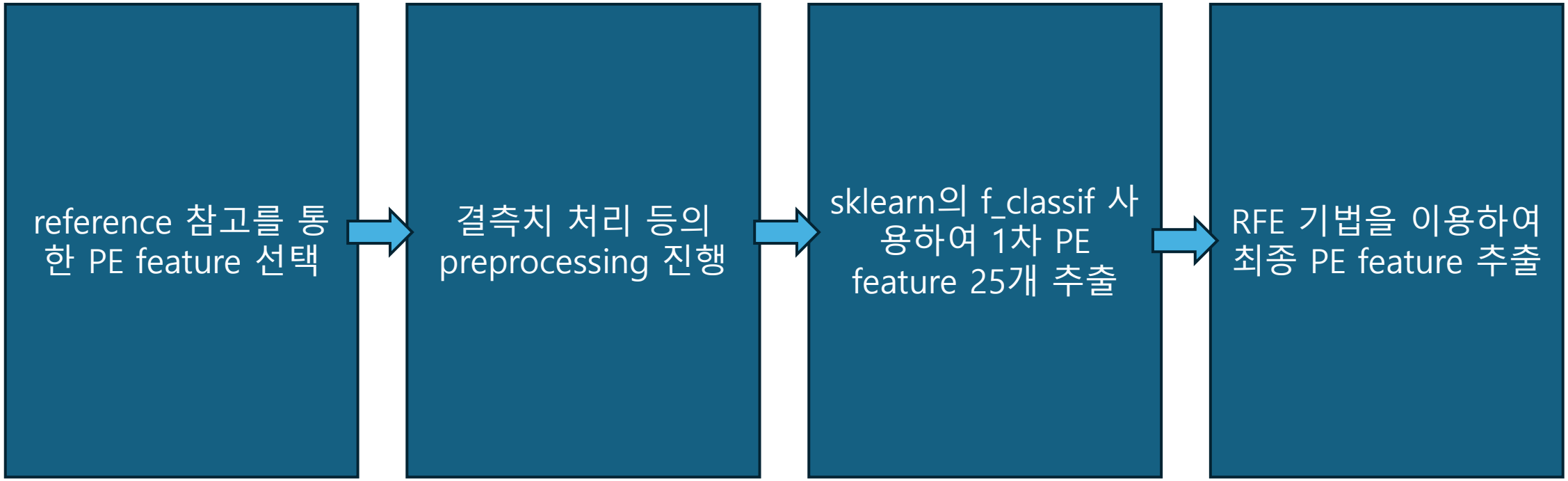
빅데이터응용보안 중간과제

2020250473 보건정책관리학부 이서영

Workflow


reference 참고를 통한 PE feature 선택
결측치 처리 등의 preprocessing 진행
sklearn의 f_classif 사용하여 1차 PE feature 25개 추출
RFE 기법을 이용하여 최종 PE feature 추출

데이터셋 EDA

데이터셋 EDA

Data overview

trojan – 2000

backdoor – 2000

worm – 1200

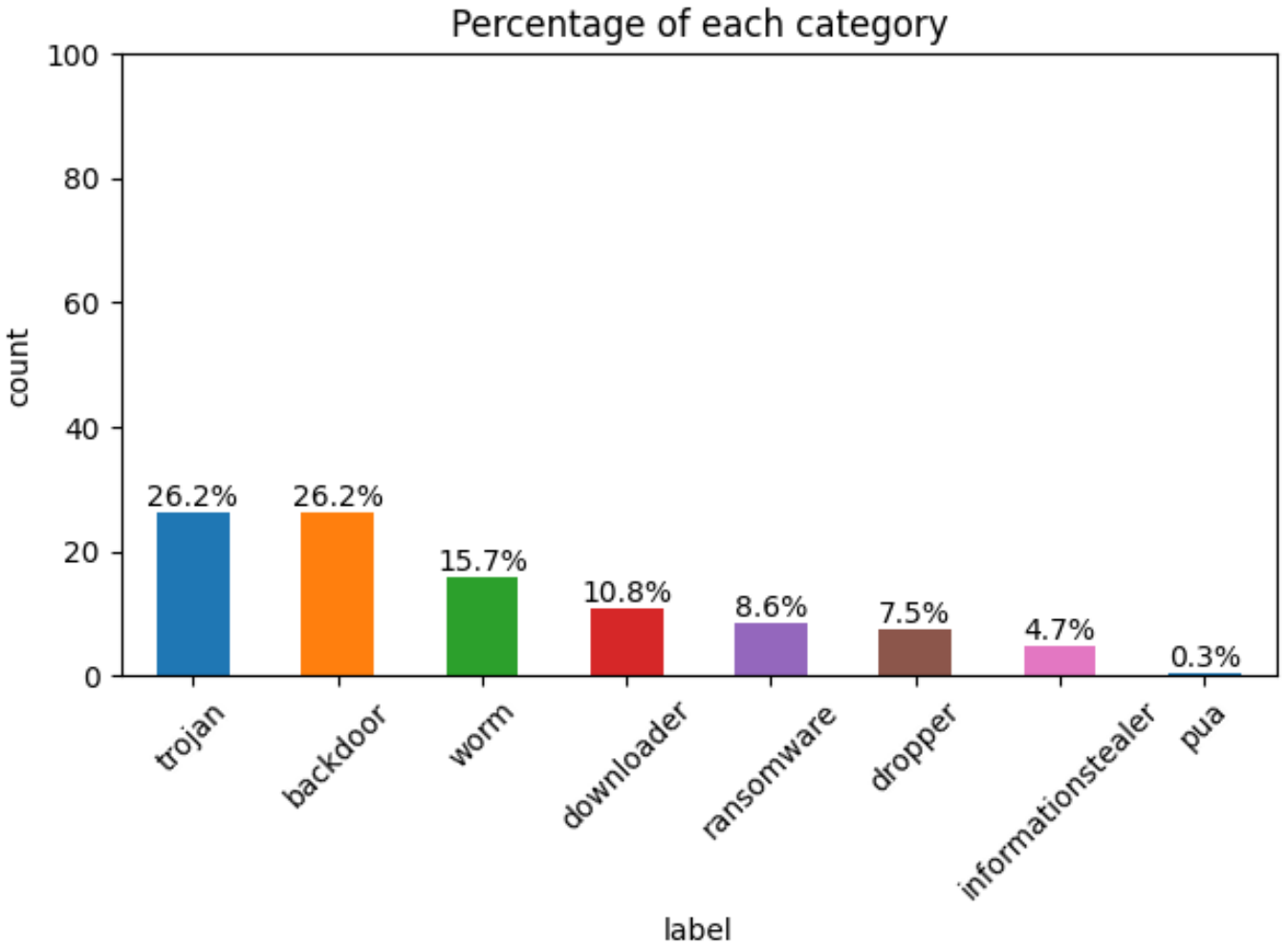
downloader – 824

ransomware – 656

dropper – 572

informationstealer – 358

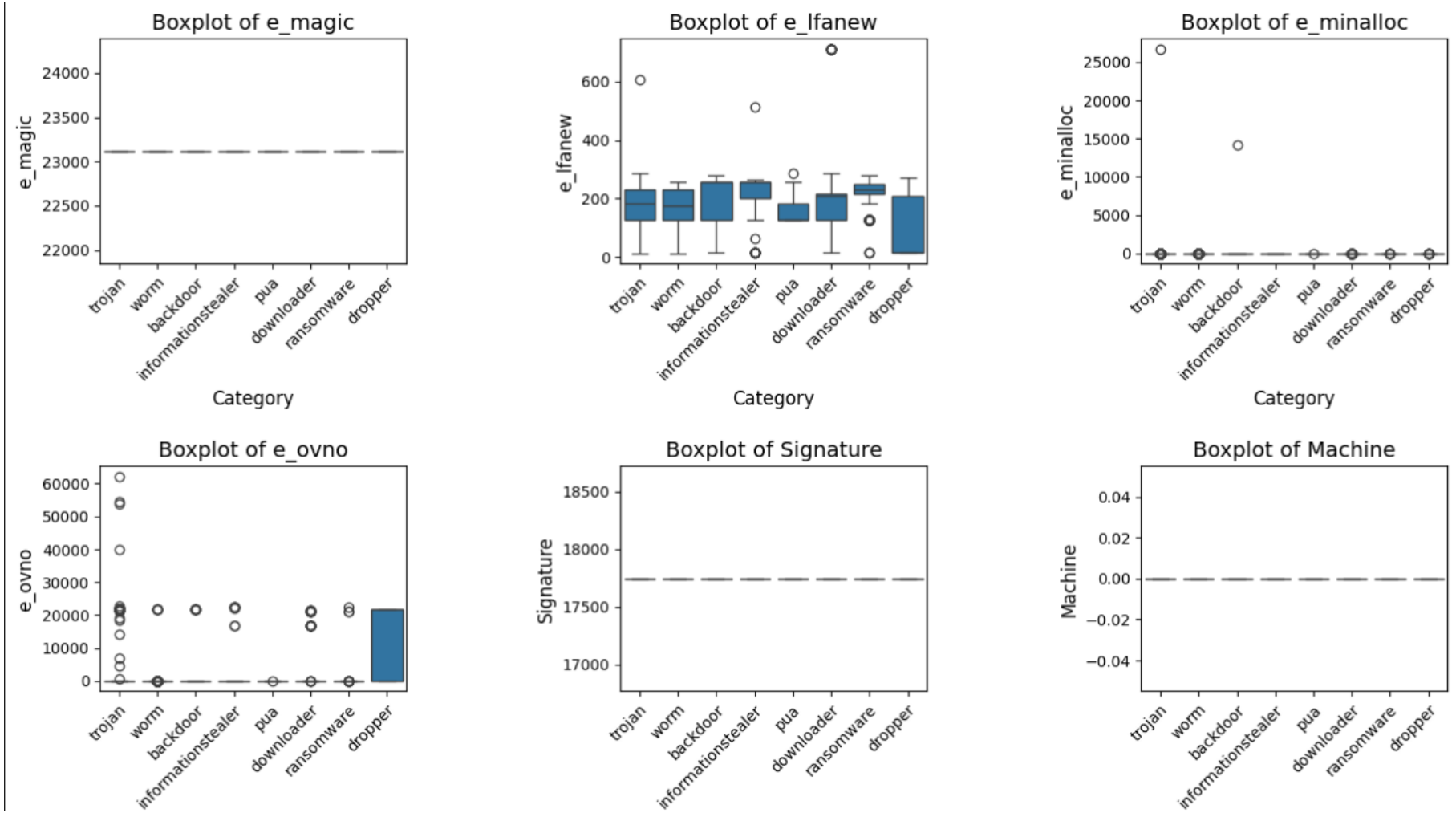
pua - 23



데이터셋 EDA

Box-plot of features

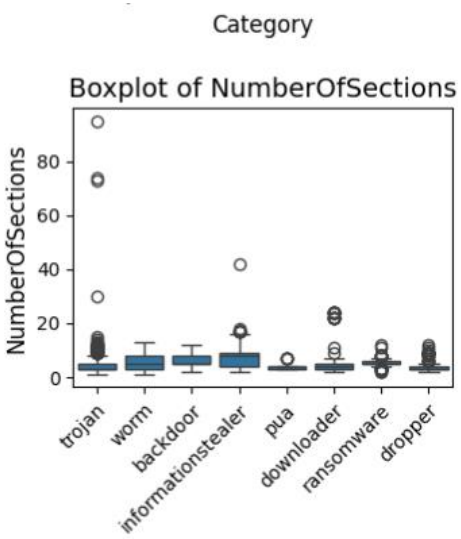
- 각 카테고리 별로 PE feature distribution을 확인
- 악성코드 클래스에 따라 차이를 보이는 PE feature : e_lfanew, e_ovno

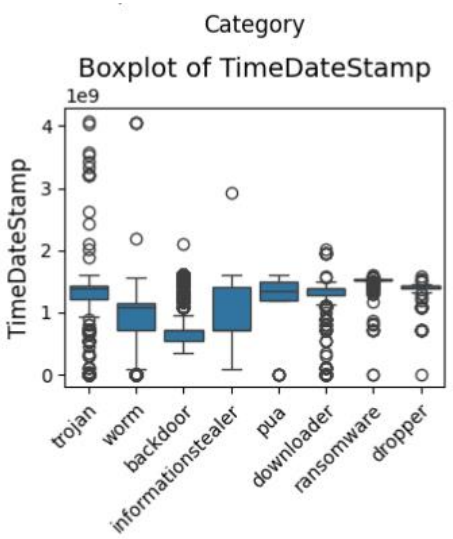


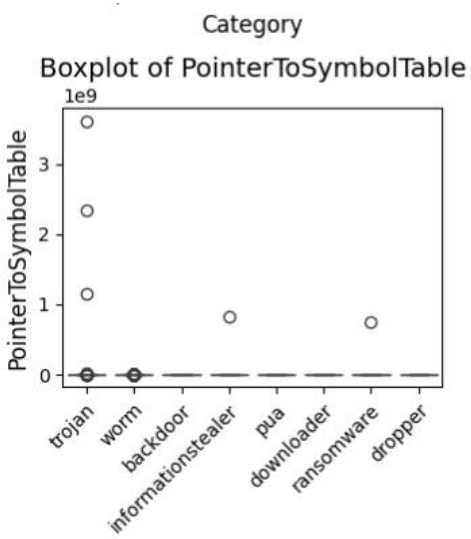
데이터셋 EDA

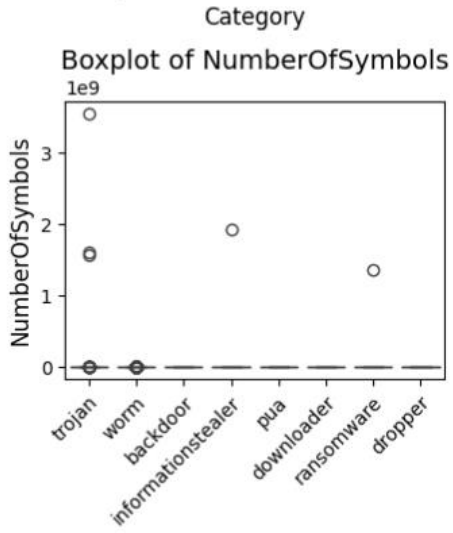
Box-plot of features

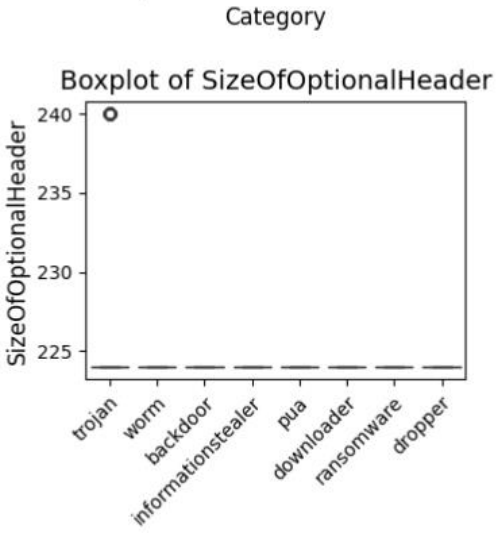
- 각 카테고리 별로 PE feature distribution을 확인
- 악성코드 클래스에 따라 차이를 보이는 PE feature : TimeDateStamp, Characteristics

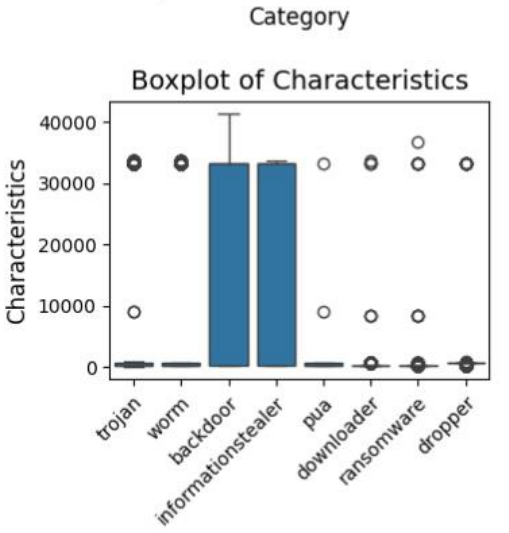








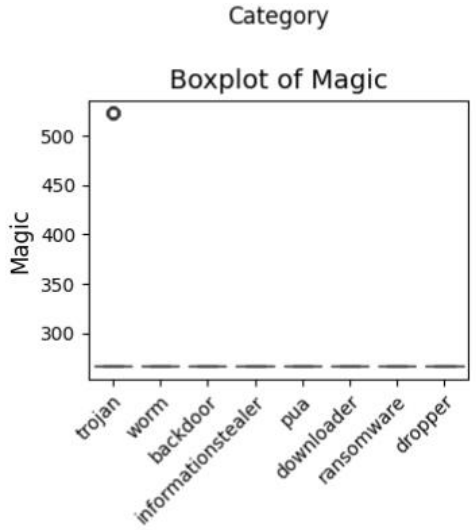


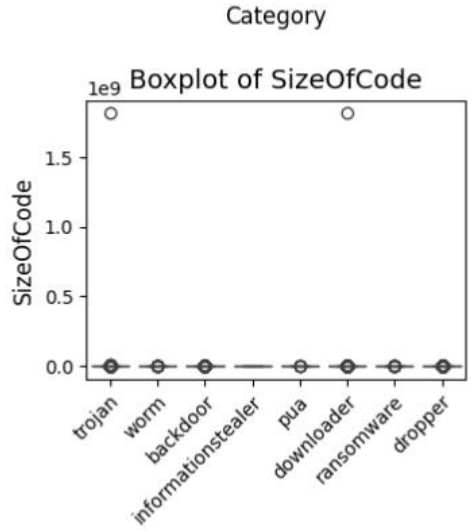


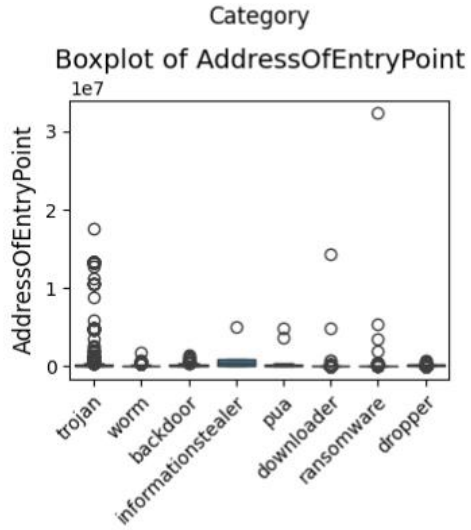
데이터셋 EDA

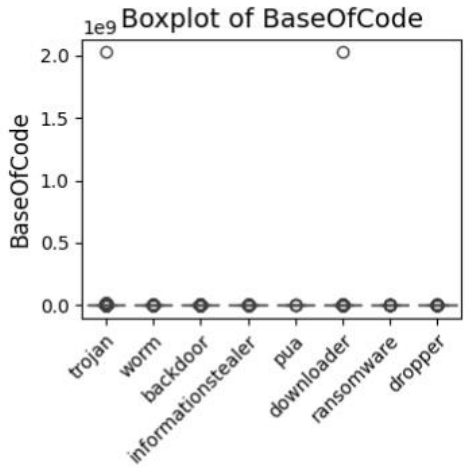
Box-plot of features

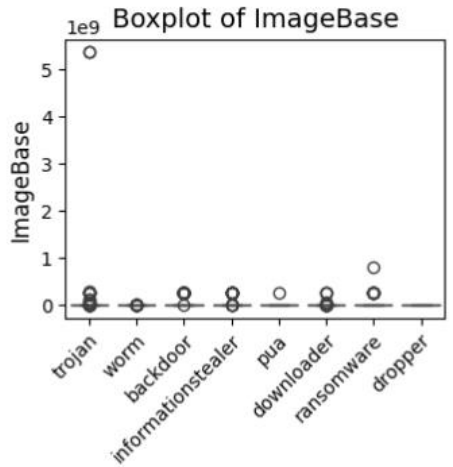
- 각 카테고리 별로 PE feature distribution을 확인
- 악성코드 클래스에 따라 차이를 보이는 PE feature : AddressOfEntryPoint

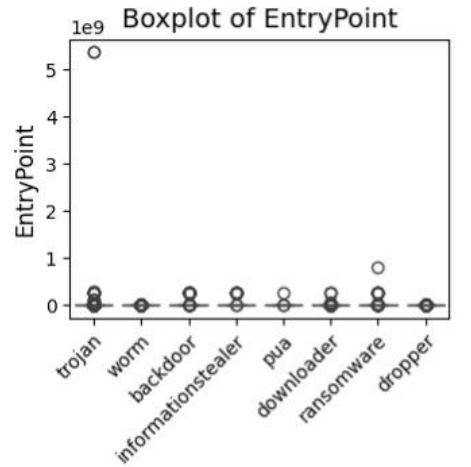








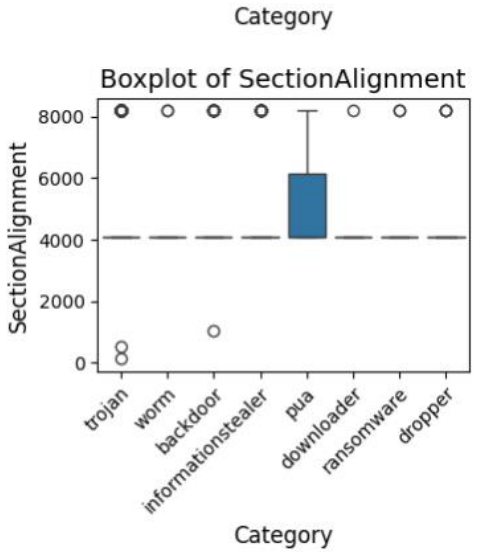


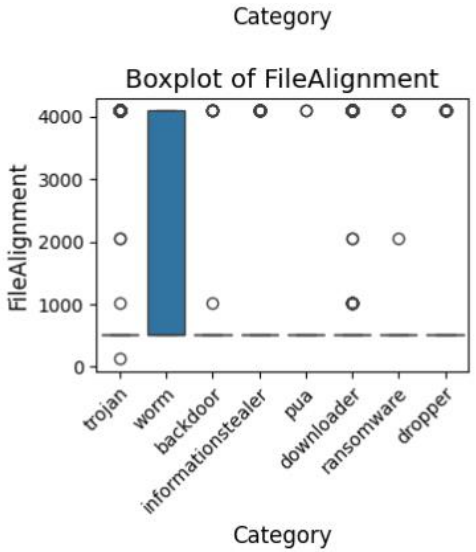


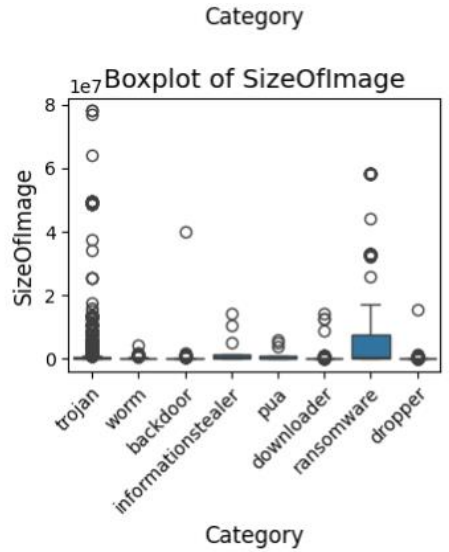
데이터셋 EDA

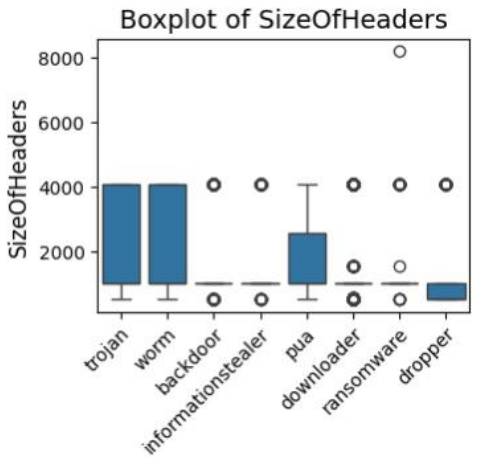
Box-plot of features

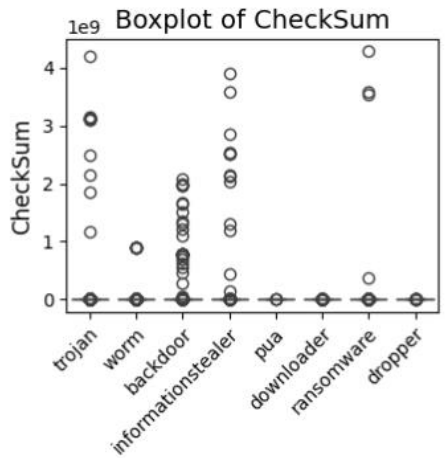
- 각 카테고리 별로 PE feature distribution을 확인
- 악성코드 클래스에 따라 차이를 보이는 PE feature : SizeOfImage, SizeOfHeaders

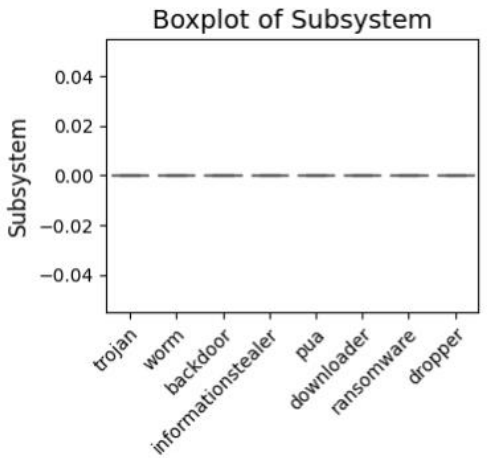






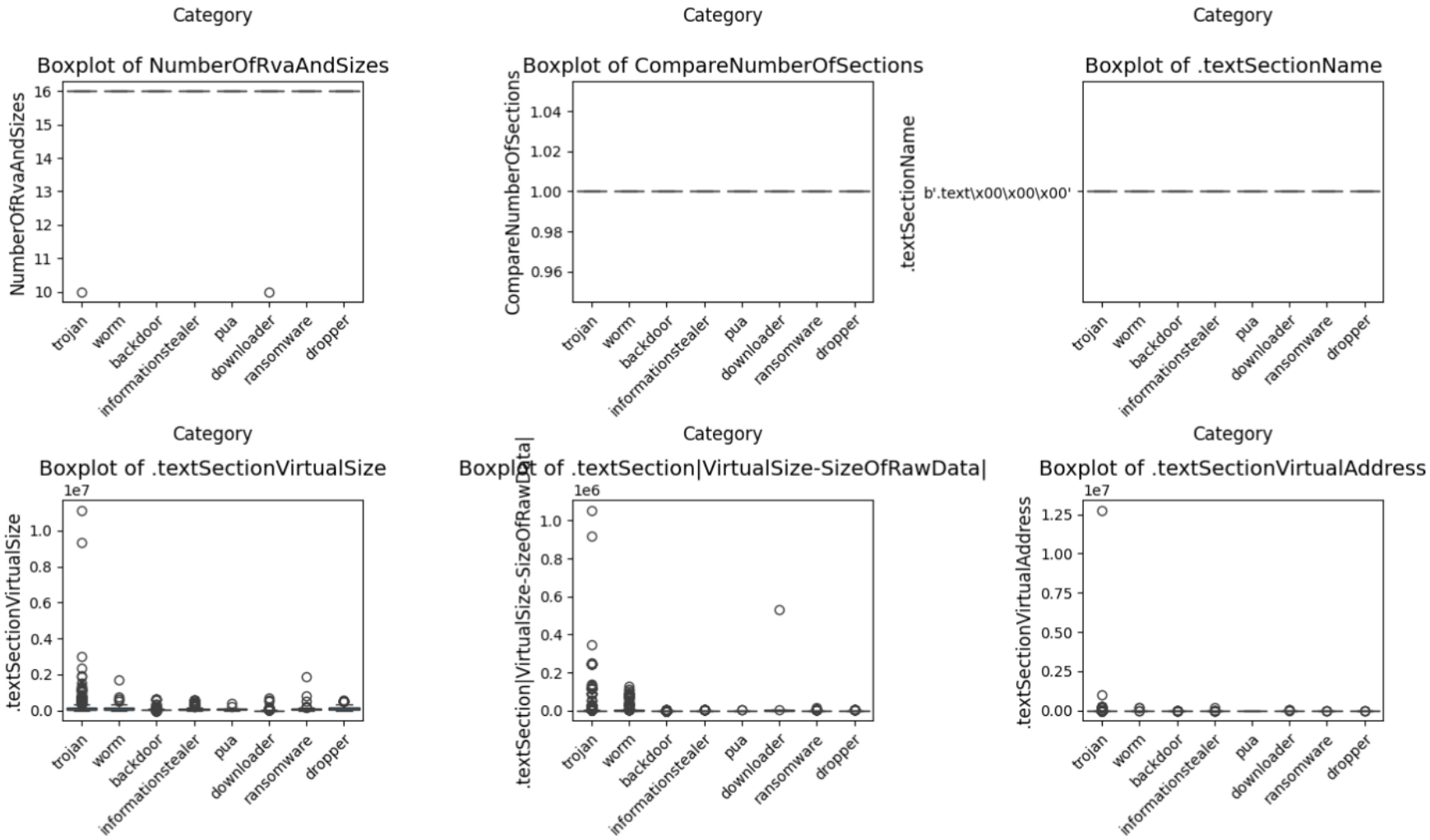






데이터셋 EDA

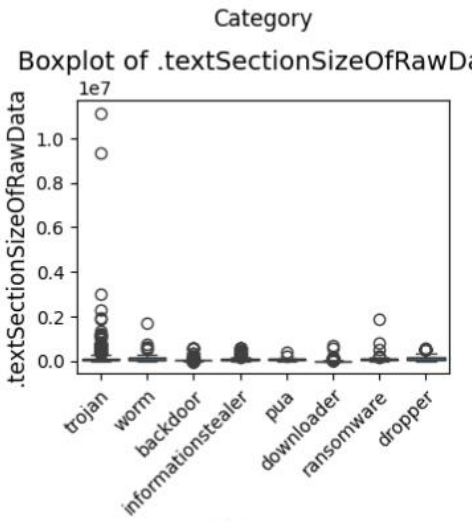
Box-plot of features
- 각 카테고리 별로 PE feature distribution을 확인

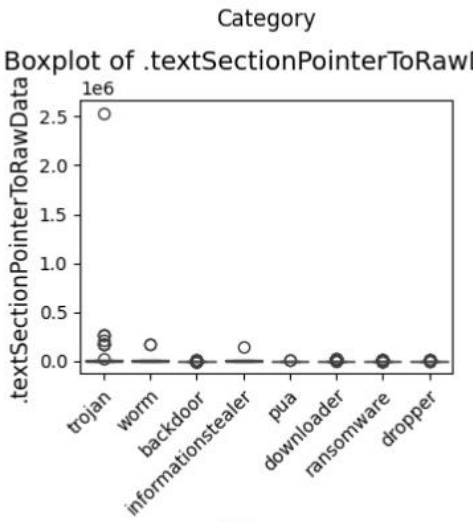


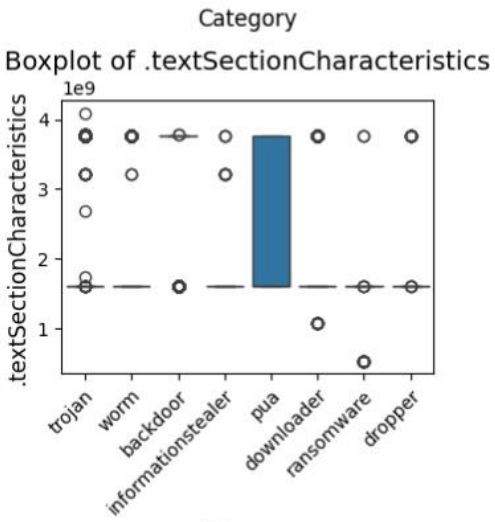
데이터셋 EDA

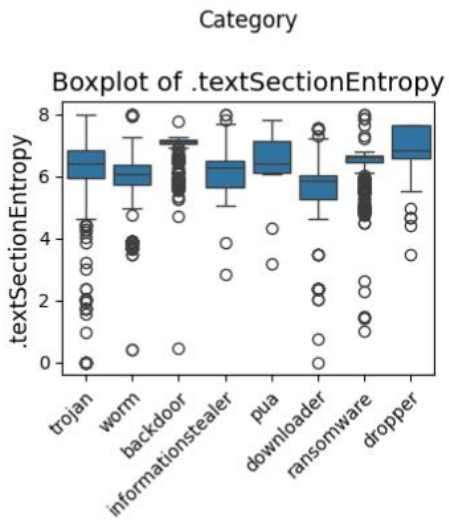
Box-plot of features

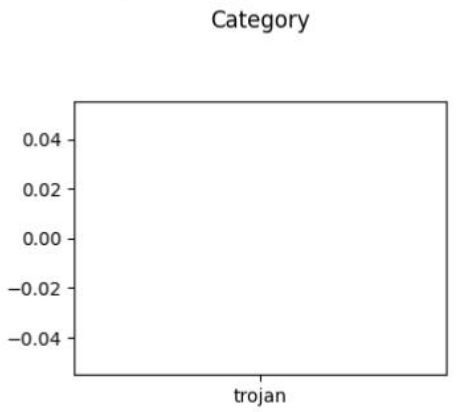
- 각 카테고리 별로 PE feature distribution을 확인
- 악성코드 클래스에 따라 차이를 보이는 PE feature : .textSectionEntropy

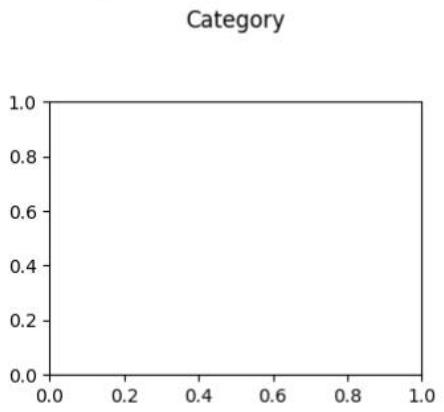












Reference 참고를 통한 PE feature 선택

PE feature selection & description

필드 값	전처리 방법	필드 값에 대한 설명 [7]	reference
e_magic	value	The e_magic field (a WORD) needs to be set to the value 0x5A4D. There's a #define for this value, named IMAGE_DOS_SIGNATURE. In ASCII representation, 0x5A4D is MZ, the initials of Mark Zbikowski, one of the original architects of MS-DOS.	[4]
a_ifanew	value	contains the file offset of the PE header	[4], [6]
e_minalloc	value		[6]
e_ovno	value		[6]
Signature	value		[4]
Machine	value	The target CPU for this executable. Common values are: IMAGE_FILE_MACHINE_I386 0x014c // Intel 386 IMAGE_FILE_MACHINE_IA64 0x0200 // Intel 64	[1], [2], [3]
NumberOfSections	value	Indicates how many sections are in the section table. The section table immediately follows the IMAGE_NT_HEADERS.	[1], [3]
TimeDataStamp	value	Indicates the time when the file was created. This value is the number of seconds since January 1, 1970, Greenwich Mean Time (GMT). This value is a more accurate indicator of when the file was created than is the file system date/time	[1], [2], [3]

[7] "An In-Depth Look into the Win32 Portable Executable File Format - Part 1" , Delphibasics, last modified Mar 15, 2010, accessed April 28, 2024, https://www.delphibasics.info/home/delphibasicsarticles/anin-depthlookintothewin32portableexecutablefileformat-part1

PE feature selection & description

필드 값	전처리 방법	필드 값에 대한 설명 [7]	reference
NumberOfSymbols	value	Number of symbols in the COFF symbol table, if present. COFF symbols are a fixed size structure, and this field is needed to find the end of the COFF symbols. Immediately following the COFF symbols is a string table used to hold longer symbol names.	[6]
SizeOptionalHeader	value	The size of the optional data that follows the IMAGE_FILE_HEADER.	[1], [3]
Characteristics	value	A set of bit flags indicating attributes of the file.	[1], [3], [6]
Magic	value	A signature WORD, identifying what type of header this is.	[1], [3], [4]
SizeOfCode	value	The combined total size of all sections with the IMAGE_SCN_CNT_CODE attribute.	[1], [3], [6]
AddressOfEntryPoint	value	The RVA of the first code byte in the file that will be executed. For DLLs, this entrypoint is called during process initialization and shutdown and during thread creations/destructions. This field can be set to 0 in DLLs, and none of the previous notifications will be received. The linker /NOENTRY switch sets this field to 0.	[1], [3], [4]
BaseOfCode	value	The RVA of the first byte of code when loaded in memory.	[1], [3], [6]
Imagebase	value	The preferred load address of this file in memory. The loader attempts to load the PE file at this address if possible (that is, if nothing else currently occupies that memory, it's aligned properly and at a legal address, and so on). If the executable loads at this address, the loader can skip the step of applying base relocations (described in Part 2 of this article). For EXEs, the default ImageBase is 0x400000. For DLLs, it's 0x10000000.	[1], [3], [6]

[7] "An In-Depth Look into the Win32 Portable Executable File Format - Part 1" , Delphibasics, last modified Mar 15, 2010, accessed April 28, 2024, https://www.delphibasics.info/home/delphibasicsarticles/anin-depthlookintothewin32portableexecutablefileformat-part1

PE feature selection & description

필드 값	전처리 방법	필드 값에 대한 설명 [7]	reference
SectionAlignment	value	The alignment of sections when loaded into memory. The alignment must be greater or equal to the file alignment field (mentioned next). The default alignment is the page size of the target CPU.	[1], [3], [6]
FileAlignment	value	The alignment of sections within the PE file. For x86 executables, this value is usually either 0x200 or 0x1000. The default has changed with different versions of the Microsoft linker.	[1], [3], [6]
SizeOfImage	value	SizeOfImage contains the RVA that would be assigned to the section following the last section if it existed. This is effectively the amount of memory that the system needs to reserve when loading this file into memory. This field must be a multiple of the section alignment.	[1], [3]
SizeOfHeaders	value	The combined size of the MS-DOS header, PE headers, and section table. All of these items will occur before any code or data sections in the PE file. The value of this field is rounded up to a multiple of the file alignment.	[1], [3], 6]
Subsystem	value	An enum value indicating what subsystem (user interface type) the executable expects. This field is only important for EXEs.	[1], [3], [6]
NumberOfRvaAndSizes	value	At the end of the IMAGE_NT_HEADERS structure is an array of IMAGE_DATA_DIRECTORY structures. This field contains the number of entries in the array. This field has been 16 since the earliest releases of Windows NT.	[1], [3]

[7] "An In-Depth Look into the Win32 Portable Executable File Format - Part 1" , Delphibasics, last modified Mar 15, 2010, accessed April 28, 2024, https://www.delphibasics.info/home/delphibasicsarticles/anin-depthlookintothewin32portableexecutablefileformat-part1

PE feature selection & description

필드 값	전처리 방법	필드 값에 대한 설명 [7]	Reference
VirtualSize	.text, .data, .rsrc, .rdata, .reloc 에서의 value		[2], [3], [4]
VirtualAddress	.text, .data, .rsrc, .rdata, .reloc 에서의 value	In executables, indicates the RVA where the section begins in memory. Should be set to 0 in OBJs.	[2], [3], [4]
SizeOfRawData	.text, .data, .rsrc, .rdata, .reloc 에서의 value	The size (in bytes) of data stored for the section in the executable or OBJ. For executables, this must be a multiple of the file alignment given in the PE header. If set to 0, the section is uninitialized data.	[2], [3], [4]
PointerToRawData	.text, .data, .rsrc, .rdata, .reloc 에서의 value	The file offset where the data for the section begins. For executables, this value must be a multiple of the file alignment given in the PE header.	[3], [4]
Characteristics	.text, .data, .rsrc, .rdata, .reloc 에서의 value	Flags OR'ed together, indicating the attributes of this section. Many of these flags can be set with the linker's /SECTION option	[3], [4], [6]
Name	.text, .data, .rsrc, .rdata, .reloc 에서의 value	packed 된 경우 일반적인 섹션의 이름과 다름	[2], [3], [4]
Entropy	.text, .data, .rsrc, .rdata, .reloc 에서의 value	Entropy 값으로 packed 되었는지 판단할 때 사용	[2]
TotalNumberOfFunctionInIAT			[2], [3], [5]
TotalNumberOfFunctionInEAT			[3], [5]

[7] "An In-Depth Look into the Win32 Portable Executable File Format - Part 1" , Delphibasics, last modified Mar 15, 2010, accessed April 28, 2024, https://www.delphibasics.info/home/delphibasicsarticles/anin-depthlookintothewin32portableexecutablefileformat-part1

PE feature selection & description

필드 값	전처리 방법	필드 값에 대한 설명	사용 이유
PointerToSymbolTable	value	The file offset of the COFF symbol table. COFF symbol tables are relatively rare in PE files, as newer debug formats have taken over. Prior to Visual Studio .NET, a COFF symbol table could be created by specifying the linker switch /DEBUGTYPE:COFF. COFF symbol tables are almost always found in OBJ files. Set to 0 if no symbol table is present.	해당 feature를 사용하는 reference를 찾지 못했으나, domain knowledge가 없는 분석가의 입장에서 feature를 이유 없이 제거하는 것 또한 근거가 빈약하다 판단되어 추후 RFE 수행을 통해 최종 PE feature 결정 여부를 판단하려 함
Entrypoint	value		해당 feature를 사용하는 reference를 찾지 못했으나, domain knowledge가 없는 분석가의 입장에서 feature를 이유 없이 제거하는 것 또한 근거가 빈약하다 판단되어 추후 RFE 수행을 통해 최종 PE feature 결정 여부를 판단하려 함
Checksum	value	The checksum of the image. The CheckSumMappedFile API in IMAGEHLP.DLL can calculate this value. Checksums are required for kernel-mode drivers and some system DLLs. Otherwise, this field can be 0. The checksum is placed in the file when the /RELEASE linker switch is used.	checksum이 0이 아닌 경우, system DLL 파일일 가능성이 존재. 악성코드 공격자는 시스템에 대한 지속적인 공격, 메모리 공간 접근 목적, 분석 난이도 상승 등의 이유로 악성코드를 DLL로 배포하는 경우가 많고 [8], 해당 feature가 DLL 파일인지 판단하는 근거로 사용될 수 있기에 feature 후보에 포함시킴.
Detection	value	https://github.com/packing-box/pypackerdetect 툴을 사용하여 추출한 파일 내 Packing 된 부분의 개수	malware의 종류에 따라 Packing 된 부분의 개수가 다를 가능성이 있다고 판단

[8] Monnappa K A, 악성코드 분석 시작하기 : 윈도우 악성코드 분석에 필요한 개념과 도구, 테크닉, 에이콘출판주식회사, p.125-126

Reference

[1] Kumar, A., Kuppusamy, K. S., & Aghila, G. (2019). A learning model to detect maliciousness of portable executable using integrated feature set. Journal of King Saud University-Computer and Information Sciences, 31(2), 252-265.
[2] Zhou, H. (2018, August). Malware detection with neural network using combined features. In China cyber security annual conference (pp. 96-106). Springer, Singapore
[3] Shafiq, M. Z., Tabish, S. M., Mirza, F., & Farooq, M. (2009, September). Pe-miner: Mining structural information to detect malicious executables in realtime. In International workshop on recent advances in intrusion detection (pp. 121-141). Springer, Berlin, Heidelberg.
[4] Zatloukal F, Znoj J (2017) Malware detection based on multiple pe headers identification and optimization for specific types of files. J Adv Eng Comput 1:153. https://doi.org/10.25073/jaec.201712.64
[5] B. Sun, Q. Li, Y. Guo, Q. Wen, X. Lin and W. Liu, "Malware family classification method based on static feature extraction," 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 507-513, doi: 10.1109/CompComm.2017.8322598
[6] Hasan H. Al-Khshali, Muhammad Ilyas, Osman N. Ucan. (2020). Effect of PE File Header Features on Accuracy. IEEE Xplore

방법론

방법론 - f_classif 와 RFE(Recursive Feature Elimination)

RFE (Recursive Feature Elimination) [9] [11]
- 재귀적 특성 제거 방법으로, 모델을 사용하여 특성의 중요도를 평가하고 중요도가 낮은 특성을 제거하는 과정을 반복
- 초기에는 모든 특성을 사용하여 모델을 학습시키고, 각 특성의 중요도를 평가
- 중요도가 가장 낮은 특성을 제거하고, 남은 특성으로 모델을 다시 학습
- 해당 과정을 원하는 특성 개수가 될 때까지 반복
- domain knowledge가 없는 분석가의 입장에서 적합한 방법론이라 판단

f_classif (in sklearn)
- ANOVA F-값을 계산하여 특성의 중요도를 평가하는 방법
- 각 특성에 대해 F-값을 계산하여 특성과 타겟 변수 간의 선형 상관관계를 측정
- F-값이 높을수록 해당 특성이 타겟 변수와 강한 선형 상관관계를 가지는 것으로 해석
- 계산된 F-값을 기준으로 특성의 중요도를 평가하고 순위를 매김

[9] B. Sani Mahmoud, B. Ahmad, A. B. Garko, "A Machine Learning Model for Malware Detection Using Recursive Feature Elimination (RFE) For Feature Selection and Ensemble Technique," IOSR Journal of Computer Engineering, vol. 24, no. 1, pp. 23-30
[11] R. Islam, M. I. Sayed, S. Saha, M. J. Hossain, and M. A. Masud, "Android malware classification using optimum feature selection and ensemble machine learning," in Internet of Things and Cyber-Physical Systems, vol. 3, pp. 100-111

방법론 - f_classif 와 RFE(Recursive Feature Elimination)

본 연구의 방법론

- 앞서 reference 참고를 통해 뽑은 PE feature 중에서 타겟 변수와 선형 상관관계가 높은 25개의 feature 를 f_classif를 이용하여 추출 후, RFE를 통해 최종 PE feature 선택

방법론의 선택 이유

1. 계산 효율성 향상

- RFE는 특성 제거 과정에서 반복적으로 모델을 학습시키므로 계산 비용이 높을 수 있음
- f_classif는 각 특성에 대해 독립적으로 F-값을 계산하므로 계산 속도가 빠름
- f_classif를 사용하여 사전에 특성의 중요도를 평가하고 순위를 매긴 후, 상위 특성을 선택하여 RFE에 적용하면 반복 횟수를 줄일 수 있음

2. 통계적 유의성 고려

- RFE는 모델의 성능을 기반으로 특성의 중요도를 평가하므로, 통계적 유의성을 직접적으로 고려하지 않음
- f_classif는 F-값을 통해 특성과 타겟 변수 간의 선형 상관관계의 통계적 유의성을 평가
- 통계적으로 유의한 특성을 사전에 선택하여 RFE에 적용하면, 통계적 유의성을 고려한 특성 선택이 가능해짐

방법론 – Decision Tree model

본 연구에서 사용할 모델은 Decision tree 모델

- 분류와 회귀 문제에 사용되는 비선형 예측 모델. 데이터를 학습하여 일련의 질문을 기반으로 한 결정 규칙을 형성하고, 이를 통해 예측을 수행
- 각 노드에서 특성의 값에 따라 데이터를 분할하고, 최종 노드(리프 노드)에서 클래스 레이블을 할당
- 계층적 구조를 통해 클래스를 효과적으로 구분
- 데이터 분포에 대한 가정이 필수적이지 않음

결측치 처리

결측치 처리

결측치 처리 방법 ideation

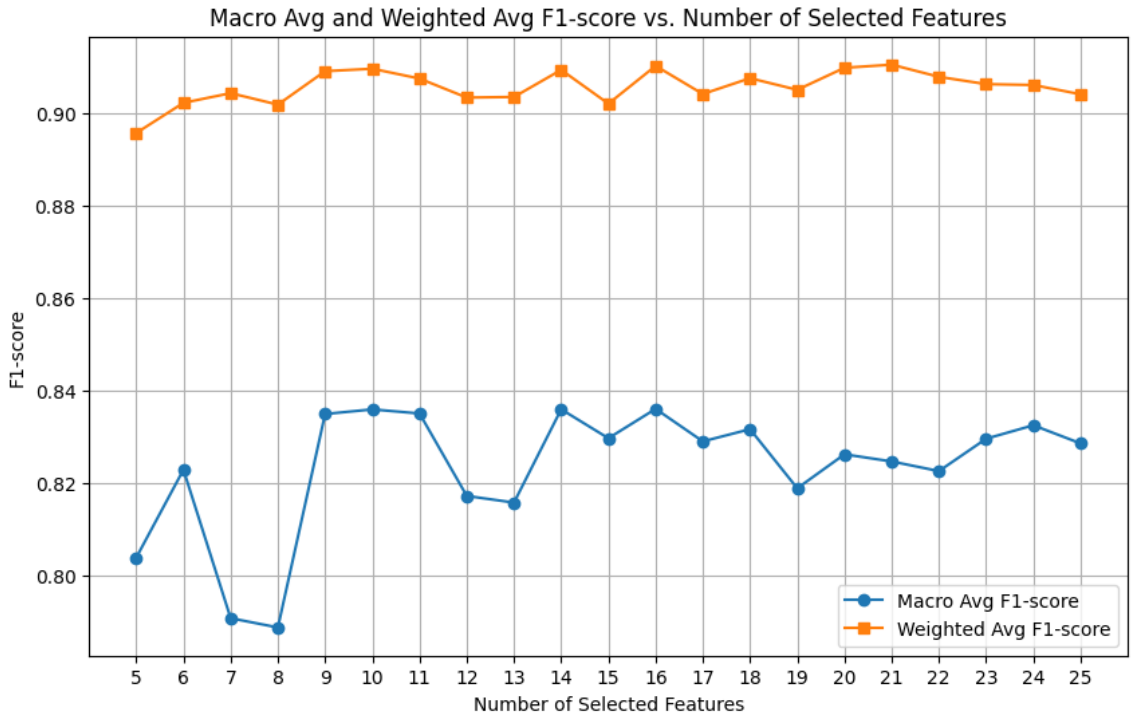
A) 모든 결측치를 “Missing” 이라는 값으로 하나의 범주화

B) 결측치가 문자형 데이터인 경우 “Missing” 이라는 값으로 범주화, 숫자형 데이터인 경우 중앙값으로 대체

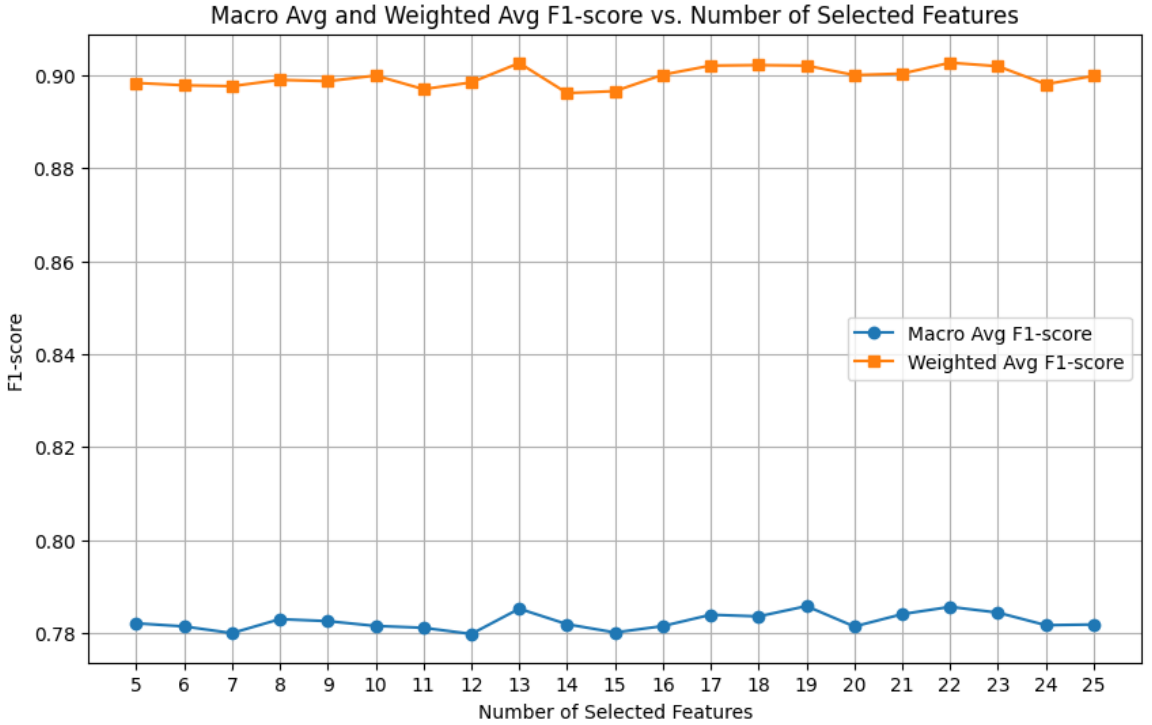
maleware analysis에서, 결측치 값은 그 자체로 악성코드 분류에 도움이 되는 하나의 범주일 수 있음. 탐지를 피하기 위해 데이터를 숨기거나, 비정상적인 시스템 동작이 결측치의 발생으로 이어질 수 있기 때문.
따라서 A 방법을 사용하기로 결정

결측치 처리

실제로 RFE를 통해 feature의 개수에 따른 정확도를 비교하였을 때에도 결측치를 모두 하나의 범주로 처리한 A 모델의 정확도가 전반적으로 높게 나온 것을 확인



A) 모든 결측치를 "Missing" 이라는 값으로 범주화

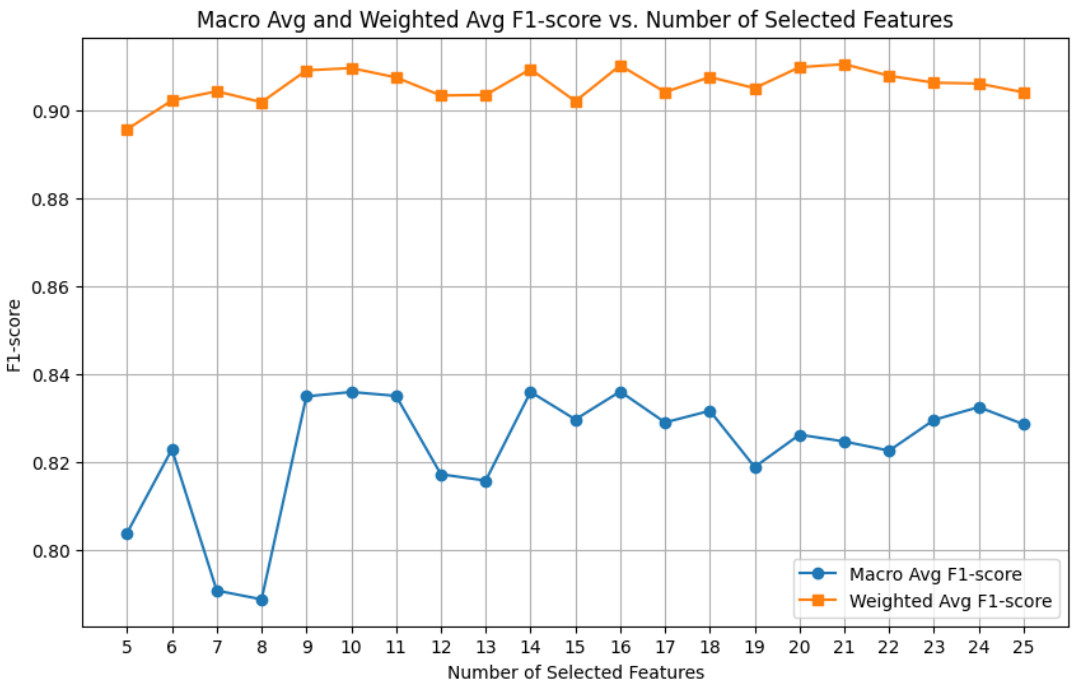


B) 결측치가 문자형 데이터인 경우 "Missing" 이라는 값으로 범주화, 숫자형 데이터인 경우 중앙값으로 대체

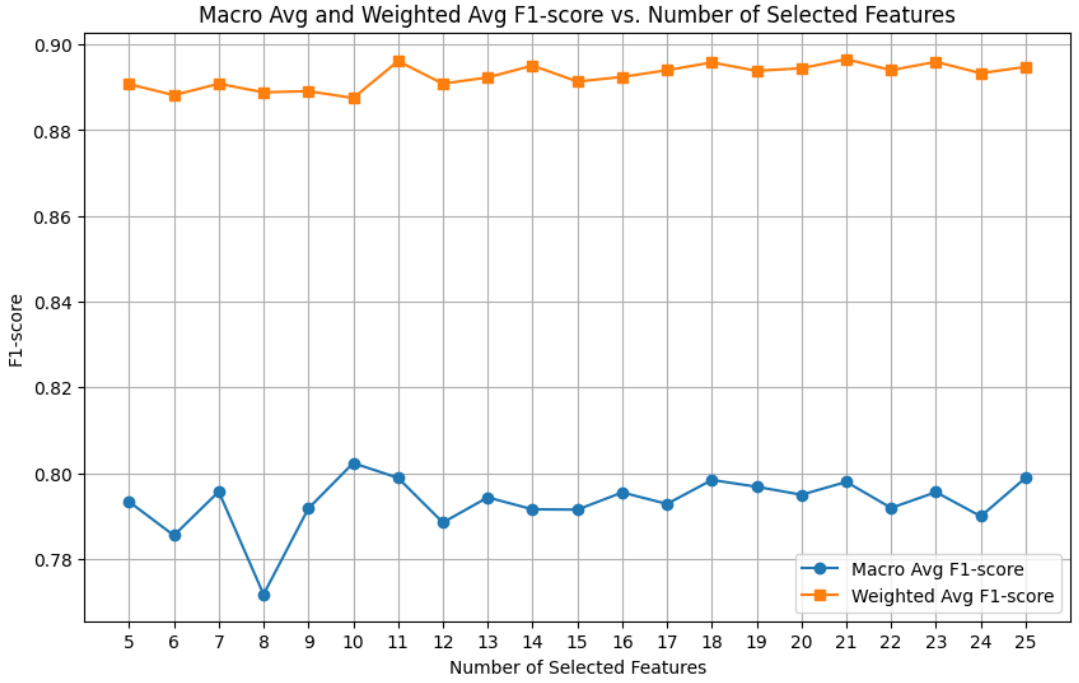
최종 PE feature 추출

최종 PE feature 추출

- Number of Selected Features 값을 5에서 25까지 1씩 늘려가며 prediction을 수행함
- TrainSet과 TestSet의 비율은 8:2 로 설정 (7:3보다 accuracy가 높은 것을 확인)



<test_size=0.2>



<test_size=0.3>

최종 PE feature 추출

- Number of Selected Features 값이 16일 때 Weighted Average P1-score가 가장 높으므로 해당 feature 들을 최종 feature로 선정함

Final Selected PE Features (n=16)	
Detections	AddressOfEntryPoint
e_ovno	BaseOfCode
NumberOfSections	ImageBase
TimeDateStamp	EntryPoint
PointerToSymbolTable	SectioAlignment
NumberOfSymbols	SizeOfImage
SizeOfOptionalHeader	.textSectionName
Characteristics	.textSection|VirtualSize-SizeOfRawData|

Number of Selected Features	Macro Avg F1-score	Weighted Avg F1-score	Number of Selected Features	Macro Avg F1-score	Weighted Avg F1-score
5	0.803	0.895	**16**	**0.835**	**0.910**
6	0.822	0.902	17	0.828	0.904
7	0.790	0.904	18	0.831	0.907
8	0.788	0.901	19	0.818	0.904
9	0.834	0.908	20	0.826	0.909
10	0.835	0.909	21	0.824	0.910
11	0.834	0.907	22	0.822	0.907
12	0.817	0.903	23	0.829	0.906
13	0.815	0.903	24	0.832	0.905
14	0.835	0.909	25	0.828	0.903
15	0.829	0.901			

Reference

[1] Kumar, A., Kuppusamy, K. S., & Aghila, G. (2019). A learning model to detect maliciousness of portable executable using integrated feature set. Journal of King Saud University-Computer and Information Sciences, 31(2), 252-265.
[2] Zhou, H. (2018, August). Malware detection with neural network using combined features. In China cyber security annual conference (pp. 96-106). Springer, Singapore
[3] Shafiq, M. Z., Tabish, S. M., Mirza, F., & Farooq, M. (2009, September). Pe-miner: Mining structural information to detect malicious executables in realtime. In International workshop on recent advances in intrusion detection (pp. 121-141). Springer, Berlin, Heidelberg.
[4] Zatloukal F, Znoj J (2017) Malware detection based on multiple pe headers identification and optimization for specific types of files. J Adv Eng Comput 1:153. https://doi.org/10.25073/jaec.201712.64
[5] B. Sun, Q. Li, Y. Guo, Q. Wen, X. Lin and W. Liu, "Malware family classification method based on static feature extraction," 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 507-513, doi: 10.1109/CompComm.2017.8322598
[6] Hasan H. Al-Khshali, Muhammad Ilyas, Osman N. Ucan. (2020). Effect of PE File Header Features on Accuracy. IEEE Xplore
[7] "An In-Depth Look into the Win32 Portable Executable File Format - Part 1" , Delphibasics, last modified Mar 15, 2010, accessed April 28, 2024, https://www.delphibasics.info/home/delphibasicsarticles/anin-depthlookintothewin32portableexecutablefileformat-part1
[8] Monnappa K A, 악성코드 분석 시작하기 : 윈도우 악성코드 분석에 필요한 개념과 도구, 테크닉, 에이콘출판주식회사, p.125-126
[9] B. Sani Mahmoud, B. Ahmad, A. B. Garko, "A Machine Learning Model for Malware Detection Using Recursive Feature Elimination (RFE) For Feature Selection and Ensemble Technique," IOSR Journal of Computer Engineering, vol. 24, no. 1, pp. 23-30
[10] R. Senapati, K. Shaw, S. Mishra, and D. Mishra, "A Novel Approach for Missing Value Imputation and Classification of Microarray Dataset," Procedia Engineering, vol. 38, pp. 1067-1071, 2012, presented at the International Conference on Modeling, Optimization and Computing (ICMOC-2012).
[11] R. Islam, M. I. Sayed, S. Saha, M. J. Hossain, and M. A. Masud, "Android malware classification using optimum feature selection and ensemble machine learning," in Internet of Things and Cyber-Physical Systems, vol. 3, pp. 100-111