Find the difference of each opcode normalized frequency $\mathbf{D(o_j)}$.

$$D(o_j) = |F_b(o_j) - F_m(o_j)|$$

end for
return **n** number of opcodes with highest **D(o)**.

Regular Method: Flow Chart



Data Preprocessing: Generation of Malware

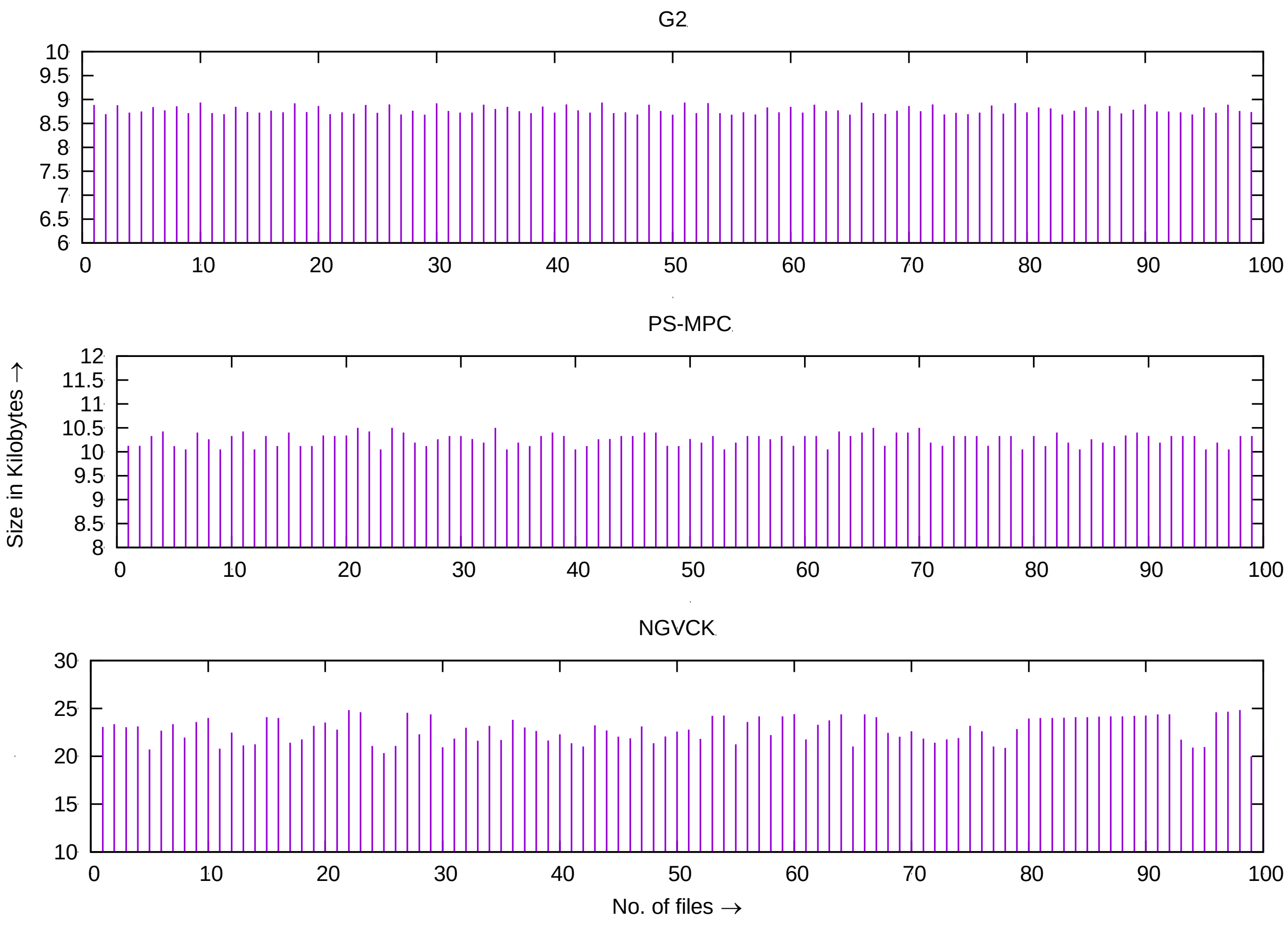
- Malware generator kit used to generate malware

  - G2 (Generation second malware kit).

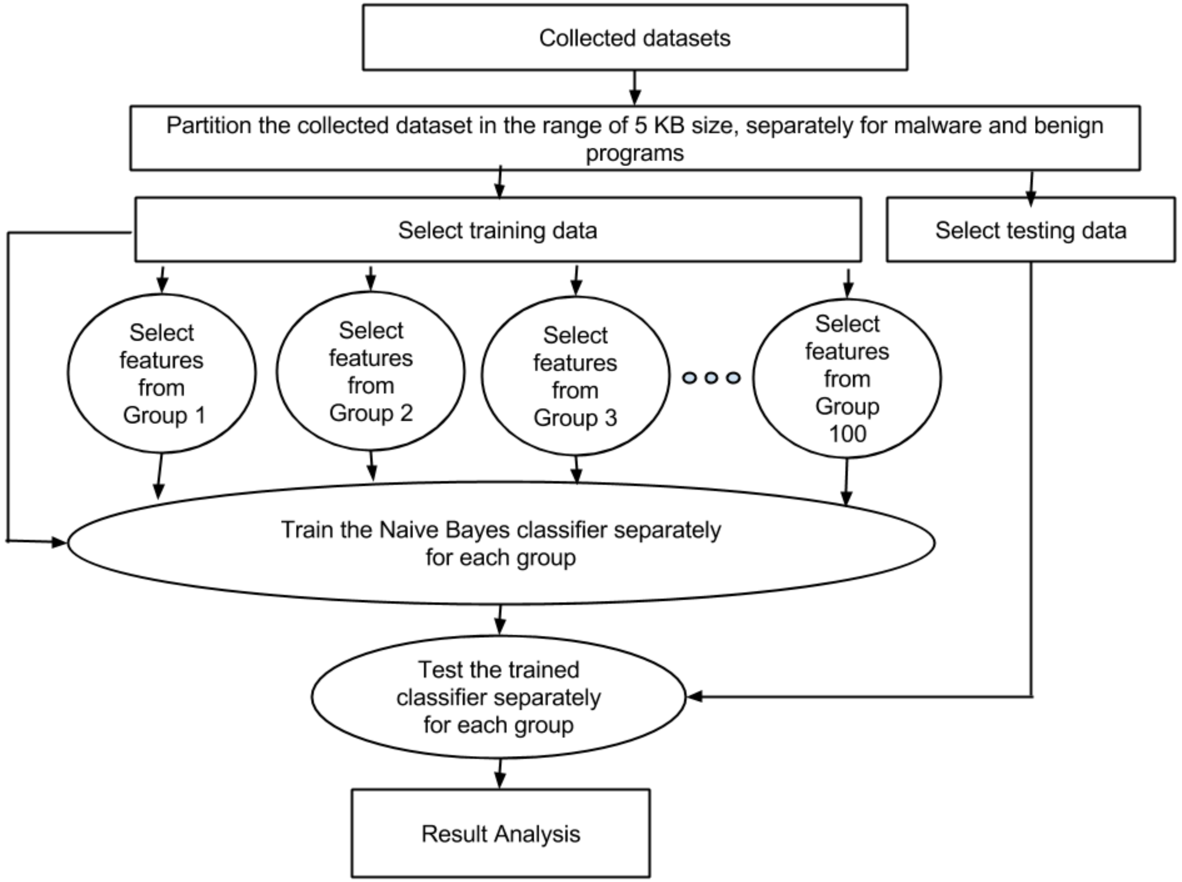
  - PS-MPC (Phalcon-Skism Mass-Produced Code Generator)

  - NGVCK (Next Generation Virus Creation Kit)

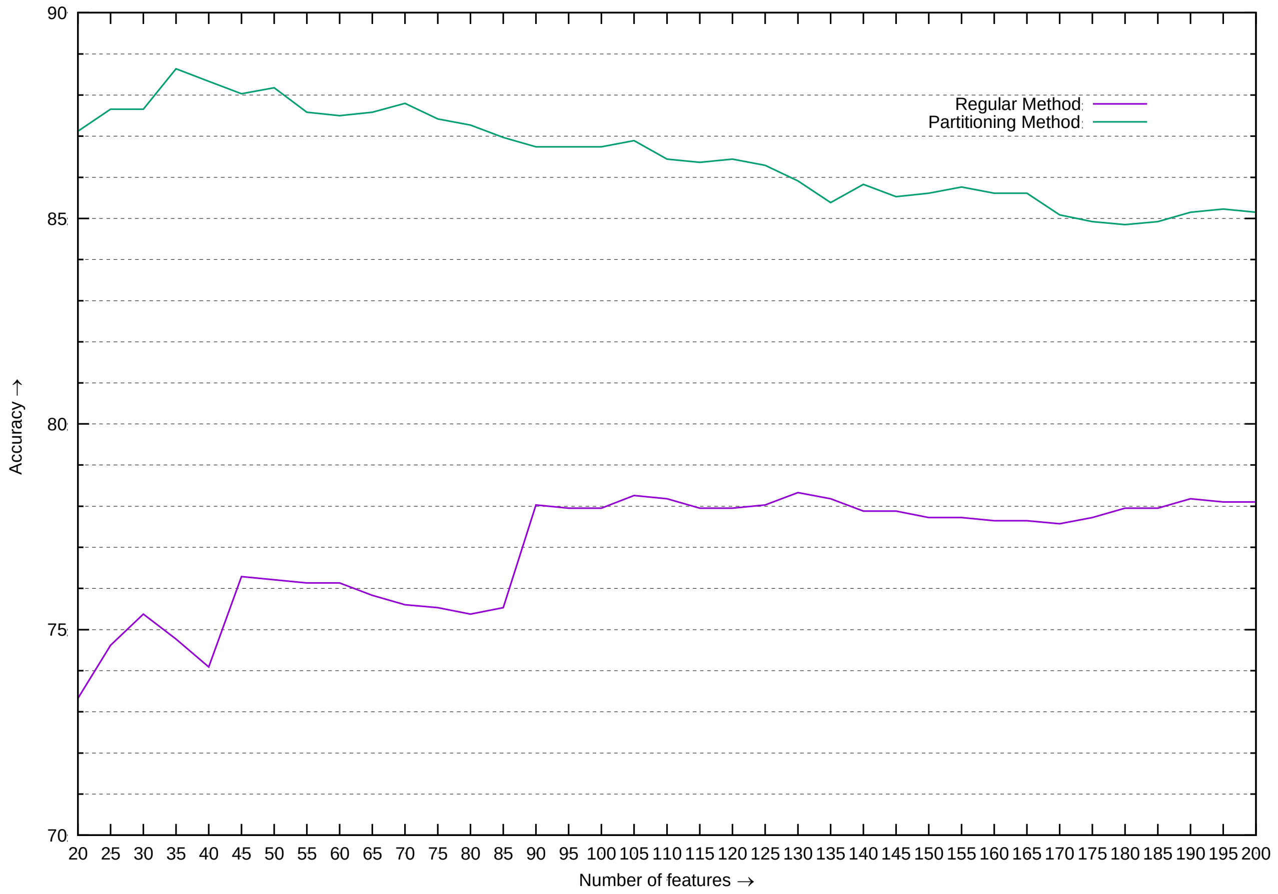
Data Preprocessing: Size of the Generated Malware



Group-wise Classification: Flow Chart



Detection Accuracy

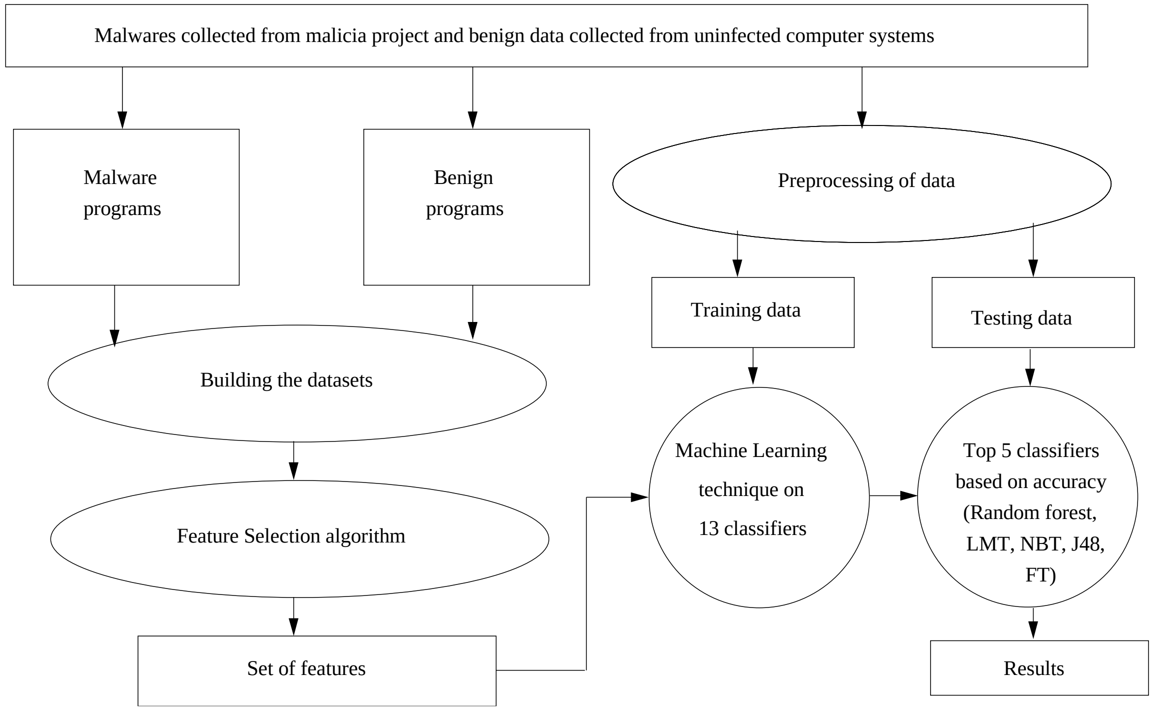


Sharma, A., Sahay, S. K.

And Kumar A., **Springer,**

Advanced Computing and Communication Technologies, 2016,

421-431



Case Studies

Group-wise classification to improve the detection accuracy of Android malicious apps, *International Journal of Network Security* (in press), 2018**.**

An investigation of the classifiers to detect android malicious apps, **Springer,** *Information and Communication Technology*, 207-217, 2017**.**

An effective approach for classification of advanced malware with high accuracy, *International Journal of Security and Its Applications*, 10 (4): 249-266, 2016**.**

Grouping the Executables to Detect Malware with High Accuracy, **Elsevier,** *Procedia Computer Science* 78: 667-674, 2016**.**

Improving the detection accuracy of unknown malware by partitioning the executables in groups, **Springer,** *Advanced Computing and Communication Technologies*, 2016, 421-431**.**