JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 26 No. 2, pp. 61-68, February 2021
https://doi.org/10.9708/jksci.2021.26.02.061

RFA: Recursive Feature Addition Algorithm for Machine Learning-Based Malware Classification

Ji-Yun Byeon*, Dae-Ho Kim*, Hee-Chul Kim*, Sang-Yong Choi*

*Student, Dept. of Cyber Security, Yeungnam University College, Daegu, Korea
*Student, Dept. of Cyber Security, Yeungnam University College, Daegu, Korea
*Student, Dept. of Cyber Security, Yeungnam University College, Daegu, Korea
*Assistant Professor, Dept. of Cyber Security, Yeungnam University College, Daegu, Korea

[Abstract]

Recently, various technologies that use machine learning to classify malicious code have been studied. In order to enhance the effectiveness of machine learning, it is most important to extract properties to identify malicious codes and normal binaries. In this paper, we propose a feature extraction method for use in machine learning using recursive methods. The proposed method selects the final feature using recursive methods for individual features to maximize the performance of machine learning. In detail, we use the method of extracting the best performing features among individual feature at each stage, and then combining the extracted features. We extract features with the proposed method and apply them to machine learning algorithms such as Decision Tree, SVM, Random Forest, and KNN, to validate that machine learning performance improves as the steps continue.

▸ **Key words**: machine learning, feature, recursive, malware, classification

[요 약]

최근 악성코드와 정상 바이너리를 분류하기 위해 기계학습을 이용하는 기술이 다양하게 연구되고 있다. 효과적인 기계학습을 위해서는 악성코드와 정상 바이너리를 식별하기 위한 Feature를 잘 추출하는 것이 무엇보다 중요하다. 본 논문에서는 재귀적인 방법을 이용하여 기계학습에 활용하기 위한 Feature 추출 방법인 RFA(Recursive Feature Addition) 제안한다. 제안하는 방법은 기계학습의 성능을 극대화 하기 위해 개별 Feature를 대상으로 재귀적인 방법을 사용하여 최종 Feature Set을 선정한다. 세부적으로는 매 단계마다 개별 Feature 중 최고성능을 내는 Feature를 추출하여, 추출한 Feature를 결합하는 방법을 사용한다. 제안하는 방법을 활용하여 Decision tree, SVM, Random forest, KNN등의 기계학습 알고리즘에 적용한 결과 단계가 지속될수록 기계학습의 성능이 향상되는 것을 검증하였다.

▸ **주제어**: 기계학습, 특성인자, 재귀적, 악성코드, 분류

• First Author: Ji-Yun Byeon, Corresponding Author: Sang-Yong Choi
*Ji-Yun Byeon (jiyun.byeon999@gmail.com), Dept. of Cyber Security, Yeungnam University College
*Dae-Ho Kim (jhk1a1a@naver.com), Dept. of Cyber Security, Yeungnam University College
*Hee-Chul Kim (yyyyyyyyyy0816@gmail.com), Dept. of Cyber Security, Yeungnam University College
*Sang-Yong Choi (csyong95@gmail.com), Dept. of Cyber Security, Yeungnam University College
• Received: 2020. 12. 31, Revised: 2021. 01. 28, Accepted: 2021. 01. 28.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

최근 악성코드는 인터넷 상의 가장 심각한 위협으로 떠오르고 있다. ENISA(European Union Agency for Cybersecurity)의 발표에 따르면, 2020년 가장 심각한 위협은 바이러스, 랜섬웨어, 웜, 스파이웨어와 같은 악성코드이다[1]. 이러한 악성코드를 탐지하기 위한 기술로 최근 기계학습이 대두되고 있다. 기계학습은 학습방법에 따라 지도학습(Supervised Learning), 비지도학습(Unsupervised Learning), 강화 학습(Reinforcement Learning)등이 있으며, 악성코드 탐지를 위해서는 지도학습 기법 중 심층신경망(DNN), 합성곱 신경망(CNN), 군집화(SVM)의 방법이 비지도학습에서는 군집화(K-평균)등의 방법이 연구되고 있다[2-3].

기계학습을 이용하여 악성코드를 잘 분류하기 위해서는 악성코드를 대표할 수 있는 특징을 잘 추출하는 것이 무엇보다 중요하다. 일반적으로 기계학습의 성능은 어떤 데이터를 입력하는지에 따라 많은 차이가 나며, 기계학습에 사용되는 입력데이터를 Feature라 한다. 즉, 기계학습의 성능을 향상시키기 위해서는 유용한 Feature 선택하는 과정(Feature Selection or Feature Extraction)이 매우 중요하다[4].

본 논문에서는 기계학습에 활용할 수 있는 Feature 추출 방법을 제안한다. 다양한 Feature추출 방법이 있지만 본 논문에서 제안하는 방법은 제귀적인 Feature 추출 방법으로 데이터로부터 추출한 개별 Feature를 반복적인 실험을 통해 가장 고성능을 내는 Feature Set을 만들어내는 방법이다. 제안하는 방법을 활용하여 기계학습 알고리즘에 적용하여 실험을 통해 효과를 검증하였다.

본 논문의 2장에서는 대표적인 기계학습 알고리즘과 기존에 연구되고 있는 Feature 추출 방법에 대해 서술하며, 3장에서는 본 논문에서 제안하는 RFA(Recursive Feature Addition)의 세부적인 내용을 기술한다. 4장에서는 4개의 기계학습 알고리즘을 이용하여 실험을 통해 제안하는 RFA의 효과를 검증한다.

II. Preliminaries

1. Machine Learning Algorithm for Malicious Code Classification

기계학습에서 지도학습으로 사용되는 다양한 알고리즘이 있다. 일반적으로 분류(Classification)를 하기 위한 지도학습과 최적의 결과를 예측하기 위한 학습 알고리즘이 있다. 예측을 위한 알고리즘은 예측 변수(Predictor Variable)라고 하는 특성(Feature)을 사용해 최적인 결과를 예측하기 위한 시도이다. 이러한 학습의 방법을 일반적으로 회귀(Regression)라고 한다. 본 논문에서 실험을 위해 사용하는 지도 학습 알고리즘들은 Decision Tree, RandomForest, SVM, KNN을 사용한다.

1.1 Decision Tree

결정 트리(Decision Tree)는 의사결정을 위한 규칙과 규칙에 따른 결과를 트리 구조로 구조화한 의사 결정 지원 도구로 시작되었다. 결정트리는 과학분야에서 주로 활용되며, 의사 결정을 위해 목표에 가장 가까운 결과를 낼 수 있는 방안을 찾기 위해 사용된다[5].

1.2 Random Forest

랜덤 포레스트(Random Forest)는 분류와 회귀 분석과 같은 목적을 위해 사용하는 앙상블 학습 방법의 한 종류로, 훈련 과정에서 다수의 결정트리를 구성하고, 이로부터 분류 또는 평균 예측치를 출력한다[6].

1.3 SVM

서포트 벡터 머신(Support Vector Machine)은 패턴인식, 자료분석을 위한 기계 학습의 분야이며, 주로 분류 또는 회귀 분석에 사용한다. SVM은 두 개의 그룹 중 어느 하나에 속하는 데이터 집합에서, 주어진 데이터 집합의 특성을 분석하여, 이를 바탕으로 하여 새로이 입력되는 데이터가 어떤 분야에 속하는지를 판단할 수 있는 비확률적 이진 선형 분류 모델을 만든다. 만들어진 모델은 데이터가 매핑된 공간상에서 경계로 표시되며, SVM은 가장 큰 폭의 경계를 찾아가는 알고리즘이다. SVM은 선형 분류, 비선형 분류에서 복합적으로 사용할 수 있다[7].

1.4 KNN

K-최근접 이웃 알고리즘(K-Nearest Neighbors, KNN)은 분류나 회귀에 사용되는 비모수 방식이다. KNN에서는 입력이 특징 공간에서 주어진 K개와 가장 가까운 훈련 데이터로 구성된다. 출력은 KNN을 분류에 사용할 것인가, 회귀에 사용할 것인가에 따라 다르다. 분류에서 사용할 경우출력은 소속된 항목이 될 수 있다. K개의 최근접 이웃사이에서 K는 제일 공통적인 항목에 할당되는 객체이며, 과반이상의 의결에 의해 분류될 수 있다. KNN을 회귀에 사용할 경우 출력결과는 객체의 특성값으로, 이 값은 K개의 최근접 이웃이 가진 모든 값의 평균값이다. KNN은 가장 간단한 기계 학습 알고리즘 중 하나에 속한다[8].

2. Feature Extraction Method

각 기계학습을 이용한 연구들 중 사용되고 있는 Feature 추출 방법은 대표적으로 현재 후보인자와의 상관성을 기반으로 하는 CFS기법, 배열을 이용해 특성인자를 추출하는 Autoencoder 기법, 시그니쳐를 생성하여 두 블록을 비교하는 V-gram기법, 피어슨상관계수를 이용해 특징을 추출하는 기법, API 호출정보를 Feature로 추출하는 기법이 있다. 또한 모바일 분야에서는 툴을 사용하여 권한을 추출하는 방법 등 여러 연구에서 좋은 성능을 내기 위해 여러 방법들을 사용하는 것을 알 수 있다.

2.1 Feature Extraction using the CFS

CFS 기법은 여러 가지 후보 Feature중 상관성을 기반으로 예측모델에 사용될 대표적인 Feature를 추출하기 위한 기계학습 기법의 하나다. CFS 기법은 Best-first 방법을 사용한다. 즉, 추정이 필요한 결과와 상관성이 크고, 동시에 다른 속성과의 상관성이 낮은 속성을 탐색하는 접근을 사용한다. CFS 기법의 이점은 일반적으로 속도가 빠르다. 이는 불필요한 변수 집합을 축소하고 연관선 있는 속성만을 효과적으로 추출하기 때문이다.[9].

CFS에서 파생한 기법의 하나로 mRMR은 각 속성들 사이의 상호 정보를 기반으로 하여 계산하기 때문에 속성 사이의 의존성이 결과에 주로 영향을 준다. 또한, 결과 출력을 위한 가중치 조절이 힘들어 속성의 개수가 많으면 연산 성능이 저하되는 단점이 있다.[10]

2.2 Feature Extraction based on Stacked Autoencoder

Autoencoder란 비지도 신경망(unsupervisedneural network)으로 입력층(input layer)과 출력층(output layer)의 뉴런의 숫자를 같게 하고, 입력 및 출력 값이 동일하게 되도록 학습하는 신경망이다. 이를 위해 엔트로피값을 세분화(0.5단위)하여 Y축의 범위를 0~15로 확장함으로 더 세밀한 분석이 가능하도록 하였다. 결과적으로 섹션명인 X축의 범위(0~15), 엔트로피인 Y축의 범위(0~15)를 선정하여 (0,0)에서 (15,15)까지 총 256개의 격자를 생성하며, 격자별 발생횟수를 산출한다. 이 후, 16x16구조의 2차원 배열을 1차원 배열로 변환, 256개의 Feature 값을 선택하여 활용한다. autoencoder를 병렬로 연결하여 추출한 Feature를 학습시키는 방법과 전체 Feature를 대상으로 ExtraTreeesClasifier를 사용하여 Importance를 추출하여 의미 있는 Feature를 선택하는 방법을 사용한다[11].

2.3 V-gram

V-gram은 크게 2가지 단계로 진행된다. 첫 번째로 하나의 기본 블록 내의 opcode를 순서대로 연결하여 하나의 연결을 만든다. 만들어진 연결은 블록의 기본 길이에 따라 가변적일 수 있다. 두 번째로, 이 연결은 SHA-256 해시 알고리즘을 사용하여 시그니쳐를 생산한다. 해시 알고리즘의 특성상 이 시그니쳐는 해당 블록을 대표하는 고유한 값이 된다. 그 이후 모든 기본블록에 대해 이 단계를 적용한다[12].

2.4 Static Analysis and Stacking Techniques

PE(Portable Execution) 파일은 윈도우 OS에서 사용하는 실행파일을 말하며, DLL, 폰트, 객체 등을 의미한다. PE에서 필요한 정보는 대부분 PE header 부분에 있다. PE header 특징 추출을 위해 ClaMP(Classification of Malware with PE header) 스크립트를 사용하여 PE header내 DOS header에서 6개, FILE header에서 17개, OPTIONAL header에서 37개 총 60개의 기본특징과 PE header의 값을 한 번 더 가공하여 재생산한 7개의 Derived 특징을 추출할 수 있다. 추출한 PE header 중 범주형 데이터인 packer_type column을 제거하여 pe_header 특징을 생성하고, 동일한 column을 one-hot encoding 하여 가공한 pe_packer를 생성한다. 마지막으로 피어슨 상관 계수를 활용하여 pe_top을 생성한다[13].

2.5 API Call Time Interval

API 호출 정보에서 시간간격을 계산하여 특징 값으로 만든다. 전체 호출된 API를 단순히 시간간격을 구하는 것 이외에 단계를 초기, 중기. 후기의 3단계로 구분하여 특징을 추출하였다. 이는 API를 호출한 횟수가 많아질수록 변화량이 작은 시간간격이 증가 할 수 있고, 결국 평균 시간차를 계산할 경우 변화량이 큰 호출정보의 의미를 축소시킬 수 있으므로 이를 방지하기 위해 단계별 구분을 시도하였다. 전체 호출 횟수가 악성코드에 따라 무작위의 횟수가 발생하므로 발생 횟수를 그룹으로 묶어 단계를 적용하였다. 많은 악성코드가 실행초기 탐지 회피를 위해 동작하는 초기 부분과 실제 주요한 파일의 내용이 동작하는 중각부분, 그리고 파일의 행위를 마무리하는 마지막 부분으로 구분하여 초기, 중기, 후기로 나누는 방법을 적용하였다. API 전체 call을 3단계로 나누는 방법은 3으로 나눈 나머지를 고려하여 경우의 수를 모두 실험에 적용해보고 성능결과가 가장 우수한 방법을 채택하는 방식으로 구분하였다. 예를 들면, 전체 호출 건수가 10개일 경우, (3,3,4), (3,4,3), (4,3,3)의 세가지 경우를 도출하고 이를 모두 적용

하여 최적의 방법을 선정하였다. 3개의 단계를 나눈 후, 각 단계별 특징 값을 도출 하였다. 각 단계별 time interval의 표준편차, 평균, 최대값 등을 계산하였고, 부가적으로 첫 번째 API name, 마지막 API name 등을 찾아내어 One-hot-encoding을 실시한다[14].

2.6 APK Parser

이 방법은 정적분석을 위해 사용되며, 분석을 위해 약 2000개의 어플리케이션에 대해 특징추출을 위해 apk parse를 이용하였다. apk parse는 오픈소스이며, APK파일에 대한 MD5 해쉬 값, 파일크기, SDK 버전, 라이브러리 정보, 안드로이드 권한 및 컴포넌트 정보 등을 추출할 수 있다. 이 방법에서는 악성코드에서 주로 사용되는 130여개의 권한정보를 특징으로 선택하였으며 입력데이터로는 APK파일에서 추출한 권한목록이 활용된다[15].

III. RFA(Recursive Freature Addition)

1. System Architecture

제안하는 시스템은 악성코드의 분류를 위해 악성과 정상 PE파일의 필드를 추출하고, 추출된 정보를 기반으로 악성코드 분류에 의미가 있는 feature를 우선적으로 선별한다. 이후 악성코드 탐지 모델의 성능 향상을 위해 Feature Selection 방법으로 재귀 변수 추가법(RFA, Recursive Feature Addition)을 사용하여 최종 Feature를 선정하고 모델에 최종적으로 적용시킨다. Fig. 1은 시스템 모델의 전체적인 과정을 나타낸 그림이다.

시스템에 필요한 Dataset을 제작하기 위해, 우선적으로 악성과 정상 판단에 영향을 준다고 판단한 PE파일의 필드를 선정하였고, e_magic, e_lfanew, Signature, Machine, TimeDateStamp, NumberOfSections, Characteristics, SizeOfOptionalHeader, Magic, SizeOfCode, AddressOfEntryPoint, ImageBase, SectionAlignment, FileAlignment과 section을 포함한 17개 초기 Feature를 선별하였다.

이후 악성 및 정상 PE파일에서 각 필드들의 데이터를 CSV파일로 파싱하는 파서를 사용하여 Dataset을 제작하였다. 정상 Dataset은 VMware 가상머신에 설치한 Windows7 32bit clean OS에서 Pcomodel을 사용하여 PE파일 중 exe파일 2,532개, dll파일 9,000개를 추출, 총 11,532개의 RAW Dataset을 제작하였으며, 악성 Dataset은 KAIST에서 받은 악성코드 10,000개를 활용하였다.

1.1 Data Preprocessing

제작된 Dataset은 이상 값 처리, 결측 값 처리, 데이터 인코딩, 데이터 스케일링 4단계로 진행하였다. 이상 값 처리과정에서는 제작하고자 하는 모델을 왜곡할 가능성이 많고 연관성이 없다고 판단한 Feature인 e_magic, Signature, Machine, TimeDateStamp를 pandas에서 제공하는 drop()을 이용하여 제거하였다. 결측 값 처리과정에서는 pandas에서 제공하는 누락 데이터에 특정 값을 채우는 함수 fillna()를 사용하여 평균값으로 결측 값을 처리하였다. 데이터 인코딩 과정에서는 LabelEncoding을 사용하여 기존 데이터의 문자열을 정수형 데이터로 인코딩하였다. 데이터 스케일링과정에서는 StandardScaler를 사용하여 각 Feature의 평균을 0, 분산을 1로 변경하고 모든 특성들이 같은 스케일을 갖게 만든 후, 평균을 제거하고 데이터를 단위 분산으로 조정하였다.

1.2 Feature Selection

각 알고리즘 마다 최상의 정확도를 내는 Feature를 선정하기 위해 본 논문에서 제안하는 재귀 변수 추가법(RFA) 알고리즘은 Fig. 2 와 같다.

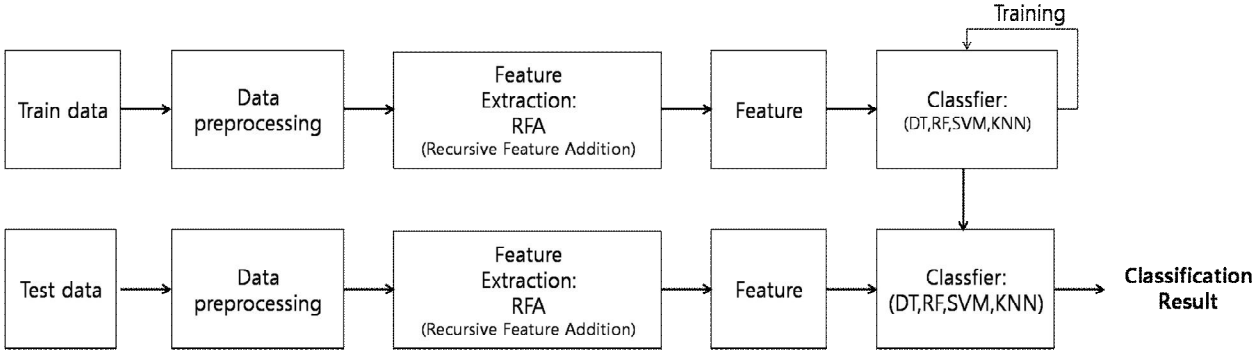

Fig. 1. System Architecture

Feature를 선정하는 과정은 Build Classifier, Feature Addition 의 2단계가 반복적으로 이루어진다.

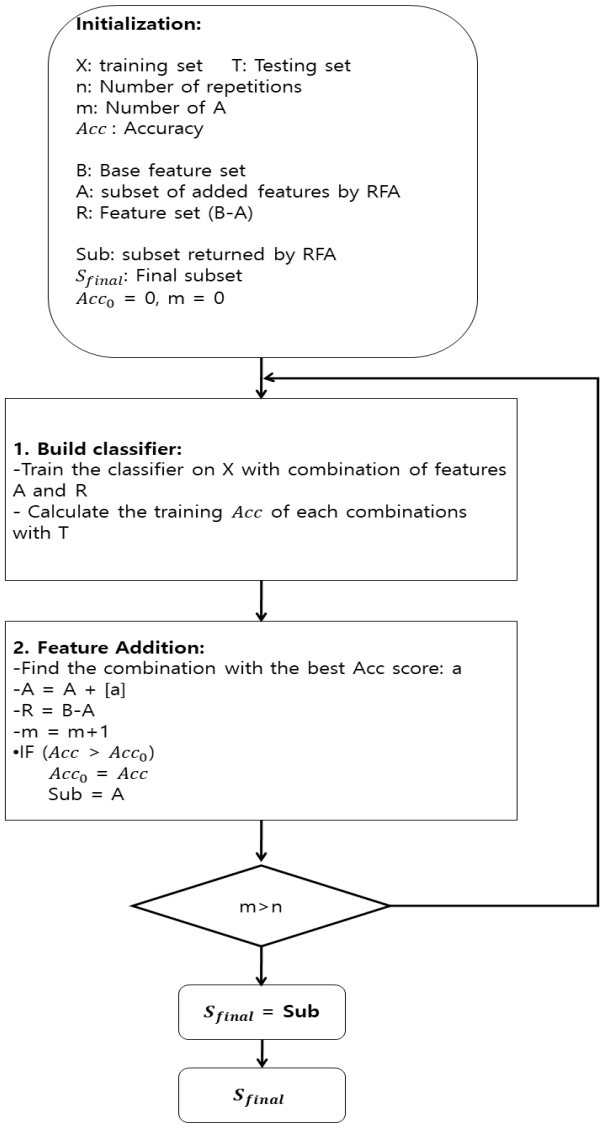

Fig. 2. Feature Selection Process

1.2.1 Build Classifier

RFA알고리즘에 의해 추가된 Feature의 subset인 A와 Base feature set B에서 A를 제외한 나머지 Feature set인 R의 combination을 모델의 Feature로 지정하고, Training set X로 학습시킨다. 각각의 Feature combination으로 생성된 모델을 Testing set T를 이용하여 정확도 Acc를 확인한다.

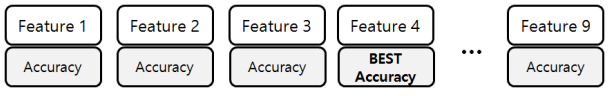

Fig. 3. Step1. Build Classifier

1.2.2 Feature Addition

Build classifier 과정을 통해 가장 높은 정확도 Acc의 점수를 보이는 Feature combination a를 찾는다. 찾은 Feature combination a를 subset A로 설정하고, Base feature set B에서 A를 제외한 나머지 Feature들을 R로 설정한다. subset A에 포함된 Feature의 수 m을 m+1로 변경한다. 만약 Acc값이 기존의 Acc_0값보다 클 경우, Acc_0의 값을 Acc의 값으로 변경하고. 출력되는 subset Sub의 값을 subset A로 변경한다(Fig. 4).

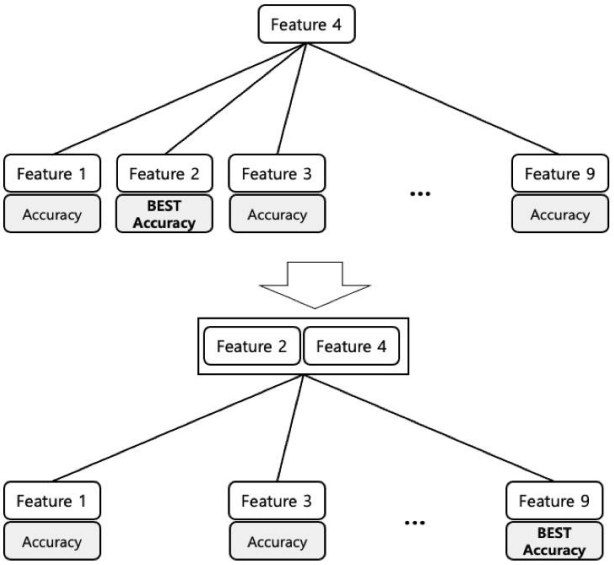

Fig. 4. Step2. Feature Addition

1.2.3 Loop Process

가장 첫 번째 loop에서는 데이터 셋의 Feature 필드들을 RFA 알고리즘의 base feature set으로 지정하고, 각각의 Feature를 기계학습 알고리즘의 단일 Feature로 선정하여 모델을 생성한 후, 정확도를 출력한다. 이때, 가장 정확도 점수가 높은 모델의 Feature만을 A로 지정하고, 지정된 Feature를 제외한 나머지 Feature들은 R로 지정한다. 두 번째 loop때는, A에 포함된 모든 Feature와 R에 포함된 각각의 Feature를 합쳐서 기계학습 알고리즘의 Feature로 선정하여 모델을 생성한 후, 정확도를 확인한다. 첫 번째 loop와 마찬가지로, 가장 정확도가 높은 모델의 Feature를 A로 지정하고, A로 지정된 Feature를 제외한 나머지 Feature들은 R로 지정한다. 이 방법을 원하는 Feature의 수 n개를 출력할 때까지 n번 반복하여 단계적으로 Feature의 combination을 A에 추가하며 진행하면 최종적으로 알고리즘에서 가장 최상의 정확도를 내는 Feature의 subset S_{final}을 구하게 된다.

RFA 사용하여 Feature selection을 하면, n번 이후에는 정확도가 크게 상승하지 않고 유지될 것이다. 유지되는 값(n)이 가장 최적의 결과를 나타낼 수 있는 파라미터가 될 수 있다. 우리는 다음 장에서 실험을 통해, 제안한 방법이 효과가 있음을 검증한다.

IV. Test and Result

1. Test Environment

본 논문에서 제안한 특성추출 방법을 검증하기 위한 실험 환경은 Table. 1과 같이 CPU i7-9700F, RAM 16GB 사양이며, Windows 10 Pro와 개발언어 파이썬(python) 3.8.3버전을 사용하였고, 기계학습을 위해 파이썬 기반 기계학습 라이브러리인 사이킷런(Sci-kit Learn)을 사용하여 실험을 진행하였다.

Table 1. Test Environment

Name	Specufucation
OS	Windows 10 Pro (64bit)
CPU	Inter(R) Core(TM) i7-9700F, 3.00GHz
RAM	16.0GB
GPU	NVIDIA GeForce GTX 1660 SUPER

2. Test Procedures

실험 데이터는 정상파일 10,000개와 악성파일 10,000개로 총 20,000개의 파일을 대상으로 실시하였다. 정상파일은 windows7 32bit clean OS에서 PE파일, 악성파일은 KAIST CSRC(Cyber Securiry Research Center)에서 제공받은 악성코드 데이터를 사용하였다. 데이터의 정상파일과 악성파일의 비율은 1:1로 균등하게 하였으며 실험 시 학습데이터(train data)와 검증데이터(test data)는 7:3의 비율로 구성하였다. 실험은 우연에 의한 테스트 결과의 오류를 방지하기 위해 학습과 테스트를 포함하여 7:3의 비율로 무작위로 총 10회를 실시하였다. 7:3로 나눈 데이터를 활용하여, 비율 7의 데이터를 모델을 학습시키는 Train에 적용하고, 나머지 3의 데이터를 모델 성능을 검증하는 Test에 적용하였다. 정확도(Accuracy)뿐만 아니라 민감도(Sensitivity), 재현률(Recall), F1-score 모두 10회씩 측정하였다. 10회의 평균 정확도로 제안하는 방법에 따라 feature의 개수를 순차적으로 증가하여 n번 진행하였다. 알고리즘은 Decision Tree, Random Forest, SVM, KNN 등 총 4개의 알고리즘을 사용하였다.

3. Test Result and Analysis

먼저 제안하는 RFA에 따라 개별 특성을 추출하여 매 실험에서 가장 좋은 성능을 나타내는 특성을 1개씩 순차적

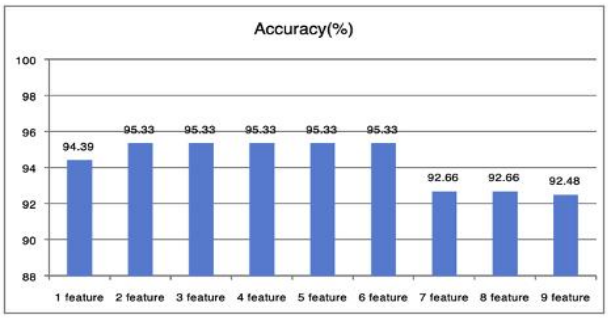

Fig. 5(a) – Decision Tree

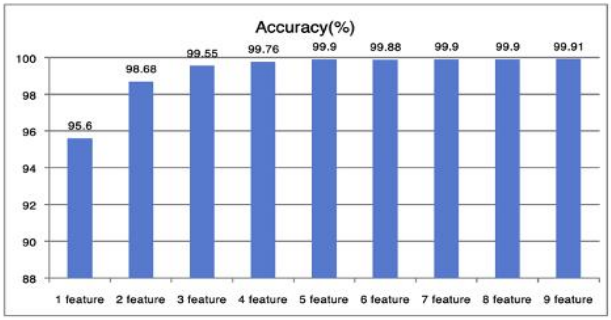

Fig. 5(b) – Random Forest

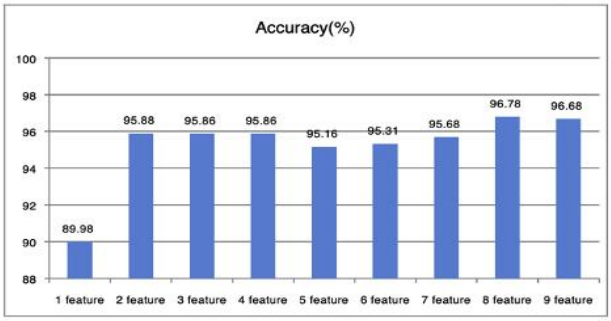

Fig. 5(c) – SVM

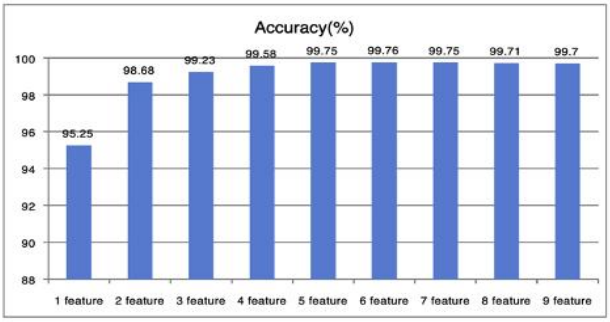

Fig. 5(d) – KNN

Fig. 5. Test result for each Algorithm

으로 증가시키면서 실험 결과 Fig. 5(a)와 같이, Decision Tree 에서는 2개 ~ 5개 특성수 구간에서 정확도가 93.55%로 가장 높게 나타났으며, 7개 이상이 되면 정확도가 다소 떨어지는 경향을 보였다. Fig. 5(b)에서 Random Forest 에서는 4개 특성인자를 사용할 때까지 순차적으로 정확도가 증가하다가 5개 부터는 99.9%의 정확도에서 큰 차이가 없는 것으로 나타났다. Fig. 5(c)에서 SVM에서는 2개 ~ 4개 특성 수 구간에서 정확도가 유사했으며, 이후 소폭 감소 후 8개에서 다시 증가하는 경향을 보였다. Fig. 5(d)를 보면 KNN에서는 4개까지 정확도가 증가하다가 그 이후 큰 변화 없이 높은 정확도가 유지되고 있다.

이러한 결과로 볼 때, 3장에서 가정한 바와 같이 RFA를 사용할 때 최소한의 Feature를 사용하여 최대 99.9%의 학습 성능을 도출할 수 있는 최적의 Feature Selection 방법으로 활용이 가능함이 검증되었다. 다만, RFA의 경우 최적의 Feature를 선정하기 위해 매 단계를 반복해야 하는 비용이 추가된다. 하지만 전처리를 통해 선정된 Feature을 모두 사용하는 것 대비 더 나은 성능을 낼 수 있는 Feature를 선정할 수 있다는 장점이 있다.

V. Conclusions

최근 정보보안 분야에서는 악성코드, 비정상 트래픽을 포함한 전체 데이터에서 악성과 정상 데이터를 효과적으로 분류하기 위해 기계학습을 활용하는 접근이 활발히 연구되고 있다. 또한 기계학습에서는 어떤 Feature를 사용하는가에 따라 모델의 성능에 지대한 영향을 준다. 본 논문에서는 Feature를 선택함에 있어 보다 효과적으로 높은 성능을 낼 수 있는 Feature Selection 방법인 RFA를 제안하였다. 제안한 방법을 사용하여 정상 실행파일과 악성코드를 대상으로 4가지의 기계학습 알고리즘에 적용하여 실험한 결과 가장 효과적인 성능을 보여주는 Feature를 잘 선택할 수 있다는 것이 검증되었다. 다만, 본 연구를 수행함에 있어 악성코드와 정상 파일을 분류하기 위한 특성 추출의 과정에서 PE파일의 섹션의 수가 파일마다 상이하여 수작업으로 일부 분류하는 작업이 포함되었다. 향후 이 부분은 개선이 필요할 것으로 판단된다. 본 연구의 결과를 활용한다면 악성코드 뿐만 아니라 트래픽, 로그 등 다양한 데이터에서 기계학습에 활용할 수 있는 효과적인 Feature를 추출할 수 있을 것으로 기대한다. 다만, 본 논문에서 제안한 방법은 최종적인 Feature를 선택하기 위해 전처리가 완료된 각각의 데이터를 활용하여 수회 동일한 알고리즘을 실행해야 한다는 한계점이 있다. 향후 이러한 한계점을 개선하고, 최소한의 실행으로 가장 효과적인 Feature를 추출할 수 있는 연구를 지속할 것이다.

REFERENCES

[1] ENISA Threat Landscape 2020, *https://online.flippingbook.com/view/165705/*, Oct. 2020.

[2] Dong-Geun Lee. "Analysis of Malware Detection Techniquesbased on Machine Learning." Graduate School of Soonchunhyang University, Feb. 2018

[3] El Merabet, Hoda, and Abderrahmane Hajraoui. "A survey of malware detection techniques based on machine learning." International Journal of Advanced Computer Science and Applications. Vol. 10 No. 1, pp. 366-373. 2019.

[4] Feature engineering, *http://www.incodom.kr/%EA%B8%B0%EA%B3%84%ED%95%99%EC%8A%B5/feature_engineering.*

[5] Decision Tree, *https://ko.wikipedia.org/wiki/%EA%B2%B0%EC%A0%95_%ED%8A%B8%EB%A6%AC*

[6] Random forest, *https://ko.wikipedia.org/wiki/%EB%9E%9C%EB%8D%A4_%ED%8F%AC%EB%A0%88%EC%8A%A4%ED%8A%B8*

[7] SVM, *https://ko.wikipedia.org/wiki/%EC%84%9C%ED%8F%AC%ED%8A%B8_%EB%B2%A1%ED%84%B0_%EB%A8%B8%EC%8B%A0*

[8] KNN, *https://ko.wikipedia.org/wiki/K-%EC%B5%9C%EA%B7%BC%EC%A0%91_%EC%9D%B4%EC%9B%83_%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98*

[9] Woo-Seok Go, Chun-Gyeong Yoon, Han-Pil Rhee, Soon-Jin Hwang, Sang-Woo LEE, “A Study on the prediction of BMI(Benthic Macroinvertebrate Index) using Machine Learning Based CFS(Correlation-based Feature Selection) and Random Forest Model“, Journal of Korean Society on Water Environment, Vol.35, No.5, pp.425-431, September, 2019. DOI:.10.15681/KSWE.2019.35.5.425

[10] Sung-Guk Choi, “A Study on the Prediction of Intrusion Types Using a Support Vector Machine”, Yonsei University, Feb. 2016.

[11] Hong-bi Kim, Tae-jin Lee, “Stacked Autoencoder Based Malware Feature Refinement Technology Research”, Journal of Korea Institute of Information Security & Cryptology, Vol.30, No.4, pp-593-603, Aug. 2020. DOI:10.13089/JKIISC.2020.30.4.593

[12] Seong-Min Jeong, Hyeon-Seok Kim, Young-Jae Kim, Myung-Keun Yoon, “V-gram: Malware Detection Using Opcode Basic Blocks and Deep Learning”, Journal of KIISE, Vol.46, No.7, pp.599-605, July, 2019. DOI:10.5626/JOK.2019.46.7.599

[13] Jin-Young Cho, Eun-Gi Ko, Hye-Bin Yoo, Mi-Ri Cho, Chang-Jin

Seo, “A Study on Malware Detection System Using Static Analysis and Stacking”, The Transactions of the Korean Institute of Electrical Engineers, Vol.69P, No.3, pp.187-192, September, 2020. DOI:10.5370/KIEEP.2020.69.3.187

[14] Young-Min Cho, Hun-Yeong Kwon, “Machine Lerning Based Malware Detection Using API Call Time Interval”, Journal of The Korea Institute of Information Security & Cryptology, Vol.30, No.1, pp.51-58, Feb, 2020. DOI:10.13089/JKIISC.2020.30.1.51

[15] Seong-Eun Kang, Nguyen Vu Long, Sou-hwan Jung, “Android Malware Detection Using Permission-Based Machine Learning Apporach”, Journal of The Korea Institute of Information Security & Cryptology, Vol.28, No.3, pp.617-623, Jun, 2018. DOI:10.13089/JKIISC.2018.28.3.617

Authors

Ji-Yun Byeon is a student attending a bachelor's degree in cyber security division at Yeungnam University College in 2021. Ji-Yun Byeon is interested in malware and machine learning among cyber security fields.

Dae-Ho Kim is a student attending a bachelor's degree in cyber security division at Yeungnam University College in 2021. Dae-Ho Kim is interested in malware, machine learning and artificial intelligence among cybers ecurity fields.

Hee-Chul Kim is a student attending a bachelor's degree in cyber security division at Yeungnam University College in 2021. Hee-Chul Kim is interested in malware, machine learning and network security among cybers ecurity fields.

Sang-Yong Choi received his B.S. degree in Mathematics and M.S. degree in Computer Science, both from Hannam University in 2000 and 2003, and Ph.d degree in Interdisciplinary of Information Security from Chonnam National University in 2014, Dr. Choi is a assistant professor at the Dept. of Cyber Security in Yeungnam University College, Daegu, Korea. His research interests are in web security, network security and cloud computing security.