

머신러닝을 이용한 악성코드 탐지

포비 조진영 고은기 유혜빈 조미리



INDEX



개요



데이터 유출 및 삭제, 금전적 요구 등
다양한 유형의
사이버 침입 문제 발생 증가

변종/신종 악성코드 증가

악성코드의 기하급수적인 증가에 비해
현저히 부족한 악성코드 분석가

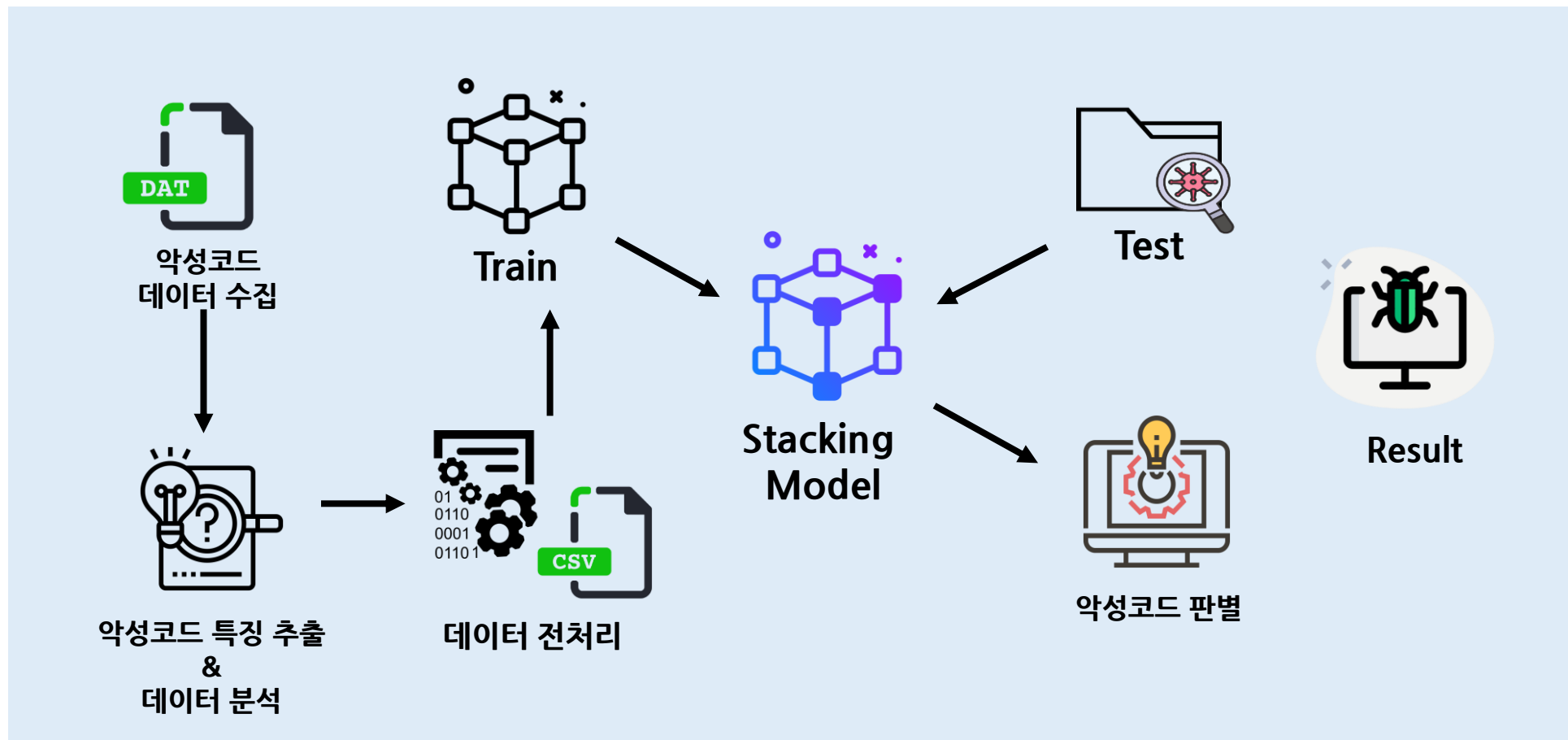


지능형 악성코드를 사전에 탐지하고 대응하는 것이 중요

따라서 머신러닝을 이용한 악성코드 분석 방식을 통해 다양한 악성코드 탐지에 적용하고자 함

개요

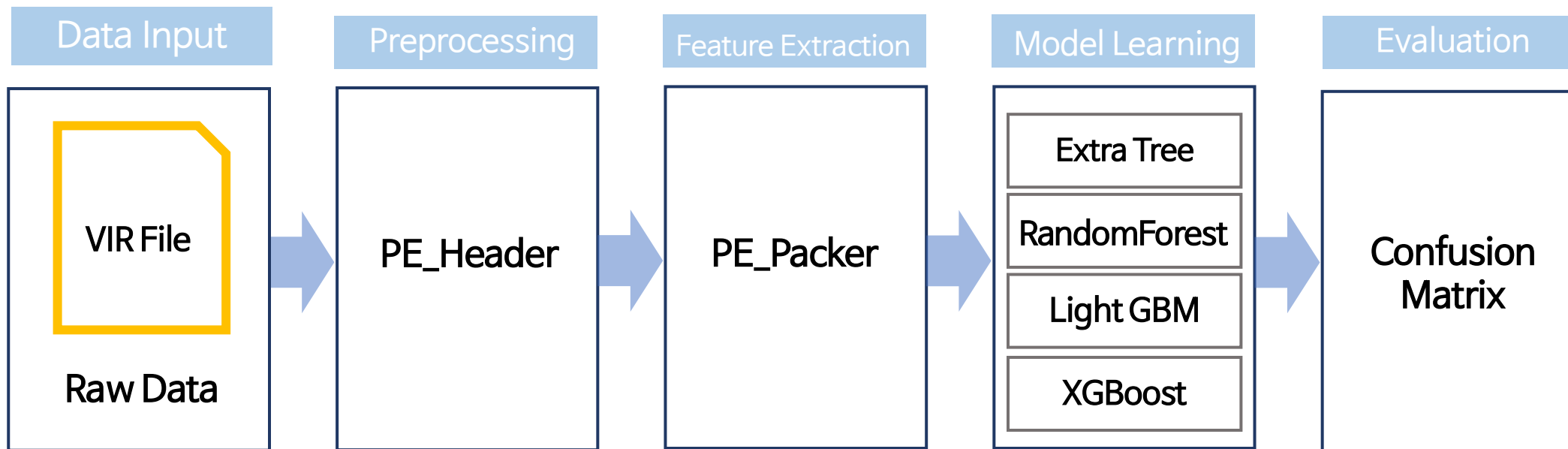
구상도



기술의 독창성

- 기존의 작품은 대부분 정적 분석을 위한 데이터 추출 시 pe헤더의 일부만을 가공하여 사용하지만
본 작품은 N-gram방식, pe헤더 전체 데이터에 원-핫인코딩, 피어슨의 상관계수 등을 사용하여
정제된 데이터를 생성함. 이를 머신 러닝 모델에 적용 가장 정확도가 높은 특징을 사용.
- 모델 성능을 높이기 위해 서로 다른 단일 모델과 혼합하여 사용하는 방식인 스택킹 기법을 사용하여 모델링.
- 더 나아가 기존 작품들은 대부분 머신 러닝을 이용하여 악성코드를 분류하는 과정 까지만 수행 하였지만
본 작품은 실제 사용을 위한 GUI프로그램을 설계하여 악성파일로 분류한 파일들을 삭제할 수 있도록 구현.

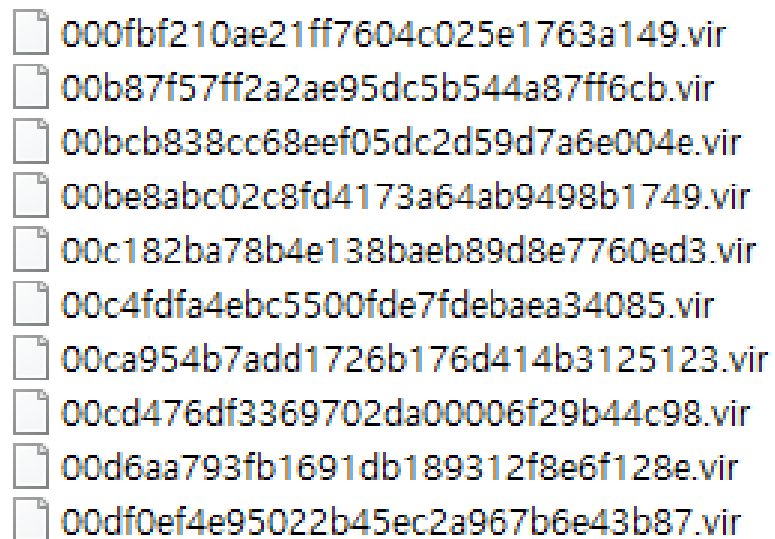
시스템 구성도



악성코드 수집

❖ 모델 학습 데이터 셋

- 2018년 정보보호 R&D 데이터 챌린지 악성코드 탐지 트랙

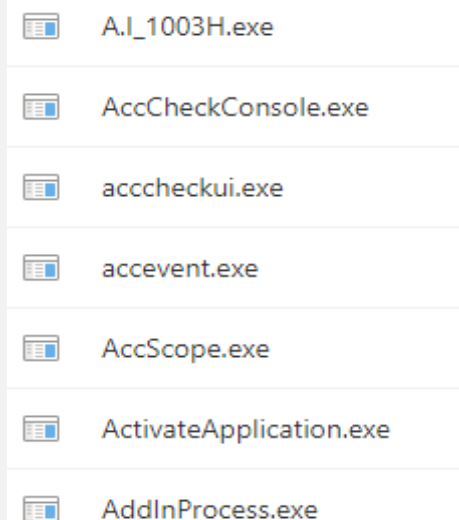


000fbf210ae21ff7604c025e1763a149.vir
00b87f57ff2a2ae95dc5b544a87ff6cb.vir
00bcb838cc68eef05dc2d59d7a6e004e.vir
00be8abc02c8fd4173a64ab9498b1749.vir
00c182ba78b4e138baeb89d8e7760ed3.vir
00c4fdfa4ebc5500fde7fdebaea34085.vir
00ca954b7add1726b176d414b3125123.vir
00cd476df3369702da00006f29b44c98.vir
00d6aa793fb1691db189312f8e6f128e.vir
00df0ef4e95022b45ec2a967b6e43b87.vir

→ 10500개의 악성 / 4500개의 정상 데이터

❖ 모델 검증 데이터 셋

- Malwaredb 사이트 등



A.I_1003H.exe
AccCheckConsole.exe
acccheckui.exe
accevent.exe
AccScope.exe
ActivateApplication.exe
AddInProcess.exe

→ 총 378개의 악성 / 466개의 정상 데이터

Data preprocessing

데이터 분석 방식

- 정적분석

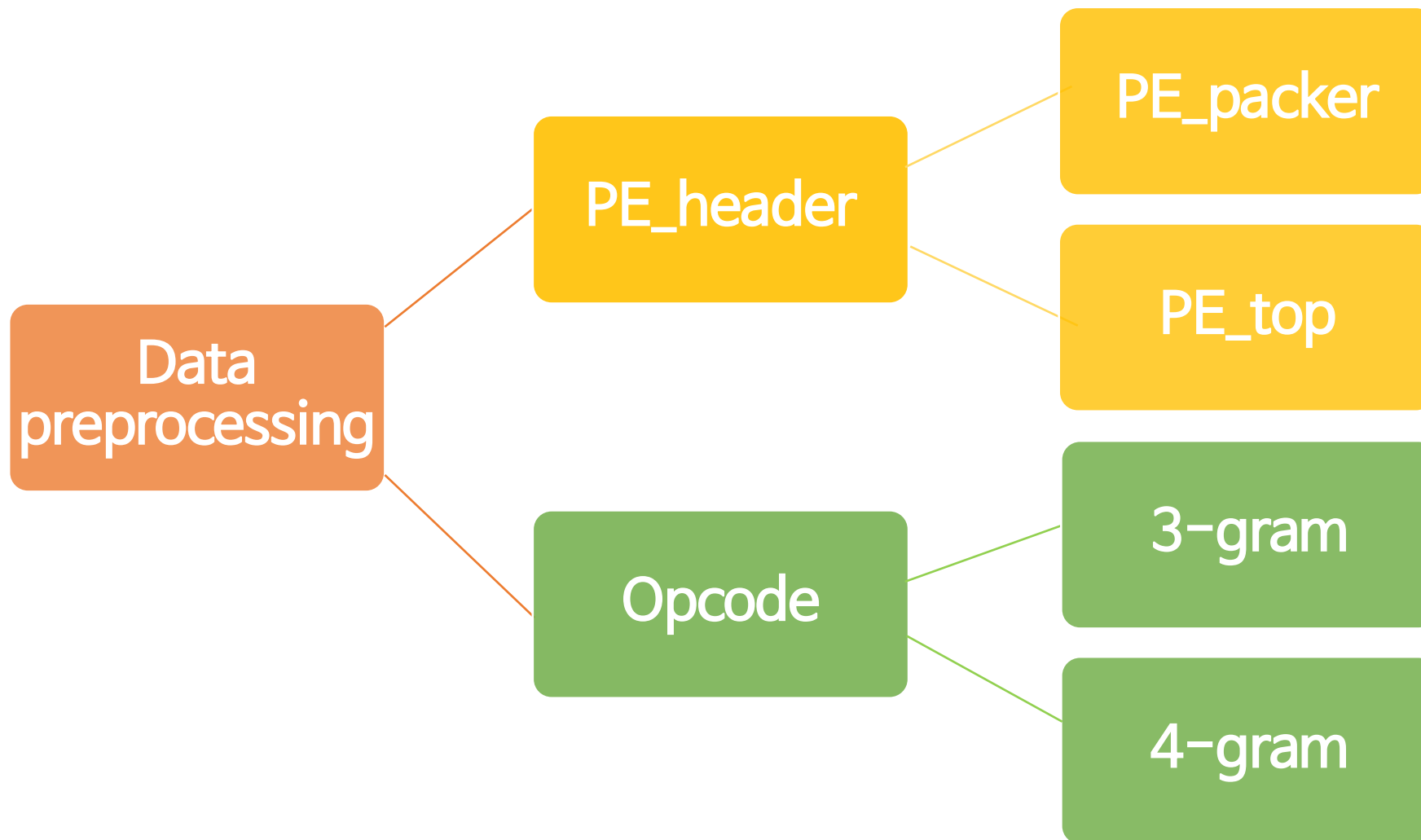
: 악성코드 분석 시 가장 먼저 진행, 프로그램 자체를 실행시키지 않음
프로그램의 기능을 파악하기 위한 코드 및 프로그램 구조 분석

- 동적 분석

: 악성코드를 실행시키면서 악성코드의 행위, 실제 실행되는 코드 위주로 분석

⇒ 동적분석은 실제로 실행시켜야 함으로 **감염 위험성이 있음**
따라서 **정적분석을 통한 데이터 특징 정보 추출**

Data preprocessing



Data preprocessing - PE header

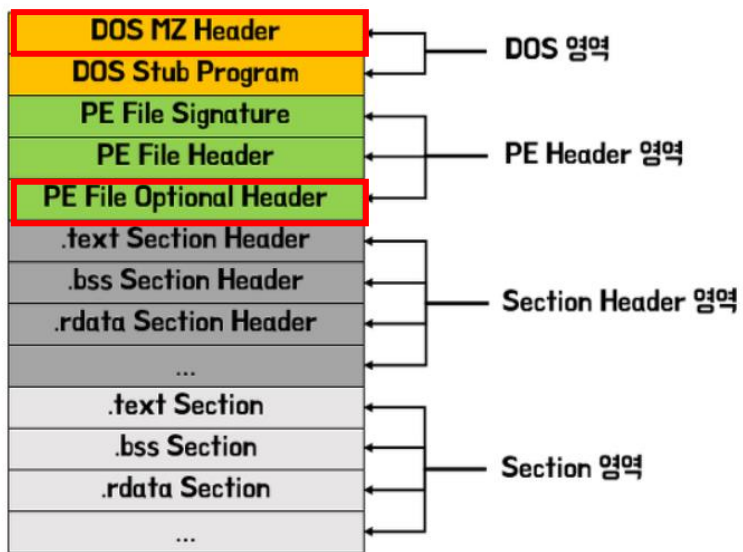


그림1. PE구조 구분

PE (Portable Executable File Format)

: 모든 윈도우 프로그램이 가지고 있는 부분으로

실행파일을 실행하기 위한 각종 정보와 로딩 시 필요한 정보,
메모리에 적재될 각종 리소스 할당 정보 등 포함

ClaMP오픈소스를 사용하여

PE header의 DOS header에서 6개,
FILE헤더에서 17개,
OPTIONAL header에서 37개 총 60개의 Raw특징과
필드 값을 한번 더 가공한 7개의 Deriven특징 추출

Data preprocessing - PE header

```
(cap) j201721380@VirtualBox:~/Downloads$ python pe_headers.py
Enter type of sample( malware(1)|benign(0))>>3
hash: 0a22db17a1e5adaf5af0d397700084da
Successfully Data extracted and written to file: 0a22db17a1e5adaf5af0d397700084da
Processed 1 files
hash: 0abf1ee7bfc6ce6d1a8779b30a9dde6
Successfully Data extracted and written to file: 0abf1ee7bfc6ce6d1a8779b30a9dde6
Processed 2 files
```

1. Packer중 범주형 데이터를 제거한 **PE_header** 특징
2. Packer특징을 one-hot-encoding한 **PE_packer** 특징
3. class와 0.0 ~ 0.3 까지 상관계수를 갖는 Columns를 추출한 **PE_top** 특징

filename	MD5	e_cblp	e_cp	e_cparhdr	e_maxallo	e_sp	e_lfanew	NumberO	packer_type	E_text	E_data	filesize	E_file	fileinfo	class
d9083937f	d9083937f	144	3	4	65535	184	224	5	NoPacker	6.41462	6.374763	121348	7.074258	0	0
d6db8f9e0	d6db8f9e0	144	3	4	65535	184	128	3	NoPacker	0	0	1142012	7.932987	0	0
ccb45a121	ccb45a121	144	3	4	65535	184	264	5	NoPacker	5.776291	2.996614	581636	5.615077	0	0
bbaab0f80	bbaab0f80	144	3	4	65535	184	128	3	NoPacker	6.324629	2.051228	49508	7.040551	0	0
de99a025e	de99a025e	144	3	4	65535	184	600	7	NETexecut	6.623654	0	290820	6.556785	0	0
c5078a7f6	c5078a7f6	144	3	4	65535	184	216	4	NoPacker	6.644056	0.337406	818468	7.991148	0	0
d86e1bf5b	d86e1bf5b	144	3	4	65535	184	128	3	NoPacker	5.929845	0	462852	5.844465	0	0
bc50cf4e0	bc50cf4e0	144	3	4	65535	184	128	4	NoPacker	6.466231	2.678912	1190514	7.920582	0	0
beb88ca3c	beb88ca3c	144	3	4	65535	184	200	3	NoPacker	6.651937	3.4085	748028	6.957228	0	0

Data preprocessing - PE header

상관 분석

- 상관 분석

: 확률론과 통계학에서 두 변수 간 어떤 선형적 또는 비선형적 관계를 갖고 있는 지를 분석하는 방법

- 상관계수

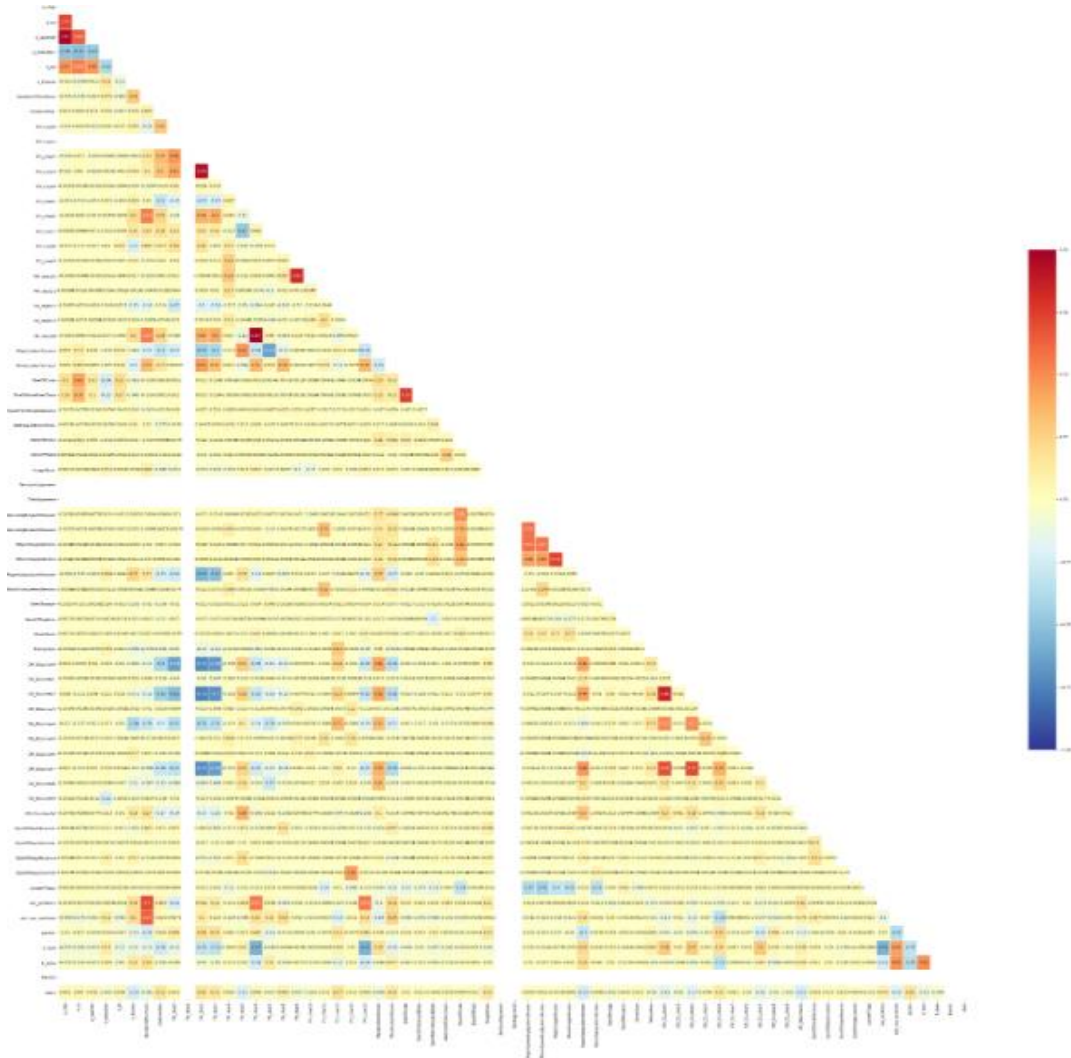
: 상관관계 정도를 파악해 두 변수 간 연관된 정도를 나타냄

- 피어슨 상관 계수

$$r(\text{상관계수 값}) = \frac{\text{X와 Y가 함께 변하는 정도}}{\text{X와 Y가 각각 변하는 정도}}$$

r 값	관계
+0.7 ~ +1.0	강한 양적 상관관계
+0.3 ~ +0.7	뚜렷한 양적 상관관계
+0.1 ~ +0.3	약한 양적 상관관계
-0.1 ~ +0.1	상관관계 거의 없음
-0.3 ~ -0.1	약한 음적 상관관계
-0.7 ~ -0.3	뚜렷한 음적 상관관계
-1.0 ~ -0.7	강한 음적 상관관계

Data preprocessing - PE header



데이터가 많아 값을 한눈에 보기 위해 가시화
class와 0.0~0.3까지의 상관계수를 가지는
45개의 column만을 따로 추출하여 pe_top특징으로 사용

Data preprocessing - Opcode

N-gram : 문자열에서 N개의 연속된 요소를 추출하는 방법

4dbed6:	34 45	xor	\$0x45,%al
4dbed8:	72 72	jb	0x4dbf4c
4dbeda:	56	push	%esi
4dbedb:	61	popa	
4dbedc:	72 00	ib	0x4dbede
4dbede:	00 00	add	%al, (%eax)
4dbee0:	5f	pop	%edi
4dbee1:	5f	pop	%edi
4dbee2:	76 62	jbe	0x4dbf46
4dbee4:	61	popa	

- 연속한 N개의 opcode를 묶어 하나의 패턴으로 인식,
같은 패턴의 수를 카운트 해 상위 100개 빈도를 가지는 패턴만을 특징으로 사용
- N의 수가 커질수록 패턴의 사이즈가 커지고 개수를 카운트하는 확률이 줄어들어
희소문제가 생길 수 있기 때문에 N의 값이 3과 4인 경우만 테스트

Data preprocessing - Opcode

N-gram 추출 결과

```
j201721380@VirtualBox:~/Downloads$ python ngram.py
1 file processed (0a22db17a1e5ada15af0d397700084da.vir), 1046 patterns extracted
2 file processed (0abf1ee7bfc6ce6d1a8779b30a9dde64.vir), 2376 patterns extracted
3 file processed (0a0afad8e8f344e0bc67dd683189abcc.vir), 4437 patterns extracted
4 file processed (00e1faccfb4d45193d342d36fbb88914.vir), 8498 patterns extracted
```

3-gram

	1	filename	MD5	mov	mov	mov	int3	int3	int3	push	push	push	class
2	0003c3dd2	0003c3dd2		10					0			0	1
3	000590557	000590557		10				157				210	1
4	0006c0992	0006c0992		0				0				0	0
5	000fbf210	000fbf210		359				5				765	1
													0

4-gram

	filename	MD5	int3	int3	int3	int3	mov	mov	mov	mov	add	add	add	add	class
	0003c3dd2	0003c3dd2				0				7				21	1
	000590557	000590557				72				132				6	1
	0006c0992	0006c0992				0				0				6	0
	000fbf210	000fbf210				4				135				3	1
															0

→ 하지만 파일의 특성에 따라 N-gram 추출이 불가능한 경우가 존재

Data preprocessing - Model Training

→ 가장 효과적인 특징을 찾기 위해 각 특징들을 단일 모델에 넣어 정확도 확인

사용한 단일 모델

- Logistic Regression
: 데이터가 범주에 속할 확률을 0에서 1사이의 값으로 예측하여 더 가능성이 높은 범주로 분류하는 지도 학습 알고리즘
- Support Vector Machine
: 두 부류 사이에 존재하는 여백을 의미하는 마진을 최대화하여 일반화 능력을 극대화하는 모델
- Random Forest
: 여러 개의 작은 결정 트리가 예측한 값들 중 가장 많은 값 혹은 평균값을 최종 예측 값으로 정함
- XGBoost
: 여러 개의 Decision Tree를 조합해서 사용하는 Ensemble 알고리즘

Data preprocessing - Model Training

추출한 전체 데이터 셋 15,000개 중 80%를 학습에 사용하고 20%를 테스트에 사용

	Logistic Regression	SVM	Random Forest	XGBoost	AVG
pe_header	0.708	0.709	0.939	0.933	0.822
pe_packer	0.708	0.709	0.943	0.934	0.824
pe_top	0.707	0.709	0.927	0.919	0.816
4gram	0.743	0.735	0.804	0.811	0.773
3gram	0.773	0.730	0.823	0.819	0.786
pe_4gram	0.690	0.696	0.909	0.931	0.807
pe_3gram	0.691	0.696	0.908	0.923	0.804

▪ pe_header를 가공한 특징 > N-gram 단일 특징

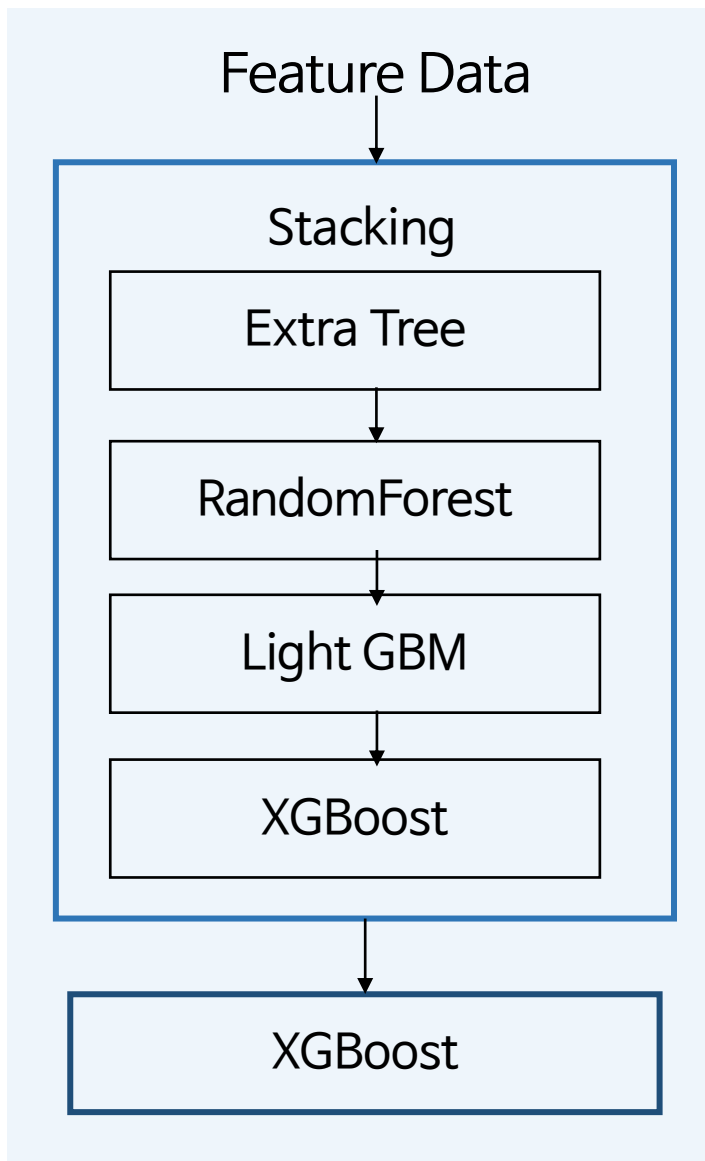
▪ pe_packer 특징 > pe_header 특징

→ 따라서 pe_packer특징을 특징 데이터로 사용

❖ pe_ngram

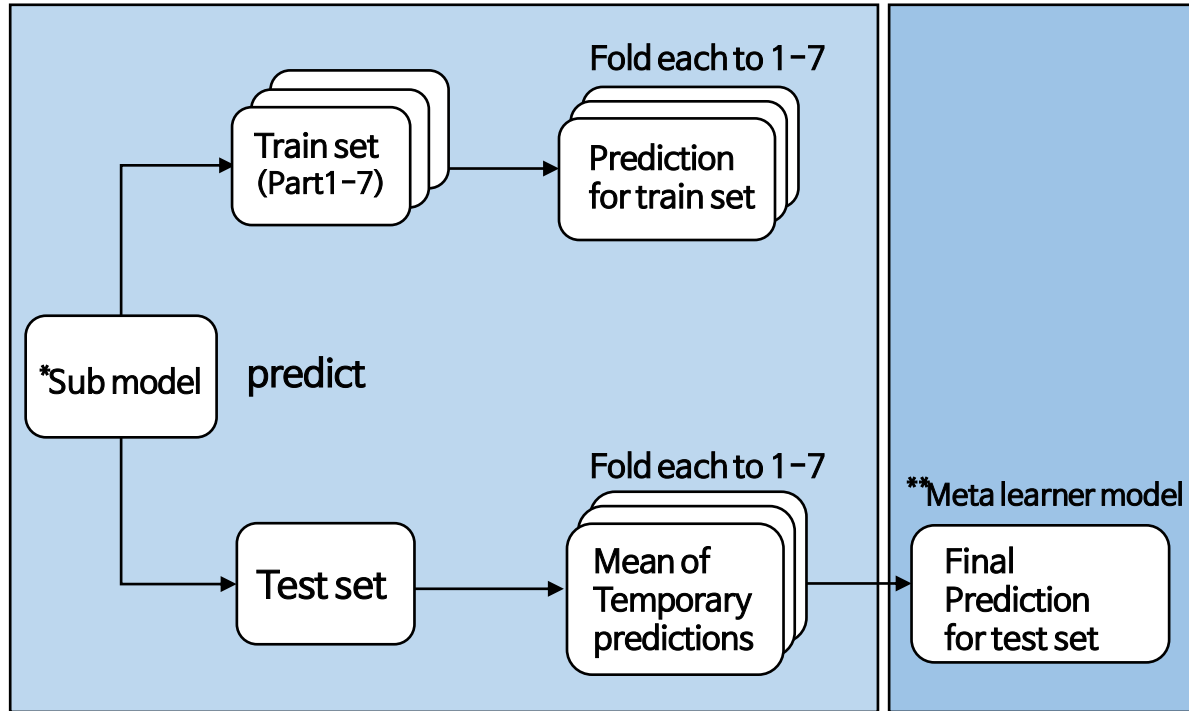
= pe_packer특징 + N-gram특징을 열 병합해 하나로 합친 특징 데이터

Modeling - Staking



- 다른 앙상블 기법과 달리 Staking은 서로 다른 종류의 단일 모델 혼합 하여 각 모델의 예측 값을 토대로 학습하는 기법
- 훈련 데이터셋을 이용하여 **Sub-model의 예측결과를 생성**하고 이를 입력 값으로 하여 **Meta-learner 최종 예측 값**을 내는 방식의 알고리즘

Modeling - Staking



* Sub model = ExtraTree / RandomForest / LightGBM / XGBoost

** Meta learner model = XGBoost

- 선택된 모델마다 K-fold로 K번 prediction을 한 후 예측치들의 평균을 결과 예측치로 사용
- 4개의 모델을 사용하고 7-fold로 하여 테스트

ET : 0.9362
RF : 0.9436
LXGB : 0.9426
XGB : 0.9396

도출한 결과 값을 최종 Classifier 입력한 결과
0.9529의 값으로 단일 모델 만을 사용한 경우보다
정확도가 개선 됨

Model Evaluation

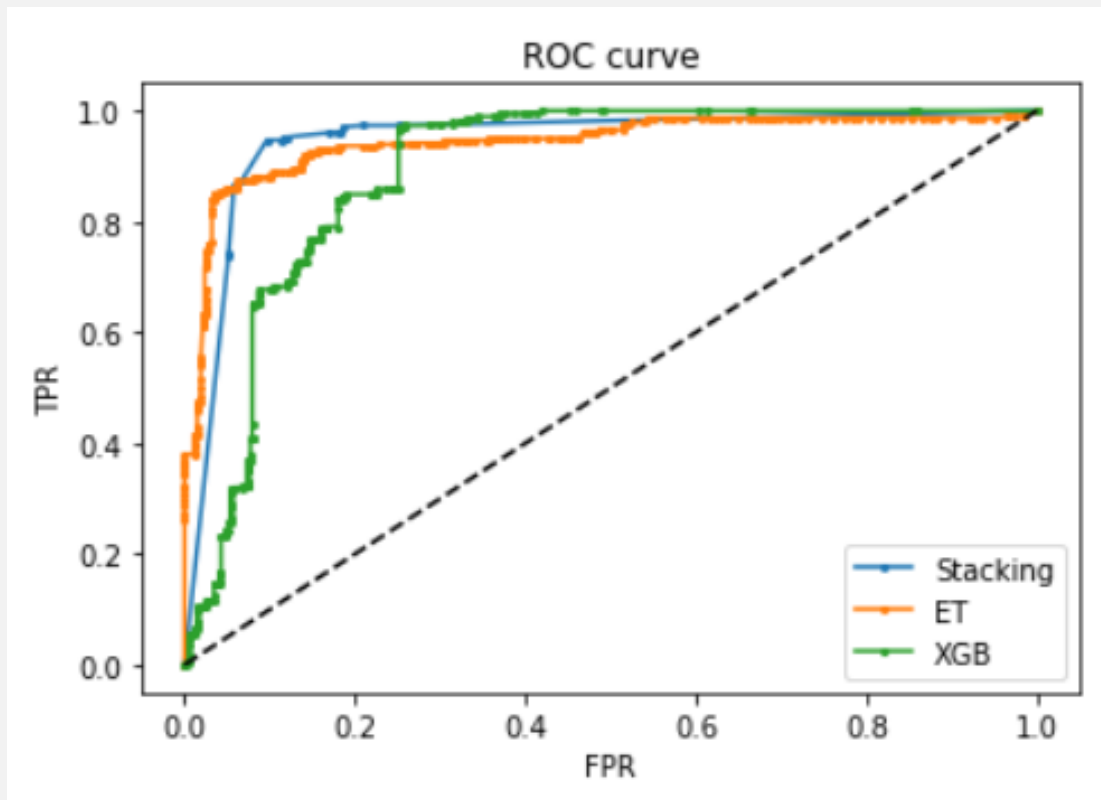
Model의 성능을 평가하기 위해 Confusion Matrix를 사용

Model	정밀도 (PRE)	탐지율 (TPR)	오탐율 (FPR)	정확도 (ACC)	F-점수
ExtraTree Model	87.6%	88.3%	10.0%	89.2%	88.00
XGBoost Model	78.9%	84.1%	18.2%	82.8%	81.42
Stacking Model	88.8%	94.7%	9.6%	92.3%	91.90

- 정밀도 (PRE): 악성 파일이라고 판단한 파일 중 실제 악성 파일에 해당하는 비율
- 탐지율 (TPR): 실제 악성 파일을 악성 파일로 정확하게 판단한 비율 , 높을 값을 가질수록 좋은 시스템
- 오탐율 (FPR): 실제 정상 파일을 악성 파일로 판단한 비율, 낮은 값을 가질수록 좋은 시스템
- F점수: 정밀도와 재현율의 가중조화평균, 탐지율과 정밀도가 균형을 이룰 때 F점수의 값이 높아짐

Model Evaluation

성능 평가 지표 실험 결과에 대한 ROC Curve → 그래프 아래의 면적이 클수록 좋은 결과 의미

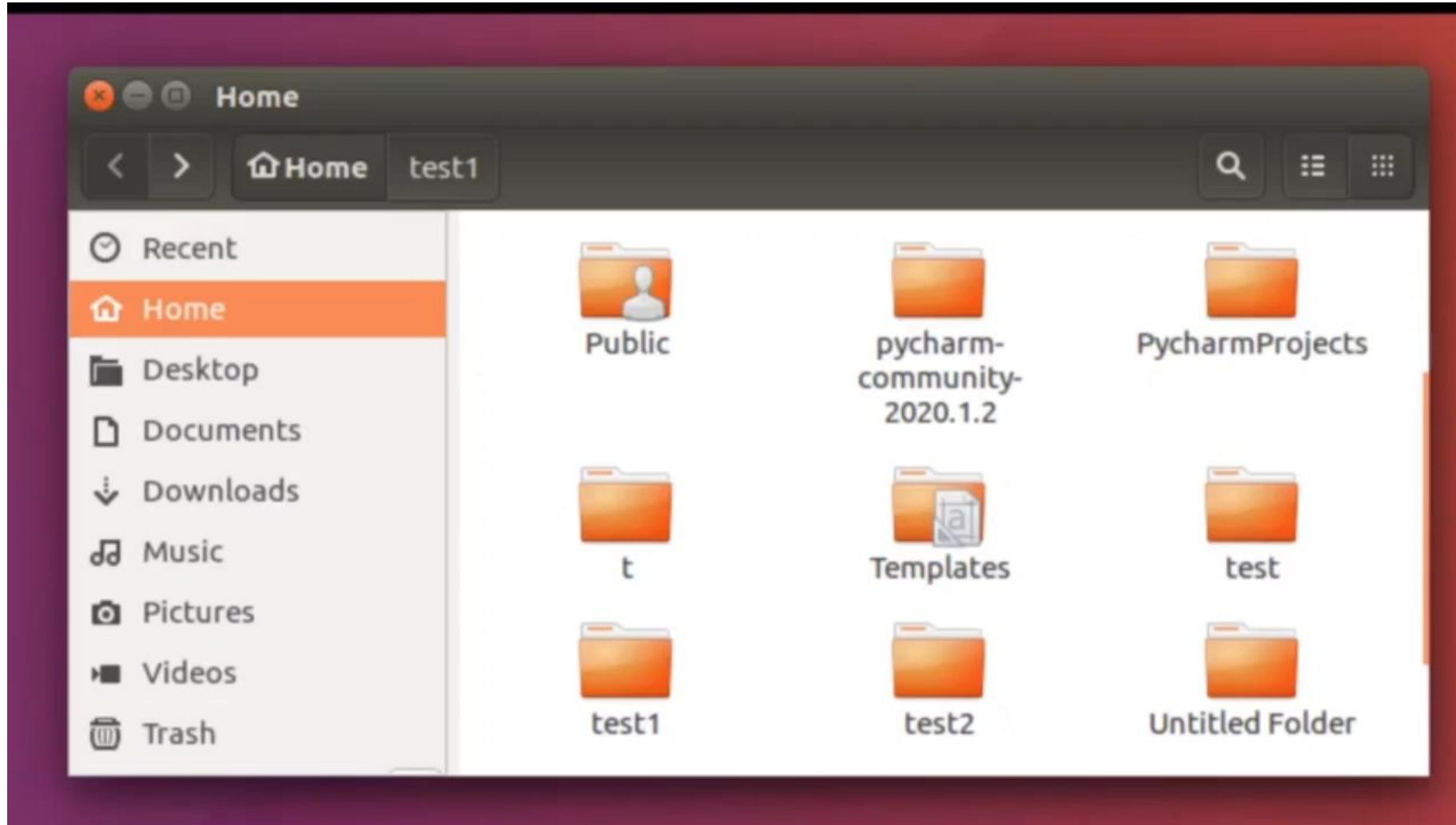


스태킹 앙상블 모델의 탐지율(TPR) 94.7%,
오탐율(FPR) 9.6%, 분류 정확도(ACC) 92.3%,
정밀도(PRE) 88.8%, F점수는 91.9의 성능을 가짐.

단일 모델의 평균 악성코드 탐지율(TPR) 약 86.2%,
오탐율(FPR) 14.1%, 분류 정확도(ACC) 86.0%,
정밀도(PRE) 83.25%, F점수는 84.71의 성능을 가짐.

→ ROC curve 결과로 단일 모델을 사용한 경우보다 **스태킹 모델을 사용한 경우**가 더 좋은 결과를 낼 수 있음

GUI 실행 영상



삭제 전 : 32Items, 17.3MB 삭제 후 : 15Items, 2.3MB

기대효과

- 향후 대량의 신 변종 악성코드를 빠르고 정확하게 탐지 가능 및 실시간 탐지, 치료까지 가능한 머신러닝, 딥러닝 기반의 악성코드 탐지 시스템 등 다양한 연구 발전 가능
- 감염의 우려가 있어 정적 분석으로 특징 데이터를 추출하였지만 동적 분석, 자동화 분석 등을 통해 더 유의미한 특징 데이터를 추출한다면 정확도를 더욱 높일 수 있을 것으로 기대함