سامانه‌های پردازشی و ارتباطی چندرسانه‌ای هوشمند
Intelligent Multimedia Processing and Communication Systems (IMPCS)

Fall 2024, 5 (3), 171-183
DOR:

Received: 30 Aug 2024
Accepted: 14 Nov 2024

مقاله پژوهشی

The Application of the CatBoost Algorithm for Malware Category Detection

Mohsen Ghasemi*

Assistant Professor, Department of Computer Engineering, Larestan Branch, Islamic Azad university, Larestan, Iran. **Corresponding Author*, Email: Mohsen.Ghasemi@iau.ac.ir

Abstract

Introduction: Android is the most widely used mobile operating system. Its popularity, open-source nature, and extensive app ecosystem have made it a target for malwares. While many studies focus on detecting Android malware, fewer have explored malware classification by attack types. This classification is essential for understanding malicious patterns. This paper proposes using the CatBoost algorithm to detect categories of Android malware, employing a hybrid analysis that combines static and dynamic features. The Kronodroid dataset, which contains 14 distinct malware category labels, serves as the benchmark for evaluation.

Method: A novel feature selection method called VFCE is introduced. Four feature selection techniques - variance, F-test, Chi-Square, and Extra Tree - were applied sequentially to the dataset. By selecting 50, 100, and 200 features using these methods along with VFCE, a total of 15 distinct feature sets were generated. The dataset was divided into training (70%) and testing (30%) subsets. I evaluated the performance of eight classification algorithms: CatBoost, Random Forest, Decision Tree, Support Vector Machine, Logistic Regression, Multi-Layer Perceptron, Bagging, and K-Nearest Neighbors across these feature sets.

Results: The VFCE feature selection method produced better results when selecting 100 features, especially in combination with the CatBoost algorithm. I compared the performance of various algorithms, including CatBoost, using the proposed feature selection method. They were assessed based on several metrics: accuracy, precision, recall, F-measure, false positive rate (FPR), root mean squared error (RMSE), training time, and testing time. CatBoost outperformed the other algorithms and previous studies on this dataset, achieving an accuracy of 93.28%, precision of 93.32%, recall of 93.28%, F-measure of 93.19%, and a rapid testing time of 0.07 seconds.

Discussion: This study examines the use of the CatBoost algorithm for Android malware category detection. The proposed VFCE feature selection method enhances both accuracy and speed. The CatBoost algorithm, in conjunction with the proposed feature selection method, improves accuracy, precision, recall, F-measure, RMSE, and testing time.

Keywords: Android Operating System, Malware Category, Machine Learning, CatBoost.

سامانه‌های پردازشی و ارتباطی چندرسانه‌ای هوشمند
Intelligent Multimedia Processing and Communication Systems (IMPCS)
انجمن علمی تجارت الکترونیکی ایران
واحد لارستان

کاربرد الگوریتم CatBoost در شناسایی دسته بدافزار اندروید

تاریخ دریافت: ۱۴۰۳/۰۶/۰۹
تاریخ پذیرش: ۱۴۰۳/۰۸/۲۴

دورهٔ پنجم، پاییز ۱۴۰۳
شمارهٔ سوم، صص: ۱۷۱-۱۸۳

محسن قاسمی*

استادیار، گروه کامپیوتر، واحد لارستان، دانشگاه آزاد اسلامی، لارستان، ایران. Mohsen.Ghasemi@iau.ac.ir

چکیده: اندروید پراستفاده‌ترین سیستم‌عامل تلفن همراه است و بسیار مورد هدف بدافزارها قرارمی‌گیرد. از الگوریتم‌های یادگیری ماشین برای تشخیص بدافزار اندروید، بسیار استفاده شده‌است. علاوه بر تشخیص مخرب‌بودن یک برنامه، شناسایی نوع حمله نیز مهم است. تعیین دسته بدافزار اندروید، گام مهمی برای درک دقیق تهدیدی است که در برابر آن آسیب‌پذیر هستیم. در این مقاله، از الگوریتم CatBoost و از رویکرد ترکیبی با ویژگی‌های ایستا و پویا برای تشخیص دسته بدافزار اندروید استفاده‌می‌شود. نتایج این الگوریتم و چند الگوریتم دیگر یادگیری‌ماشین، روی مجموعه‌داده Kronodroid با یکدیگر مقایسه‌می‌شوند. برای انتخاب ویژگی‌ها، چهار روش انتخاب ویژگی، به‌طور متوالی روی مجموعه‌داده اعمال‌می‌شود. برای ارزیابی مجموعه ویژگی پیشنهادی (VFCE)، براساس تعداد ویژگی و هرکدام از روش‌های انتخاب ویژگی، ۱۵ مجموعه ویژگی را ایجادکرده و دقت الگوریتم‌ها، روی آن‌ها مقایسه می‌شوند. روش پیشنهادی VFCE با ۱۰۰ ویژگی، دقت بهتری دارد. سپس نتایج الگوریتم‌ها با معیارهای دقت، صحت، فراخوانی، معیارF، نرخ‌مثبت کاذب، ریشه میانگین‌مربعات‌خطا، زمان‌آموزش و زمان‌تست، مقایسه می‌شوند. همچنین نتایج با سایر کارهای مرتبط روی این مجموعه‌داده مقایسه می‌شوند. الگوریتم CatBoost با دقت ۹۳,۲۸٪، صحت ۹۳,۳۲٪، فراخوانی ۹۳,۲۸٪، معیارF، ۹۳,۱۹٪ و زمان‌تست ۰,۰۷ ثانیه عملکرد بهتری از سایر الگوریتم‌ها دارد.

واژه‌های کلیدی: سیستم‌عامل اندروید، دسته بدافزار، یادگیری‌ماشین، CatBoost .

۱. مقدمه

بدافزار موبایل، نرم‌افزار مخربی اســت که به‌طور خاص برای هدف قرار دادن دســتگاه‌های تلفن همراه مانند گوشــی‌های هوشــمند و تبلت‌ها طراحی شده‌است. بدافزار به هر نوع کد مخربی اطلاق‌می‌شود که بدون اطلاع یا رضــایت کاربر، یکپارچگی و عملکرد ســیســتم تلفن همراه را تحت تأثیر قرارمی‌دهد. انواع بدافزارها شــامل باج‌افزار، تروجان، کرم، جاسوس‌افزار، روت‌کیت و بات‌نت است [۱].

اندروید یک سیستم‌عامل منبع‌باز برای دستگاه‌های تلفن همراه و یک پروژه متن‌باز به رهبری گوگل است. سیستم‌عامل اندروید براساس یک نسخه تغییریافته لینوکس ساخته‌شده‌است و عمدتا در دستگاه‌های موبایل استفاده‌می‌شود. اندروید، ۳,۳ بیلیون کاربر در سال ۲۰۲۳ و ۷۱,۷۴٪ سهم بازار سیستم عامل‌های موبایل را دارد. ۹۶,۹٪ برنامه‌های آن مجانی است. در مارس ۲۰۲۳، ۲۶۷۳۲۹۲ برنامه در فروشگاه گوگل دردسترس بوده‌است [۲]. با سیاست گوگل مبنی بر هسته منبع‌باز، نویسندگان بدافزار به‌طور بالقوه می‌توانند دانش خوبی از پلت فرم تلفن همراه به‌دست‌آورند و مجرمان سایبری فرصت‌می‌یابند تا بدافزارهای تلفن همراه را ایجاد و منتشرکنند. علاوه بر این، با افزایش تعداد کاربرانی که برنامه‌ها را دانلود و نصب می‌کنند، احتمال نصب بدافزار موبایل نیز افزایش می‌یابد [۳].

دســتگاه‌های اندرویدی به‌طور مداوم توســط بدافزارها مورد حمله قرارمی‌گیرند و آلوده می‌شوند. درفصل اول سال ۲۰۲۴، بیشتر از ۱۰,۱ میلیون نمونه حمله موبایل به‌وسیله کاسپرا سکای گزارش شده‌است. تعداد نمونه‌های شناسایی شده بدافزار و نرم‌افزارهای ناخواسته اندروید در همین فصل، به ۳۸۹۱۷۸ بسته نصب شده رسید [۴]. بیش از ۹۸ درصد از حملات سایبری به دستگاه‌های تلفن همراه، مربوط به اندروید است [۵]. بدافزار اندرویدی می‌تواند باعث عملیاتی مانند دسترسی‌های غیرمجاز، سرقت اطلاعات، جاسوسی، ارسال پیامک غیرمجاز و دانلود اضافی شود [۶].

در دهه اخیر، حجم بسیاری از تحقیقات برای تشخیص بدافزارهای اندروید، از یادگیری‌ماشــین اســتفاده‌می‌کنند. اکثریت قریب به اتفاق مطالعات تشــخیص بدافزار اندروید مبتنی بر یادگیری‌ماشــین، کارایی بالایی را گزارش‌می‌کنند [۷]. برای تشــخیص بدافزار با اســتفاده از الگوریتم‌های یادگیری‌ماشــین، برنامه‌ها باید از قبل پردازش شــوند تا ویژگی‌هایی که رفتار آن‌ها را به بهترین شــکل توصــیف‌می‌کنند، استخراج شوند. این کار با استفاده از تکنیک‌های تحلیل نرم‌افزاری پویا یا ایســتا انجام‌می‌شــود. در تجزیه و تحلیل پویا، رفتار یک برنامه شبیه سازی می شود. تجزیه و تحلیل ایستا بر اساس بررسی فایل‌های موجود در بســته برنامه اســت و نیاز به اجرای کد ندارد. این تکنیک‌ها مزایا و معایب خاص خود را دارند. از یک طرف، از طریق تحلیل پویا، دســترســی به کدهایی که در زمان‌اجرا بارگذاری و اجرامی‌شــوند، امکان‌پذیر است. بااین‌حال، موفقیت این تجزیه و تحلیل، از نظر پوشش کد، تا حد زیادی به مکانیسم شبیه‌سازی و عدم وجود تکنیک‌های فرار از مجازی‌سازی، بستگی دارد [۸]. از سوی دیگر، تجزیه و تحلیل ایستا، قادر اســت تمام اطلاعات موجود در برنامه‌ها را ارزیابی‌کند، اما موفقیت آن در فقدان تکنیک‌های فرار مانند مبهم‌ســازی یا بارگذاری پویای کد است. تحلیل دینامیکی، زمان و توان‌محاسباتی و منابع بیشتر نسبت به تحلیل ایستا مصرف‌می‌کند، به همین دلیل کمتر مورد توجه محققین قرارمی‌گیرد [۷]. رویکرد ترکیبی، با هدف به‌دســت‌آوردن مزایای هر دو رویکرد ایســتا و پویا، آن‌ها را با هم ترکیب می‌کند [۶]. در این تحقیق از رویکرد ترکیبی استفاده‌می‌شود.

تشخیص مخرب‌بودن یک برنامه به‌تنهایی کافی نیست. نوع حمله می‌تواند برای جلوگیری از آســیب مورد نظر بســیار مهم باشــد تا برای تهدید، اقدامات مناسبی انجام‌داد [۹]. دسته‌بندی بدافزارها، بینش‌های عمیق‌تری در مورد رفتار و تکامل انواع مختلف بدافزار ارائه‌می‌دهد. انواع مختلف بدافزار، رفتارها و اهداف متفاوتی دارند. با دسته‌بندی بدافزارها، ســیســتم های امنیتی می‌توانند دفاع های خود را به‌طور موثرتری تنظیم‌کنند و حفاظت بهتری در برابر تهدیدات خاص ارائه‌دهند [۱۰]. بنابراین، ضروری است که راه‌حلی داشته‌باشیم که یک گام فراتر رفته و بدافزار را به خانواده یا دسته خاصی نسبت‌دهد [۱۱].

مطالعات زیادی برای بررسی اثربخشی مدل‌های مختلف یادگیری ماشین برای شناسایی بدافزار اندرویدی، انجام‌شده‌است. اکثر این مطالعات از مجموعه‌داده‌هایی استفاده‌می‌کنند که با استفاده از تجزیه و تحلیل ایستا یا پویا جمع‌آوری‌شده‌اند؛ اما استفاده از رویکردهای تجزیه و تحلیل ترکیبی به‌طور کامل موردتوجه قرارنگرفته است. در [۱۲]، با استفاده از رویکردهای تجزیه و تحلیل ترکیبی، یک مدل یادگیری‌ماشین نظارت‌شده برای تشخیص بدافزار اندروید و تشخیص دسته بدافزار اندروید، ارائه‌شد. از مجموعه‌داده Kronodroid که شامل ویژگی‌های ایستا و پویا است، استفاده‌کردند. این مجموعه‌داده شامل ۲۴۰ خانواده بدافزار است. با کمک چندین مخزن آنلاین، بدافزارها در ۱۵ گروه، دسته‌بندی شدند. با الگوریتم‌های Extra Tree و اطلاعات متقابل، تعداد ۳۰، ۵۰ و ۱۰۰ ویژگی برتر را روی مجموعه‌داده عادی و نرمال‌شده انتخاب و ۱۲ نمونه داده ایجادکردند. دقت الگوریتم‌های درخت تصمیم (DT)، جنگل‌تصادفی (RF)، K نزدیکترین‌همسایگی(KNN) و ماشین-بردارپشتیبان (SVM) را روی این نمونه داده‌ها، با هم مقایسه نمودند. نمونه داده با۵۰ ویژگی انتخاب‌شده با Extra Tree و الگوریتم RF، بالاترین دقت را در هر دو مورد شناسایی بدافزار و شناسایی دسته بدافزار به‌دست‌آورد. با تنظیم پارامترهای ۴ الگوریتم کلاسبندی مذکور، RF، دقت ۹۸,۰۳٪ برای تشخیص بدافزار و ۸۷,۵۶٪ برای تشخیص دسته بدافزار را به‌دست‌آورد.

در این تحقیق، از مجموعه‌داده و دسته‌بندی بدافزار ارائه‌شده در [۱۲] استفاده‌می‌شود. برای انتخاب ویژگی، چهار روش انتخاب ویژگی به‌صورت متوالی استفاده‌می‌شود. الگوریتم‌های CatBoost، پرسپترون چندلایه (MLP)، لجستیک‌رگرسیون (LR) و Bagging نیز به مدل یادگیری اضافه می‌شوند.

CatBoost یک کتابخانه منبع‌باز با کارایی بالا برای تقویت گرادیان در درخت‌های تصمیم است که یاندکس در سال ۲۰۱۷ ارائه‌کرد. این الگوریتم؛ در مقایسه با سایر الگوریتم‌های مبتنی بر درخت تصمیم افزایش گرادیان، می‌تواند ویژگی‌های کلاسبندی را به‌طور موثر و معقول پردازش‌کند و مشکلات سوگیری گرادیان و شیفت (توزیع خطا) پیش‌بینی را حل‌کند. در نتیجه بیش‌برازش را کاهش‌داده و دقت کلاسبندی و توانایی تعمیم مدل را بهبودمی‌بخشد [۱۳].

در ادامه، در بخش۲، سوابق و کارهای گذشته آورده شده‌است. در بخش ۳، متدلوژی، مجموعه‌داده و روش انتخاب ویژگی تشریح می‌شود. در بخش۴، نتایج تجربی الگوریتم‌ها نمایش داده می‌شود و الگوریتم‌های پیاده سازی شده با یکدیگر و با نتایج تحقیق دیگران مقایسه می شوند. در پایان نیز نتیجه‌گیری و کارهای آینده بیان می‌شود.

۲. کارهای گذشته

تحقیقات بسیاری در زمینه تشخیص دسته و خانواده بدافزارهای اندروید با استفاده از یادگیری‌ماشین انجام‌شده‌است. در بخش قبل، تحقیق واحد و قدیر [۱۲] آورده‌شد. در ادامه چند مورد از آخرین این کارها آورده می‌شود.

در [۱۳]، چارچوب شناسایی سریع بدافزار اندروید (FAMD) معرفی شد. با استخراج دنباله‌های آپ‌کدها و مجوزها از نمونه برنامه‌ها، مجموعه ویژگی‌های اصلی ساخته‌می‌شوند. سپس آپ‌کدها با تکنیک N-Gram پیش‌پردازش می‌شوند و با الگوریتم فیلتر مبتنی بر همبستگی سریع بر اساس عدم قطعیت متقارن، ابعاد ویژگی‌ها کاهش می‌یابد. در این تحقیق، برای اولین بار از کلاسبند یادگیری‌ماشین CatBoost برای تشخیص بدافزار اندروید استفاده‌شد. نویسندگان نشان دادند که روش پیشنهادیشان از KNN، RF و XGBoost بهتر عمل می‌کند و زمان مصرف کمتری دارد. آن‌ها از دو مجموعه‌داده استفاده‌کردند. یک مجموعه‌داده شامل ۲۵۷۳۷ برنامه (۱۲۹۸۹ بدافزار و ۱۲۷۴۸ برنامه امن) را جمع‌آوری‌کردند و برای آزمایش‌ها از آن استفاده‌کردند. ۷۰٪ داده‌ها، برای آموزش و ۳۰٪ برای تست استفاده‌شد. برای مقایسه چارچوب پیشنهادیشان با دیگر کارها، از مجموعه‌داده Drebin استفاده‌کردند. علاوه بر شناسایی بدافزار، کلاسبندی خانواده بدافزار را نیز در این چارچوب موردتوجه قراردادند. ۲۰ خانواده برتر بدافزار در این مجموعه‌داده را در نظر گرفتند و کلاسبندی خانواده بدافزار را با چارچوب ارائه‌شده، انجام‌دادند. الگوریتم CatBoost، بهترین عملکرد را داشت و دقت ۹۷,۴٪ و ۹۷,۳۸٪ را به‌ترتیب، برای شناسایی بدافزار و شناسایی خانواده بدافزار اندروید، به‌دست‌آورد.

در [۱۱]، براساس شبکه‌عصبی مصنوعی، یک چارچوب برای شناسایی بدافزار و شناسایی خانواده بدافزار بنام MalDozer، پیشنهاد شد. در این چارچوب، دنباله‌های خام فراخوانی‌های API از کد اسمبلی استخراج‌می‌شود. فراخوانی‌های API برداری‌شده با استفاده از شبکه عصبی کانولوشنال، آموزش داده‌می‌شوند. در طول آموزش، MalDozer می‌تواند به‌طور خودکار الگوهای مخرب را با استفاده از دنباله فراخوانی‌های متد خام در کد اسمبلی، تشخیص‌دهد. MalDozer می‌تواند خانواده بدافزار اندروید را مشخص‌کند.

در [۱۴]، یک روش تشخیص بدافزار اندروید بنام ICCDetector پیشنهاد شد. از ارتباطات و تعاملات بین مولفه‌های داخل برنامه (ICC) یا در سراسر مرزهای برنامه و اینتنت‌ها، به عنوان ویژگی‌هایی برای شناسایی بدافزار استفاده شد. از مجموعه‌داده شامل ۵۲۶۴ بدافزار و ۱۲۰۲۶ برنامه امن، ۱۲۱۶۲۱ ویژگی استخراج شد. با روش انتخاب ویژگی مبتنی بر همبستگی، ویژگی‌های اضافی حذف‌شدند. برای کلاسبندی از SVM و اعتبارسنجی متقابل ۱۰ فولد استفاده‌شد. ارزیابی روی این مجموعه‌داده، دقت ۹۷,۴٪ و نرخ‌مثبت‌کاذب ۰,۶۷٪ را نتیجه‌داد. همچنین، با توجه به ویژگی‌های مؤلفه‌های داخل برنامه، بدافزارهای شناسایی شده توسط ICCDetector، به پنج دسته بدافزار جدید کلاسبندی شدند.

در [۱۵]، یک یک مدل تشخیص بدافزار مبتنی بر یادگیری تصویر عمیق به نام DIDROID، برای شناسایی دسته بدافزار اندروید پیشنهاد شد. فرضیه این بود که تبدیل داده‌های ویژگی‌های ایستا به تصاویر، می‌تواند به شناسایی چندکلاسه برنامه‌های بدافزار اندروید کمک‌کند. نویسندگان، از مجموعه‌داده CCS-CICAndMal2020 با ۴۰۰هزار برنامه (۲۰۰۰۰۰ بدافزار و ۲۰۰۰۰۰ برنامه امن) که خود در تولیدش مشارکت داشتند استفاده‌نمودند. مجموعه‌داده شامل ۱۹۱ خانواده بدافزار است که در ۱۲دسته، برچسب‌گذاری‌شده‌اند. برای انتخاب ویژگی‌ها، از Extra Tree و برای کلاسبندی از شبکه‌عصبی کانولوشنال، استفاده‌شد. نتایج آزمایش‌ها روی داده‌های تست، نشان داد که ۹۳,۳٪ (۳۱۹۰۱ نمونه) بدافزارها، به‌درستی در رده‌های مربوطه کلاسبندی شدند. ۶,۷٪ (۲۲۹۹ نمونه)، اشتباه کلاسبندی شدند.

در [۱۶]، یک تکنیک تجزیه و تحلیل پویا مبتنی بر آنتروپی به نام EntropLyzer پیشنهاد شد. تجزیه و تحلیل پویا، در یک محیط شبیه‌سازی‌شده روی مجموعه‌داده CCCS-CIC-AndMal2020 انجام‌شد که شامل ۱۴۷ خانواده و ۱۲ دسته بدافزار اندروید است. شش دسته از ویژگی‌ها برای کلاسبندی و شناسایی بدافزار اندروید استخراج شد. از آنتروپی شانون برای شناسایی تغییرات رفتاری در خانواده‌های بدافزار و رتبه‌بندی این ویژگی‌ها استفاده‌شد. از الگوریتم‌های مختلف یادگیری ماشین (بیز ساده، DT، RF)، برای کلاسبندی دسته و خانواده بدافزار استفاده‌شد. بهترین نتایج را DT با صحت ۰,۹۸۴، فراخوانی و معیارF، ۰,۹۸۳ به‌دست‌آورد.

یادگیری‌ماشین بانظارت، نیاز به تعداد قابل‌توجهی برنامه‌امن و بدافزار دارد که از قبل شناسایی و برچسب‌گذاری‌شده‌باشند. داده‌های برچسب‌گذاری‌شده در این زمینه، گران و به‌دست‌آوردن آن‌ها دشوار است. داده‌های بدون برچسب، فراوان و ارزان هستند. در [۱۷]، براساس یاگیری نیمه‌نظارت‌شده برای شبکه‌های‌عصبی عمیق، مجموعه‌ای از نمونه‌های برچسب‌گذاری‌شده و بدون برچسب را آموزش دادند. این

جدول ۱ : تحقیقات اخیر برای تشخیص دسته بدافزار اندروید با استفاده از مجموعه‌داده‌های مختلف [۱۲]

مرجع	الگوریتم و سال	مجموعه‌داده و سال انتشار آن	بازه زمانی	رویکرد (ایستا، پویا، ترکیبی)	روش نتخاب ویژگی	شناسایی دسته بدافزار
[۲۰]	Deep ANN (2021)	CICANDMAL2019	-	ترکیبی	-	دقت رویکرد ایستا: ٪۹۲,۵ دقت رویکرد پویا: ٪۸۰,۳
[۲۱]	Machine learning (2018)	Updroid	۲۰۱۸-۲۰۱۴	ترکیبی	-	دقت: ٪ ۹۶,۳۷
[۱۷]	Deep neural network with pseudolabel (2020)	CICMALDroid2020	۲۰۱۸-۲۰۱۷	پویا	-	۴ دسته بدافزار با Fmeasure: 97.8%
[۱۶]	Machine learning (2021)	CCS-CICAndMal2020	-	پویا	Shannon Entropy	۱۲ دسته با صحت: ٪۹۸,۴
[۲۲]	Machine learning (2017)	MacAfee, Drebin, Praguard	-	ایستا	Mean decrease impurity (MDI)	دقت: ٪ ۹۹,۲۶
[۲۳]	Machine learning (2018)	Drebin, Genome VirusShare	۲۰۱۷-۲۰۰۹	پویا	-	۹ دسته بدافزار با Fmeasure: 97.8%
[۱۸]	Deep neural network with pseudo label stack auto encoder (2022)	CICMALDroid2020	۲۰۱۸-۲۰۱۷	ترکیبی	Variance, Mutual information	۵ دسته با صحت: ٪۹۸,۴
[۱۲]	Machine learning (random forest)	Subset of 2020 KronoDroid	۲۰۲۰-۲۰۰۸	ترکیبی	Extra tree , Mutual information	۱۵ دسته با دقت ٪۸۷,۶

چارچوب به عنوان ورودی، رفتارهای مشاهده‌شده پویا را می‌گیرد تا با استفاده از یک تکنیک نیمه‌نظارت‌شده، کلاسبندی دسته بدافزار را فعال‌کند. نویسندگان، مجموعه‌داده CICMalDroid2020، را از نمونه برنامه‌های چند منبع، در فاصله دسامبر ۲۰۱۷ تا دسامبر ۲۰۱۸، ایجادکردند. داده‌ها شامل برنامه‌های‌امن و چهار دسته بدافزار هستند. این مجموعه‌داده شامل ویژگی‌های ایستا و پویاست. آن‌ها مدل پیشنهادیشان (بنام PLDNN) را با روش یادگیری‌ماشین نیمه‌نظارتی انتشار برچسب و سایر الگوریتم‌های رایج یادگیری‌ماشین مقایسه کردند. نتایج نشان داد که این مدل می‌تواند، برنامه‌های اندرویدی را با توجه به دسته بدافزار، با معیار F، ٪۹۷,۸۴ و نرخ‌مثبت‌کاذب، ٪۲,۷۶ کلاسبندی کند که بهتر از روش انتشار برچسب است. در ادامه، در [۱۸]، با ترکیب تحلیل ایستا و پویا، روش نیمه‌نظارت‌شده بنام PLACE برای کلاسبندی دسته بدافزار پیشنهاد شد. نتایج نشان داد که مدل پیشنهادی، دقت ٪۹۸,۲۸ و نرخ مثبت‌کاذب ٪۱,۱۶ را به‌دست‌آورد. همچنین نتایج نشان دادند که با ٪۱ داده‌های برچسب‌گذاری‌شده، مدلی پیشنهادی دقت ٪۹۵,۱۹ را دارد که عملکرد خوبی است.

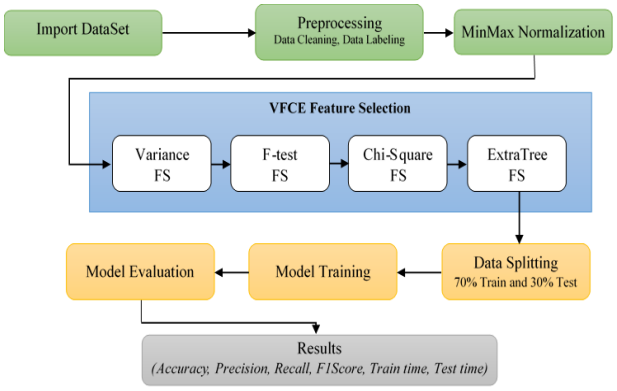
بدافزارهای هوشمند، برای پنهان کردن عملکرد خود و فرار از موتورهای ضد بدافزار، از تکنیک‌های مبهم سازی استفاده‌می‌کنند. یک رویکرد مبهم سازی می‌تواند نسخه‌های بدافزاری تولیدکند که می‌تواند از استراتژی‌های تشخیص بدافزار، فرار کند و دقت تشخیص را به‌طور چشمگیری کاهش دهد. برنامه‌های‌امن نیز برای محافظت از حریم خصوصی کد یا مالکیت معنوی، از مبهم‌سازی استفاده‌می‌کنند. در [۱۹]، رویکردی برای تشخیص تکنیک‌های مبهم‌سازی بدافزارها با یادگیری‌عمیق و با استفاده از مکانیزم رای‌گیری تجمعی پیشنهادشد. از مجموعه‌داده Kronodroid که برای ارزیابی استفاده‌شد. انتخاب ویژگی‌ها با رویکرد Chi-Squared انجام شد و ۲۰ ویژگی از ۴۸۹ ویژگی انتخاب شدند. نتایج نشان‌داد که با زیرمجموعه کوچکی از ویژگی‌ها، می‌توان بخوبی برنامه های غیرمبهم را کلاسبندی کرد، ولی پس از اعمال مبهم‌سازی، کلاسبندی تغییر شدیدی دارد که نشان‌دهنده اهمیت نسبی تکنیک‌های مبهم‌سازی است.

در جدول ۱، تحقیقات اخیر برای تشخیص دسته بدافزار اندروید با استفاده از مجموعه‌داده‌های مختلف، آورده‌شده‌است.

۳. متدولوژی

۳. ۱. توصیف محیط

تمام آزمایش ها با کامپیوتر در برنامه پایتون ۱۲,۳,۱ و در محیط ویندوز ۱۰، ۶۴ بیتی انجام‌شده‌است. از نظر سخت‌افزاری از Intel Core i7-7700HQ 2.8 GHz Processor with 16 GB RAM استفاده‌شده است. در شکل ۱، مراحل و متدولوژی، ترسیم‌شده‌است.



شکل ۱: متدولوژی برای تشخیص دسته بدافزار اندروید

۳. ۲. مجموعه‌داده

در این مقاله، از مجموعه‌داده Kronodroid استفاده‌شده‌است [۲۴]. این مجموعه‌داده شامل ۷۸۱۳۷ نمونه (۴۱۳۸۲ بدافزار و ۳۶۷۵۵ برنامه امن) از سال ۲۰۰۸ تا ۲۰۲۰ است. نمونه‌ها دارای ۲۰۰ ویژگی ایستا و ۲۸۹ ویژگی پویا هستند. ویژگی‌های پویا با اجرای بدافزار بر روی دستگاه واقعی جمع‌آوری شده‌اند. در این مجموعه‌داده علاوه بر برچسب بدافزار یا امن بودن، خانواده بدافزارها نیز مشخص شده‌است.

در اولین مرحله، ۹۳۸۹۴ برنامه‌های اندروید را از ۷ مجموعه‌داده و مخزن مختلف جمع‌آوری کردند. ۵۴۸۳۴ بدافزارها را از Drebin، AMD، VirusTotalو VirusShare و ۳۹۰۶۰ برنامه‌های‌امن را از MARVIN، F-droid و APKMirror دانلود کردند. سپس تحلیل پویا را با دو پلتفرم مختلف یعنی شبیه‌ساز و دستگاه واقعی انجام‌دادند. فراخوانی‌های سیستمی برای ویژگی‌های پویا استفاده‌می‌شوند. سیستم‌عامل اندروید، درخواست‌های سرویس‌ها و منابعی را مانند کنترل فرآیند، مدیریت فایل، مدیریت دستگاه و ارتباطات، مدیریت می‌کند. فراخوان‌های سیستمی در این مجموعه‌داده، با استفاده از یک اسکریپت خودکار با استفاده از دستورات خط فرمان ADB انجام شد. این اسکریپت؛ نصب، اجرا، نظارت، گزارش‌گیری و حذف برنامه های جمع‌آوری‌شده را انجام‌می‌دهد. برای فراخوانی‌های سیستمی، از ابزار monkey استفاده‌کردند. پس از نصب، هر برنامه ۶۰ ثانیه اجرامی‌شود. در زمان‌اجرا، برنامه بررسی می شود و فراخوانی‌های سیستمی صادر شده بوسیله ابزار strace ثبت می‌شود. فایل log ایجادشده، با استفاده از ADB، دخیره‌می‌شود. پس از تحلیل پویا، دو مرحله متوالی تحلیل ایستا انجام‌می‌شود. در مرحله اول، فایل برنامه (apk) و مانیفست آن بررسی و اطلاعات لازم استخراج می‌شوند. فایل مانیفست شامل داده‌هایی از مجوزهای امنیتی (۱۶۶ درخواست مجوز)، سرویس‌ها، فعالیت ها، ویژگی های سخت‌افزاری و نرم‌افزاری مورد نیاز برای دسترسی به منابع است. ویژگی‌های ایستا مورد نظر از فایل مانیفست با استفاده از ابزارهای AndroGuard، Android Asset Packaging Tool (aapt) استخراج‌می شوند. با استفاده از ابزار ExifTool، متاداده مانند نام فایل، زمان آخرین تغییر و زمان اولین تغییر واکشی می‌شود. مرحله دوم تحلیل ایستا با استفاده از آنتی‌ویروس VirusTotal انجام‌می‌گیرد. یک سرویس پیمایشگر آنتی‌ویروس است که برنامه را می‌گیرد و براساس نتایج ۵۰ آنتی‌ویروس، گزارش تشخیص بدافزار، خانواده بدافزار، نرخ تشخیص بدافزار و ... را می‌دهد. پس از تحلیل پویا و ایستا، داده‌های به‌دست‌آمده برای هر برنامه پردازش شده و ویژگی‌ها استخراج می‌شوند. در کل ۴۸۹ ویژگی (۲۰۰ ویژگی ایستا و ۲۸۹ ویژگی پویا) برای هر برنامه به‌دست می‌آید [۲۴].

نسخه اصلی مجموعه‌داده Kronodroid، شامل برچسب دسته بدافزار نیست. در [۱۲]، با کمک چندین مخزن آنلاین، ۲۴۰ خانواده بدافزار این مجموعه‌داده را در ۱۵ دسته برچسب گذاری کردند. در مرحله برچسب گذاری داده، هر بدافزار را با نام دسته بدافزار برچسب‌گذاری کردند. برچسب دسته بدافزار از مخازن آنلاین VirusTotal، F-Secure و FortiGuard به‌دست آمده است. نسخه اصلاح شده و بهبود یافته نهایی مجموعه‌داده در Github در دسترس قرارگرفته‌است [۲۵]. ویژگی‌های غیرعددی sha256 و HighestModDate و EarliestModDate که تأثیری در خروجی ندارند، حذف شده‌اند. ۴۷۹ ویژگی باقی مانده است. مقادیر عددی با تکنیک MinMax نرمال می‌شوند.

در جدول ۲، تعداد بدافزار در هر دسته نمایش داده‌شده‌است. بدافزارهایی که دسته آن‌ها شناسایی نشده‌است در Blank-Category قرارگرفته‌اند. به هر دسته، عددی صحیح اختصاص یافته‌است که جانشین نام آن‌ها در مجموعه‌داده شده‌است. دسته برنامه‌های‌امن (Benign) با عدد ۱۶، مشخص‌شده‌اند.

جدول ۲: تعداد نمونه هر دسته بدافزار

ردیف	دسته بدافزار	تعداد	مقدار عددی متناظر
۱	Adware	۱۱۱۸۳	۱
۲	Trojan	۷۵۴۱	۲
۳	Trojan-SMS	۶۳۴۷	۳
۴	Riskware	۵۳۵۸	۴
۵	Trojan-spy	۴۳۳۶	۵
۶	Ransomware	۲۰۲۹	۶
۷	Trojan-banker	۱۶۸۱	۷
۸	Trojan-Backdoor	۲۲	۸
۹	Scareware	۴۲	۹
۱۰	PUA	۸۸۱	۱۰
۱۱	File-infector	۴۴۲	۱۱
۱۲	Spyware	۲۶۰	۱۲
۱۳	Trojan-dropper	۴۶	۱۳
۱۴	Trojan/Riskware	۱۰۴۸	۱۴
۱۵	Blank-category	۱۶۶	۱۵
۱۶	Benign	۳۶۷۵۵	۱۶

هر بدافزار با توجه به سه مخزن، دسته‌بندی شده‌است. یک چالش، عدم توافق آنتی‌ویروس‌ها و برنامه‌های امنیتی در مورد خانواده و دسته بدافزارهاست. برخی بدافزارها، به وسیله برنامه های امنیتی در خانواده‌های مختلف، برچسب‌گذاری می‌شوند. رأی‌گیری بین نتایج چند برنامه امنیتی یا اولویت‌دهی به نتایج آنتی‌ویروس‌ها را می‌توان به‌کار برد. مثلا اگر یک بدافزار به وسیله VirusTotal و F-Secure در دسته Riskware و به وسیله FortiGuard در دسته Spyware، قرارگیرد، با مکانیزم رای‌گیری، در دسته Riskware قرارمی‌گیرد.

چالش دیگری که در افزایش تعداد دسته‌بندی‌ها باید در نظر گرفت، نامتوازن شدن مجموعه داده است. یعنی تعداد بدافزارها در دسته‌های مختلف، تفاوت زیاد داشته‌باشند. در دنیای واقعی نیز انواع بدافزار متوازن نیستند و برخی انواع بدافزار بیشتر از انواع دیگر هستند. در صورتی که بخواهیم مجموعه‌داده را متوازن‌کنیم، باید تعدادی از بدافزارها را حذف‌کنیم. در صورت حذف، تعداد نمونه‌های مجموعه‌داده

کاهش می‌یابد. در این مجموعه‌داده نیز مشکل نامتوازن بودن تا حدی وجود دارد. به همین دلیل در محاسبه معیارهای ارزیابی از میانگین وزنی استفاده‌شده‌است. در جدول ۱، تحقیقات مشابه روی مجموعه‌داده‌های مختلف آورده شده‌است. در این تحقیقات نیز تعداد ۴، ۹، ۵ و ۱۲ دسته در نظر گرفته شده‌است. این مجموعه‌داده، ۱۵ دسته بدافزار دارد که از بقیه بیشتر است.

برای تنوع بیشتر، می‌توان از نمونه برنامه‌های بیشتر و مخازن دیگر نیز برای دسته‌بندی استفاده‌کرد. همچنین با توجه به تکامل بدافزارها، خانواده و دسته آن‌ها باید در بازه‌های کم زمانی، به‌روزرسانی شوند.

۳. ۳. انتخاب ویژگی

از آنجایی که تعداد زیادی ویژگی در مجموعه‌داده وجود دارد، یافتن مرتبط‌ترین ویژگی‌ها، نقشی حیاتی در یادگیری‌ماشین دارد و به آموزش دقیق و مؤثر مدل کمک‌می‌کند. هدف اصلی فرآیند انتخاب ویژگی، انتخاب مرتبط‌ترین ویژگی‌ها و حذف ویژگی‌های نامربوط و زائد است. انتخاب ویژگی، باعث‌می شود که مدل، سریع‌تر و با حافظه کمتر آموزش ببیند [۲۶]. در اینجا، چهار روش انتخاب ویژگی، به‌صورت متوالی روی مجموعه‌داده اعمال‌می شود. در شکل ۲، شبه کد الگوریتم انتخاب ویژگی آمده است. هدف این است که زنجیره‌ای از روش‌های انتخاب ویژگی استفاده‌شود که در کنار کاهش ابعاد، دقت و سرعت خوبی نیز داشته باشد. همچنین نشان دهیم که روش زنجیره‌ای متشکل از چند روش انتخاب ویژگی، نتایج بهتری از هرکدام از روش‌ها به‌تنهایی دارد.

در کنار این چهار روش، PCA، mutual information و Logistic regression نیز برای انتخاب ویژگی برر سی شدند. از بین این روش‌ها، براساس دقت و سرعت، چهار روش واریانس، F-test، Chi-squared و Extra Tree انتخاب‌شدند. ترتیب اعمال هرکدام از این چهار روش نیز مهم است. انجام به‌ترتیب واریانس، F-test، Chi-squared و Extra Tree، نتایج بهتری داشتند.

روش واریانس روی مجموعه‌داده پس از نرمال‌سازی، انجام‌شده‌است که بیشتر از ۱۸۰ ویژگی را حذف‌می‌کند. ابتدا واریانس هر ویژگی محاسبه‌می‌شود و سپس ویژگی‌هایی که واریانس بالاتری دارند، انتخاب‌می‌شوند. این کار باعث‌می‌شود که ویژگی‌های کم‌اهمیت و با واریانس پایین حذف شوند و مدل یادگیری‌ماشین با ویژگی‌های مهم‌تر و اطلاعات بیشتر آموزش داده‌شود. روش F-test، امکان مقایسه چند گروه را به‌طور هم‌زمان فراهم‌می‌کند و در سناریوهایی با بیش از دو گروه کارآمد است. در اینجا نیز چند کلاس (گروه) از بدافزارها را داریم. روش F-test، سرعت بالایی دارد. روش Extra Tree در کارهای دیگر نیز نتایج خوبی داشته‌اند. در [۱۲]، Extra Tree و mutual information روی مجموعه‌داده Kronodroid برای تشخیص دسته بدافزار، با هم مقایسه شدند که Extra Tree نتایج بهتری داشت. در [۲۷]، سه روش انتخاب ویژگی chi-square، F-test و mutual information انجام‌شد که chi-square نتیجه بهتری داشت. F-test در رتبه بعدی قرارگرفت.

در کارهای دیگر مانند [۱۹، ۲۸] نیز، از chi-square استفاده‌شده‌است.

نام این روش را VFCE می‌نامیم که از قراردادن حرف اول هرکدام از چهار روش انتخاب ویژگی به‌دست آمده‌است. ابتدا ویژگی‌هایی را که واریانس صفر یا نزدیک به صفر دارند، حذف‌می‌کنیم. ۳۰۰ ویژگی باقی می‌ماند.

تابع SelectKBest درکتابخانه Scikit-Learn پایتون، k ویژگی با بالاترین امتیاز را نگه‌می‌دارد و بقیه را حذف‌می‌کند. با این تابع، ابتدا ۲۵۰ ویژگی از ۳۰۰ ویژگی مرحله قبل که امتیاز F-test بیشتر دارند، انتخاب‌می‌کنیم. در واقع، F-value بین هر ویژگی و برچسب (کلاس) را محاسبه‌می شود و ۲۵۰ ویژگی با مقدار بالاتر، انتخاب‌می شود. مجددا با Chi-Square، ۲۲۰ ویژگی از ۲۵۰ ویژگی مرحله قبل را انتخاب‌می‌کنیم. Chi-Square روشی آماری برای آزمون استقلال دو متغیر است. در انتخاب ویژگی، Chi-Square بررسی‌می‌کند که یک ویژگی به چه میزان مستقل از برچسب است.

```
VFCE_Feature_Selection (X, Y)
    X_Var = VarianceThreshold (threshold = 0).fit_transform(X)
    X_Ftest = SelectKBest (f_classif, k = 250).fit_transform (X_Var, Y)
    X_Chi = SelectKBest (chi2, k = 220).fit_transform (X_Ftest, Y)
    Model =ExtraTrees (max_features = 2) .fit (X_Chi, Y)
    X_new = TopFeatures (Model. feature_importances, 100)
Return  X_new
```

شکل ۲: شبه کد انتخاب ویژگی VFCE

از کلاسبندی Extra Tree، برای انتخاب نهایی ویژگی‌ها استفاده می‌شود. Extra Tree یک نوع تکنیک یادگیری تجمعی است که چندین درخت‌تصمیم را به‌صورت موازی ایجادمی‌کند. هر درخت با استفاده از یک زیرمجموعه تصادفی از ویژگی‌ها و نقاط داده، ساخته می‌شود. این کلاسبند، نتایج چندین درخت‌تصمیم بدون همبستگی در یک جنگل را جمع‌آوری می‌کند تا نتیجه کلاسبندی آن را به‌دست

جدول ۳: مجموعه ویژگی‌های ایجادشده

الگوریتم انتخاب ویژگی	نام مجموعه ویژگی	تعداد ویژگی‌ها
Variance	Variance 50	۵۰
	Variance 100	۱۰۰
	Variance 200	۲۰۰
F_Test	F_Test 50	۵۰
	F_Test 100	۱۰۰
	F_Test 200	۲۰۰
Chi-Square	Chi-Square 50	۵۰
	Chi-Square 100	۱۰۰
	Chi-Square 200	۲۰۰
Extra Tree	Extra Tree 50	۵۰
	Extra Tree 100	۱۰۰
	Extra Tree 200	۲۰۰
Combination Variance, F-Test, Chi-Square and Extraree Algorithms sequentially	VFCE 50	۵۰
	VFCE 100	۱۰۰
	VFCE 200	۲۰۰

آورد [۱۵]. در طول ساخت جنگل، به‌طور ضمنی ویژگی‌ها را بر اساس سهم آن‌ها در کاهش ناخالصی و پیش‌بینی‌های دقیق، رتبه‌بندی می‌کند. در اینجا تعداد مورد نظر (۵۰،۱۰۰ و ۲۰۰) ویژگی بهتر را انتخاب‌می‌کنیم.

برای ارزیابی روش ترکیبی انتخاب ویژگی، مجموعه ویژگی های هر کدام از روش های انتخاب ویژگی نیز، جداگانه روی مجموعه‌داده اعمال‌می‌شود. مجموعه ویژگی‌های متفاوت با ۲۰، ۳۰، ۵۰، ۱۰۰ و ۲۰۰ ویژگی برتر امتحان‌شدند. در جدول ۳، ۱۵ مجموعه ویژگی متفاوت که نتایج کلاسبندی بهتری داشتند، انتخاب‌شدند.

۳. ۴. الگوریتم کلاسبندی CatBoost

CatBoost یک کتابخانه منبع‌باز با کارایی بالا برای تقویت گرادیان در درخت‌های تصمیم است که توسط یاندکس در سال ۲۰۱۷ ارائه شد. CatBoost الگوریتم بهبودیافته درخت‌تصمیم افزایش گرادیان (GBDT) است [۱۳].

در الگوریتم‌های بوستینگ کلاسیک، گرادیان‌های مورد استفاده در هر مرحله، با استفاده از همان مجموعه‌داده - که مدل فعلی بر روی آن‌ها ساخته‌شده‌است - تخمین زده می‌شوند و برای آموزش و به‌دست‌آوردن کلاسبند پایه، استفاده‌می‌شوند. این عمل، باعث سوگیری گرادیان، توزیع آن و بیش‌برازش می‌شود. به جای آن، CatBoost از مفهوم تقویت مرتب (Ordered Boosting) استفاده‌می‌کند. فرض کنید مجموعه‌داده $\boldsymbol{D} = \{(X_i, y_i)\}_{i=1..n}$ با n نمونه، مشاهده شده‌است؛ به‌طوری که $X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,m})$ برداری از m ویژگی است. ویژگی‌ها می‌توانند، عددی یا غیرعددی باشند. $y_i \in \mathbb{R}$ مقدار برچسب است. از $\boldsymbol{D}$ برای آموزش استفاده‌می‌نماییم. تعدادی جایگشت تصادفی از داده آموزش ایجادمی‌شود. فرض کنید که $\sigma = (\sigma_1, \dots, \sigma_n)$ یک جایگشت تصادفی روی نمونه‌های آموزش است و n مدل پشتیبان مختلف $s_1, \dots, s_n$ را داریم، طوری‌که مدل s_i، فقط با استفاده از i نمونه اول جایگشت، آموزش می‌بینند. در هر مرحله، برای به‌دست‌آوردن باقیمانده برای نمونه jام، از مدل s_{j-1} استفاده‌می‌شود [۲۹].

در CatBoost هر مقدار غیرعددی با میانگین برچسب همان مقدار جایگزین می‌شود. یعنی اگر $\sigma = (\sigma_1, \dots, \sigma_n)$ یک جایگشت تصادفی روی داده آموزش باشد، $x_{\sigma_q,k}$ با رابطه (۱)، جایگزین‌می‌شود.

$$x_{\sigma_q,k} = \frac{\sum_{j=1}^{i-1}\left[x_{\sigma_j,k} = x_{\sigma_i,k}\right] y_{\sigma_j} + a.p}{\sum_{j=1}^{q-1}\left[x_{\sigma_j,k} = x_{\sigma_i,k}\right] + a} \qquad (۱)$$

که منظور از [.]، آیورسون براکت است. یعنی اگر $x_{\sigma_j,k} = x_{\sigma_i,k}$ باشد، $\left[x_{\sigma_j,k} = x_{\sigma_i,k}\right] = 1$ و در غیراین‌صورت صفر است. p مقدار پیشینی و معمولاً برابر با مقدار میانگین برچسب برای رگرسیون و احتمال پیشینی کلاس مثبت، برای کلاسبندی دودویی است. a مقداری بزرگتر از صفر است. با جایگذاری مقادیر غیرعددی با مقادیر رابطه بالا، ممکن است اطلاعات را از دست بدهیم. ترکیب ویژگی‌ها این مشکل را کم می‌کند و ویژگی قویتری را ایجادمی‌کند. ترکیب همه ویژگی‌های غیرعددی در مجموعه داده، با توجه به رشد نمایی تعداد ترکیبات، مطلوب نیست. CatBoost ترکیب ویژگی‌ها را هنگام ساختن یک انشعاب جدید در درخت فعلی مورد نظر قرارمی‌دهد. در اولین قسمت ساخت درخت، ترکیب ویژگی‌ها انجام‌نمی شود ولی برای ایجاد قسمت‌های بعدی درخت، تمام ویژگی‌های غیرعددی و ترکیبات موجود در درخت جاری را با تمام ویژگی های غیر عددی مجموعه‌داده، ترکیب‌می‌کند. مقادیر ترکیب ویژگی ها را طبق رابطه (۱) به مقدار عددی تبدیل‌می‌کند. همچنین CatBoost، ترکیبات ویژگی‌های عددی و غیر عددی را نیز تولید می‌کند. ویژگی های غیر عددی را مانند ویژگی‌های عددی به‌خوبی مدیریت‌می‌کند و عملکرد بهتری از الگوریتم‌های مبتنی بر تقویت گرادیان مانند XGBoost، LightGBM و H2O دارد [۳۰].

۳. ۵. معیارهای ارزیابی

الگوریتم‌های کلاسبندی، براساس معیارهای عمومی ارزیابی داده‌کاوی یعنی دقت، صحت، فراخوانی، معیارF، نرخ‌مثبت کاذب (FPR) و ریشه‌میانگین‌مربعات‌خطا (RMSE)؛ که به‌ترتیب براساس روابط (۲) تا (۷) به‌دست‌می‌آیند و همچنین براساس زمان‌آموزش و زمان‌تست با یکدیگر مقایسه‌می‌شوند.

در این کلاسبندی چندکلاسی، معیارها را برای هر کلاس، به‌دست آورده و میانگین وزنی هر معیار محاسبه‌می‌شود. برای هر کلاس c_i:

TP: تعداد نمونه‌هایی است که به‌درستی، بعنوان کلاس c_i کلاسبندی شده‌اند.

TN: تعداد نمونه‌هایی است که به‌درستی، به‌عنوان کلاسی غیر از c_i کلاسبندی شده‌اند.

FP: تعداد نمونه‌هایی است که به‌اشتباه، به‌عنوان کلاس c_i کلاسبندی شده‌اند.

FN: تعداد نمونه‌هایی است که به‌اشتباه، به‌عنوان کلاسی غیر از c_i کلاسبندی شده‌اند.

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \qquad (۲)$$

$$Precision = \frac{TP}{TP+FP} \qquad (۳)$$

$$Recall = \frac{TP}{TP+FN} \qquad (۴)$$

$$F\_measure = \frac{2*Precision*Recall}{Precision+Recall} \qquad (۵)$$

$$FPR = \frac{FP}{FP+TN} \qquad (۶)$$

$$RMSE = \sqrt{1-r^2} \times sd \qquad (۷)$$

r خطای پیش‌بینی و sd انحراف معیار است. خطای بین داده آموزش و تست را خطای پیش‌بینی می‌گویند.

۳. ۶. تنظیم پارامترها

در این تحقیق برای مقایسه کارایی الگوریتم CatBoost روی مجموعه‌داده Kronodroid، از الگوریتم های کلاسبندی RF، SVM، KNN، DT، MLP، LR و Bagging نیز استفاده‌می‌شوند.

در کتابخانه Scikit-Learn پایتون، هایپرپارامترهای پیش‌فرضی برای هر مدل یادگیری در نظر گرفته‌اند. معمولا این مقادیر، برای مسأله مورد نظر بهینه نیستند. تعیین بهترین هایپر پارامتر ها معمولا کار غیرممکنی است ولی با آزمون و خطا می‌توان به مقادیر مناسبی د ست‌یافت. برای این منظور، به نتایج تجربی نیاز ا ست. تعداد زیادی از ترکیبات هایپرپارامترها، مورد آزمون قرارمی‌گیرد و کارایی هر مدل به د ست‌آمده، ارزیابی می شود. جستجوی‌گرید، یک الگوریتم بهینه سازی است که بهترین پارامترها را از فهرستی از گزینه‌های پارامتری، انتخاب می‌کند. این تابع روش آزمون و خطا را خودکار می‌کند.

GridSearchCV یک کتابخانه پایتون است که فرآیند انتخاب بهترین پارامترها برای یک الگوریتم یادگیری‌ماشین را تسهیل می‌کند. این تابع، در زمان‌آموزش یک مجموعه‌داده، همه ترکیبات ممکن از پارامتر ها را ارز یابی می‌کند و بهترین ترکیب را در نظرمی‌گیرد. در [۱۲]، بهترین پارامترهای الگوریتم‌های DT، SVM، KNN و RF با این تابع مشخص شده‌اند. در اینجا نیز برای این الگوریتم‌ها، از همان مقادیر پارمترها استفاده‌می‌شود. مقادیر پارامترهای الگوریتم‌های CatBoost، LR، Bagging وMLP نیز با این تابع مشخص‌می‌شوند. در جدول ۴، مقادیر پارامترهای الگوریتم‌های استفاده‌شده، نمایش داده شده‌است.

برای بررسی دقیق‌تر تأثیر پارامترها، در جدول ۵، تأثیر مقادیر پارام‌ترهای Loss_function، Eval_metric و Learning_rate در الگوریتم CatBoost آمده‌است. نتایج ۸ معیار ارزیابی با ۱۱ ترکیب از

جدول ۴: تنظیم پارامترهای الگوریتم‌ها

الگوریتم	مقدار پارامترها
CatBoost	loss_function = 'MultiClass', eval_metric = 'AUC', learning_rate = 0.3, verbose = True
RF	bootstrap =True, max_depth =300, max_features = 'log2'
DT	Criterion = 'gini' , max_depth = 19
SVM	C = 20, kernel = 'rbf', probability = True
KNN	n_neighbors = 1
MLP	max_iter =100, activation = 'relu', solver = 'adam', learning_rate = 'adaptive'
LR	C = 0.1, penalty = None, solver = 'saga', multi_class = 'multinomial'
Bagging	max_features: 0.5, max_samples: 0.5, n_estimators: 500

مقادیر پارامترهای الگوریتم CatBoost نمایش داده شده‌است. هر کدام از پارامترها با لیستی از مقادیر مختلف، به تابع GridSearchCV داده شده‌است تا الگوریتم CatBoost را با ترکیب‌های مختلف پارامترهای داده شده، اجراکند و بهترین ترکیب مقادیر پارامترها را انتخاب‌کند. از بین ترکیب‌های مختلف پارامترهای، ۱۱ ترکیب در این جدول نمایش داده شده‌است. بهترین نتایج بولد و دومین بهترین نتایج با رنگ قرمز مشخص‌شده‌اند. بهترین ترکیب در سطر آخر جدول آمده‌است که همان ترکیبی است که به‌وسیله GridSearchCV انتخاب شده‌است.

۳. ۷. آموزش مدل

داده‌ها به دو بخش، ۷۰٪ برای آموزش و ۳۰٪ برای تست تقسیم شده‌اند. در شکل ۳، شبه کد کلاسبندی با تقسیم ۷۰٪ آمده‌است. مجموعه‌داده در قالب فایل اکسل ا ست. هر سطر مربوط به ویژگی‌های یک برنامه است. در هر سطر، ستون آخر، کلاس آن برنامه و بقیه ستون‌ها، ویژگی‌های برنامه را مشخص‌می‌کنند. ماتریس X، ۴۷۸ ستون اول داده‌ها را شامل می شود. ستون آخر، در بردار Y، بعنوان برچسب داده‌ها قرارمی‌گیرد. کلاسبندهای مورد استفاده در لیست *Classifiers*

جدول ۵: تاثیر پارامترهای الگوریتم CatBoost در تشخیص دسته بدافزار اندروید

پارامترها			دقت	صحت	فراخوانی	معیارF	نرخ مثبت کاذب	ریشه میانگین مربعات‌خطا	زمان آموزش	زمان تست
loss_function	Eval_metric	Learning_rate								
MultiClass	AUC	۰,۱	۰,۹۲۸۲	۰,۹۲۸۱	۰,۹۲۸۲	۰,۹۲۷۳	۰,۳۶۲۸	۱,۳۹۹۴	۶۴۰,۸۶۱	۰,۴۵۳
MultiClass	AUC	۰,۵	۰,۹۲۹۴	۰,۹۲۸۹	۰,۹۲۹۴	۰,۹۲۸۷	۰,۳۸۳۳	**۱,۳۵۴۳**	۳۷۳,۲۱	۰,۰۷۸
MultiClass	AUC	۰,۸	۰,۸۸۱۶	۰,۸۸۲۳	۰,۸۸۱۶	۰,۸۸۱۸	**۰,۱۸۴۹**	۱,۸۹۵	۳۵۸,۸۴۷	**۰,۰۶۷**
MultiClassOneVsAll	MCC	۰,۳	۰,۹۳۱۱	۰,۹۳۱۱	۰,۹۳۱۱	۰,۹۳۰۲	۰,۳۰۷۵	۱,۳۹۴	۳۸۱,۵۸	۰,۰۸۴
MultiClassOneVsAll	AUC	۰,۱	۰,۹۲۵۵	۰,۹۲۵۵	۰,۹۲۵۵	۰,۹۲۴۶	۰,۳۵۷۵	۱,۴۲۱۸	۳۷۱,۶۳۲	۱,۸۷۵
MultiClassOneVsAll	AUC	۰,۵	۰,۹۳۱۹	۰,۹۳۱۳	۰,۹۳۱۹	۰,۹۳۱۱	۰,۲۷۸۲	۱,۳۶۹۶	۴۰۱,۷۲۵	۰,۰۷۸
MultiClass	MCC	۰,۳	۰,۹۳۱۲	۰,۹۳۰۹	۰,۹۳۱۲	۰,۹۳۰۴	۰,۲۴۷۹	۱,۳۷۷۷	۳۹۳,۵۲۴	۰,۰۹۴
MultiClass	MCC	۰,۵	۰,۹۳۱۱	۰,۹۳۰۶	۰,۹۳۱۱	۰,۹۳۰۴	۰,۳۱۳۵	۱,۳۹۱۷	۳۷۲,۲۱	۰,۰۹۴
MultiClassOneVsAll	AUC	۰,۳	۰,۹۳۲۷	۰,۹۳۲۹	۰,۹۳۲۷	**۰,۹۳۲۲**	۰,۳۹۴۱	۱,۳۵۹۷	۴۴۰,۷۷۵	۰,۱۰۹
None	None	None	۰,۹۲۶۱	۰,۹۲۶	۰,۹۲۶۱	۰,۹۲۵	۰,۲۸۳۷	۱,۴۲۶	۳۷۳,۴۰۵	۰,۰۷۸
MultiClass	**AUC**	**۰,۳**	**۰,۹۳۲۸**	**۰,۹۳۳۲**	**۰,۹۳۲۸**	۰,۹۳۱۹	۰,۳۶۱۳	۱,۳۶۶۲	**۳۴۰,۲۵۳**	۰,۰۷

قراردارند. هر عنصر لیست، یک کلاسبند است و همه کلاسبندهای لیست با حلقه For، آموزش و تست را انجام‌می‌دهند. با استفاده از تابع *train_test_split*، تقسیم داده‌ها به ۷۰٪ آموزش و ۳۰٪ *X_train* و *X_test* هستند. *Params* پارامترهای کلاسبند *Classifier* را مشخص‌می‌کند. *Model* با تنظیم پارامترهای کلاسبند ایجادمی‌شود. آموزش با تابع *fit* انجام‌می شود. تابع *predict*، کلاس داده‌های تست را طبق مدل ایجادشده، پیش‌بینی می‌کند. تابع *time*، زمان فعلی را بر حسب ثانیه مشخص‌می‌کند. به کمک این تابع، زمان‌آموزش *Traintime* و زمان‌تست *Testtime*، مشخص‌می شوند. تابع *CalcResults*، معیارهای ارزیابی را محاسبه و در *Results* قرارمی‌دهد.

Inputs: *Dataset, Params:* params of each classifier, *Classifiers*: list of classifiers

Output: *Results*: performance metrics (*Accuracy, Precision, Recall, FMeasure, Traintime, Testtime*)

```
X =Dataset [1:478]
Y =Dataset ['Category']
X = Feature_Selection (X, Y)
X_train, X_test, Y_train, Y_test = train_test_split (X, Y, train_size=0.70, shuffle=True)
For Classifier in Classifiers:
    Model =Classifier (Params)
    Tick = time ( )
    Model.fit (X_train, Y_train)
    Traintime = time ( ) – Tick
    y_predict= Model.predict (Xtest)
    Testtime = time ( ) - Traintime
    Results=CalcResults (Xtest, Ytest, y_predict, Model)
```

شکل ۳: شبه کد الگوریتم‌های کلاسبندی با تقسیم ۷۰٪

۴. آزمایش‌ها و نتایج

۴. ۱. نتایج

در جدول ۶، دقت الگوریتم‌های کلا سبندی پیاده سازی شده، روی ۱۵ مجموعه ویژگی انتخاب شده از مجموعه‌داده Kronodroid، نمایش داده شده‌ا ست. بهترین نتیجه هر الگوریتم، بولد شده‌ا ست. دومین بهترین نتایج با رنگ قرمز مشخص‌شده‌اند. بالاترین دقت را الگوریتم CatBoost با مجموعه ویژگی VFCE 100 دارد. دومین بهترین دقت نیز با الگوریتم CatBoost و مجموعه ویژگی Extra Tree 200 به‌دست‌آمده‌ا ست. طبق نتایج الگوریتم ها در جدول ۶، مجموعه ویژگی VFCE 100 دقت بالاتری از سایر مجموعه ویژگی‌ها دارد. بالاترین دقت الگوریتم‌های RF، SVM، Bagging و CatBoost با مجموعه ویژگی VFCE 100 به‌دست آمده‌است. مجموعه ویژگی VFCE 200 در رتبه دوم قراردارد. نتایج نشان‌می‌دهند که روش انتخاب ویژگی پیشنهادی VFCE، نتایج بهتری به‌دست‌می‌آورد. از مجموعه ویژگی VFCE 100 برای مقایسه نتایج الگوریتم‌ها براساس معیارهای ارزیابی، استفاده‌می‌شود.

جدول ۶: دقت کلاسبندی تشخیص دسته بدافزار در مجموعه ویژگی‌های مختلف

روش انتخاب ویژگی	DT	KNN	SVM	LR	MLP	Bagging	RF	CatBoost
Variance 50	۰,۹۰۰۱	۰,۹۰۳۸	۰,۹۱۷۴	۰,۸۵۸۱	۰,۹۰۷۵	۰,۹۰۹	۰,۹۲۰۷	۰,۹۱۹۲
Variance 100	۰,۹۰۲۲	۰,۹۰۷۵	۰,۹۱۹۸	۰,۸۶۵۰	۰,۹۱۰۹	۰,۹۱۵۲	۰,۹۲۳۳	۰,۹۲۴۵
Variance 200	۰,۹۰۱۴	۰,۹۱۰۲	۰,۹۲۰۳	۰,۸۶۸۲	۰,۹۱۱۷	۰,۹۲۲۲	۰,۹۲۶۵	۰,۹۳۰۷
F_Test 50	**۰,۹۰۵۸**	۰,۹۰۹۹	۰,۹۰۹۶	۰,۸۵۲۲	۰,۹۰۱	۰,۹۲۲۱	۰,۹۳	۰,۹۲۸۷
F_Test 100	۰,۹۰۲۵	۰,۹۰۹۴	۰,۹۱۷۲	۰,۸۶۲۸	۰,۹۰۹۳	۰,۹۲۰۳	۰,۹۲۷۲	۰,۹۲۸۹
F_Test 200	۰,۹۰۴۵	۰,۹۱۰۶	۰,۹۲۰۶	۰,۸۶۸۳	۰,۹۱۳۳	۰,۹۱۳۳	۰,۹۲۳۷	۰,۹۳۱۹
Chi-Square 50	۰,۹۰۱۹	۰,۹۰۷۵	۰,۹۲۰۴	۰,۸۵۷۶	۰,۹۰۶۹	۰,۹۱۶۱	۰,۹۲۶۶	۰,۹۲۳
Chi-Square 100	۰,۹۰۳۴	۰,۹۰۹۴	۰,۹۲۰۱	۰,۸۶۴۸	۰,۹۱۱۹	۰,۹۲۲۷	۰,۹۲۴۸	۰,۹۲۴۹
Chi-Square 200	۰,۹۰۳۵	**۰,۹۱۲**	۰,۹۲۰۶	۰,۸۶۸۲	۰,۹۱۳۲	۰,۹۲۳۱	۰,۹۲۵۵	۰,۹۲۰۶
Extra Tree 50	۰,۹۰۱	۰,۹۰۳۹	۰,۹۱۳۱	۰,۸۵۵۳	۰,۹۰۳۱	۰,۹۲۱۶	۰,۹۳۰۷	۰,۹۳۰۶
Extra Tree 100	۰,۹۰۴	۰,۹۰۸۳	۰,۹۲۰۶	۰,۸۶۰۲	۰,۹۱۱۴	۰,۹۲۳۳	۰,۹۳	۰,۹۳۰۶
Extra Tree 200	۰,۹۰۱۸	۰,۹۱	۰,۹۲۰۴	۰,۸۶۸۴	**۰,۹۱۳۴**	۰,۹۲۴۴	۰,۹۲۵۴	۰,۹۳۲۱
VFCE 50	۰,۹۰۰۹	۰,۹۰۳۱	۰,۹۱۸	۰,۸۵۴۹	۰,۹۰۴۶	۰,۹۱۱۷	۰,۹۲۰۷	۰,۹۲۱۵
VFCE 100	۰,۹۰۳۶	۰,۹۱۱۴	**۰,۹۲۰۸**	۰,۸۶۲۳	۰,۹۱۱۷	**۰,۹۲۸۷**	**۰,۹۳۰۹**	**۰,۹۳۲۸**
VFCE 200	۰,۹۰۲۲	**۰,۹۱۲**	۰,۹۲۰۷	**۰,۸۶۸۸**	۰,۹۱۳۲	۰,۹۲۴۰	۰,۹۲۴۹	۰,۹۳۰۵

جدول ۷: نتایج کلاسبندی تشخیص دسته بدافزار با مجموعه ویژگی VFCE 100

الگوریتم	دقت	صحت	فراخوانی	معیارF	نرخ‌مثبت کاذب	ریشه‌میانگین مربعات‌خطا	زمان‌آموزش	زمان‌تست
DT	۰,۹۰۳۶	۰,۹۰۳۶	۰,۹۰۳۶	۰,۹۰۳۵	۰,۰۱۵۸	۱,۷۲۶۴	۳,۰۰۸	**۰,۰۳۱**
KNN	۰,۹۱۱۴	۰,۹۱۰۷	۰,۹۱۱۴	۰,۹۱۱	۰,۳۳۵۳	۱,۵۶۴۸	**۰,۰۷۸**	۵,۸۵۹
SVM	۰,۹۲۰۸	۰,۹۲۰۱	۰,۹۲۰۸	۰,۹۱۹۸	۰,۰۳۴۷	۱,۵۲۳۳	۴۳۰,۴۶	۶۱,۰۲۷
LR	۰,۸۶۲۳	۰,۸۶۰۹	۰,۸۶۲۳	۰,۸۵۹۲	۰,۰۴۶۶	۱,۸۷۶۹	۵۱,۳۳۳	۰,۰۳۸
MLP	۰,۹۱۱۷	۰,۹۱۱۴	۰,۹۱۱۷	۰,۹۱۰۶	۰,۰۴۲۴	۱,۵۹۷۶	۷۰,۵۳۵	۰,۰۶۲
Bagging	۰,۹۲۸۷	۰,۹۲۸۴	۰,۹۲۸۷	۰,۹۲۶۷	۰,۰۶۲۶	۱,۴۱۶۲	۳۳۰,۴۱۶	۳۵,۹۶۱
RF	۰,۹۳۰۹	۰,۹۳۱۲	۰,۹۳۰۹	۰,۹۲۸۸	**۰,۰۱۳**	۱,۴۰۹۳	۱۸,۰۶	۰,۸۷۹
CatBoost	**۰,۹۳۲۸**	**۰,۹۳۳۲**	**۰,۹۳۲۸**	**۰,۹۳۱۹**	۰,۳۶۱۳	**۱,۳۶۶۲**	۳۴۰,۲۵۳	۰,۰۷

در جدول ۷، نتایج الگوریتم‌های کلا سبندی برای ت شخیص د سته بدافزار اندروید، براســاس معیارهای ارزیابی دقت، صــحت، فراخوانی، معیارF، نرخ مثبت کاذب، ریشــه‌میانگین مربعات خطا، زمان آموزش و زمان تســت نشــان‌داده‌شــده‌اســت. از مجموعه ویژگی VFCE 100 اســتفاده‌شــده‌است. بهترین نتایج بولد شــده‌اند. دومین بهترین نتایج با رنگ قرمز مشخص‌شده‌اند.

جدول ۸: مقایسه الگوریتم‌های پیاده‌سازی‌شده با نتایج دیگر

الگوریتم	دقت	صحت	فراخوانی	معیارF
DT [۱۲]	۰,۸۳۰۹	۰,۷۴۱۴	۰,۷۰۸۷	۰,۷۲۴۷
SVM [۱۲]	۰,۸۴۷۸	۰,۷۸۴	۰,۷۱۳۴	۰,۷۴۷
KNN [۱۲]	۰,۸۴۹۲	۰,۷۷۲۷	۰,۷۳۶۷	۰,۷۵۲
RF [۱۲]	۰,۸۷۵۶	۰,۹۰۱۴	۰,۷۴۲	۰,۸۱۳۹
Proposed DT	۰,۹۰۰۹	۰,۹۰۱	۰,۹۰۰۹	۰,۹۰۰۷
Proposed KNN	۰,۹۰۳۱	۰,۹۰۲۴	۰,۹۰۳۱	۰,۹۰۲۷
Proposed SVM	۰,۹۱۸	۰,۹۱۷۲	۰,۹۱۸	۰,۹۱۶۹
Proposed LR	۰,۸۵۴۹	۰,۸۵۱۴	۰,۸۵۴۹	۰,۸۵۰۴
Proposed MLP	۰,۹۰۴۶	۰,۹۰۶	۰,۹۰۴۶	۰,۹۰۴۳
Proposed Bagging	۰,۹۱۱۷	۰,۹۱۷۲	۰,۹۱۳۴	۰,۹۱۶۹
Proposed RF	۰,۹۲۰۷	۰,۹۲۰	۰,۹۲۰۷	۰,۹۱۹۴
Proposed CatBoost	**۰,۹۲۱۵**	**۰,۹۲۱**	**۰,۹۲۱۵**	**۰,۹۲۰۸**

الگوریتم‌هــای CatBoost و RF، بــه‌ترتیــب؛ بهترین نتــایج را در معیارهای دقت، صــحت، فراخوانی، معیارF و ریشــه‌میانگین‌مربعات‌خطا دار ند. الگوریتم های KNN و DT، به‌ترتیب؛ کمترین ز مان‌آموزش را دار ند. DT و CatBoost، به‌ترتیب؛ کمترین ز مان‌تســت را دار ند. در مجموع، CatBoost و RF، به‌ترتیب، رتبه اول و دوم را دارند. SVM و CatBoost بیشــترین ز مان‌آموزش را دار ند. RF ز مان‌آموزش بســیار کمتری از CatBoost دارد. یادآوری می شود که آموزش برای ۷۰٪ داده و ت ست برای ۳۰٪ دیگر داده‌ها، انجام شده‌ا ست. با توجه به اینکه مدل پس از آموزش ایجادمی‌شــود و پس از آن برای پیش‌بینی اســتفاده می شود، زمان‌ت ست اهمیت بیشتری دارد. زمان‌ت ست DT و CatBoost به‌ترتیب، ۰,۰۳۱ و ۰,۰۷ ثانیه ه ستند که زمان خوبی ه ستند. RF نیز ز مان‌آموزش کمتر از یک ثانیه را به‌دســت‌آورده‌اســت. RF و DT، بــه‌ترتیــب، بهترین نرخ‌مثبــت‌کــاذب را دارنــد. CatBoost بــدترین نرخ‌مثبت‌کاذب را دارد.

نتایج کلی نشان‌می‌دهند که الگوریتم CatBoost با دقت ۹۳,۲۸٪، صحت ۹۳,۳۲٪، فراخوانی ۹۳,۲۸٪ و معیارF، ۹۳,۱۹٪ و ریشه میانگین مربعات خطا ۱,۳۶۶۲ عملکرد بهتری از ســایر الگوریتم‌ها دارد. نقطه ضعف آن در نرخ مثبت کاذب و زمان آموزش است.

۴. ۲. مقایسه با دیگر کارها

با توجه به اینکه در این مقاله از مجموعه‌داده و همچنین دســته بدافزارهایی که در [۱۲] ایجاد شــده‌اســت، اســتفاده‌می‌شــود، نتایج به‌دست‌آمده با آن مقایسه‌می شود. مقاله مذکور در سال ۲۰۲۴ انتشار یافته است و تاکنون دسته بدافزارهای مشخص‌شده در این مقاله در کار دیگری بررسی نشده‌است. نتایج در جدول ۸، نمایش داده‌شده است. در [۱۲]، ۵۰ ویژگی برتر انتخاب‌شده با الگوریتم Extra Tree بهترین نتایج را دا شت. برای مقایسه عادلانه، تعداد ویژگی‌های انتخاب شده را ۵۰ در نظرگرفته و نتایج الگوریتم‌ها روی مجموعه ویژگی VFCE 50 که بدتر از VFCE 100 هســتند، در جدول آورده‌شــده‌اســت. نرخ‌مثبت‌کاذب، ریشــه‌میانگین‌مربعات‌خطا، زمان‌آموزش و زمان‌تســت، در [۱۲] گزارش نشده‌است.

بهترین نتایج هر معیار ارزیابی، به‌ترتیب، بولد و قرمز شده‌ا ست. در اینجا نیز، در همه معیارها، مدل پیشــنهادی CatBoost بهترین نتایج را به‌دست‌آورده‌است. الگوریتم RF در رتبه دوم قراردارد.

۵. نتیجه‌گیری

در این مقاله، تشــخیص دســته بدافزار اندروید با الگوریتم CatBoost و به روش ترکیبی (ویژگی‌های ایســتاو پویا) انجام‌شــد. از دســته‌بندی بدافزارها در [۱۲] روی مجموعه‌داده Kronodroid اســتفاده‌شــد. برای انتخاب ویژگی ها، روش VFCE ارائه‌شــد. در این روش، چهار روش انتخاب ویژگی (وار یانس، F-test، Chi-Square و Extra Tree) به‌طور متوالی روی مجموعه‌داده اعمال می‌شــود. براســاس تعداد ۵۰، ۱۰۰ و ۲۰۰ ویژگی در هرکدام از چهار روش انتخاب ویژگی و روش پیشنهادی VFCE، ۱۵ مجموعــه ویژگی را ایجــادکرده و دقــت الگوریتم‌هــای

CatBoost، RF، SVM، KNN، DT، MLP، LR و Bagging، روی این مجموعه ویژگی‌ها، مقایسه‌می‌شوند. مجموعه‌داده به دو بخش ۷۰٪ برای آموزش و ۳۰٪ برای تست تقسیم می‌شود. نتایج الگوریتم‌های کلاسبندی، روی مجموعه ویژگی VFCE 100 براساس معیارهای دقت، صحت، فراخوانی، معیارF، نرخ‌مثبت‌کاذب، ریشه‌میانگین‌مربعات خطا، زمان‌آموزش و زمان‌تست، مقایسه شدند. نتایج مقایسه الگوریتم‌های پیاده‌سازی‌شده، نشان‌می‌دهد که:

- روش انتخاب ویژگی پیشنهادی VFCE با تعداد ۱۰۰ ویژگی، دقت بهتری از چهار روش دیگر انتخاب ویژگی به‌دست‌آورد.
- الگوریتم CatBoost، با دقت ۹۳,۲۸٪، صحت ۹۳,۳۲٪، فراخوانی ۹۳,۲۸٪، معیارF، ۹۳,۱۹٪ و ریشه‌میانگین‌مربعات‌خطا ۱,۳۶۶۲، عملکرد بهتری از سایر الگوریتم‌ها دارد. الگوریتم RF در رتبه دوم این معیارها قراردارد.
- DT و CatBoost، با زمان‌تست کمتر از یک ثانیه، به‌ترتیب؛ کمترین زمان‌تست را دارند.
- نقطه ضعف CatBoost در نرخ‌مثبت‌کاذب و زمان‌آموزش است.

در [۱۲]، الگوریتم‌های RF، SVM، KNN، DT روی این مجموعه-داده، پیاده‌سازی شده‌اند. مقایسه نتایج به‌دست‌آمده با نتایج [۱۲]، نشان‌می‌دهد که:

- الگوریتم CatBoost، عملکرد بهتری از دیگر الگوریتم‌ها دارد.
- الگوریتم های RF، SVM، KNN و DT با روش انتخاب ویژگی پیشنهادی VFCE، نتایج بهتری از همین الگوریتم‌ها در [۱۲]، به‌دست‌می‌آورند؛ درنتیجه، روش انتخاب ویژگی پیشنهادی VFCE، موثر است.

۶. مسیر کارهای آتی و تحقیقات بعدی

مجموعه‌داده Kronodroid، شامل خانواده بدافزارها نیز است. دسته بدافزار براساس خانواده بدافزار ایجاد شده‌است. در آینده می‌توان، تشخیص خانواده بدافزار را با CatBoost و روش انتخاب ویژگی VFCE، انجام‌داد. همچنین با بهبود الگوریتم CatBoost و ترکیب آن با الگوریتم های دیگر مانند DT، نرخ‌مثبت کاذب و زمان‌آموزش را بهبود داد.

روش انتخاب ویژگی‌های پیشنهاد شده را در حوزه‌های دیگر مانند تشخیص بات نت نیز می‌توان، به‌کاربرد.

با توجه به تکامل بدافزارها، خانواده و دسته آن‌ها باید در بازه‌های کم زمانی، به‌روزرسانی شود. می‌توان، در مجموعه‌داده Kronodroid، نمونه برنامه‌های بیشتری را اضافه کرد تا مجموعه‌داده، متوازن‌تر شود و همچنین از مخازن دیگر نیز برای دسته‌بندی بدافزارهای این مجموعه‌داده، استفاده‌کرد.

References

[1] M. Alazab, "Profiling and classifying the behavior of malicious codes", Journal of Systems and Software, vol. 100, pp. 91-102, 2015.

[2] R. Shewale, 20 Android Statistics For 2024 (Market Share & Users). Retrieved December 27, 2023. Available: https://www.demandsage.com/android-statistics

[3] M. Alazab, M. Alazab, A. Shalaginov, A. Mesleh, and A. Awajan, "Intelligent mobile malware detection using permission requests and API calls", Future Generation Computer Systems, vol. 107, pp. 509-521, 2020.

[4] A. Kivva, Android malware and unwanted software statistics for Q1 2024, Securelist, 2024. Available: https://securelist.com/it-threat-evolution-q1-2024-mobile-statistics/112750.

[5] A. Guerra-Manzanares, "Machine learning for android malware detection: mission accomplished? a comprehensive review of open challenges and future perspectives", Computers & Security, vol. 138, p. 103654, 2024.

[6] R. A. Yunmar, S. S. Kusumawardani, and F. Mohsen, "Hybrid android malware detection: a review of heuristic-based approach", IEEE Access, vol. 12, pp. 41255-41286, 2024.

[7] A. Muzaffar, H. R. Hassen, M. A. Lones, and H. Zantout, "An in-depth review of machine learning based Android malware detection", Computers & Security, vol. 121, p. 102833, 2022.

[8] A. Afianian, S. Niksefat, B. Sadeghiyan, and D. Baptiste, "Malware dynamic analysis evasion techniques: a survey", ACM Computing Surveys (CSUR), vol. 52, no. 6, pp. 1-28, 2020.

[9] I. Rajabizadeh, and N. Modiri, "Strategic, tactical, technical, and operational requirements of executive organizations for developing cyber security vulnerability calculator", Intelligent Multimedia Processing and Communication Systems (IMPCS), vol. 2, no. 3, pp. 13-23, 2021. [Online]. Available: https://sanad.iau.ir/fa/Article/903356.

[10] H. H. R. Manzil and S. Manohar Naik, "Android malware category detection using a novel feature vector-based machine learning model", Cybersecurity, vol. 6, pp. 1-11, 2023.

[11] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: automatic framework for android malware detection using deep learning", Digital Investigation, 24, pp. S48-S59, 2018.

[12] M. Waheed and S. Qadir, "Effective and efficient android malware detection and category classification using the enhanced kronodroid dataset", Security and Communication Networks, vol. 2024, no. 1, p. 7382302, 2024.

[13] H. Bai, N. Xie, X. Di and Q. Ye, "Famd: a fast multifeature android malware detection framework, design, and implementation", in IEEE Access, vol. 8, pp. 194729-194740, 2020, doi: 10.1109/ACCESS.2020.3033026

[14] K. Xu, Y. Li and R. H. Deng, "ICCDetector: icc-based malware detection on android", IEEE Transactions on Information Forensics and Security, vol. 11, no. 6, pp. 1252-1264, June 2016, doi: 10.1109/TIFS.2016.2523912.

[15] A. Rahali, A. H. Lashkari, G. Kaur, L. Taheri, F. GAGNON, and F. Massicotte, "DIDroid: android malware classification and characterization using deep image learning", Proceedings of the 2020 10th International Conference on Communication and Network Security, Tokyo, Japan, 2021. Available: https://doi.org/10.1145/3442520.3442522.

[16] D. S. Keyes, B. Li, G. Kaur, A. H. Lashkari, F. Gagnon and F. Massicotte, "EntropLyzer: android malware classification and characterization using entropy analysis of dynamic characteristics", 2021 Reconciling Data

Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS), Hamilton, Canada, pp. 1-12, 2021.
[17] S. Mahdavifar, A. F. Abdul Kadir, R. Fatemi, D. Alhadidi and A. A. Ghorbani, "Dynamic android malware category classification using semi-supervised deep learning", 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Canada, 2020.
[18] S. Mahdavifar, D. Alhadidi, and A. A. Ghorbani, "Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder", Journal of Network and Systems Management, vol. 30, no. 1, p. 22, 2022.
[19] S. Aurangzeb and M. Aleem, "Evaluation and classification of obfuscated android malware through deep learning using ensemble voting mechanism", Scientific Reports, 13, 3093, 2023.
[20] S. I. Imtiaz, S. Rehman, A. R. Javed, Z Jalil, X. Liu, and W. S. Alnumay, "DeepAMD: detection and identification of android malware using high-efficient deep artificial neural network", Future Generation Computer Systems, vol. 115, pp. 844-856, 2021.
[21] K. Aktas and S. Sen, "Updroid: updated android malware and its familial classification", In Secure IT Systems: 23rd Nordic Conference, NordSec 2018, Oslo, Norway, 2018, Proceedings 23 (pp. 352-368). Springer International Publishing, 2018.
[22] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, "Droidsieve: fast and accurate classification of obfuscated android malware", In Proceedings of the seventh ACM on conference on data and application security and privacy, 2017.
[23] H. Cai, N. Meng, B. Ryder, and D. Yao, "Droidcat: effective android malware detection and categorization via app-level profiling". IEEE Transactions on Information Forensics and Security, vol. 14, no. 6, pp. 1455-1470, 2018.
[24] A. Guerra-Manzanares, H. Bahsi, and S. Nomm, "Kronodroid: time-based hybrid-featured dataset for effective android malware detection and characterization", Computers & Security, vol. 110, p. 102399, 2021.
[25] M. Waheed and S. Qadir, Kronodroid improved dataset, 2022. Retrieved June 10, 2024 from https://github.com/semw/kronodroid_improved_hybrid_detection_v2.git.
[26] Y. P. Khah, M. H. Shirvani, and H. Motameni, "Metaheuristic algorithms for feature selection in intrusion detection systems: a systematic review", Intelligent Multimedia Processing and Communication Systems (IMPCS), vol.4, no.3, pp. 63-92, 2023. [Online]. Available: http://sanad.iau.ir/fa/Article/1123897.
[27] C. Ding, N. Luktarhan, B. Lu, and W. Zhang, "A hybrid analysis-based approach to android malware family classification", Entropy, vol. 23, no. 8, p. 1009, 2021.
[28] X. Wang, L. Zhang, K. Zhao, X. Ding, and M. Yu, M, "MFDroid: a stacking ensemble learning framework for Android malware detection", Sensors, vol. 22, no. 7, p. 2597, 2021.
[29] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features", Advances in Neural Information Processing Systems, vol. 31, pp. 6638-6648, 2018.
[30] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: gradient boosting with categorical features support", arXiv: 1810.11363, 2018.