Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.30, NO.3, Jun. 2020 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.3.357

악성 안드로이드 앱 탐지를 위한 개선된 특성 선택 모델

부 주 훈,[1†] 이 경 호[2‡]
[1,2]고려대학교 정보보호대학원(대학원생, 교수)

Advanced Feature Selection Method on Android Malware Detection by Machine Learning

Joo-hun Boo,[1†] Kyung-ho Lee[2‡]
[1,2]Graduate School of Information Security, Korea University(Graduate student, Professor)

요 약

2018년 시만텍 보고서에 따르면, 모바일 환경에서 변종 악성 앱은 전년도 대비 54% 증가하였고, 매일 24,000개의 악성 앱이 차단되고 있다. 최근 연구에서는 기존 악성 앱 분석 기술의 사용 한계를 파악하고, 신·변종 악성 앱을 탐지하기 위하여 기계학습을 통한 악성 앱 탐지 기법이 연구되고 있다. 하지만, 기계학습을 적용하는 경우에도 악성 앱의 특성을 적절하게 선택하여 학습하지 못하면 올바른 결과를 보일 수 없다. 본 연구에서는 신·변종 악성 앱의 특성을 찾아낼 수 있도록 개선된 특성 선택 방법을 적용하여 학습 모델의 정확도를 최고 98%까지 확인할 수 있었다. 향후 연구를 통하여 정밀도, 재현율 등 특정 지표의 향상을 목표로 할 수 있다.

ABSTRACT

According to Symantec's 2018 internet security threat report, The number of new mobile malware variants increased by 54 percent in 2017, as compared to 2016. And last year, there were an average of 24,000 malicious mobile applications blocked each day. Existing signature-based technologies of malware detection have limitations. So, malware detection technique through machine learning is being researched to detect malware variant. However, even in the case of applying machine learning, if the proper features of the malware are not properly selected, the machine learning cannot be shown correctly. We are focusing on feature selection method to find the features of malware variant in this research.

Keywords: Android, Malware, Machine Learning, Feature Selection

I. 서 론

2018년 가트너에 따르면 전 세계 모바일 운영체제(Operating System) 시장 점유율은 안드로이드(88%), iOS(11.9%) 및 기타 OS 플랫폼(0.1%)으로 구성된다[1].

안드로이드 운영체제가 가장 높은 점유율을 차지함에 따라 모바일 위협의 대부분이 안드로이드 운영체제를 대상으로 하고 있다. 2017년과 2018년을 비교하면, 공격적인 광고 기반 악성 앱(malware)이 49% 증가하고 가짜 앱(fake app)이 24% 증가하였고, 특히 은행서비스의 악성 앱은 150% 증가하였다[2]. 그리고 2018년 시만텍의 "인터넷보안 위협 보고서"에 따르면 변종 악성 앱은 54% 증가하였고, 매일 24,000개의 악성 앱이 차단되고 있다[3].

이렇듯 악성 앱은 계속해서 증가하고 있으며, API 호출 구조 변경, 쓰레기 코드 삽입 등을 통해 악성 앱 탐지를 회피한다.

이러한 악성 앱에 대한 대응책으로 여러 가지 방법들이 제시되었지만, 그중에서 코드의 시그니처를

Received(10. 08. 2019), Modified(1st: 01. 23. 2020, 2nd: 03. 03. 2020), Accepted(03. 23. 2020)
† 주저자, mrboo84@korea.ac.kr
‡ 교신저자, kevinlee@korea.ac.kr(Corresponding author)

기반으로 블랙 리스트 혹은 화이트 리스트를 작성하고 이를 이용해 악성 앱을 탐지하는 방법이 주로 사용되었다. 하지만 이러한 기법은 기존에 알려진 악성 앱을 탐지하는 데는 적합하지만 새롭게 등장한 악성 앱은 시그니처가 변경되어 탐지하는 데는 한계가 있다. 한편 역공학이 제시하는 다양한 분석 방법을 이용해 코드의 악성 여부를 전문가가 직접 밝혀내는 방법이 있다. 이는 악성 앱을 가장 정확하게 분석하는 방법이지만 역공학 분야의 전문적인 지식을 가진 전문가를 필요로 할 뿐 아니라 앱을 분석하는데 많은 시간이 요구되어 대량의 악성 앱을 실시간에 분석하기는 어렵다는 단점이 있다[4][5].

최근 기계학습 분야의 성장으로 시그니처 기반의 악성 앱 탐지 기법이 가진 단점인 신변종 탐지와 오탐율 문제점을 해결하고자 하는 다양한 연구들이 진행되고 있다[6-12]. 하지만, 기계학습을 이용하는 경우에도 입력값인 특성 집합에 의존하여 정상 앱과 악성 앱을 구분하여야 한다[4]. 그 결과, 학습에 사용된 특성이 일반적인 경우 모든 앱을 악성으로 분류하게 되고, 특성이 구체적인 경우 동일한 특성을 가진 악성 앱을 분류하지만, 변종 악성 앱의 경우 특성이 변경되어 분류할 수 없게 된다[13][21]. 이렇듯, 기계학습을 이용해서 학습을 시키고 악성 앱을 판별하기 위해서는 적합한 기계학습의 선택 못지않게 사용하는 특성 집합이 중요한 의미를 가진다[10].

II. 관련 연구

2.1 안드로이드 앱의 정적 분석

안드로이드 앱은 APK 파일 형식을 가지고 있으며, APK 파일은 DEX 파일, 자원(resources), 자산(assets), 인증서 정보와 매니페스트(manifest) 파일을 포함하고 있다[14].

DEX 파일은 안드로이드 앱이 실행되기 위한 실제 실행 코드가 기록된 파일로 보통 APK 파일 내부에 Classes.dex 파일로 존재한다. DEX 파일은 Header, String IDs, Type IDs, Proto IDs, Field IDs, Method IDs, Class Defs, Data, Link Data 섹션(Section)으로 구성되어 있으며, 전체 문자열 정보, 반환 값 정보, 메소드 정보, 앱에 있는 모든 클래스에 대한 정보가 저장되어 있다. 이렇듯, DEX 파일을 통해 클래스의 타입, 상위 클래스 존재 여부, 인터페이스 존재 여부 등의 정보를 파악할 수 있다 있다 [15][16].

매니페스트 파일에는 많은 정보를 포함하고 있다. 첫 번째로, 시스템에서 고유한 앱 식별자로 활용되는 앱의 패키지 이름을 선언한다. 두 번째로, 앱에서 사용하는 소프트웨어 및 하드웨어 기능 선언한다. 세 번째로, 앱의 구성 요소(activity, service, broadcast receiver, provider)의 기본 속성을 정의한다. 뿐만 아니라, 앱이 처리할 수 있는 기기 구성의 종류, 그리고 구성 요소가 어떻게 시작되는지 설명하는 인텐트 필터(intent filter)와 같은 기능을 선언한다. 네 번째로, 앱이 시스템 또는 다른 앱의 보호된 부분에 액세스하기 위해 필요한 권한(permission)을 선언한다. 권한은 민감한 사용자 데이터와 특정 시스템에 접근하고자 하는 경우 반드시 선언하여야 한다. 권한 단계에 따라서 시스템에서 자동으로 권한을 부여하거나, 사용자가 권한 요청을 승인하여야 사용자 데이터와 특정 시스템에 접근할 수 있다[29].

따라서 DEX 파일 분석을 통해 string, class, method의 영역에 접근하여 API 및 API흐름까지 추출할 수 있으며, 매니페스트 파일 분석을 통하여 앱이 실행 조건 및 앱이 접근하는 정보를 추출할 수 있다. 이러한 정보는 Androguard를 이용하여 추출이 가능하다[26].

2.2 기계학습을 활용하기 위한 특성 선택 방법

특성 선택(feature Selection)은 데이터 셋의 여러 가지 항목 중 기계학습 모델이 학습하고 결과값을 도출해낼 때에 가장 큰 연관성을 갖고 있는 항목을 골라내는 과정이다. 특성 선택을 통하여 학습시간 단축, 과적합(overfitting) 문제 해결, 모델 간 소화 등의 효과를 얻을 수 있다[18][19].

특성 선택 방법은 특성의 중요도를 평가한 랭킹을 기반으로 가장 중요도가 낮은 특성부터 하나씩 제거해 나가는 필터(filter) 방법, 기계학습 알고리즘에서 어떠한 특성 집합이 가장 좋은 성능을 갖는지 찾아내는 래퍼(wrapper) 방법, 모델 자체에 특성 선택 기능이 추가되어 있는 임베디드(embedded) 방법, 이렇게 총 3가지 방법으로 나뉜다[17].

필터 방법은 개별 특성과 라벨과의 관계성을 상관계수나 정보 이론적인 수치로 측정하여 순위를 매긴 후 주어진 임계치를 기준으로 특성들을 선택하거나 제거하는 방법이다. 필터 방법은 계산량이 적어 속도

가 빠르다는 장점이 있으나 상관성이 높은 비슷한 특성들이 중복해 사용될 수 있고 특성 조합이 주는 창발적(emergent) 특성을 무시하게 되는 단점이 거론된다. 이에 반해 래퍼 방법은 사용하는 기계학습 방법의 성능을 직접 목적함수로 사용하여 이를 극대화하는 특성 조합을 해공간으로부터 탐색하는 방법을 사용한다. 단점으로는 주어진 특성 집합으로부터 지수승개의 가능한 특성 조합을 대상으로 탐색을 해야 하므로 많은 시간이 걸린다는 점과 사용하는 기계학습 방법에 과적합 문제를 발생시킬 수 있다는 점이 거론된다. 하지만 필터 방법에 비해 명시적으로 분류기의 성능을 사용한다는 점에서 일반적으로 탐지율이 높은 것으로 알려져 있다[4][24]. 임베디드 방법은 래퍼 방법과 동일하게 특성 집합을 평가하지만 평가 시기가 기계학습과 동시에 일어난다. 임베디드 방법은 기계학습과 동시에 특성 선택이 수행되기 때문에 래퍼 방법의 큰 시간 복잡도를 다소 해결할 수 있지만 여전히 분류 정확도로 특성 집합을 평가하기 때문에 많은 시간이 걸리는 작업이다[27]. 이렇듯, 기계학습의 정확도 향상을 위해서는 특성을 고려하여 적절한 특성 선택 방법을 사용하여야 한다[19].

2.3 구글 지원 앱 검증 방법(SafetyNet)

구글은 안드로이드 앱이 보안 위협으로부터 앱을 보호할 수 있는 기능 제공한다. 기기 조작, 잘못된 URL, 잠재적으로 유해한 앱, 가짜 사용자 등 보안 위협으로부터 앱을 보호할 수 있는 일련의 서비스와 API를 제공한다.

앱을 실행하는 기기가 안드로이드 호환성 테스트를 충족하는지 확인하여 기기의 무결성 검사를 진행할 수 있고, URL 확인을 사용하여 URL이 알려진 위협을 가하는지 여부를 파악할 수 있다. 뿐만 아니라, 악성 트래픽으로부터 앱을 보호하기 위해 reCAPTCHA 사용할 수 있다.

특히, Verify Apps API를 이용하는 경우, 앱이 기기의 앱 인증 기능과 상호작용하여 잠재적으로 위험한 앱으로부터 기기를 보호할 수 있다. 이 기능을 사용하면 중요한 정보를 숨기거나 왜곡하여 구글의 소프트웨어 정책 또는 개발자 정책을 위반한 앱을 탐지할 수 있으며, 기기에 다른 소스에서 받은 잠재적으로 위험한 앱을 탐지할 수 있다[30].

2.4 기계학습을 이용한 악성 안드로이드 앱 탐지

“2017 정보보호 R&D 데이터 챌린지”의 예선에서 제공한 학습용 데이터 셋을 가지고 진행된 타 연구를 볼 때, 정상 앱과 악성 앱이 가지고 있는 권한 정보와 API정보를 분석하여 기계학습에 활용하였다[28]. 사용한 특성 선택 방법은 명확하게 공개되지 않았지만, 정상 앱과 악성 앱에 사용한 권한의 빈도수를 분석하여 빈도 수의 차이가 크거나, 악성 앱에서 사용 빈도 수가 높은 권한을 특성을 활용한 것으로 보인다. 또한, 악성 앱에서 권한을 사용하는 빈도수가 높은 것을 활용하여 앱이 가지고 있는 권한 수를 새로운 특성으로 생성하여 분석에 활용하였다.

심층 신경망(deep neural network)을 이용하여 추출한 권한 특성 22개와 권한의 빈도수, 그리고 API 특성 78개를 입력 값으로 활용한 결과 97~98%의 정확도를 보였다.

III. 악성 앱 탐지 모델

기계학습은 특성 집합에 의존하여 정상 앱과 악성 앱을 선택하기 때문에, 기계학습을 기반으로 하는 악성 앱 자동 분류기의 성능은 선택된 특성 조합의 구성 및 크기에 좌우된다[10].

본 연구에서는 안드로이드 APK 파일 내의 권한(permission) 정보와 인텐트(intent) 정보에 대한 정적 분석을 통해 악성 앱 탐지를 위한 특성을 연구하고, 해당 특성을 기계학습 모델의 학습 데이터로 활용하기 위한 특성 선택 단계에서 개선된 특성 선택 방법을 제시하여 안드로이드 APK파일의 특성 집합에서 악성 앱의 특성을 선택하여 기계학습의 정확도를 높일 수 있도록 하였다.

악성 앱 탐지 모델의 학습 과정은 데이터 전처리(data preprocessing), 특성 추출(feature extraction), 특성 선택(feature selection), 모델 학습(model learning), 모델 평가(model evaluation)로 구분되어 진행되며, 특성 선택 단계를 “3.2 특성 선택”과 “3.3 가중치 적용 특성 선택”로 구분하여 비교 검증하였다.[20][23].

3.1 앱 데이터 추출 및 전처리

악성 앱 분석을 위한 데이터 셋은 “2017 정보보호 R&D 데이터 챌린지” 예선에서 제공한 학습용

(train) 데이터 셋(정상 APK파일 1500개, 악성 APK파일 500개)을 활용하였다.

Androguard를 이용하여 "Fig. 1"과 같이 매니페스트 파일이 가지고 있는 정보(권한, 인텐트 필터)를 추출하여 앱 데이터 분석을 수행하였다[22][26]. 학습용 데이터 셋에서 추출한 권한 정보 164개와 인텐트 정보 115개를 "Fig. 2"와 같이 정리하여 기계학습에 활용하였다.

```
For apk in apk_list
    app, _, _ = androguard.AnalyzeAPK(apk)
    name = app. get_app_name ()
    permission = app.get_permissions()
    actions = app.get_intent_fileter()

    p_map.insert(name, p)
    a_map.insert(app, a)
Next
```

Fig. 1. Pseudocode of preprocessing

index	Permission 1	Permission 2	…	B/M
1	0	0	…	1
2	1	1	…	0
…	…	…	…	…
2000	0	0	…	1

Fig. 2. Example of pre-processed dataset

3.2 특성 선택

악성 앱 탐지를 위한 특성 선택 방법으로는 필터 방법의 알고리즘인 Chi-Squre, ANOVA F-value와 래퍼 방법인 REF, 임베디드 방법인 Lasso 네 가지 알고리즘을 이용하여 25개의 특성을 선택하였으며, 특성 선택 방법별 순위는 "Table 1, 2"과 같다.

Table 1. Result of feasure selection (Chi-squared, ANOVA F-value)

	Chi-squared	ANOVA F-value
1	(P)SEND_SMS	(P)READ_PHONE_STATE
2	(P)READ_SMS	(A)BOOT_COMPLETED
3	(P)INSTALL_PACKAGES	(P)SEND_SMS
4	(P)RECEIVE_SMS	(P)RECEIVE_SMS
5	(A)BOOT_COMPLETED	(P)RECEIVE_BOOT_COMPLETED
6	(P)PERSISTENT_ACTIVITY	(P)READ_SMS
7	(P)FORCE_STOP_PACKAGES	(P)WRITE_EXTERNAL_STORAGE
8	(PG)SYSTEM_TOOLS	(P)ACCESS_WIFI_STATE
9	(P)EXPAND_STATUS_BAR	(P)GET_TASKS
10	(A)SET_WALLPAPER	(P)INSTALL_PACKAGES
11	(P)SET_WALLPAPER_HINTS	(P)MOUNT_UNMOUNT_FILESYSTEMS
12	(A)PHONE_STATE	(P)WAKE_LOCK
13	(A)DATA_SMS_RECEIVED	(P)READ_LOGS
14	(P)READ_EXTERNAL_STORAGE	(P)INTERNET
15	(P)CHANGE_WIFI_STATE	(P)RESTART_PACKAGES
16	(P)WRITE_SETTINGS	(P)CHANGE_WIFI_STATE
17	(P)MODIFY_PHONE_STATE	(A)PACKAGE_REMOVED
18	(P)READ_LOGS	(P)KILL_BACKGROUND_PROCESSES
19	(P)RESTART_PACKAGES	(P)ACCESS_COARSE_LOCATION
20	(P)KILL_BACKGROUND_PROCESSES	(P)VIBRATE
21	(P)RECEIVE_BOOT_COMPLETED	(P)WRITE_APN_SETTINGS
22	(P)BIND_APPWIDGET	(A)PACKAGE_ADDED
23	(P)WRITE_APN_SETTINGS	(P)PERSISTENT_ACTIVITY
24	(A)PACKAGE_ADDED	(P)FORCE_STOP_PACKAGES
25	(P)GET_TASKS	(PG)SYSTEM_TOOLS

Table 2. Result of feasure selection (Lasso, RFE)

	Lasso	RFE
1	(A)MAIN	(A)MAIN
2	(P)ACCESS_WIFI_STATE	(P)ACCESS_WIFI_STATE
3	(P)INTERNET	(P)INTERNET
4	(P)READ_PHONE_STATE	(P)READ_PHONE_STATE
5	(P)ACCESS_NETWORK_STATE	(P)ACCESS_NETWORK_STATE
6	(P)WRITE_EXTERNAL_STORAGE	(P)WRITE_EXTERNAL_STORAGE
7	(P)ACCESS_COARSE_LOCATION	(P)ACCESS_COARSE_LOCATION
8	(P)WAKE_LOCK	(P)WAKE_LOCK
9	(P)RECEIVE_BOOT_COMPLETED	(P)RECEIVE_BOOT_COMPLETED
10	(P)VIBRATE	(P)VIBRATE
11	(P)ACCESS_FINE_LOCATION	(P)ACCESS_FINE_LOCATION
12	(A)BOOT_COMPLETED	(A)BOOT_COMPLETED

	Lasso	RFE
13	(P)MOUNT_UNMOUNT_FILESYSTEMS	(A)SEND
14	(A)VIEW	(P)SET_WALLPAPER
15	(A)USER_PRESENT	(P)MOUNT_UNMOUNT_FILESYSTEMS
16	(P)CALL_PHONE	(P)BIND_WALLPAPER
17	(P)GET_ACCOUNTS	(A)VIEW
18	(P)GET_TASKS	(A)USER_PRESENT
19	(P)RECEIVE_SMS	(P)CHANGE_WIFI_STATE
20	(P)SEND_SMS	(P)BIND_DEVICE_ADMIN
21	(P)READ_SMS	(P)RESTART_PACKAGES
22	(P)READ_EXTERNAL_STORAGE	(P)READ_LOGS
23	(A)PACKAGE_ADDED	(P)CALL_PHONE
24	(P)ACCESS_LOCATION_EXTRA_COMMANDS	(P)GET_ACCOUNTS
25	(P)SYSTEM_ALERT_WINDOW	(P)CAMERA

3.3 가중치 적용 특성 선택

"3.2 특성 선택"에서 선택된 특성들을 조합한 특성 선택 방법을 추가 제안하고, 제안한 방법을 사용한 모델과 "3.2 특성 선택"을 사용한 모델의 악성 앱 탐지 결과를 비교 분석하였다.

첫 번째 방법으로 위에서 선택한 4개의 특성 선택 방법에서 1회 이상 선택된 특성을 사용하는 방법, 두 번째 방법으로 4개의 특성 선택 방법에서 2회 이상 선택된 특성을 사용하는 방법, 세 번째 방법으로 첫 번째 방법에서 선택된 횟수를 가중치로 적용하는 방법, 네 번째 방법으로 두 번째 방법에서 선택된 횟수를 가중치로 적용하는 방법을 적용하였다.

```
FS_result_list = [Chi_result, ANOVA_result,
                  Lasso_result, RFE_result]

For FS_result in FS_result_list                    (1)
    For Feature in FS_result
        Result.add(Feature)
        Result_weight[Feature] ++
    Next
Next

For Feature in Result_count                        (2)
    IF Result_weight[Feature] == 1
        THEN Result.drop(Feature)
NEXT

Feature.multiply(Feature, Result_weight)           (3)
                                                   (4)
```

Fig. 3. Pseudocode of weighted feature selection method

각 단계별 의사코드(pseudocode)는 "Fig. 3"와 같다.

각 알고리즘 별로 특성이 선택된 횟수는 "Table 3."과 같다. 제안 방법 (3), (4)에서는 특성이 선택된 횟수를 가중치로 적용하기 때문에 4회 선택된 "Boot"관련 권한을 높은 가중치로 적용하며, "Wallpaper" 등 1회 선택된 권한을 낮은 가중치로 적용하거나 특성에서 제외하여 학습에 활용하였다.

Table 3. Weight of results

Weight	Permission
4	(A)BOOT_COMPLETED
	(P)RECEIVE_BOOT_COMPLETED
3	(P)READ_PHONE_STATE
	(P)ACCESS_WIFI_STATE
	(P)WRITE_EXTERNAL_STORAGE
	(P)INTERNET
	(P)SEND_SMS
	(P)RECEIVE_SMS
	(P)WAKE_LOCK
	(P)READ_SMS
	(P)ACCESS_COARSE_LOCATION
	(P)MOUNT_UNMOUNT_FILESYSTEMS
	(P)VIBRATE
	(P)CHANGE_WIFI_STATE
	(P)GET_TASKS
	(P)READ_LOGS
	(P)RESTART_PACKAGES
	(A)PACKAGE_ADDED
2	(A)MAIN
	(P)ACCESS_NETWORK_STATE
	(P)INSTALL_PACKAGES
	(P)ACCESS_FINE_LOCATION
	(P)PERSISTENT_ACTIVITY
	(A)VIEW
	(P)FORCE_STOP_PACKAGES
	(A)USER_PRESENT
	(PG)SYSTEM_TOOLS
	(P)READ_EXTERNAL_STORAGE
	(P)KILL_BACKGROUND_PROCESSES
	(P)CALL_PHONE
	(P)GET_ACCOUNTS
	(P)WRITE_APN_SETTINGS
1	(P)EXPAND_STATUS_BAR
	(A)SET_WALLPAPER
	(P)SET_WALLPAPER_HINTS
	(A)PHONE_STATE
	(A)DATA_SMS_RECEIVED
	(A)SEND
	(P)SET_WALLPAPER
	(P)BIND_WALLPAPER
	(P)WRITE_SETTINGS
	(A)PACKAGE_REMOVED
	(P)MODIFY_PHONE_STATE
	(P)BIND_DEVICE_ADMIN
	(P)BIND_APPWIDGET
	(P)ACCESS_LOCATION_EXTRA_COMMANDS
	(P)CAMERA
	(P)SYSTEM_ALERT_WINDOW

3.4 모델 생성

전체 2000개의 데이터를 3:1의 비율로 학습 데이터 셋(train set)과 검증 데이터 셋(vaildation set)으로 구분하여 기계학습에 활용하였다.

기계학습은 RF(Random Forest), AdaBoost, Bagging, C-SVM(C-Support Vector Machines), MLP(Multi-layer Perceptron), K-NN(K-nearest neighbors vote), K-Means clustering, 7가지 알고리즘을 이용하여 8가지 특성 선택 방법과 조합하여 총 56개의 모델을 구성하여 평가하였다. 기계학습은 파이썬(python) 기계학습 패키지인 사이킷런(scikit-learn)을 이용하여 구현하였으며, 각 기계학습 알고리즘에서 사용된 설정값은 "Table 4."와 같다.

Table 4. Setting of Machine Learning

ML	Setting
RF	50 decision tree classifier
AdaBoost	50 estimators (decision tree classifier)
Bagging	50 subset (decision tree classifier)
C-SVM	Penalty parameter(C)=1.0 kernel=RBF (Radial Basis Function)
MLP	Hidden layer=100 activation=relu optimizaion=adam batch size=200 learning rate=0.001
K-NN	uniform weights
K-Means	2 clusters

3.5 모델 평가

각 기계학습 모델의 악성 앱 분류 성능을 정확도(Accuracy), 정밀도(precision), 재현율(recall), 그리고 정밀도와 재현율의 조화평균 값인 F-Measure를 이용하여 수치를 표한하고, ROC curve(Receiver Operating Characteristic curve), AUC(Area under the ROC Curve)로 비교 분석하였다[25]. 그리고, 동일 데이터 셋으로 진행된 연구에서 선택된 22개의 특성을 이용하여 악성 분류 모델을 생성하고 본 연구와 비교 분석하였다[28].

정확도 분석 결과는 "Table 5."과 같으며, MLP와 (1) 특성 선택 방법을 이용하였을 때 가장 높은 정확도인 98%를 보였고, 평균적으로 (3) 특성 선택 방법을 이용하였을 때 가장 높은 정확도인 94.9%를 보였다. 정밀도와 재현율 분석 결과 (1) 특성 선택 방법을 이용하였을 때 평균 정밀도가 96.1%, 평균 재현율이 89.5%로 가장 높은 수치를 보이고 있음을 "Table 6."과 "Table 7."을 통하여 확인할 수 있다.

F-Measure 수치인 "Table 8."을 보면, (3) 특성 선택 방법을 이용하였을 때 가장 높은 수치인 97.4%를 보였으며, 평균적으로 (1) 특성 선택 방법을 이용하였을 때 가장 높은 수치인 92.7%를 보였다. 또한, 일반적인 특성 선택 방법보다 (1)~(4) 특성 선택 방법을 이용하였을 때 평균적으로 높은 F-Measure 수치를 보이고 있다.

평균적으로 높은 정확도를 보인 기계학습 알고리즘 4개의 ROC Curve, AUC 값을 비교해볼 때, 제안한 특성 선택 방법 4개 모두 높은 수치를 보였으며, 그 결과는 "Fig.8~11"과 같다. 가장 낮은 수

Table 5. Accuracy(%)

	previous study	Chi-squared	ANOVA	Lasso	RFE	(1)	(2)	(3)	(4)
RF	92.8	92	96.2	97	98	97.8	97.4	97.2	97.8
Ada Boost	92	91.2	92.4	96	93	95.8	94.8	94.6	93.8
Bag ging	93.2	92.4	95.2	96.2	95.6	97	97.4	97.6	97.2
C-SVM	86.6	91.2	90	87.4	90.6	92.6	92	95	95.6
MLP	93.8	91.6	95	95.6	95.4	98	96.4	97.8	96.4
K-NN	91.2	91.2	94	94.2	95.2	95.2	94.6	95.6	94.2
K-Means	85.6	78.8	86	78.6	84	82	86.6	86.8	87
Avg.	**90.7**	**87.8**	**90.8**	**92.1**	**93.1**	**94.0**	**94.1**	**94.9**	**94.5**

Table 6. Precision(%)

	previous study	Chi-squared	ANOVA	Lasso	RFE	(1)	(2)	(3)	(4)
RF	92.2	82.8	92.5	93.7	92.3	96.6	96.8	94.9	95.6
Ada Boost	93.4	90.1	88.7	90.4	90.6	94.2	90.8	93.4	92.9
Bag ging	95.5	83.6	92	92.5	88.7	94.3	93.3	97.7	95.2
C-SVM	87.2	87.9	91.4	98.4	90.4	95.6	94.7	95.3	95.3
MLP	94.5	85.9	91.8	91.5	89.2	97.3	95.8	96.2	93.5
K-NN	93	90	90.9	91	91.6	94.7	91.2	92.7	90.8
K-Means	84.9	100	94.5	100	87.6	100	100	89.8	85.4
Avg.	**91.5**	**88.63**	**91.7**	**93.9**	**90.1**	**96.1**	**94.7**	**94.3**	**92.7**

Table 7. Recall(%)

	previous study	Chi-squared	ANOVA	Lasso	RFE	(1)	(2)	(3)	(4)
RF	98.3	81.3	91.9	93.8	94	97.4	95.7	96.1	98.5
Ada Boost	96.3	71	79.8	92.5	80.7	92.4	87.9	86.1	84.6
Bag ging	95.5	82.1	89.3	92.4	95.4	96.6	96.7	97.1	93.7
C-SVM	95.9	67.5	65.5	50.4	71	74.3	72.9	83.7	86.9
MLP	97.3	77.9	88.2	92	92	95.2	91.6	95.2	94
K-NN	95.8	75.9	83.3	84.3	89.3	85.7	89.1	94.2	89.5
K-Means	97.8	12.4	44.	14.8	44.1	84.6	61.4	51.2	63.7
Avg.	**96.7**	**66.9**	**77.4**	**74.3**	**80.9**	**89.5**	**85**	**86.2**	**87.3**

Table 8. F-Measure(%)

	previous study	Chi-squared	ANOVA	Lasso	RFE	(1)	(2)	(3)	(4)
RF	95.2	82.1	92.3	93.8	93.2	97.0	96.3	95.5	97.1
Ada Boost	94.8	79.5	84.1	91.5	85.4	93.4	89.3	89.6	88.6
Bag ging	95.5	82.9	90.7	92.5	92.0	95.5	95.0	97.4	94.5
C-SVM	91.3	76.4	76.3	66.7	79.6	83.7	82.4	89.2	90.9
MLP	95.8	81.8	90.0	91.8	90.7	96.3	93.7	95.7	93.8
K-NN	94.4	82.4	87.0	87.6	90.5	90.0	90.2	93.5	90.2
K-Means	90.3	22.1	60.1	25.9	58.8	91.7	76.1	65.2	73.1
Avg.	**93.9**	**76.3**	**84.0**	**83.0**	**85.3**	**92.7**	**89.6**	**90.1**	**89.9**

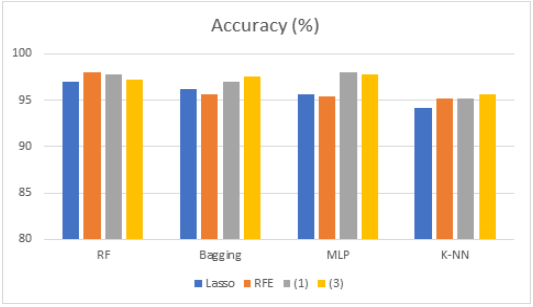

Fig. 4. Comparison of accuracy

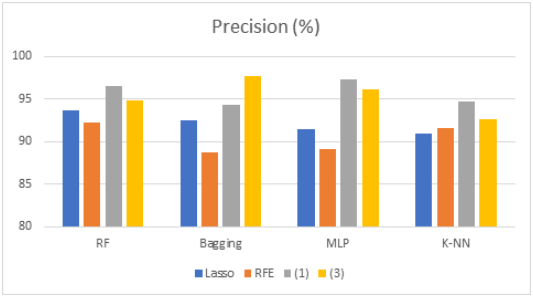

Fig. 5. Comparison of precision

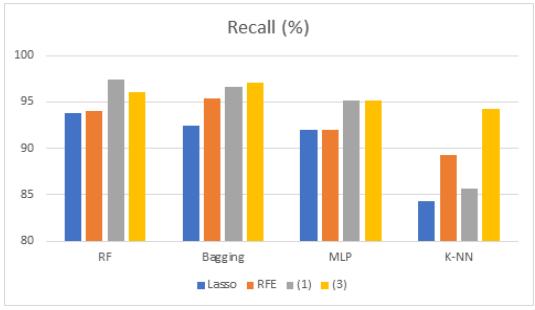

Fig. 6. Comparison of recall

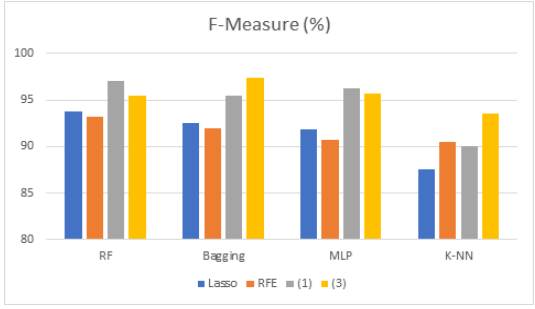

Fig. 7. Comparison of F-measure

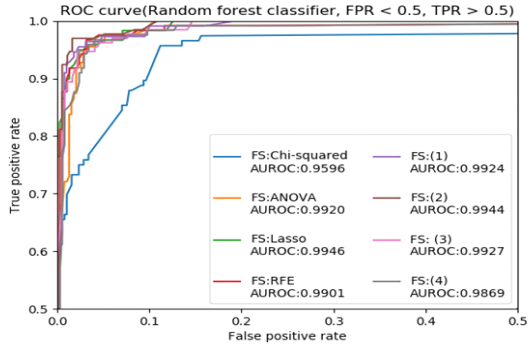

Fig. 8. ROC curve, AUROC(RF)

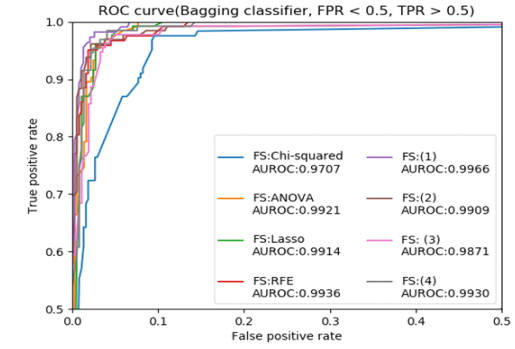

Fig. 9. ROC curve, AUROC(Bagging)

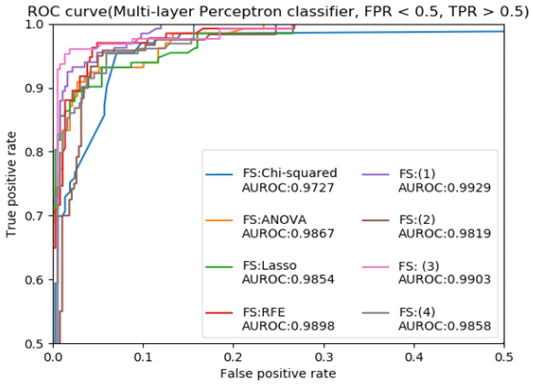

Fig. 10. ROC curve, AUROC(MLP)

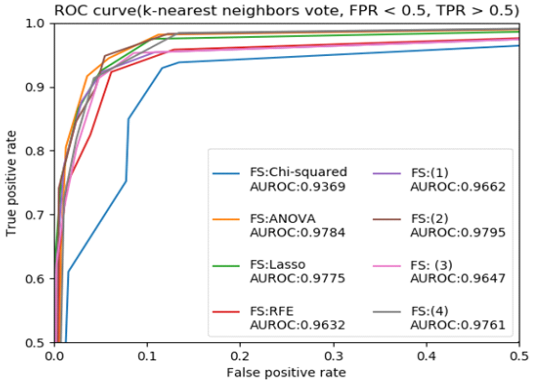

Fig. 11. ROC curve, AUROC(K-NN)

치를 보이는 K-NN을 제외하는 경우 가장 낮은 AUC값이 0.9858의 수준을 보이고 있다.

3.6 결과 분석

기계학습을 이용하여 악성 앱을 탐지하는 경우 기계학습의 설정 값의 변화를 주지 않더라도, 입력 데이터인 특성 선택 방법의 변화만으로도 악성 앱 탐지 정확도를 높일 수 있는 것을 확인하였다. 뿐만 아니라, 알려진 특성 선택 방법을 활용하여 서로 조합하여 특성 선택 방법으로 사용하는 경우에 기존 특성 선택 방법과 비교하여 평균적으로 높은 정확도를 보이는 것을 확인할 수 있었다.

동일 데이터 셋으로 진행된 대회의 발표자료를 볼 때 97~98%의 정확도(accuracy)를 보이고 있다[28]. 해당 연구결과를 참고하여 특성 선택 방법을 사용하는 경우 약 93%의 정확도를 보였으며, 본 연구에서 활용한 특성 선택 방법을 사용하는 경우 약 97%의 정확도를 보여 약 4% 정도의 성능 향상을 보였다. 이전 연구와 전처리 과정과 학습에 사용하는 알고리즘 등이 서로 달라 제안한 방법과 명확한 비교가 어렵지만, 이전 연구에서 보여준 정확도와 본 연구에서 보여진 정확도가 유사한 수준을 보이고 있음을 확인할 수 있다. 또한, 기존 연구에서는 특성 선택을 위하여 직접 데이터를 분석하고 학습 알고리즘에 맞는 최적화 과정을 진행한 것에 비해 제안한 방법을 이용하는 경우, 데이터의 특성을 분석하는 시간을 조금 더 절약할 수 있다.

결론적으로 기계학습의 정확도를 높이기 위해서는 기계학습의 설정 값 뿐만 아니라, 특성 선택 방법도 개선하여야 한다. 더불어, 서로 다른 특성 선택 방법을 가중치를 적용하여 조합하는 것도 좋은 특성 선택 방법이 될 수 있다.

IV. 결 론

악성 앱을 정확하게 탐지하게 위해서는 빠르게 변화하는 악성 앱의 특성을 찾아내고 기계학습 모델에 적용이 필요하다. 따라서 변화하는 악성 앱의 특성을 새롭게 선택하고 이를 모델에 적용해야할 필요가 있다. 이를 위해, 본 연구에서는 악성 앱 탐지를 위한 특성 선택 방법을 제안하였다.

향후, 다양한 특성 선택 방법의 조합과 가중치를 부과하여 악성 앱 탐지율을 높일 수 있을 것으로 예상된다. 또한, 특정 데이터 셋에서 과적합되지 않도록 지속적으로 데이터 셋을 갱신하고, 능동적으로 특성 탐지 및 모델 구현이 요구된다. 뿐만 아니라, 구글 지원 앱 검증 방법 등과 같이 알려진 악성 앱 탐지 방법과 교차 검증을 통하여 모델의 성능 향상을 기대해 볼 수 있다.

본 연구에서는 기계학습 모델에 적용할 학습용 및 검증용 데이터 셋이 제한되어, 특성이 변경된 신종 악성 앱을 가정하고 탐지하기에 어려움이 있었다. 또한, 모델의 평가를 정확도를 높이는데 초점을 맞추었다는 한계가 있다. 향후, 백신에서 기계학습을 활용하기 위해서는 정밀도(Precision)에 초점을 맞추어 정상 앱이 악성 앱으로 오인 받는 부분(False Positive) 부분을 최소화하여 정상 앱이 삭제되거나, 격리되는 경우가 없어야 할 것이다.

References

[1] Gartner, "Gartner Says Huawei Secured No. 2 Worldwide Smartphone Vendor Spot, Surpassing Apple in Second Quarter 2018" https://www.gartner.com/en/newsroom/press-releases/2018-08-28-gartner-says-huawei-secured-no-2-worldwide-smartphone-vendor-spot-surpassing-apple-in-second-quarter, (accessed 09. 08. 2019).

[2] Avast blog, "Mobile threats today" https://blog.avast.com/avast-mobile-threat-predictions, (accessed 09. 08. 2019).

[3] Symantec, "Symantec internet security threat report," ISTR-23-2018, p.2, Mar. 2018.

[4] Jin-Gul Joo, In-Seon Jeong, Seung-Ho Kang, "An Optimal Feature Selection Method to Detect Malwares in Real Time Using Machine Learning," Journal of Korea, Multimedia Society, vol. 22, no. 2, pp. 203-209, Feb. 2019.

[5] Darell JJ Tan, Tong-Wei Chua, Vrizlynn LL Thing, et al. "Securing android: A survey, taxonomy, and challenges," ACM Comput. Surv, vol. 47, no. 4, Article 58, p. 45, May. 2015.

[6] K. Rieck, T. Holz, C. Willems, P. Dussel and P. Laskov, "Learning and classification of malware behavior," Proceeding of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 108-125, Jul. 2008.

[7] B. Sun, Q. Li, Y. Quo, Q. Wen, X. Lin and W. Liu, "Malware family classification method based on static feature extraction," Proceeding of the 3rd IEEE International Conference on Computer and Communications (ICCC), pp. 507-513, Dec. 2017.

[8] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," Proceedings of the 6th ACM conference on data and application security and privacy, pp. 183-194, Mar. 2016.

[9] R. Veeramani, and N. Rai. "Windows API based malware detection and framework analysis," International Conference on Networks and Cyber Security, pp. 25-29, Jan. 2012.

[10] M.H. Nguyen, D.L. Nguyen, X.M. Nguyen, and T.T. Quan, "Auto-Detection of Sophisticated Malware Using Lazy-Binding Control Flow Graph and Deep Learning," Computers and Security, vol. 76, pp.128-155, Jul. 2018.

[11] C.I. Rene and J. Abdullah, "Malicious Code Intrusion Detection Using Machine Learning And Indicators of Compromise," International Journal of Computer Science and Information Security, vol. 15, no. 9, pp. 160-171, Sep. 2017.

[12] P. Singhal and N. Raul, "Malware Detection Module Using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks," International Journal of Network Security and Its Applications, vol. 4, no. 1, pp. 61-67, Feb. 2012.

[13] Karthik Raman, "Selecting Features to Classify Malware," InfoSec Southwest, Mar. 2012.

[14] Yingxian Chang et al, "Vulnerability Parser: A Static Vulnerability Analysis System for Android Applications," Journal of Physics: Conference Series, vol. 1288, 012053,

no. 1, Aug. 2019.
[15] The Android Open Source Project, "DexDalvik Executable Format" https://source.android.com/devices/tech/dalvik/dex-format, (accessed 09. 08. 2019).
[16] Keith Makan and Scott Alexander-Bown, Android Security Cookbook, Packt Publishing, Dec. 2013.
[17] L L et al, "Static analysis of android apps: A systematic literature review," Information and Software Technology, vol. 88, pp. 67-95, Aug. 2017.
[18] Isabelle Guyon, André Elisseeff, "An Introduction to Variable and Feature Selection," Journal of Machine Learning Research 3, pp. 1157-1182, Mar. 2003.
[19] Isabelle Guyon, André Elisseeff, "An introduction to feature extraction," Feature extraction, Studies in Fuzziness and Soft Computing, vol, 207, pp. 1-25, 2006.
[20] Lei Wang, Xin Yan, et al, "Prediction of RNA-Protein Interactions by Combining Deep Convolutional Neural Network with Feature Selection Ensemble Method," Journal of Theoretical Biology, vol. 461, pp. 230-238, Jan. 2019.
[21] Scott Schneider, Kent Griffin, "Modeling goodware characteristics to reduce false positive malware signatures," US Patent 8,028,338, Sep. 2011.
[22] Zarni Aung, Win Zaw, "Permission-Based Android Malware Detection," International Journal of Scientific & Technology Research, vol. 2, ISSUE 3, pp. 228-234, Mar. 2013.
[23] Christina Eusanio, "Machine Learning for the Detection of Mobile Malware on Android Devices," Proceedings of the Second Annual Data Science Symposium, Mercyhurst University, Erie, PA, pp. 19-23, May. 2019.
[24] Vipin Kumar, Sonajharia Minz "Feature Selection: A literature Review," Smart Computing Review, vol. 4, no. 3, pp. 211-229, Jun. 2014.
[25] Fairuz Amalina Narudin, Ali Feizollah, Nor Badrul Anuar, Abdullah Gani, "Evaluation of machine learning classifiers for mobile malware detection," Soft Computing, pp. 343 - 357, Jan. 2016.
[26] Androguard, "androguard", https://github.com/androguard/androguard, (accessed 09. 08. 2019).
[27] Joo-ho In, Jung-ho Kim, Soo-hoan Chae, "Combined Feature Set and Hybrid Feature Selection Method for Effective Document Classification," Journal of Internet Computing and Services,v.14 no.5, pp. 49 - 57, Oct. 2013.
[28] Information Security R&D Data Challenge, "Mobile malware" https://www.kisis.or.kr/kisis/subIndex/282.do, (accessed 09. 08. 2019).
[29] Android Developer, "Manifest" https://developer.android.com/guide/topics/manifest/manifest-intro, (accessed 09. 08. 2019).
[30] Android Developer, "Safetynet" https://developer.android.com/training/safetynet, (accessed 09. 08. 2019).

〈저자소개〉

부 주 훈 (Joohun Boo) 정회원
2012년 8월: 중앙대학교 컴퓨터공학부 졸업
2016년 3월~현재: 고려대학교 정보보호학과 석사과정
〈관심분야〉 정보보호, 데이터분석

이 경 호 (Kyung-ho Lee) 종신회원
1989년 8월: 서강대학교 수학과 학사
1997년 8월: 서강대학교 정보통신대학원 석사 졸업
2009년 8월: 고려대학교 정보보호대학원 박사 졸업
2017년 2월~2019년 2월: 고려대학교 정보전산처장
2011년~현재: 고려대학교 정보보호대학원 교수
〈관심분야〉 정보보호 정책, 개인정보보호 정책, 위험관리, 머신러닝, 블록체인