논문 2024-3-3 http://dx.doi.org/10.29056/jsav.2024.09.03

# K-평균 클러스터링 기반 안드로이드 악성 앱 탐지 기법의 지속가능성 개선

정성원*, 안석현**, 조성제*† , 김동재**, 황영섭***

# Enhancing Sustainability of an Android Malware Detection Technique using K-means Clustering

Seongwon Jeong*, Seokhyun Ann**, Seong-je Cho*† , Dongjae Kim**, Young-sup Hwang***


## 요 약

전통적인 기계학습 기반 안드로이드 악성 앱 탐지 기법은 개념 드리프트(concept drift)로 인해 새로운 유형의 악성 앱들을 탐지하는데 한계가 있다. 즉, 전통적인 기계학습 기반의 악성 앱 탐지 기법은 지속 가능하지 않을 수 있다. 개념 드리프트는 시간이 지남에 따라 악성 앱 특징의 진화와 이로 인한 기계학습 기반 탐지 모델의 성능 저하를 말한다. 본 논문에서는 API 호출 정보 및 기계학습을 사용하여 안드로이드 악성 앱을 탐지하는 방법의 지속가능성을 개선하는 기법을 제안한다. 제안 기법에서는, 먼저 K-평균 클러스터링으로 앱들을 그룹화한 후, 각 그룹별로 악성 앱을 탐지하는 분류모델을 개발한다. K-평균 클러스터링 과정에서는 최적의 k 값을 찾는 엘보우 방법(elbow method), 그리고 클러스터별 분류기에 임계값 지정과 초매개변수의 최적화 과정을 적용한다. 분류모델로는 랜덤 포레스트, K-최근접 이웃, AdaBoost를 사용한다. 실험 결과, 랜덤 포레스트 분류기가 가장 높은 성능을 보였는데, 마이크로-평균 방식으로 산출된 F1 점수와 AUT 수치가 기존의 전통적인 랜덤 포레스트 모델보다 각각 20.1%p, 20.4%p 개선되었다.

## Abstract

Traditional machine learning-based Android malicious app(malware) detection techniques have limitations in detecting new types of malware due to concept drift. In other words, traditional machine learning-based malware detection techniques may not be sustainable. Concept drift refers to the evolving nature of malware features over time and the resulting degradation in the performance of machine learning-based detection models In this paper, we propose a technique to improve the sustainability of the method for detecting Android malware using API call information and machine learning. In the proposed technique, apps are first grouped using K-means clustering, and then classification models are applied to detect malicious apps for each group. In the K-means clustering, the elbow method is used to find the optimal k value, and thresholding and hyperparameter optimization processes are applied to the classifiers for each cluster. The classifiers include random forest, K-nearest neighbor, and AdaBoost. The experimental results show that the random forest classifier showed the highest performance, with the F1 score and AUT value calculated by the micro-means method being improved by 20.1%p and 20.4%p, respectively, compared to the traditional random forest model.


**한글키워드 :** 기계학습, K-평균 클러스터링, 악성 앱 탐지, 지속가능성, 안드로이드 앱, 개념 드리프트

**keywords :** machine learning, K-means clustering, malicious app detection, sustainability, Android app, concept drift

* 단국대학교 소프트웨어학과
** 단국대학교 일반대학원 인공지능융합학과
*** 선문대학교 컴퓨터공학부
† 교신저자: 조성제(email: sjcho@dankook.ac.kr)
접수일자: 2024.09.07. 심사완료: 2024.09.17.
게재확정: 2024.09.20.

## 1. 서 론

기계학습 기반 악성코드 앱 탐지는 기존 시그니처(signature) 기반 백신 도구들과 함께, 안드로이드 악성 앱들을 탐지하는 방안으로 널리 사용되고 있다. 안드로이드 악성 앱을 탐지하는 기법과 도구들의 사용이 일반화됨에 따라, 이러한 탐지 기법과 도구들을 회피하거나 우회하려는 방법들도 개발되고 있다[1, 2].

즉, 최근 악성 앱들은 더욱 정교화되고 복잡해지고 있으며, 기존 보안 체계를 무력화시킬 수 있는 기법들을 적용하고 있다. 글로벌 보안회사 Kaspersky는 2024년 7월, Mandrake라는 고급 안드로이드 악성코드의 새로운 변종이 Google Play에서 탐지를 회피하였음을 보고한 바 있다[3].

한편, 전통적인 기계학습 기반 안드로이드 악성 앱 탐지 기법은 개념 드리프트(concept drift)로 인해 새로운 유형의 악성 앱들을 탐지하는데 한계가 있다[4, 5]. 이러한 악성 앱의 패턴 변화를 감지하기 위해 기계학습 기반 안드로이드 악성 앱 탐지 기법의 지속가능성(sustainability)을 개선하려는 연구들이 진행되고 있다[4-6].

본 논문에서는 API(Application Programming Interface) 호출 정보 및 기계학습 기반의 안드로이드 악성 앱 탐지와 지속가능성을 고려한 기법을 제안한다. 안드로이드 악성 앱을 탐지하는 기존의 기계학습 연구들에서 API 호출을 특징정보로 사용하였으며[4-12], 본 연구 역시 API를 특징정보로 채택하였다. 본 논문은 우리 연구진의 선행 연구[7]를 확장한 결과물이다.

기존 논문[7]에서는 먼저 K-평균 클러스터링(K-means)으로 앱들을 특징적으로 그룹화한 후, 각 그룹 별로 악성 앱을 탐지하는 분류기를 학습시키는 과정을 통해 시간적 편향으로 인한 개념 드리프트에 대응하는 모델을 개발했다. 모델은 과거의 2014년~2016년까지의 데이터를 학습시키고 2019~2021년의 데이터를 예측할 때 API 호출 과 빈도의 특성이 반영된 클러스터들의 분포 변화를 파악하였다. 위의 과정에서 본 논문은, K-means 과정에서는 최적의 $k$값을 찾는 엘보우 방법(elbow method), 그리고 클러스터별 분류기에 임계값 지정과 초매개변수의 최적화 과정을 적용한다. 분류모델로는 랜덤 포레스트, K-최근접 이웃(KNN), AdaBoost를 사용하여 지속가능성을 개선한다. 클러스터링 단계에서는 안드로이드 앱들의 분포 추이를 보이고, 분류 단계에서는 제안 기법의 성능을 평가한다. 실험 결과, 랜덤 포레스트 분류기가 가장 높은 성능을 보였는데, 마이크로-평균 방식으로 산출된 F1 점수와 AUT(Area Under Time)[4] 수치가 기존의 전통적인 랜덤 포레스트 모델보다 각각 20.1%p, 20.4%p 개선되었다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구들을 간단히 정리한다. 3장에서는 본 논문에서 제안하는 K-means 기반의 안드로이드 악성 앱 탐지 기법을 설명한다. 4장에서 실험설계 방법을 기술하고 5장에서는 실험 결과와 함께 주요 결과에 대해 논의한다. 6장에서 결론을 맺고 향후 연구 방향에 대해 제시한다.

## 2. 관련 연구

정재민 등[9]은 Google Android의 공개 정보와 Arp[10] 및 Aafer[11]에 의해 악성 소프트웨어 탐지에 효과적인 1,848개의 공식 API 호출을 보고하고 랜덤 포레스트(Random Forest) 모델을 이용하여 분류를 진행하였다.

Pendlebury[4] 등은 기계학습 기반 악성 앱 탐지가 해결된 문제인지에 대해 논의하며, 실험적 편향(공간적 편향(Spatial bias)과 시간적 편향(Temporal bias))으로 인해 예측 품질이 감소하

지 않고 성능이 부풀려짐을 지적했다. 공간적 편향은 현실의 도메인을 고려하지 않고 정상/악성 앱의 비율을 실험에 적용해 생기는 편향이고, 시간적 편향은 미래의 지식을 학습시켜 시간적으로 일관되지 않은 결과를 제공하는 편향이다. 그들은 시간이 지남에 따라 진화하는 악성 앱과 현실의 도메인을 고려, 현실을 모방하는 시간 및 공간 조건에서 분류를 수행한다. 또한 그들은 부풀려진 성능을 제한하는 전략으로 산출된 성능지표로 지속가능성을 평가하는 $AUT$를 제안했다.

조우상 등[5]은 2014~2015년 데이터를 사용해 학습된 모델이 시간이 지나면서 지속가능하지 않음을 밝혔으며, 특히 2019~2020년의 악성 앱에서 자주 사용된 API 호출이 정상 애플리케이션에서도 반복적으로 사용하여 모델의 성능이 감소함을 확인했다.

또한, 조우상 등[6]은 2014년부터 2020년까지의 데이터를 연도별로 조합하여 학습하고 2019~2021년과 같은 미래의 데이터에서도 좋은 성능을 유지할 수 있는지를 평가하였으며, 데이터의 분포가 크게 달라지는 3개 연도의 데이터 조합에서 가장 높은 성능과 지속가능성의 향상된 결과를 보고하였다.

정성원, 이승민등[7, 12]은 정상/악성 여부와 관계없이 유사한 API 호출 정보를 기반으로 클러스터링 기법을 적용한 이후 할당된 데이터를 바탕으로 분류모델을 학습시켜 악성 앱을 탐지하는 모델을 제안하였다. 제안 모델은 클러스터링 단계에서의 개체수, 클래스 불균형을 보고하였으며, 현실을 모방하는 시간적 조건에서 AUT 수치를 소폭 개선하였다.

기존 연구[7]와 본 논문의 주요 차이점은 클러스터링, 분류 단계로 이루어진 기존 모델의 최적화이며, 그 과정은 최적의 $k$값을 찾는 엘보우 방법, 클러스터별 분류기에 임계값 지정, 초매개변수의 탐색 과정을 포함한다.

## 3. K-평균 클러스터링 기반 안드로이드 악성 앱 탐지 기법

K-평균 클러스터링(K-means clustering)은 데이터들을 거리 기준으로 $k$개의 군집으로 나누는 비지도학습 방식이다. 본 모델의 클러스터링 단계에서는 APK의 정상 및 악성 여부와 관계없이, L2 Distance를 기준으로 유사한 API 호출 정보를 갖는 APK들을 $k$개의 클러스터에 그룹화하여 할당한다. 위의 방식을 통해 나눠진 클러스터들은 API 호출 종류 및 빈도의 차이를 반영하며, 이는 편향되지 않은 데이터를 기반으로 악성 앱 탐지 분류기를 적용하여 개념 드리프트의 발생을 우회할 수 있다는 점에서 지속가능성 개선의 측면에서 효과적이다.

유사한 API 호출을 가지는 데이터들은 $k$개의 클러스터에 할당된다. 학습 시에 할당된 훈련 데이터들은 클러스터별로 분류기를 학습시키며, 예측 시에 할당된 테스트 데이터들은 훈련 데이터로 기학습된 분류기들을 이용해 정상, 악성 클래스로 이진 분류된다.

### 3.1 K-평균 클러스터링에서 최적의 k값 탐색

최적의 $k$값(클러스터 수)을 결정하기 위해 엘보우 기법 (Elbow Method)을 사용한다[13-15]. 엘보우 기법은 클러스터 수에 따른 Within-cluster Sum of Squares (WSS) 값을 그래프에 플로팅 후, 그래프에서 팔꿈치 모양으로 꺾인 엘보우 지점을 찾아 최적의 $k$값을 선택한다. WSS값은 데이터 포인트와 클러스터의 중심점(centroid, $\mu$) 간의 거리 제곱합을 의미하며, 클러스터 내의 분산을 측정할 수 있다. 임의의 클러스터 $C_k$의 WSS값 $WSS_k$는 아래의 수식과 같이 정의한다.

$$WSS_k = \sum_{x_i \in C_k} \| x_i - \mu \|^2$$

엘보우 지점(Elbow Point)은 $k$를 증가시키면서 WSS의 감소율이 둔화하는 지점이다. 이 지점에서 WSS의 감소율이 급격히 완만해지며, 이 지점을 통해 가장 적합한 $k$값을 결정할 수 있다. 그림 1은 WSS의 플로팅 결과를 보이는데, $k$=3의 지점에서 완만한 경사 구간으로 전환되므로 본 모델의 최적의 $k$값은 3으로 판단하였다.

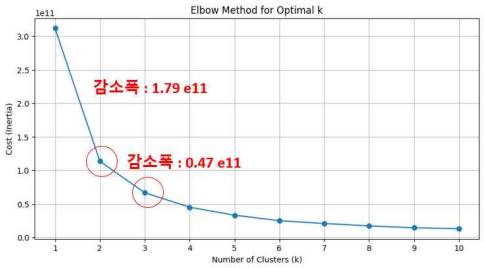


그림 1. 엘보우 기법을 통한 최적 클러스터 수 결정
Fig. 1. Determining the Optimal Number of Clusters Using the Elbow Method

### 3.2 클러스터별 악성 앱 탐지 모델

최적의 $k$값을 3으로 설정하여 데이터를 학습 및 예측 시, $k$개의 분류기에서 학습 및 예측이 수행되며 Random Forest (RF), K-Nearest Neighbors (KNN), AdaBoost의 여러 분류 알고리즘을 적용하여 최종 멀웨어 탐지 성능을 평가하고 비교한다. 그림 2는 K-means 분류모델의 구조를 나타낸다. 본 연구에서는 Scikit-Learn에서 제공하는 모델을 사용하였다[16].

또한 분류기들을 최적화시키기 위해 학습 단계에서 분류 모델들의 임계값(Threshold)과 초매개변수(Hyper Parameter)의 조절을 고려하였다. 임계값은 분류 모델의 출력확률에 대해 클래스 레이블을 결정하는 기준값이며, 초매개변수는 모델의 학습 과정이 시작되기 전에 설정되는 복잡도, 학습 방법, 알고리즘등을 조정하는 매개변수이다. 따라서 적절한 임계값과 초매개변수를 탐색하는 것은 분류의 정확도를 향상시킬 수 있다. 본 실험에서는 임계값의 경우 기존 K-means 기반 분류모델에서는 악성 앱을 정상 앱으로 분류

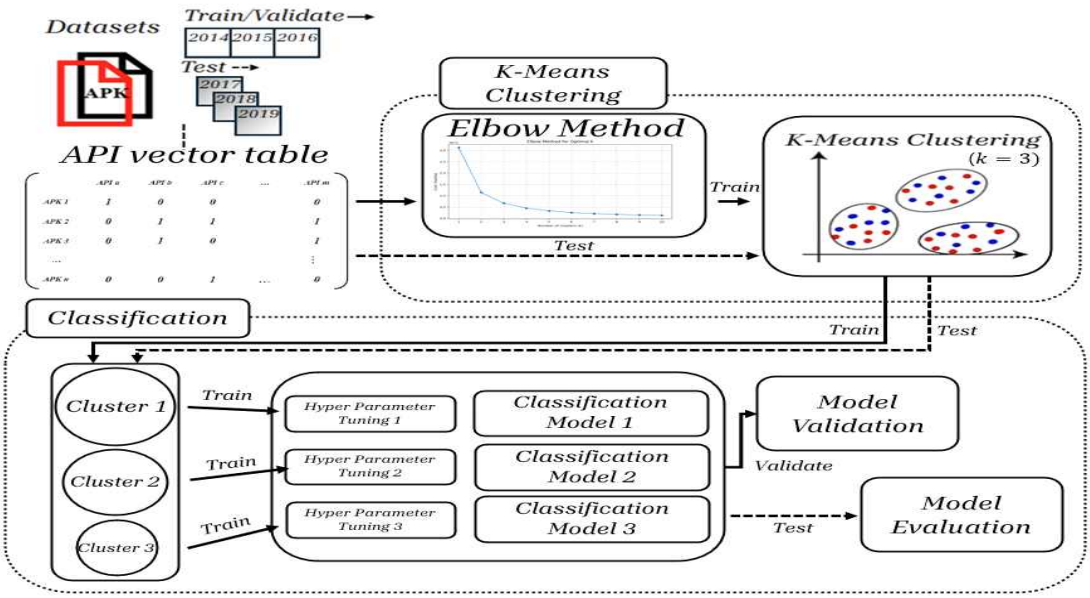


그림 2. K-평균 클러스터링 기반 악성 앱 탐지 모델의 구조
Fig 2. Structure of a Model for Detecting Malicious Apps using K-means Clustering

하는 FP의 비율이 현저히 높았음을 고려하여[7], 클러스터별로 분류모델들의 임계값을 0.50~0.90 범위의 값으로 지정하였다. 또한 초매개변수의 경우 그리드 탐색(Grid Search)을 통하여 최적의 경우를 선택했다. 그리드 탐색 과정에서는 5겹 교차검증을 수행하였으며, 모델별로 선정된 초매개변수와 임계값은 표 1과 표 2와 같다.

표 1. 초매개변수 최적화 결과
Table 1. Hyper-Parameter Optimization Results

| Hyper-Parameter | | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|---|
| RF | n_estimators | 300 | 100 | 300 |
| | max_depth | None | None | None |
| KNN | n_neighbors | 5 | 3 | 3 |
| Ada | n_estimators | 200 | 50 | 100 |
| | learning_rate | 1.0 | 0.01 | 0.1 |

표 2. 클러스터별 임계값 지정 ($k$=3)
Table 2. Cluster-Specific Thresholds ($k$=3)

| | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|
| RF | 0.80 | 0.78 | 0.78 |
| KNN | 0.80 | 0.76 | 0.78 |
| Ada | 0.50 | 0.60 | 0.60 |

## 4. 실험 설계

### 4.1 데이터셋

본 실험은 AndroZoo의 데이터셋을 사용한다[17]. Google Play를 비롯한 다양한 도메인에서 앱을 수집하여 APK 파일의 분석을 지원한다. AndroZoo는 현재 약 2,400만 개의 APK 데이터를 보유하고 있다[18].

#### 4.1.1 데이터 수집

본 실험에서 사용할 데이터는 훈련, 검증, 테스트 데이터로 나뉜다. 훈련 데이터와 검증 데이터는 2014년부터 2016년까지의 앱을 연도별로 정상 앱과 악성 앱을 8,000개씩 추출하여, 6,000개와 2,000개로 각각 분할한다. 테스트 데이터는 2019년부터 2021년까지의 앱을 연도별로 정상 앱과 악성 앱 6,000개씩을 테스트 데이터로 선정하여 실험을 진행한다. 정상 앱과 악성 앱의 비율은 1대1이며, 총 84,000개의 데이터를 수집했다.

표 3은 시간적 편향을 고려한 훈련, 검증, 테스트 데이터의 분할을 나타낸다.

표 3. 실험에 사용될 연도별 정상 앱 및 악성 앱 데이터 분포
Table 3. Distribution of Benign and Malicious App Data by Year for Experimentation

| | 연도 | 정상 앱 | 악성 앱 |
|---|---|---|---|
| Train | 2014 | 6,000 | 6,000 |
| | 2015 | 6,000 | 6,000 |
| | 2016 | 6,000 | 6,000 |
| Validate | 2014 | 2,000 | 2,000 |
| | 2015 | 2,000 | 2,000 |
| | 2016 | 2,000 | 2,000 |
| Test | 2019 | 6,000 | 6,000 |
| | 2020 | 6,000 | 6,000 |
| | 2021 | 6,000 | 6,000 |
| Total | | 42,000 | 42,000 |

#### 4.1.2 APK의 클래스, 연도 분류

정상, 악성 클래스 분류는 AndroZoo에서 제공하는 메타 데이터 vt_detection을 기준으로 했다. vt_detection은 VirusTotal의 검사 결과를 기록한 것으로 0이면 정상 앱으로 분류하고 3 이상일 경우 악성 앱으로 분류한다. 각 APK의 연도 분류는 dex_date를 기준으로 했다. dex_date는 APK 파일의 생성 및 수정 일자를 나타내는 메타데이터이다.

정상 앱 : $vt\_detection = 0$
악성 앱 : $vt\_detection \geq 3$

### 4.2 데이터 전처리

APK는 안드로이드 앱의 배포 형식이며 소스 코드, 리소스, 메타데이터등의 정보를 포함하고 있다. 이때, APK 내의 API 호출은 안드로이드 SDK의 일부로 제공되는 dexdump[19]를 이용하여 추출할 수 있다. 이후 악성 앱 탐지에 효율적이라 알려진 1,848개의 API 특징정보에 매핑한다[9]. 추출한 특징정보를 column 명으로, 각 앱을 row명으로 하는 벡터 테이블을 구성한다. 벡터 테이블의 각 요소는 특정 API의 호출 빈도수를 의미한다.

## 5. 실험 결과 및 논의

### 5.1 클러스터의 데이터 불균형

표 4는 2014~2016년의 훈련 데이터를 K-means로 학습시킨 결과이다 ($k$=3). 클러스터링 결과 하나의 거대 클러스터 $C_1$이 생성되며, 클러스터들의 개체수 불균형과 클래스 불균형을 확인할 수 있다. 이러한 클래스 불균형은 앱의 API 호출 패턴이 대부분 2014~2016년의 앱들은 $C_1$과 같은 패턴을 보이며, 그 외에 $C_2$, $C_3$ 와 같은 호출 패턴이 특이적으로 구분되어 있음을 의미한다.

표 5는 2014~2016년의 데이터를 K-means 학습 후 2019~2021년의 데이터를 K-means 예측한 결과를 보인다. 표 4와 비교했을 때, 클래스 불균형이 더 심화된 것을 확인할 수 있다. 가장 크기가 큰 $C_1$은 2019~2021년 악성 앱 비율의 평균을 계산했을 때 2014~2016년의 악성 앱 비율보다 약 15.4%p 증가한 것으로 나타났다. 두 번째로 큰 $C_2$의 정상 앱 비율의 평균값은 2014~2016년의 정상 앱 비율보다 약 22.5%p 높은 수치를 보였으며, $C_3$의 악성 앱 비율의 평균값은 2014~2016년의 악성 앱 비율과 비교할 때 약 61.1%p 증가하였다. 이는 2014~2016년의 데이터를 통해 만들어진 클러스터가 2019~2021년에 사용되는 앱의 API 정상 호출 패턴의 경향을 높은 비율로 반영하고 있음을 의미해 시간적 편향을 잘 반영하고 있음을 의미한다.

표 4. 2014~2016년 K-means 학습 결과 ($k$=3)
Table 4. K-means Training Results from 2014 to 2016 ($k$=3)

| | 개체수 | 비율 | 악성 앱 개체수 | 클러스터 내 악성 앱 비율 |
|---|---|---|---|---|
| $C_1$ | 22,563 | 0.6268 | 12,246 | 0.5427 |
| $C_2$ | 9,956 | 0.2765 | 3,483 | 0.3498 |
| $C_3$ | 3,481 | 0.0966 | 2,271 | 0.6524 |

표 5. 2019~2021년 K-means 예측 결과 ($k$=3)
Table 5. K-means Prediction Results for 2019 to 2021 ($k$=3)

| | $C_i$ | 유형 | 개체수 | 클러스터 내 비율 |
|---|---|---|---|---|
| 2019 | $C_1$ | 정상 | 2,379 | 0.3043 |
| | | 악성 | 5,438 | 0.6957 |
| | $C_2$ | 정상 | 1,150 | 0.6882 |
| | | 악성 | 521 | 0.3118 |
| | $C_3$ | 정상 | 2,296 | 0.9140 |
| | | 악성 | 216 | 0.0860 |
| 2020 | $C_1$ | 정상 | 1,802 | 0.2518 |
| | | 악성 | 5,354 | 0.7482 |
| | $C_2$ | 정상 | 1,603 | 0.6336 |
| | | 악성 | 927 | 0.3664 |
| | $C_3$ | 정상 | 2243 | 0.9693 |
| | | 악성 | 71 | 0.0307 |
| 2021 | $C_1$ | 정상 | 1,608 | 0.3540 |
| | | 악성 | 2,934 | 0.6460 |
| | $C_2$ | 정상 | 1,941 | 0.4024 |
| | | 악성 | 2,883 | 0.5976 |
| | $C_3$ | 정상 | 2,615 | 0.9928 |
| | | 악성 | 19 | 0.0072 |

### 5.2 성능 평가지표

본 모델의 정확도, F1-Score등의 성능 평가지표는 혼동행렬(Confusion Matrix)을 기반으로 산출하며, $C_i$의 혼동행렬 요소를 아래와 같이 정의한다.

- $TP_i$ : 악성 앱을 악성 앱으로 탐지한 수
- $TN_i$ : 정상 앱을 정상 앱으로 탐지한 수
- $FP_i$ : 정상 앱을 악성 앱으로 탐지한 수
- $FN_i$ : 악성 앱을 정상 앱으로 탐지한 수

표 4, 표 5를 참고하였을 때, K-means 클러스터링 시 개체수 불균형과 클래스 불균형을 확인할 수 있다. 따라서 본 실험에서는 모든 예측 결과를 하나의 혼동행렬로 합산한 후 정밀도(Precision), 재현율(Recall), F1-score를 계산하는 마이크로 평균(Micro-Average) 방식을 사용한다[20]. 마이크로 평균 방식은 각 클러스터의 개체수 불균형과 정상/악성 클래스의 불균형을 고려하여 계산할 수 있다. 아래는 마이크로 평균 방식으로 정밀도, 재현율, F1-Score를 산출하는 수식이다.

$$\mathrm{Pr}^{\mu} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} (TP_i + FP_i)}$$

$$Re^{\mu} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} (TP_i + FN_i)}$$

$$F_1^{\mu} = \frac{2 \cdot Pr^{\mu} \cdot Re^{\mu}}{Pr^{\mu} + Re^{\mu}}$$

F1-Score는 재현율과 정밀도의 조화 평균으로, 두 지표를 하나의 측정값으로 나타내어 모델의 성능을 효과적으로 평가할 수 있다. 이를 바탕으로 실험에서는 각 클러스터에서 혼동행렬의 구성 요소들을 바탕으로 하나의 혼합행렬로 합산 후 $F_1^{\mu}$과 Pendlebury등[4]이 제안한 $AUT$(Area Under Time)를 계산하여 평가한다.

$AUT$는 모델의 지속가능성을 나타내는 지표로 [0, 1]의 범위의 값을 가지며, 1에 가까울수록 모델의 지속가능성이 높음을 의미한다.

$$AUT(f, N) = \frac{1}{N-1} \sum_{k=1}^{N-1} \frac{[f(x_{k+1}) + f(x_k)]}{2}$$

여기서 $f(x_k)$는 $k$ 지점에서 평가된 성능지표이며, F1-Score, 재현율, 정밀도 등이 사용될 수 있다. N은 테스트 슬롯의 수를 의미한다. 본 실험에서는 $f$의 성능 평가 지표로 $F_1^{\mu}$를 사용하였고, 2019년부터 2021년도까지 3개 연도에 대해 실험을 진행했으므로 $AUT(F_1^{\mu}, 3Y)$에 해당한다.

### 5.3 최적화된 분류모델의 검증 및 성능 비교

표 6은 최적화된 K-means 기반 멀웨어 분류 모델의 지속가능성을 평가 및 비교하기 위해, 클러스터링 이후 최적화된 분류기들의 $F_1^{\mu}$으로 $AUT$를 산출하여 비교한다.

표 7은 2014~2016년의 학습 데이터를 이용하여, 임계값이 지정된 분류기에 그리드 탐색 과정과 5겹 교차검증을 수행하고, 2014~2016년의 검증 데이터를 이용해 $F_1^{\mu}$로 성능을 검증한 결과를 보인다.

표 6의 테스트 결과와 표 7의 검증 결과를 비교 시 RF와 KNN의 경우 일반화 성능이 준수하

표 6. 분류기 간 성능 비교 ($k$=3)
Table 6. Performance Comparison Among Classification Models

| 연도 | Micro-Averaging F1-Score($F_1^{\mu}$) | | |
|---|---|---|---|
| | *RF* | *KNN* | *AdaBoost* |
| 2019 | 0.9058 | 0.8687 | 0.5291 |
| 2020 | 0.9097 | 0.8663 | 0.5072 |
| 2021 | 0.8913 | 0.8487 | 0.5072 |
| Avg. | 0.9023 | 0.8612 | 0.5145 |
| *AUT* | 0.9041 | 0.8625 | 0.5127 |

지만. AdaBoost의 경우 $F_1^{\mu}$이 약 47.5%의 감소했다. 이는 AdaBoost가 미래 시점의 테스트 데이터를 예측 시 최적의 성능을 발휘하지 못할 수 있음을 의미한다.

표 7. 검증 데이터에 대한 분류기 성능지표
Table 7. Performance Metrics for Classification Models on Validation Data

| | | $C_1$ | $C_2$ | $C_3$ | Overall. |
|---|---|---|---|---|---|
| RF | $F_1^{\mu}$ | 0.8976 | 0.9632 | 0.8966 | 0.9167 |
| KNN | $F_1^{\mu}$ | 0.8844 | 0.9149 | 0.8389 | 0.8900 |
| Ada | $F_1^{\mu}$ | 0.9850 | 1.0000 | 0.9978 | 0.9896 |

2019~2021년의 실험 결과 임계값과 초매개변수가 지정된 3개의 모델 중 RF, KNN, AdaBoost 순으로 높은 지속가능성을 보였다. 또한 세 분류 모델은 가장 지속가능성 수치가 높았던 RF 모델을 비롯하여, 2014~2016년의 데이터를 검증하였을 때 모두 균일한 일반화 성능을 보였다.

표 8은 3개의 분류기 중 가장 성능이 우수했던 RF 모델의 분류 결과를 합산하여 연도별로 나타낸다. 모든 테스트 슬롯에서, RF는 6,000개의 악성 앱 중 5,000개가 넘는 악성 앱을 정확하게 탐지했다. 하지만, 2021년의 재현율($Re^{\mu}$)을 비교하면 2019~2020년의 재현율의 평균값을 계산했을 때보다 약 5.2%p 감소하였다. 이는 2019년과 2020년에 비해 2021년에 모델이 더 많은 악성 앱을 정상 앱으로 잘못 예측했음을 의미한다.

표 8. 혼동행렬 합산 결과 ($k$=3)
Table 8. Total Confusion Matrix Results ($k$=3)

| | 예측 / 실제 | 정상 앱 | 악성 앱 |
|---|---|---|---|
| 2019 | 정상 앱 | 5,347 (TN) | 653 (FP) |
| | 악성 앱 | 478 (FN) | 5,522 (TP) |
| 2020 | 정상 앱 | 5,282 (TN) | 718 (FP) |
| | 악성 앱 | 366 (FN) | 5,634 (TP) |
| 2021 | 정상 앱 | 5,430 (TN) | 570 (FP) |
| | 악성 앱 | 734 (FN) | 5,266 (TP) |

표 9는 2019년 테스트 데이터를 RF 모델로 예측하였을 때 클러스터별 혼동행렬을 나타낸다. 가장 크기가 컸던 $C_1$은 94.49%의 정확도로 3개의 클러스터 중 가장 높은 수치를 보였다. 하지만, $C_2$와 $C_3$의 정확도는 각각 73.61%, 89.69%였는데, 특히 2014~2016년에 할당된 개체수가 적어진 $C_2$의 정확도는 $C_1$ 대비 20.88%p 낮았다.

또한 $C_2$와 $C_3$의 정밀도($\mathrm{Pr}_2$, $\mathrm{Pr}_3$)는 각각 31.28%, 24.07%의 수치를 보였는데, 이는

2019~2021년의 데이터를 K-means 예측했을 때, $C_2$와 $C_3$의 정상 앱 비율이 높음에도, RF 모델은 $C_2$와 $C_3$에서 높은 비율의 정상 앱을 악성 앱이라 탐지하였음을 의미한다.

표 9. 클러스터별 분류 결과 ($k$=3, dex_date=2019)
Table 9. Classification Results by Cluster ($k$=3, dex_date=2019)

| | 예측 / 실제 | 정상 앱 | 악성 앱 |
|---|---|---|---|
| $C_1$ | 정상 앱 | 2,079 (TN) | 131 (FP) |
| | 악성 앱 | 300 (FN) | 5,307 (TP) |
| $C_2$ | 정상 앱 | 1,067 (TN) | 358 (FP) |
| | 악성 앱 | 83(FN) | 163 (TP) |
| $C_3$ | 정상 앱 | 2,201 (TN) | 164 (FP) |
| | 악성 앱 | 95 (FN) | 52 (TP) |

### 5.4 기존 방법과의 지속가능성 비교

표 10은 K-means 기법을 사용하지 않고, 전통적인 방식으로 RF로 분류를 진행했을 때(Traditional RF), K-means 기반 분류모델의 최적화 과정을 거치지 않았을 때(Baseline Model), 거쳤을 때, (Optimized Model)에 따라 분류를 진행한 결과를 나타낸다($k$=3). 훈련 및 테스트 데이터는 동일한 데이터셋을 사용하였으며 전통적인 방식과, 최적화 과정을 거치지 않은 RF의 초매개변수와 임계값은 기본값(n_estimator : 100, max_depth : None, Threshold : 0.5)을 사용하였다. 실험결과, 엘보우 기법, 임계값 지정과 초매개변수의 최적화 과정을 거친 K-means 기반 분류모델은 시간적 편향을 고려한 데이터셋으로 전통적인 RF로 분류를 진행했을 때보다 20.4%p 개선된 지속가능성을 보였다. 또한 위의 과정을 거치지 않은 모델보다 18.5%p 높은 지속가능성을 보였다.

표 10. 기존 RF, 베이스라인 모델 및 최적화된 모델의 연도별 성능 비교
Fig 10. Yearly Performance Comparison of Traditional RF, Baseline Model, and Optimized Model

| | Traditional RF | Baseline Model[7] | Optimized Model |
|---|---|---|---|
| 2019 | 0.7083 | 0.7228 | 0.9058 |
| 2020 | 0.6963 | 0.7164 | 0.9097 |
| 2021 | 0.6980 | 0.7214 | 0.8913 |
| Avg. | 0.7009 | 0.7202 | 0.9023 |
| $AUT$ | 0.6997 | 0.7192 | 0.9041 |

## 6. 결론 및 향후 연구

본 논문에서는 전통적인 기계학습 기반의 안드로이드 악성 앱 탐지에서 논의되는 지속가능성 문제를 완화하기 위해 K-means 기반 분류모델을 훈련단계에서 클러스터별로 최적화하여, 미래의 데이터를 평가한 성능을 모델별로 검증 및 비교하였다.

실험과정에서 클러스터링 후 개체수 불균형과 클래스 불균형을 고려하여 마이크로-평균 방식으로 F1-Score와 $AUT$를 산출했을 때, 최종적인 악성 앱 분류과정에서 RF, KNN, AdaBoost 모델 중 RF 모델이 가장 높은 지속가능성을 보였다. 이 수치는 전통적인 RF 방식으로 악성 앱을 분류한 결과와 엘보우 기법, 임계값 지정과 초매개변수의 최적화 과정을 거치지 않은 K-means 기반 분류모델의 결과($k$=3)보다 각각 20.4%p, 18.5%p 높은 수치를 보였다.

하지만 본 모델은 K-means 기법으로 2014~2016년의 데이터 학습 시와 2019~2021년의

데이터 예측 시 데이터 포인트들의 클러스터 재배치를 API 호출 정보를 바탕으로 충분히 설명할 수 없었다는 한계점을 찾을 수 있었다. 또한 2019년과 2020년에 비해 2021년을 살펴볼 때, 마이크로 평균 방식으로 산출한 재현율이 5.2%p 감소하였으며, 이는 2021년보다 미래의 데이터가 테스트 데이터로 사용되었을 시 지속가능성 저하의 가능성이 있다는 점 또한 찾을 수 있었다.

따라서 향후 연구에는 이상 탐지(Anomaly Detection) 알고리즘을 접목하여 미래 데이터에 대해 클러스터 분포 추이와 클러스터별 데이터 포인트들의 API 호출 정보 변화에 대한 설명력을 강화하고, 모델을 재훈련해야 하는 최적의 시기를 예측할 수 있는 시계열 모델을 구축하여 안드로이드 멀웨어 탐지에서 논의되는 지속가능성 문제를 완화하고자 한다.

본 연구는 산업통상자원부(MOTIE)와 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구과제임(No. RS-2021-KP002461).
또한 2021년도 정부(과학기술정보통신부의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(no. 2021R1A2C2012574).
또한 과학기술정보통신부 및 정보통신기획평가원의 학석사연계ICT핵심인재양성사업의 연구결과로 수행되었음(IITP-2023-00259867).

## 참 고 문 헌

[1] K. Tam, A. Kimberly, N. B. Anuar, R. Salleh, and L. Cavallaro, “The evolution of android malware and android analysis techniques”, ACM Computing Surveys, vol. 49, no. 4, pp. 1-41, 2017. DOI: https://doi.org/10.1145/3017427

[2] Q. Wu, X. Zhu, and B. Liu, “A survey of android malware static detection technology based on machine learning”, Mobile Information Systems, vol. 2021, pp. 1-18, 2021. DOI: https://doi.org/10.1155/2021/8896013

[3] Kaspersky, “Kaspersky discovers new Mandrake spyware campaign undetected for two years, with over 32,000 installs on Google Play”, Kaspersky, 2024. https://www.kaspersky.com/about/press-releases/kaspersky-discovers-new-mandrake-spyware-campaign-undetected-for-two-years-with-over-32000-installs-on-google-play/

[4] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, “TESSERACT: Eliminating experimental bias in malware classification across space and time”, 28th USENIX Security Symposium (USENIX Security 19), pp. 1289-1306, 2019. DOI: https://doi.org/10.48550/arXiv.1807.07838

[5] H. Lee, S.-j. Cho, H. Han, W. Cho, and K. Suh, “Enhancing sustainability in machine learning-based android malware detection using API calls”, 2022 IEEE Fifth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Laguna Hills, CA, USA, pp. 131-134, 2022. DOI: https://doi.org/10.1109/AIKE55402.2022.00028

[6] W. S. Cho, S. J. Cho, Y. S. Hwang, and S.C. Han, “A Study on the Sustainability Improvement of Android Malicious App Detection through Year Data Set Combination Machine Learning”, The Journal of Korean Institute of Next Generation Computing, vol. 18, no. 5, pp. 47-57, 2022. DOI: https://doi.org/10.23019/kingpc.18.5.202210.005

[7] S. W. Jeong , S. H. Ann, S.-j. Cho, D. J. Kim, Y. S. Hwang, “A K-means Based Classification Model for the Sustainability of Android Malware Detection”, Workshop

on Dependable and Secure Computing, pp 172-177, 2024. URL : https://sites.google.com/view/wdsc2024/workshop-proceedings

[8] N. Peiravian and X. Zhu, “Machine learning for android malware detection using permission and api calls”, 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, pp. 300-305, 2013. DOI: https://doi.org/10.1109/ICTAI.2013.53

[9] J.M Jung, J. H. Park, S. J. Cho, S. C. Han, M. K. Park, and H. H. Cho, “Feature engineering and evaluation for android malware detection scheme”, Journal of Internet Technology, vol. 22, pp. 423-440, 2021. DOI: https://doi.org/10.3966/160792642021032202017

[10] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, “Drebin: Effective and explainable detection of android malware in your pocket”, The Network and Distributed System Security Symposium, vol. 14, pp. 23-26, 2014. DOI: https://doi.org/10.14722/ndss.2014.23247

[11] Y. Aafer, W. Du, and H. Yin, “Droidapiminer: Mining api-level features for robust malware detection in android”, International Conference on Security and Privacy in Communication Systems, Springer, Cham, pp. 86-103, 2013. DOI: https://doi.org/10.1007/978-3-319-04283-1_6

[12] S. M. Lee, S. H. Ann, S.-j. Cho, D. J. Kim, Y. S. Hwang, “A Hierarchical Clustering Hybrid Model for the Sustainability of Android Malicious Application Detection”, Workshop on Dependable and Secure Computing, pp 178-183, 2024. URL: https://sites.google.com/view/wdsc2024/workshop-proceedings

[13] Thorndike, R.L, “Who belongs in the family?”, Psychometrika, vol. 18, no. 4, pp. 267-276, 1953. DOI: https://doi.org/10.1007/BF02289263

[14] D. Arthur and S. Vassilvitskii, “K-means++: The advantages of careful seeding”, Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1027-1035, 2007. URL: http://people.eecs.berkeley.edu/~brecht/cs294docs/week2/07.arthur.kmeans.pdf

[15] T. Hastie, R. Tibshirani, and J. Friedman, “The Elements of Statistical Learning”, Springer, 2009, 532p. DOI: https://doi.org/10.1007/978-0-387-21606-5

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, and É. Duchesnay, “Scikit-learn: Machine learning in Python”, Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011. URL:https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

[17] K. Allix, T. F. Bissyande, J. Klein, and Y. L. Traon, “AndroZoo: Collecting millions of Android apps for the research community”, IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), Austin, TX, USA, pp. 468-471, 2016. DOI: https://doi.org/10.1145/2901739.2903508

[18] AndroZoo, “AndroZoo Documentation”, 2023. AndroZoo, https://androzoo.uni.lu/

[19] S. Aboy Solanes, “dexdump.cc”, Google Git, 2024. https://android.googlesource.com/platform/art/+/master/dexdump/dexdump.cc/

[20] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks”, Information Processing and Management, vol. 45, pp. 427-437, 2009. DOI: https://doi.org/10.1016/j.ipm.2009.03.002

## 저 자 소 개


정성원(Seongwon Jeong)

2020.3-현재 단국대학교 소프트웨어학과 학사과정
<주관심분야> AI를 활용한 악성코드 탐지, 인공지능 보안 등

안석현(Seokhyun Ann)

2024.8 단국대학교 소프트웨어학과 학사
2024.9-현재 단국대학교 인공지능융합학과 석사과정
<주관심분야> 운영기술 보안, AI를 활용한 악성코드 탐지 등

조성제(Seong-je Cho)

1989.2 서울대학교 컴퓨터공학과 공학사
1991.2 서울대학교 컴퓨터공학과 공학석사
1996.8 서울대학교 컴퓨터공학과 공학박사
1997.3-현재 단국대학교 소프트웨어학과/컴퓨터학과 교수
<주관심분야> 디지털포렌식, 시스템 보안 및 악성코드 분석, 인공지능 보안, 시스템 소프트웨어 등

김동재(Dongjae Kim)

2016.2 고려대학교 생명과학부 학사
2021.2 KAIST 바이오및뇌공학과 뇌인지공학프로그램 공학박사
2021.5-2022.8 New York University 박사후연구원
2022.9-현재 단국대학교 인공지능융합학과 조교수
<주관심분야> 강화학습, 기계학습, 뇌공학

황영섭(Young-Sup Hwang)

1989.2 서울대학교 컴퓨터공학과 학사
1991.2 포항공과대학교 컴퓨터공학과 석사
1997.2 포항공과대학교 컴퓨터공학과 박사
1997-2002 한국전자통신연구원 선임연구원
2002-현재 선문대학교 컴퓨터공학부 교수
<주관심분야> 딥러닝, 기계학습, 패턴인식, 영상처리