مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٩) العدد (١٢) ٢٠٢٥
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (9) No. (12) 2025

الكشف عن البرمجيات الخبيثة المتخفية في تطبيقات الأندرويد باستخدام التحليل الديناميكي

د . اناس ليلى*

م. حنين حسن**

(تاريخ الإيداع ٢٠٢٥/٨/٢١ . قُبِل للنشر في ٢٠٢٥/١٠/٢)

□ ملخّص□

مع الانتشار الواسع لأجهزة الأندرويد، أصبحت هدفاً رئيسياً لأنواع متعددة من البرمجيات الخبيثة مثل الفيروسات وأحصنة طروادة وبرامج التجسس والفدية، مما يشكل تهديداً لأمن الأجهزة وسلامة بياناتها. ورغم أن التحليل الساكن يتميز بسرعته وكفاءته، إلا أنه قد يعجز عن كشف السلوكيات الخبيثة غير الظاهرة في الكود، بينما يوفر التحليل الديناميكي فعالية أعلى في اكتشاف البرمجيات الخبيثة غير المعروفة أو المشوشة.

قمنا في هذا البحث بتقييم أداء خوارزميات كشف الهجمات على تطبيقات أندرويد بالاعتماد على تحليل سلوك الذاكرة أثناء التشغيل باستخدام بيانات CIC-AndMal2020، وذلك من خلال ثلاث تجارب رئيسية: الأولى لتقييم أداء المصنفات باستخدام مجموعة البيانات الأصلية بدون اختزال السمات، والثانية بعد اختزال السمات باستخدام تقنية اختيار السمات، والثالثة بعد دمج السمات السلوكية المستخرجة من ملفات الذاكرة **(**HPROF**)** بواسطة الأدوات Android Studio Profilerو Memory Analyzer Tool (MAT) مع البيانات الأصلية.

أظهرت النتائج أن دمج السمات السلوكية المستخرجة من الذاكرة مع البيانات الأصلية ساهم في تحسين طفيف في دقة بعض المصنفات مثل Extra Trees (من 98.02% إلى 98.57%) و Random Forest (من 97.91% إلى 98.09%)، في حين حافظ كل من SVMو KNNعلى نفس مستوى الدقة تقريباً. ويُلاحظ أن نموذج Extra Trees حقق أفضل أداء إجمالي، مع تقليل ملحوظ في زمن التدريب مقارنة بباقي النماذج.

الكلمات المفتاحية: CIC-AndMal2020، Extra Trees ، HPROF ، Android Profiler ، MAT ، تحليل الذاكرة الديناميكي ، تقنية اختيار السمات.

*مدرس – قسم النظم و الشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة اللاذقية – اللاذقية – سوريا – البريد الالكتروني Dr.inas.laila8@gmail.com

**طالبة دراسات عليا– ماجستير – قسم النظم و الشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة اللاذقية – اللاذقية – سوريا – البريد الالكتروني eng.haneen.hasan77@gmail.com

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٩) العدد (١٢) ٢٠٢٥
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (9) No. (12) 2025

Detecting Malware Hidden in Android Apps Using Dynamic Analysis

Dr. Inas Laila*

Eng. Haneen Hassan**

(Received 21/8/2025 . Accepted 2/10/2025)

□ ABSTRACT □

With the widespread availability of Android devices, they have become a primary target for various types of malware, such as viruses, Trojans, spyware, and ransomware, posing a threat to device security and data integrity. Although static analysis is fast and efficient, it may fail to detect malicious behaviors hidden in the code. Dynamic analysis, on the other hand, provides greater effectiveness in detecting unknown or obfuscated malware.

In this research, we evaluated the performance of attack detection algorithms on Android applications based on memory behavior analysis using CIC-AndMal2020 data. We conduct three main experiments: the first is to evaluate the performance of classifiers using the original dataset without feature down sampling; the second is after feature down sampling using a feature selection technique; and the third is after combining behavioral features extracted from memory files (HPROF) using the Android Studio Profiler and Memory Analyzer Tool (MAT) tools with the original dataset. The results showed that combining memory-extracted behavioral features with the original data contributed to a slight improvement in the accuracy of some classifiers, such as Extra Trees (from 98.02% to 98.57%) and Random Forest (from 97.91% to 98.09%), while both SVM and KNN maintained approximately the same level of accuracy. It is noted that the Extra Trees model achieved the best overall performance, with a significant reduction in training time compared to the other models.

Keywords: CIC-AndMal2020, Extra Trees, HPROF, Android Profiler, MAT, Dynamic Memory Analysis, Feature Selection Technique.

*Lecturer- Department of Computer Systems and Networks - Faculty of Information Engineering - Latakia University – Syria -Email Address: Dr.inas.laila8@gmail.com

**Master Student- Department of Computer Systems and Networks - Faculty of Information Engineering - Latakia University – Syria – Email Address: eng.haneen.hasan77@gmail.com

١- المقدمة

شهدت الهواتف الذكية نمواً متسارعاً خلال العقدين الماضيين، لتصبح جزءاً أساسياً من حياة المستخدمين اليومية، مع سيطرة نظام التشغيل Android كأكثر أنظمة التشغيل المحمولة انتشاراً، مما جعله هدفاً رئيسياً لهجمات البرمجيات الخبيثة [١][٢]. ومع التطور السريع في البرمجيات وتقنيات الاتصال، أصبحت البرمجيات الخبيثة أكثر تعقيداً وتخفّياً، متجاوزة العديد من آليات الكشف التقليدية، ومتنوعة في وظائفها بين سرقة البيانات، تعطيل النظام، والتجسس، ما يستدعي اعتماد منهجيات متقدمة تعتمد على تحليل سلوك التطبيقات أثناء التشغيل [3][4].

تعد تحليلات الذاكرة الديناميكية من الأدوات الواعدة في هذا السياق، حيث توفر رؤية دقيقة حول سلوك التطبيقات من خلال تتبع استخدام الموارد مثل الذاكرة وعدد الكائنات والعمليات المفتوحة، وهو ما يساعد على كشف الأنماط الخبيثة التي قد يصعب رصدها بالتحليل الساكن فقط [5][6].

نقدم في هذا البحث، إطاراً تجريبياً يعتمد على تحليل الذاكرة الديناميكي باستخدام بيانات CIC-المعززة بسمات مستخرجة من ملفاتHPROF ، مع تطبيق خوارزميات تعلم آلي تقليدية لقياس أثر هذه السمات على دقة وكفاءة النماذج. ترتكز الدراسة على ثلاثة سيناريوهات رئيسية: (١) مقارنة أداء النماذج باستخدام مجموعة البيانات الأصلية بدون اختزال السمات، (٢) مقارنة الأداء بعد اختزال السمات باستخدام تقنية اختيار السمات، و(٣) تقييم تأثير إضافة السمات السلوكية المستخرجة من الذاكرة ودمجها مع البيانات الأصلية.

وتهدف الدراسة للإجابة عن ثلاثة أسئلة رئيسية: (١) ما المصنف الذي يحقق أفضل دقة في كشف البرمجيات الخبيثة على نظام أندرويد؟ (٢) ما تأثير اختزال السمات على أداء ودقة النماذج؟ (٣) ما تأثير إضافة السمات السلوكية المستخرجة من الذاكرة على دقة وكفاءة الكشف؟

٢- الدراسات المرجعية

قدمت الدراسة [٧] بالاعتماد على مجموعة البيانات CIC-AndMal2020، منهجية للكشف عن البرمجيات الخبيثة باستخدام الشبكات العصبية الاصطناعية مع اختيار السمات بواسطة تقنية Recursive Feature Elimination (RFE) وقد أظهرت النتائج أن اختيار السمات الأمثل ساهم في رفع دقة النماذج بشكل ملحوظ مع تقليل الزمن الحسابي، مما يعزز فعالية الكشف عن الأنماط الخبيثة المعقدة ويوضح أهمية تقنيات اختزال السمات في تسريع عملية التعلم الآلي. وفي السياق نفسه، قامت الدراسة [٨] بتقييم فعالية السمات المستخرجة من التحليل الديناميكي للذاكرة في تصنيف البرمجيات الخبيثة على الأندرويد بالاعتماد على نفس مجموعة البيانات، وأكدت أن إدراج السمات الديناميكية في النماذج الإحصائية والتعلم الآلي يمكن أن يحسن دقة الكشف مقارنة بالسمات التقليدية المستندة إلى التحليل الساكن، مع تعزيز قدرة النظام على اكتشاف السلوكيات الخبيثة المخفية التي لا تظهر غالباً في الشيفرة البرمجية. كذلك، توصلت الدراسة [٩] إلى نتائج مشابهة، حيث استعانت بخوارزمية Support Vector Regression (SVR) على السمات الديناميكية، وأثبتت أن تحليل البيانات المستخرجة من ذاكرة التشغيل يساهم في تحقيق دقة عالية في التعرف على البرمجيات الخبيثة، مع قدرة جيدة على التعميم عند اختبارها على تطبيقات مختلفة ومتنوعة.

وفي إطار التطورات الحديثة في هذا المجال، ركّز الباحثون في الدراسة [١١] على تقديم إطار للكشف الديناميكي يعتمد على تقنية Conformal Prediction المدمجة مع مصنف Random Forest، حيث تميز هذا الإطار بقدرته على توفير ضمانات غير متحيزة لمستوى الثقة في قرارات التصنيف، وهو ما يمثل إضافة نوعية من

حيث موثوقية نتائج الكشف، خاصة عند التعامل مع مجموعات بيانات كبيرة تضم آلاف التطبيقات الضارة والسليمة. أما الدراسة [١٢] فقد اتجهت نحو تطوير أداة تحليل شاملة أطلق عليها اسم DroidDissector، والتي جمعت بين خصائص التحليل الثابت مثل الأذونات والـ API call graphs ، وخصائص التحليل الديناميكي مثل مكالمات النظام وحركة مرور الشبكة وملفات السجل، الأمر الذي وفر إطاراً هجيناً يمكنه الاستفادة من نقاط القوة في كلا النهجين لتعزيز دقة النماذج وكفاءتها. وفي المقابل، تناولت الدراسة [١٣] بعداً مختلفاً من التحليل الديناميكي من خلال التركيز على مرحلة الإقلاع للتطبيقات باعتبارها بيئة غنية بالمؤشرات السلوكية، حيث اعتمد الباحثون على تقنيات محاذاة التسلسلات وطرق bagging مع اختبار Wilcoxon لاتخاذ القرار، وقد أظهرت التجارب أن تحليل سلاسل مكالمات النظام في هذه المرحلة المبكرة يتيح إمكانية الكشف السريع والدقيق عن السلوكيات الخبيثة قبل أن تتمكن البرمجيات الضارة من التمويه أو إخفاء أنماطها أثناء التشغيل الكامل.

وانطلاقاً من هذه الأعمال السابقة التي أظهرت قيمة كل من اختيار السمات، والتحليل الديناميكي للذاكرة، والتكامل بين التحليل الساكن والديناميكي، وكذلك التركيز على مراحل تشغيلية محددة، قمت في هذا البحث بتوسيع نطاق التحليل الديناميكي للبرمجيات الخبيثة على الأندرويد من خلال دمج السمات السلوكية المستخرجة من ملفات HPROF ، بما يسمح باختبار أثرها على دقة وكفاءة نماذج التعلم الآلي. وتم تصميم ثلاثة سيناريوهات بحثية تشمل: استخدام البيانات الأصلية كما هي، ثم البيانات بعد اختزال السمات، وأخيراً دمج السمات السلوكية مع البيانات الأساسية، وذلك بهدف الحصول على تقييم شامل لأثر كل نهج في تحسين قدرة أنظمة الكشف على مواجهة البرمجيات الخبيثة المتخفية.

٣- أهمية البحث و أهدافه

يهدف هذا البحث إلى تقييم أداء خوارزميات التصنيف في كشف الهجمات على تطبيقات الأندرويد من خلال بناء نموذج متكامل للكشف عن البرمجيات الخبيثة والتهديدات الأمنية، يعتمد على الكشف اللاحق الذي يعتمد على تحليل بيانات الذاكرة بعد تذبيت التطبيق على الجهاز.

تتمثل أهمية هذا البحث في تقييم قدرة هذه الخوارزميات على تصنيف التطبيقات بدقة عالية مع استخدام أقل عدد ممكن من السمات (البيانات) المؤثرة، مما يساهم في تحسين كفاءة النظام وتقليل الحمل الحسابي مع الحفاظ على جودة التصنيف.

٤- طرق البحث و مواده

اعتمد هذا البحث على التحليل الديناميكي للكشف عن البرمجيات الخبيثة المتخفية في تطبيقات أندرويد، وذلك عبر مراقبة سلوك النظام أثناء التشغيل وتحليل بيانات الذاكرة في بيئة افتراضية آمنة. تم استخدام مجموعة البيانات CIC-AndMal2020 كنقطة انطلاق، والتي تضم سجلات تشغيل متعددة لتطبيقات أندرويد خبيثة وسليمة تم جمعها تحت ظروف تشغيل قريبة من البيئة الحقيقية، بهدف زيادة دقة الكشف وتعزيز موثوقية النتائج.

ولتعزيز دقة الكشف، تم استخراج سمات سلوكية إضافية من ملفات الذاكرة بصيغة HPROF ، حيث تم تسجيل هذه الملفات أثناء تنفيذ التطبيقات باستخدام Android Studio Profiler، ومن ثم تحليلها بواسطة Memory Analyzer Tool (MAT) لاستخلاص مؤشرات سلوكية دالة على النشاط الخبيث، مثل أنماط

استهلاك الذاكرة، عدد الكائنات المفتوحة، واستدعاءات الدوال. بعد ذلك، تم دمج هذه السمات المستخرجة مع السمات الأصلية لمجموعة البيانات لضمان اكتمال المعلومات المتاحة للنماذج.

تم بعد ذلك تدريب وتقييم خوارزميات التعلم الآلي باستخدام لغة Python ضمن منصة Google Colab، التي توفر بيئة مرنة لدعم معالجة البيانات الضخمة وتنفيذ التجارب بسرعة.

ولضمان تقييم شامل لأداء النماذج، تم الاعتماد على مجموعة من المقاييس الإحصائية الشائعة التي تعكس دقة النموذج وفعاليته في التمييز بين الفئات المختلفة. وتشمل هذه المقاييس [14] :

- الدقة (Accuracy)

تمثل النسبة المئوية للعينات التي تم تصنيفها بشكل صحيح من إجمالي العينات. وتعتبر مقياسا عاما لجودة النموذج، وفق العلاقة (١) :

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (١)$$

حيث:

TP = عدد العينات الإيجابية المصنفة بشكل صحيح (True Positive)

TN = عدد العينات السلبية المصنفة بشكل صحيح (True Negative)

FP = عدد العينات السلبية المصنفة بشكل خاطئ كإيجابية (False Positive)

FN = عدد العينات الإيجابية المصنفة بشكل خاطئ كسلبية (False Negative)

- الاستدعاء(Recall)

يقيس قدرة النموذج على اكتشاف الحالات الإيجابية بشكل صحيح، وهو مقياس مهم في حالات الكشف الأمني حيث يكون من الضروري تقليل الحالات غير المكتشفة(FN) ، وفق العلاقة (٢):

$$Recall = \frac{TP}{TP+FN} \qquad (٢)$$

- الدقة الإيجابية (Precision)

تعكس نسبة العينات المصنفة كإيجابية والتي كانت فعلاً إيجابية. وهي مهمة في الحالات التي يكون فيها عدد كبير من الإيجابيات الكاذبة غير مقبول (مثل الإبلاغ الخاطئ عن ملف كخبيث)، وفق العلاقة (٣):

$$Precision = \frac{TP}{TP+FP} \qquad (٣)$$

- مقياس (F1-Score)

يعتبر مقياسا توازنيا بين الدقة والاستدعاء، خصوصا عند وجود تفاوت بين عدد العينات الإيجابية والسلبية. يتم حسابه وفق المتوسط التوافقي بين الدقة والاستدعاء، وفق العلاقة (٤):

$$F1\ Score = \frac{2*Precision*Recall}{Precision+Recall} \qquad (٤)$$

تتيح هذه المنهجية المتكاملة دمج التحليل الديناميكي مع تحليل الذاكرة، وتقييم النماذج باستخدام مؤشرات معيارية دقيقة، مما يوفر أساساً قوياً لمقارنة أداء خوارزميات التعلم الآلي المختلفة وتحسين كشف البرمجيات الخبيثة المتخفية في بيئة أندرويد.

٥- مخطط البحث

تم تصميم هذا البحث من خلال ثلاثة سيناريوهات تهدف إلى تقييم فعالية خوارزميات تصنيف البرمجيات الخبيثة على نظام الأندرويد، بالاعتماد على التحليل الديناميكي ودمج السمات السلوكية المستخرجة من الذاكرة.

- السيناريو الأول: تقييم البيانات الأصلية(Original Dataset)

يهدف هذا السيناريو إلى قياس أداء النماذج التقليديةExtra Trees ، Random Forest، SVM، KNN باستخدام مجموعة البيانات الأصلية CIC-AndMal2020 بدون أي تعديل أو اختزال للسمات، لتحديد كفاءة النماذج في الكشف الأساسي عن التطبيقات الخبيثة.

- السيناريو الثاني: اختيار السمات(Feature Selection)

يركز هذا السيناريو على تقييم أثر استخدام تقنيات اختيار السمات Extra Trees Feature) (Importanceعلى أداء النماذج من حيث الدقة وكفاءة الحساب، مع تقليل الأبعاد غير الضرورية للبيانات.

- السيناريو الثالث: دمج السمات السلوكية المستخرجة من الذاكرة Memory Behavioral) Features Integration)

يهدف هذا السيناريو إلى تعزيز كفاءة النماذج من خلال دمج السمات المستخرجة من ملفات HPROFمع مجموعة البيانات الأصلية، ما يتيح للنماذج التعرف على أنماط خبيثة معقدة وتحسين دقة الكشف مع تقليل الزمن الحسابي.

٦- النتائج و المناقشة

تُعد مجموعة البيانات CCCS-CIC-AndMal-2020 (Canadian Centre for Cyber Security - Canadian Institute for Cybersecurity Android Malware 2020) من أبرز المجموعات المرجعية في مجال تحليل البرمجيات الخبيثة على منصة أندرويد. تم تطويرها وفق منهجية علمية دقيقة بالتعاون بين المركز الكندي للأمن السيبراني (CCCS) ومعهد الأمن السيبراني الكندي (CIC). تحتوي هذه المجموعة على أكثر من 200,000 عينة من التطبيقات، تضم تطبيقات حميدة (Benign) تم جمعها من متجر Google Play الرسمي، وتطبيقات خبيثة (Malware) تنتمي إلى ١٤ عائلة مختلفة من بينها برمجيات الإعلانات المتطفلة (Adware) ، وبرمجيات الفدية (Ransomware) ، وبرمجيات التجسس (Spyware)، وغيرها.

بدأ العمل بتحميل البيانات المدمجة (merged_andmal2020.csv) التي تم فيها تجميع السمات الديناميكية من ملفات التتبع قبل وبعد إعادة تشغيل النظام(Reboot) ، والتي تعكس سلوك التطبيقات أثناء التنفيذ في بيئة معزولة. يوضح الجدول(١) هذه السمات المستخرجة من تتبع استخدام الذاكرة لتطبيقات أندرويد أثناء التنفيذ.

الجدول (١) السمات المستخدمة فيCIC-AndMal2020

	السمة	الشرح
١	Memory_PssTotal	إجمالي حجم الذاكرة الفعلي المستهلك (Proportional Set Size)ويعد أفضل تقدير للذاكرة التي يشغلها التطبيق فعلياً
٢	Memory_PssClean	حجم الذاكرة المشتركة غير المعدلة مثل shared libraries
٣	Memory_SharedDirty	حجم الذاكرة المشتركة التي تم تعديلها أثناء التنفيذ
٤	Memory_PrivateDirty	حجم الذاكرة الخاصة التي تم تعديلها ولا تشارك مع تطبيقات أخرى
٥	Memory_SharedClean	ذاكرة مشتركة نظيفة لم يتم تعديلها
٦	Memory_PrivateClean	ذاكرة خاصة بالتطبيق ولكن لم يتم تعديلها بعد
٧	Memory_SwapPssDirty	الذاكرة المعدلة التي تم نقلها إلى مساحة التبديل
٨	Memory_HeapSize	إجمالي حجم heap الذي تم تخصيصه للتطبيق
٩	Memory_HeapAlloc	مقدار heap المستخدم فعلياً
١٠	Memory_HeapFree	مقدار heap غير المستخدم
١١	Memory_Views	عدد كائنات View النشطة (عناصر واجهة المستخدم)
١٢	Memory_ViewRootImpl	عدد الجذور الهيكلية للواجهات الرسومية
١٣	Memory_AppContexts	عدد كائنات السياق المستخدمة من قبل التطبيق
١٤	Memory_Activities	عدد الأنشطة (Activities) النشطة
١٥	Memory_Assets	عدد كائنات الموارد (Assets) المستخدمة
١٦	Memory_AssetManagers	عدد كائنات إدارة الموارد (AssetManager) المستخدمة
١٧	Memory_LocalBinders	عدد الاتصالات المحلية (Local Binders) عبر واجهةIPC (Inter-Process Communication)
١٨	Memory_ProxyBinders	عدد وكلاء الربط (Proxy Binders) المستخدمة للربط بين العمليات
١٩	Memory_ParcelMemory	مقدار الذاكرة المستخدمة لنقل البيانات عبر كائنات parcel
٢٠	Memory_ParcelCount	عدد كائنات parcel المستخدمة لنقل البيانات
٢١	Memory_DeathRecipients	عدد كائنات death recipients التي تراقب إنهاء العمليات المرتبطة بها
٢٢	Memory_OpenSSLSockets	عدد اتصالات OpenSSL المفتوحة ضمن التطبيق
٢٣	Memory_WebViews	عدد مكونات WebView الفعالة داخل التطبيق

تعكس هذه السمات كيفية إدارة التطبيق للذاكرة خلال وقت التشغيل، ما يساعد في تمييز التطبيقات الخبيثة التي قد تستغل الموارد بشكل غير طبيعي، مثل محاولة استنزاف الذاكرة أو تشغيل مكونات متعددة دون داعٍ.

٦-١- السيناريو الأول: تقييم أداء خوارزميات الكشف باستخدام مجموعة البيانات الأصلية CIC-AndMal2020

بدأت مرحلة المعالجة الأولية للبيانات بعد تحميلها، باستخدام لغة البرمجة Python ضمن بيئة Google Colab، تمت إزالة الأعمدة غير الضرورية مثل Hash، وReboot_State، وكذلك Category وFamily، لضمان التركيز على السمات العددية ذات الصلة بالكشف. ثم تم استخدام LabelEncoder لترميز العمود الهدف Malware_Family، الذي يمثل العائلة الخبيثة التي ينتمي إليها كل تطبيق، حيث تحول التصنيف النصي إلى تمثيل عددي لتناسب النماذج الإحصائية.

بعد ذلك، تم فصل السمات عن التصنيف المستهدف، والتأكد من أن جميع الأعمدة المعتمدة في التدريب هي رقمية بالكامل، ومن ثم تم تطبيق Min-Max Scaler لتطبيع البيانات وضمان أن جميع السمات تقع ضمن نفس النطاق العددي، مما يساعد في تحسين أداء بعض الخوارزميات مثل SVM وKNN، تم تقسيم البيانات إلى مجموعتي تدريب واختبار بنسبة ٨٠/٢٠ باستخدام طريقة train_test_split مع تثبيت (random_state=42) لضمان قابلية إعادة التجربة.

أخيراً، تم تدريب وتقييم أربع خوارزميات تعلم تقليدية شائعة الاستخدام في مجال كشف البرمجيات الخبيثة:

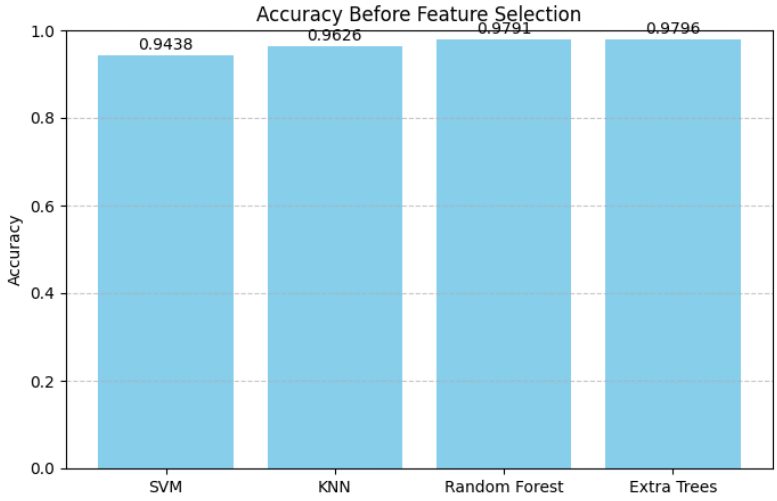
- Extra Trees Classifier : الاشجار شديدة العشوائية Extremely Randomized Trees يشبه الى حد كبير مصف الغابة العشوائية الا ان الاختلاف الأساسي بينهما يكمن في الية تقسيم البيانات للتدريب و التحقق و التي تتم هنا بطريقة عشوائية و هذا ما يقلل من العمليات الحسابية اللازمة للتدريب .
- Random Forest Classifier: هذا النموذج هو عبارة عن مجموعة من أشجار القرار و يعد من طريق التعلم التجميعي للتصنيف حيث يتم تقسيم البيانات و توزيعها على الأشجار و يكون القرار النهائي (الصنف) للغابة هو القرار المعتمد من غالبية الأشجار.
- Support Vector Machine (SVM): يعد من نماذج التصنيف الخاضعة للإشراف و يستخدم غالبا من بيانات مقسمة الى صنفين يمكن الفصل بينهما بشكل خطي.
- K-Nearest Neighbors (KNN): يعد من نماذج التصنيف الخاضعة للاشراف و يقوم بتقسيم البيانات الى عناقيد بالاعتماد على اقرب K جار و تحسب هذه المسافة باستخدام علاقة رياضية .

جرى تقييم النماذج على مجموعة الاختبار باستخدام الدقة (Accuracy) و قياسات مثل Precision و Recall و F1-score بالإضافة الى معيارين أساسيين:

- زمن التدريب(Training Time) : لقياس الوقت المستغرق في بناء النموذج من بيانات التدريب.
- زمن التنفيذ(Execution Time) : لقياس الوقت المستغرق في تصنيف بيانات الاختبار بعد التدريب.

وقد وفرت هذه النتائج الأساس لمقارنة أداء النماذج عند تطبيق تقنيات اختزال السمات أو دمج السمات السلوكية في السيناريوهات اللاحقة.

أظهرت النماذج المبدئية في الشكل (1) المبنية على الأشجار مثل Extra Trees و Random Forest تفوقاً واضحاً من حيث الدقة مقارنة بـ SVM و KNN، مما يعكس كفاءتها في التعامل مع بيانات ذات خصائص متعددة مثل بيانات الذاكرة.



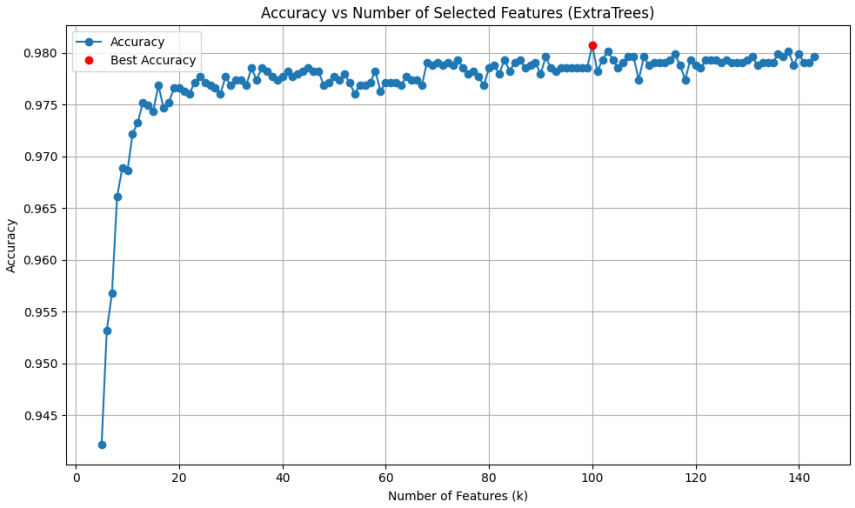
الشكل(1) دقة النماذج على مجموعة البيانات

٦-٢- السيناريو الثاني: تحسين كفاءة النماذج عبر اختزال السمات باستخدام خوارزمية Extra Trees

بعد الانتهاء من التجربة الأولى باستخدام كافة السمات المتوفرة في مجموعة البيانات، تم الانتقال إلى سيناريو جديد يهدف إلى تحسين الأداء وتقليل التعقيد الحسابي و الزمني من خلال تقليل عدد السمات المستخدمة أثناء التدريب، تم استخدام تقنية اختيار السمات تعد هذه التقنية من الخطوات الأساسية في معالجة البيانات الكبيرة، حيث تهدف إلى تحديد السمات الأكثر أهمية وتأثيراً على أداء النماذج مع التخلص من السمات غير المفيدة أو المكررة، مما يقلل التعقيد الحسابي ويسرع عملية التدريب ويحد من خطر التعلم الزائد(Overfitting) .تشمل هذه التقنيات عدة أساليب، مثل الاختيار القائم على الإحصاءات (Statistical Methods)، والاختيار القائم على النماذج (Model-based Selection)، واختيار السمات التكراري (Recursive Feature Elimination) . في هذا البحث، تم استخدام خوارزمية Extra Trees لاختيار السمات نظراً لقدرتها على قياس أهمية كل سمة بشكل مباشر من خلال شجرة القرار العشوائية، وتوفير ترتيب تنازلي للسمات حسب تأثيرها على التصنيف. هذا الأسلوب يسمح بالحفاظ على السمات الأكثر فائدة للنموذج، مع تقليل عدد السمات المستخدمة، وبالتالي تحسين كفاءة النماذج وتسريع زمن التنفيذ دون التضحية بالدقة.

بدأت مرحلة المعالجة المبدئية بتطبيق خطوات أساسية تشمل ترميز السمات النصية باستخدام Label Encoding، حذف السجلات المكررة والقيم المفقودة لضمان جودة البيانات، و إزالة الأعمدة ذات التباين الصفري التي لا تقدم أي معلومات إضافية. بعد ذلك، تم اعتماد خوارزمية Extra Trees لاختيار السمات، نظراً لقدرتها على تقدير أهمية السمات (Feature Importance) من خلال حساب مساهمتها في تقليل عدم التجانس (Impurity Reduction) ضمن أشجار القرار المكونة للنموذج، وبالتالي إنتاج ترتيب تنازلي للسمات بحسب تأثيرها على عملية التصنيف. وبهدف تحديد العدد الأمثل من السمات، تم إجراء سلسلة من التجارب باختيار أعداد مختلفة من السمات الأعلى أهمية وفق هذا الترتيب، وتقييم أداء النماذج باستخدامها. وقد بينت النتائج أن أفضل أداء تحقق عند استخدام

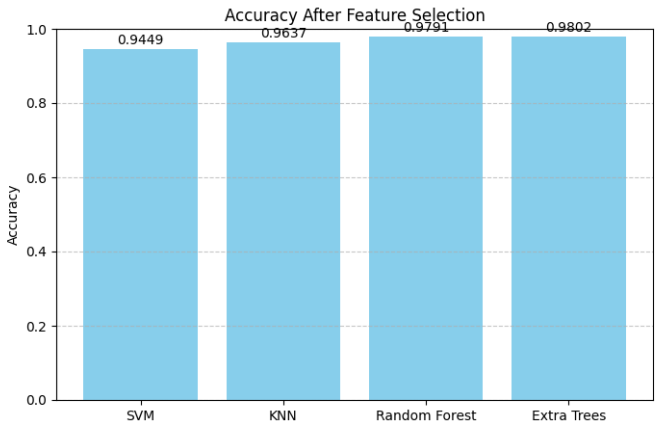
100سمة من أصل ١٤٣، حيث سجلت دقة بلغت 0.9807، مع انخفاض ملحوظ في زمن التدريب والتنفيذ. وعليه، فإن اعتماد ١٠٠ سمة يوفّر التوازن الأمثل بين حجم المعلومات المستخلصة من البيانات وكفاءة النماذج من حيث الدقة والموارد الحسابية كما هو موضح في الشكل (٢).



الشكل(2) دقة النماذج مقابل عدد السمات

تم تقسيم البيانات إلى مجموعتي تدريب واختبار بنسبة 80/20 باستخدام train_test_split مع تثبيت العشوائية لضمان قابلية إعادة التجربة. بعد ذلك، تم تدريب وتقييم النماذج Extra Trees، Random Forest، SVM، KNN باستخدام مؤشرات معيارية تشمل الدقة(Accuracy) ، الدقة الإيجابية(Precision) ، الاسترجاع(Recall) ، معامل F1 ، وزمن التدريب والتنفيذ.

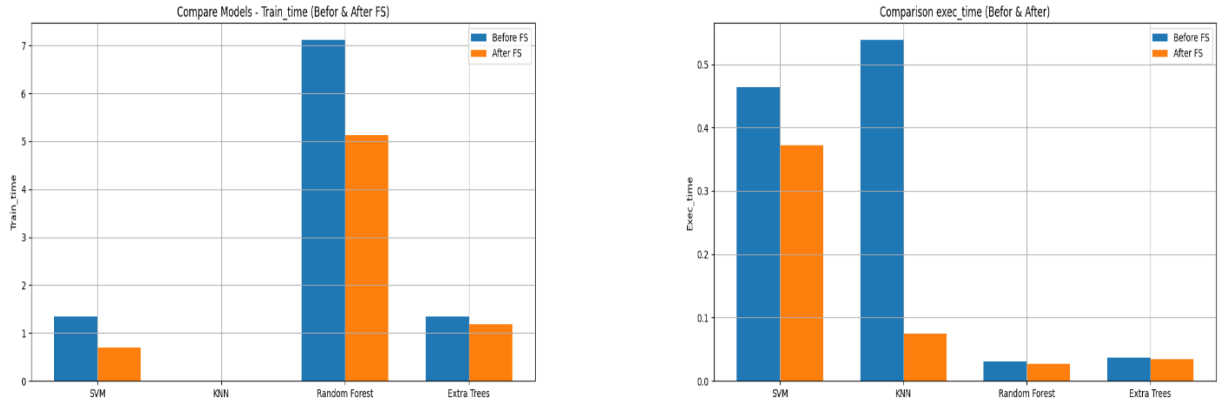
أظهرت النتائج كما هو موضح في الشكل (٣) أن استخدام السمات الأكثر أهمية حافظ على مستوى عال من الدقة لجميع النماذج، تعكس هذه النتائج فعالية اختيار السمات في تحسين كفاءة النماذج دون التأثير على جودة التصنيف، مما يجعلها خطوة مهمة عند التعامل مع مجموعات بيانات كبيرة أو تطبيقات زمنية حساسة.



الشكل(3) دقة النماذج على مجموعة البيانات بعد تخفيض السمات

نلاحظ من الشكل (4) أن تطبيق تقنية اختيار السمات (Feature Selection) باستخدام خوارزمية Extra Trees أدى إلى تحسينات ملحوظة في كفاءة النماذج من حيث زمن التدريب وزمن التنفيذ. ففيما يتعلق

بزمن التدريب، لوحظ انخفاض واضح في النماذج التي تعتمد على العمليات الحسابية المكثفة مثل SVM وRandom Forest، حيث تقلص زمن تدريب SVM من ١.٣٤٦٤ ثانية إلى ٠.٦٩٦٢ ثانية، بينما انخفض زمن تدريب Random Forestمن ٧.١١٧٨ ثانية إلى ٥.١٢٦٩ ثانية. كذلك شهدت نماذج KNN وExtra Trees انخفاضاً في زمن التدريب، وإن كان بدرجات متفاوتة نظراً لاختلاف طبيعة العمليات الحسابية لكل نموذج. أما بالنسبة لزمن التنفيذ، فقد لوحظ تحسن كبير خصوصاً في نموذجKNN ، الذي انخفضزمن تنفيذه من ٠.٥٣٨٧ ثانية إلى ٠.٠٧٤٤ ثانية، مما يعكسكفاءة أكبر في التنبؤ بعد تقليل عدد السمات. النموذجان SVM وExtra Trees أظهرا تحسناً متوسطاً في زمن التنفيذ، بينما حافظ Random Forest على زمن تنفيذه تقريباً مع انخفاض طفيف. توضح هذه النتائج أن تقليص عدد السمات لا يؤثر على الدقة ويعزز الكفاءة الزمنية للنماذج، خاصة عند التعامل مع مجموعات بيانات كبيرة أو تطبيقات تتطلب استجابة سريعة، مما يجعل استخدام تقنيات اختيار السمات خطوة فعالة لتحسين الأداء الحسابي للنماذج التقليدية في كشف البرمجيات الخبيثة.



الشكل (4) زمن التدريب و زمن التنفيذ قبل و بعد تخفيض السمات

توضح النتائج الجدولية الموضحة في الجدول (2) مقارنة أداء النماذج قبل وبعد تخفيضالسمات. نلاحظ أن الدقة بقيت ثابتة تقريباً لكل النماذج بعد استخدام السمات الأكثر أهمية، مما يشير إلى أن تخفيضالسمات لم يؤثر سلباً على جودة التصنيف.

أما بالنسبة للزمن، فقد أظهر نموذج KNN وExtra Trees تحسناً واضحاً في زمن التنفيذ بعد تخفيضالسمات، حيث انخفضزمن تنفيذ KNN من 0.5387 إلى 0.0744s ، وزمن تنفيذ Extra Trees من 0.0364s إلى 0.0320s .

جدول (٢) : نتائج اختبار المصنفات (قبل وبعد تخفيض السمات)

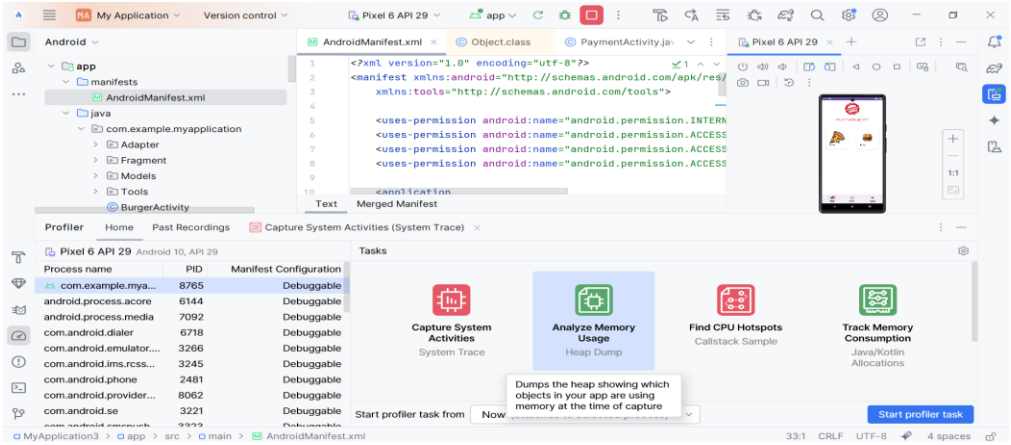
	قبل تخفيض السمات			بعد تخفيض السمات		
النموذج	الدقة	زمن التدريب	زمن التنفيذ	الدقة	زمن التدريب	زمن التنفيذ
SVM	0. 9438	1.3464s	0.4647s	0.9449	0.6962s	0.3719s
KNN	0.9626	0.0050s	0.5387s	0.9637	0.0019s	0.0744s
Random Forest	0.9791	7.1178s	0.0300s	0.9791	5.1269s	0.0270s
Extra Trees	0.9796	1.3418s	0.0364s	0.9802	1.1872s	0.0320s

٦-٣- السيناريو الثالث: تعزيز دقة الكشف بدمج السمات السلوكية المستخرجة من الذاكرة مع مجموعة البيانات CIC-AndMal2020

قمنا بتعزيز مجموعة البيانات الأصلية CIC-AndMal2020، التي تعتمد على التحليل الديناميكي، من خلال دمجها بسمات سلوكية مستخرجة من ملفات الذاكرة(HPROF) ، باستخدام الأداة Android Studio Profiler وMemory Analyzer Tool (MAT) بهدف تقييم فعالية هذه الأدوات في استخراج السمات السلوكية المتعلقة بالذاكرة. يهدف هذا التعزيز إلى تعميق التحليل الديناميكي عبر تتبع حالة الذاكرة أثناء تنفيذ التطبيقات، مما يوفر مؤشرات أكثر دقة على السلوك الخبيث شملت هذه التجربة تنفيذ 10تطبيقات Androidفي بيئة Android Studio خاضعة للمراقبة، منها 7تطبيقات حميدة و 3تطبيقات خبيثة موزعة على ثلاث فئات:

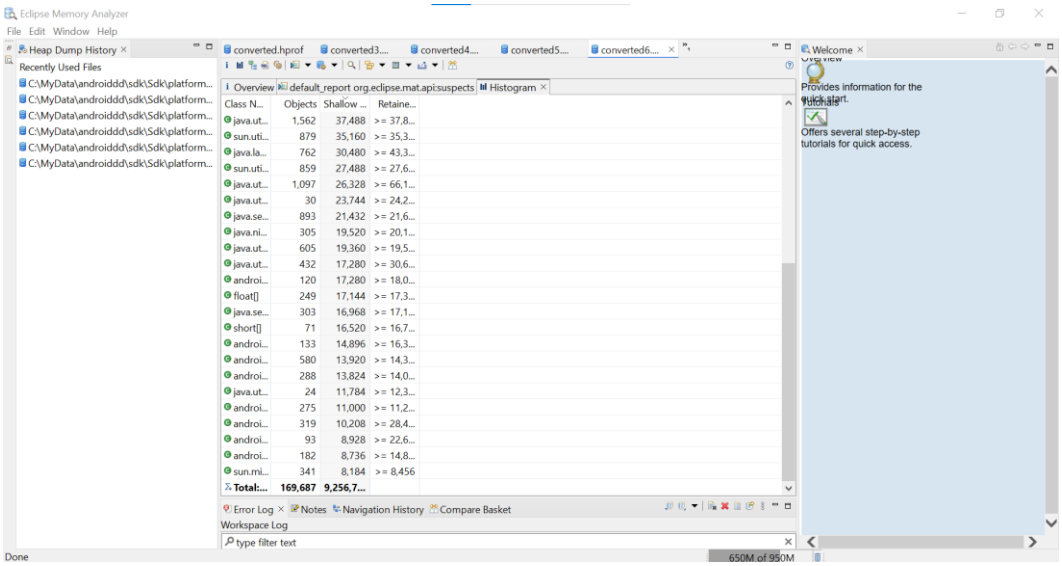
- تطبيق من نوع Trojan
- تطبيق من نوع Ransomware
- تطبيق من نوع Spyware

يبين الشكل (5) عمل أداة Android Studio Profiler لقياس السمات السلوكية الخاصة بالذاكرة أثناء تنفيذ التطبيق .



الشكل (5) قياس السمات السلوكية للتطبيق باستخدامAndroid Studio Profiler

ومن ثم تحليلها باستخدام MAT لاستخراج سمات مثل:عدد الكائنات(Objects) ، وكمية الذاكرة المحجوزة مؤقتاً(Shallow Heap) ، وكمية الذاكرة المحتجزة فعلياً (Retained Heap)لكل نوع من أنواع الكائنات كما هو مبين في الشكل (6).



الشكل (6) السمات المستخرجة للتطبيق باستخدامMAT

أظهرت النتائج المبدئية اختلافات ملحوظة في بعض السمات بين التطبيقات الحميدة والخبيثة. فعلى سبيل المثال، سجلت التطبيقات الخبيثة ارتفاعاً في عدد الكائنات من نوع byte[] و java.lang.String، واحتفاظاً أكبر بالذاكرة في كائنات مرتبطة بالتخزين أو التشفير، مما قد يعكس سلوكيات مثل تسريب البيانات أو تحميل ملفات مشبوهة. تشير هذه المؤشرات إلى إمكانية الاعتماد على تحليل الذاكرة كجزء مهم من عملية الكشف المبكر عن التطبيقات الخبيثة.

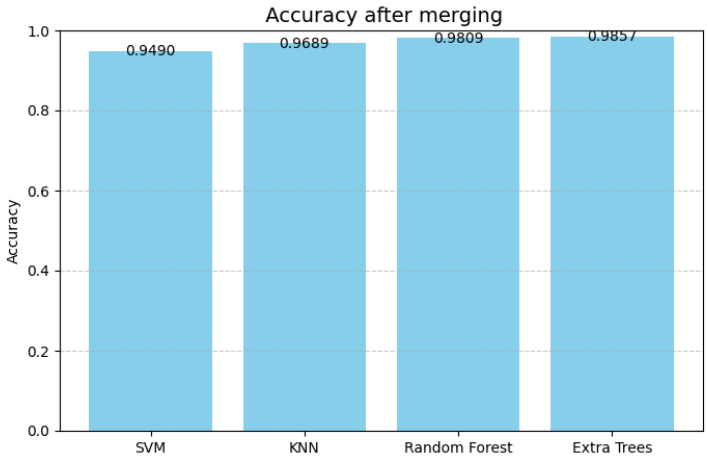
بعد استخراج السمات السلوكية المتعلقة بالذاكرة من ملفات HPROF باستخدام أداةMemory Analyzer Tool (MAT)، تم دمج هذه السمات مع مجموعة البيانات الأصلية CIC-AndMal2020 بناء على معرف التطبيق (App_Hash)لضمان التطابق الدقيق بين الصفوف.

قبل الدمج، خضعت السمات المستخرجة من الذاكرة إلى معالجة مسبقة شملت التطبيع باستخدام Min-Max Scalerلتوحيد القيم العددية، كما تم اعتماد خوارزمية Extra Trees Classifier لاختيار السمات الأكثر تأثيراً وتخفيض الأبعاد. لضمان التوازن في عدد العينات بين الفئات، تم تطبيق تقنية SMOTEلموازنة البيانات، بالإضافة إلى دمج جميع فئات Trojan في فئة موحدة لتحسين التركيز على السلوكيات العامة بدلاً من التفاصيل الدقيقة لكل عائلة خبيثة.

تم تقسيم البيانات إلى مجموعتي تدريب واختبار بنسبة ٨٠/٢٠ باستخدام train_test_split مع تثبيت العشوائية لزيادة القابلية لإعادة التجربة. بعد ذلك، تم تدريب النماذج وتقييم الأداء وقد جاءت النتائج النهائية لتؤكد فعالية دمج سمات الذاكرة كما هو موضح في الجدول (3).

جدول (٣) : نتائج اختبار المصنفات (قبل وبعد دمج السمات)

	قبل دمج السمات			بعد دمج السمات		
النموذج	الدقة	زمن التدريب	زمن التنفيذ	الدقة	زمن التدريب	زمن التنفيذ
SVM	0.9449	0.6962s	0.3719s	0.9490	0.6755s	0.2598s
KNN	0.9637	0.0019s	0.0744s	0.9689	0.0040s	0.0684s
Random Forest	0.9791	5.1269s	0.0270s	0.9809	5.1288s	0.0170s
Extra Trees	0.9802	1.1872s	0.0320s	0.9857	1.1850s	0.0120s



الشكل (7) دقة النماذج بعد دمج السمات

أظهرت النتائج أن دمج السمات السلوكية المتعلقة بالذاكرة مع مجموعة البيانات الأصلية CIC-AndMal2020 ساهم بشكل فعال في تحسين أداء نماذج التصنيف. فقد تمكنت هذه السمات من توفير مؤشرات إضافية على السلوكيات الخبيثة المرتبطة بالاستهلاك غير الطبيعي للذاكرة أو عمليات التشفير والتخزين، وهو ما انعكس في زيادة دقة النماذج وانخفاض زمن التنفيذ في بعض الحالات. برز نموذج Extra Trees كأفضل أداء، محققاً دقة بلغت ٩٨.٥٧%، يليه نموذج Random Forest بدقة ٩٨.٠٩%، وهو ما يؤكد فعالية الاعتماد على تحليل الذاكرة كنهج تكميلي للتحليل الديناميكي التقليدي. هذه النتائج تعزز أهمية دمج مصادر متعددة للسمات السلوكية من أجل رفع كفاءة أنظمة كشف البرمجيات الخبيثة وتحقيق استجابة أكثر موثوقية وفاعلية في البيئات العملية.

الاستنتاجات

أظهرت النتائج عند تقييم المصنفات باستخدام مجموعة البيانات الأصلية CIC-AndMal2020 أن:

- النماذج المبنية على الأشجار مثل Extra Trees و Random Forest حققت أعلى أداء، حيث سجل نموذج Extra Trees دقة بلغت ٩٨.٠٢% بزمن تدريب ١.٣٤ ثانية وزمن تنفيذ ٠.٠٣٦ ثانية، بينما حقق نموذج Random Forest دقة ٩٧.٩١% بزمن تدريب أطول نسبياً بلغ ٧.١١ ثانية. في المقابل، جاءت دقة كل من SVM و KNN أقل نسبياً.

أما عند تطبيق اختزال السمات باستخدام خوارزمية Extra Trees، فقد أثّر ذلك إيجابياً على كفاءة النماذج من خلال:

- تقليل عدد السمات مع الحفاظ على مستوى عالٍ من الدقة، بل وتحقيق تحسن ملحوظ في زمن التنفيذ خاصة لدى نموذج KNN (من ٠.٥٣ ثانية إلى ٠.٠٧ ثانية) ونموذج Extra Trees (من ٠.٠٣٦ ثانية إلى ٠.٠٣٢ ثانية)، مما يبرز أهمية تقنيات اختيار السمات في تسريع النماذج دون التضحية بالدقة.

وأخيراً، عند دمج السمات السلوكية المستخرجة من الذاكرة (HPROF) مع مجموعة البيانات الأصلية، سجل تحسن إضافي في الأداء، حيث:

- حقق نموذج Extra Trees أعلى دقة بلغت ٩٨.٥٧% بزمن تنفيذ ٠.٠١٢ ثانية، يليه نموذج Random Forest بدقة ٩٨.٠٩%.
- كما أظهرت التحليلات أن التطبيقات الخبيثة تترك أنماطاً مميزة في الذاكرة مثل ارتفاع عدد الكائنات من نوع byte[] وjava.lang.String، وهو ما يدعم اعتماد تحليل الذاكرة كمنهج تكميلي يعزز دقة وكفاءة نظم الكشف.

التوصيات والأعمال المستقبلية

بناء على النتائج ، توصي هذه الدراسة بما يلي:

١. اعتماد النماذج المبنية على الأشجار Extra Trees وRandom Forest في أنظمة كشف البرمجيات الخبيثة، نظراً لتوازنها بين الدقة والكفاءة الزمنية.

٢. تطبيق تقنيات اختيار السمات لتقليل الأبعاد وتحسين سرعة التنفيذ، خصوصاً في البيئات التي تتطلب استجابة فورية.

٣. توسيع نطاق السمات المستخرجة من الذاكرة ودمجها مع بيانات التحليل الديناميكي، بما يعزز قدرة النماذج على الكشف المبكر عن الهجمات الخبيثة.

٤. استثمار مؤشرات الذاكرة كعامل إضافي في أنظمة كشف هجينة تجمع بين التحليل الستاتيكي والديناميكي، لتحقيق دقة أعلى وكشف أكثر ذكاء للبرمجيات الخبيثة المتخفية.

أما على صعيد الأعمال المستقبلية، فيقترح اختبار الإطار المقترح على مجموعات بيانات أحدث وأكثر تنوعاً، بما يسمح بتقييم قدرته على التعميم ومواكبة التهديدات السيبرانية المتطورة. كما يمكن استكشاف دمج التحليل الستاتيكي مع التحليل الديناميكي في نظام كشف موحد متعدد الطبقات، للاستفادة من مزايا كل نوع من التحليل. بالإضافة إلى ذلك، يعد تطوير آليات تكيفية قادرة على تحديث السمات والنماذج بشكل مستمر مع ظهور برمجيات خبيثة جديدة، من أهم التوجهات التي ينبغي العمل عليها مستقبلاً. وأخيراً، يوصى بإجراء اختبارات في بيئات تشغيل واقعية (Real-world Environments) للتأكد من جاهزية النماذج المقترحة للتطبيق العملي في أنظمة الحماية التجارية.

المراجع

[1] Alzubaidi, L., et al. *Android Malware Detection Using Machine Learning Techniques: A Review*. Computers & Security, 2020.

[2] Hossain, M. S., et al. *Dynamic Analysis for Android Malware Detection: State-of-the-Art*. Journal of Information Security and Applications, 2021.

[3] Li, Y., et al. *Memory-based Malware Detection in Android Devices*. IEEE Transactions on Mobile Computing, *2022*.

[4] Android Malware Dataset : *CIC-AndMal2020 Dataset Accessed:* Aug.18,2025 .Available: https://www.unb.ca/cic/datasets/andmal2020.html .

[5] Zhang, J., et al. *Dynamic Memory Analysis for Malware Detection on Mobile Platform*s. Future Generation Computer Systems, *2023*.

[6] Kumar, R., et al. *Behavioral Feature Extraction for Android Malware Classificatio*n. Applied Soft Computing, 2024.

[7] Sandeep Sharma , Prachi , Rita Chhikara , Kavita Khanna , *Dynamic Analysis Based Android Malware Detection Using ANN and RFE Feature Selection*. 2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N).

[8] Saneeha Khalid, Faisal Bashir Hussain, *Evaluating Dynamic Analysis Features for Android Malware Categorization*, 2022 International Wireless Communications and Mobile Computing (IWCMC)

[9] Nahier Aldhafferi , *Android Malware Detection Using Support Vector Regression for Dynamic Feature Analysis*. Information, 15(10), 658. 19 October 2024.

[10] N. Alangari, M. El Bachir Menai, H. Mathkour, and I. Almosallam**,** *Exploring Evaluation Methods for Interpretable Machine Learning: A Survey*, Information, vol. 14, no. 8, Art. 469, 2023.

[1١] P. Papadopoulos, D. Koukis, and G. Kambourakis, *Android malware detection with unbiased confidence guarantees*, *arXiv preprint arXiv:2312.11559*, Dec. 2023.

[1٢] M. Muzaffar, M. N. Aman, and B. Sikdar, *DroidDissector: A static and dynamic analysis tool for Android malware detection*, *arXiv preprint arXiv:2308.04170*, Aug. 2023.

[1٣] J. Maestre Vidal, C. Chothia, and R. J. Anderson, *A novel pattern recognition system for detecting Android malware by analyzing suspicious boot sequences*, Feb. 2024.

[١٤] D. Powers, *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation*, Journal of Machine Learning Technologies, vol. 2, no. 1, pp. 37–63, 2011.