تصنيفُ التسلّلات إلى الشّبكات المحليّة باستخدام الغابات العشوائية و Apache Spark

الدُكتورة زينب راتب خلوف*

الملخّص

مع الاعتماد المُتزايد على النُظم المعلوماتيّة و الشبكات الحاسوبيّة جذبَ موضوعُ حماية البيانات و النُظم اهتماماً واسعاً ، وخاصّة مع تطوّر قُدرات المهاجمين وزيادة حالات الاختراق. لكن من المعروف أنّ بناء نظام قادر على تجاوز جميع الهجمات غير مُمكن في وقتنا الحالي ومن هنا تأتي أهميّة نُظُم كشف التسلّلات بأنواعها. يستندُ قسمٌ من نظم كشف التسلّل إلى تعلّم الآلة الخاضع للإشراف بحيثُ يُدرَّب النظام على بيانات مصنفّة في فئات وبناءً على مرحلة التعلّم هذه يصنف سجلات الاتصالات القادمة لتمييز التسللات إلى الشبكة.

نحقّق في هذا البحث عدّة نماذج تصنيف لاكتشاف التسللّات إلى الشبكات المحليّة باستخدام خوارزمية الغابات العشوائية والمُضمنّة في مكتبة تعلّم الآلة من أباتشي سبارك ، كما نقيّم هذه النماذج بناءً على مجموعة معايير لتقييم الأداء.

الكلمات المفتاحيّة: أمن النظم و الشّبكات الحاسوبية ، نُظم كشف التسللّات ، تعلّم الآلة ، الغابات العشوائية ، Apache Spark ، لغة سكالا ، مجموعة البيانات NSL-KDD.

* أستاذ مساعد-قسم هندسة الشبكات و النظم الحاسوبية- كلية الهندسة المعلوماتية-جامعة البعث

Classification of Intrusions in Local Area Networks (LANs) using Random Forests and Apache Spark

Dr. Zainab Khallouf*

Data and systems security has attracted increasing attention as computer systems and networks influence every aspect of modern life.

It is known that an "Attack Proof" system doesn't exist, but it is possible to detect suspicious network activities using intrusions detection systems (IDSs). IDSs that utilize supervised machine learning techniques such as classification to learn from existing data can classify new connections to detect abnormal behavior.

In this paper, we implement and evaluate several models to detect intrusions in local area networks (LANs) using the random forest algorithm provided by the machine learning library in Apache Spark. In order to evaluate the performance of the proposed framework, several performance matrices are used.

Keywords: Computer systems and networks security, intrusions detection systems, machine learning, random forest, Apache Spark, Scala, NSL-KDD data set.

* Associate Professor, department of systems and computer networks engineering, faculty of informatics engineering, Al-Baath University, Homs, Syria.

1. مقدمة والهدف من البحث:

يُعرّف كشفُ التسلّلات بأنّه عملية مُراقبة و تحليل الأحداث (Events) في الشبكة أو ضمن النظام المعلوماتي بهدف كشف أي إشارة لحدوث اختراق للنظام. تأتي هذه التسلّلات إمّا من مهاجمين ، أو من مستخدمين يحاولون الحصول على سماحيات أعلى ، أو يسيئون استخدام السماحيّات الممنوحة لهم [19].

تقسّم نُظم كشف التسلّلات (IDSs) لقسمين رئيسين [1]:

أولاً : (Anomaly Detection Systems) نُظم كشف الشذوذ: يعتمدُ هذا النموذج على تعريف سلوك طبيعي للنظام المعلوماتي بطرق إحصائية ، و من ثمّ اعتبار أي انحراف عن هذا السلوك كدليل على وجود اختراق.

ثانياً : (Signature Based Detection Systems) نُظم معتمدة على مجموعة من الأنماط المسبقة التعريف: يعتمدُ هذا النموذج على مجموعة من الأنماط المُسبقة التعريف للهجمات الممكنة على النظام ، بالإضافة لمجموعة من القواعد الواجب تطبيقها عند اكتشاف أحد هذه الأنماط في الرزم الملتقطة.

يُمكن استخدام التصنيف في نُظم كشف التسلّلات من النوع الأول (نظم كشف الشذوذ) التي تتعلم من مجموعة بيانات (Dataset) تتضمن اتصالات طبيعية "Normal" و اتصالات تابعة لمتسللين "Attack" ، وبناءً على مرحلة التعلم هذه يتمّ تصنيف سجلات الاتصالات القادمة لتمييز التسللات إلى الشبكة [1].

نحقّق في هذا البحث عدة نماذج تصنيف لاكتشاف التسللاّت إلى الشبكات المحليّة باستخدام الغابات العشوائية و مكتبة تعلّم الآلة من أباتشي سبارك (Apache Spark) وهو نظام مفتوح المصدر لمعالجة موزعّة للبيانات الكبيرة. يتميزُ سبارك بكونه متسامح مع الأخطاء ، وذو أداء مرتفع نظراً لإمكانية معالجة البيانات في الذاكرة ، بالإضافة لقابلية التوسع [2]. نقيّم النموذج المحقّق اعتماداً على عدة معايير لتقييم أداء نماذج تعلّم الآلة.

يقدّم البحث في قسمه الأول لمحة نظريّة عن التقنيات المستخدمة ويبدأ بلمحة عن تقنيات تعلّم الآلة و خوارزمية أشجار القرار التي تعدّ المُكون الأساس للغابات العشوائية ، مروراً بأباتشي سبارك ، و انتهاءً بمجموعة البيانات المُستخدمة في البحث NSL-KDD. أمّا في قسمه الثاني فيلخّص عدد من الأعمال البحثية المُتعلقة بموضوع البحث ، و في قسمه الثالث

يوضّح النماذج المحقّقة و يعرضُ تقييماً للأداء ، يليه مجموعة من التوصيات و الأعمال المستقبلية في خاتمة البحث.

2. لمحة عن تقنيات تعلم الآلة [3]:

يهدفُ تعلّم الآلة لتعليم الحواسيب كيفية التعرّف على أنماط و استنباط رؤى من البيانات و يمكن استخدام هذا العلم لحل العديد من المسائل في مجالات شتى مثل التعرّف على الصور ، التعرّف على الكلام ، مُعالجة اللغات الطبيعية ، اكتشاف الاحتيال ، النُظم الناصحة ، الروبوتية و غيرها. عند استخدام تقنية لتعلم الآلة يمكن أن نميّز بين المفاهيم الآتيّة:

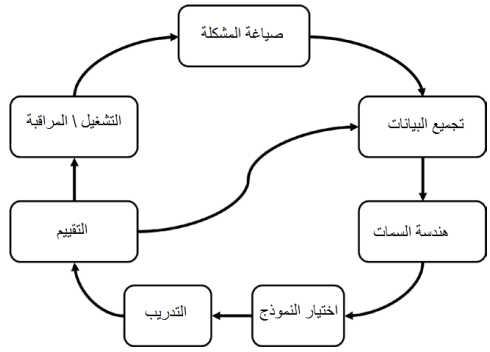
- خوارزميّة تعلّم الآلة: مجموعة خطوات تُنفّذ بشكل تكراري بهدف استنباط أنماط من مجموعة بيانات لتعلم كيفية توليد تصنيف أو تنبؤ من أجل دخل معيّن. يوجد العديد من خوارزميات تعلّم الآلة لكن التحدي يكمنُ في اختيار الخوارزمية الأنسب لحل مسألة معينة.
- الملاحظة (Observation): و تمثّل مثال من الكائن المستخدم للتعلّم.
- التسمية (Label): قيمة تُستخدم لعنونة الملاحظة (على سبيل المثال: اتصال "طبيعي" أو اتصال "هجمة").
- السمات (Features): تمثّل مجموعة الخصائص الهامة للملاحظات و التي لها التأثير الأهم في خرج التنبؤ ؛ مثلاً معدل الإرسال من عنوان IP معين.
- بيانات التدريب (Training data): قسم من البيانات يستخدم لتدريب خوارزمية تعلّم الآلة للحصول على النموذج.
- بيانات التحقق (Validation): قسم من البيانات يستخدم لتقييم أداء نموذج تعلّم الآلة خلال عملية ضبط النموذج.
- بيانات الاختبار (Test data): تُستخدم لتقييم أداء نموذج تعلّم الآلة بعد انتهاء عملية ضبط النموذج.

- النموذج (Model): ينتجُ عن تدريب خوارزمية تعلّم الآلة و يُستخدم للتنبؤ أو لاتخاذ قرارات مُتعلقة ببيانات جديدة. الهدف هو انتاج نموذج مُعمم يُنفّذ جّيداً حتى على بيانات لم يُدرب عليها.

يوجد ثلاثة أنواع رئيسية لخوارزميات تعلّم الآلة:

- التعلّم بالإشراف (Supervised Learning): في هذا النوع من التعلّم يتم تدريب الخوارزمية على بيانات معنونة (Labeled). من أمثلته: التصنيف (Classification) و هو التقنية المستخدمة في هذا البحث و الإنحدار (Regression).
- التعلّم الغير خاضع للإشراف (Unsupervised Learning): يتم تدريب الخوارزمية على بيانات غير معنونة ، صُمّم هذا النوع من التعلّم بهدف إيجاد بنية أو أنماط مخفية في البيانات وأحد أمثلته التجميع (Clustering).
- التعلم المعزز (Reinforcement Learning): في هذا النوع من التعلّم لاتتعلم الخوارزمية من البيانات و إنما من التفاعل مع البيئة من خلال سلسلة من الأفعال (Actions) ومن خلال معلومات التغذية الراجعة تقوم بتعديلات (Adjustments) بهدف تعظيم قيمة ربح ما.

تتكون عملية تعلّم الآلة من الخطوات الآتية الموضّحة في الشكل 1:



الشكل 1: خطوات تعلّم الآلة (مترجم عن [17])

1) فهم و صياغة المسألة التي ننتظر من تقنية تعلّم الآلة حلها و من الجيد في هذه المرحلة دراسة امكانية استخدام حلول بديلة لتقييم الكلفة بالإضافة للميزات و للسلبيات. عند بناء ثقة قوية بأن تعلّم الآلة هو الخيار الأفضل للوصول للهدف عندئذ ننتقل إلى الخطوة التاليّة.
2) جمع البيانات: حيث أنّ نوعيّة و كميّة البيانات المجمعّة سيكون لها تأثير مباشر على أداء النموذج المدرب و من المهم أن تكون البيانات ممثّلة قدر الإمكان للمُشكلة التي نحاول حلها.
3) هندسة السمات (Feature engineering): وهي واحدة من أكثر المراحل أهميّة و تطلباً للوقت. في هذه المرحلة تتمّ تنقية البيانات (Data cleaning) و الاستنارة بمعرفة المجال (Domain knowledge) لتحديد خصائص أساسيّة أو سمات مهمّة لتعلّم الآلة. كذلك يمكن أن يتمّ تحويل قيّم السمات المحرفية لأدلة (String Indexing) وأيضاً ترميز قيّم السمات الفئوية العددية باستخدام مرمزات مثل ترميز واحد حار (One hot encoding) لإلغاء أي علاقة ترتيب قد تعتبرها خوارزمية تعلّم الآلة موجودة بينها و في هذا البحث سنستخدم هذه التقنيات.
4) اختيار خوارزمية تعلم الآلة وتدريبها مما يتطلب الفهم الجيد للمشكلة المطلوب حلها ، مع معرفة عمليّة بالخصائص المختلفة لكل خوارزمية وامتلاك خبرة تطبيق هذه الخوارزميات على سيناريوهات مماثلة في السابق.
5) تقييم النموذج ولاتهدف هذه المرحلة إلى تحديد أداء النموذج فحسب و إنما أيضاً لتحديد متى يتوقف ضبط النموذج (Model tuning) عندما يصل أداؤه إلى جودة مقيّمه باستخدام معايير محددّة. يُمكن أن تؤدي هذه المرحلة إمّا لنموذج قابل للاستخدام أو للعودة إلى تجميع البيانات و المرور على المراحل من جديد عندما يكون أداء النموذج المولّد غير كاف.

3. أهم المقاييس لتقييم أداء نماذج التصنيف [17]:

يوجدُ عدد من مقاييس تقييم أداء نماذج التصنيف التي تهدفُ لضمان تقييم "غير متحيز" لأداء النموذج و لعل أكثرها شهرة المقاييس الآتية:

- مصفوفة الارتباك (Confusion matrix) المُوضحة في الجدول 1 ، و تعدّ أساساً لحساب قيمة عدد من المقاييس المعبّرة عن أداء نموذج التصنيف.

		التنبؤ (Predicted)	
		طبيعي (Normal)	هجمة (Anomalous)
الواقع (Actual)	طبيعي (Normal)	TN	FP
	هجمة (Anomalous)	FN	TP

الجدول 1: مصفوفة الارتباك

في الجدول 1 نميّز بين المفاهيم التاليّة:

- إيجابية حقيقيّة ((True Positive (TP)): عدد الملاحظات التي تمثّل هجمة و التي صُنفت بشكل صحيح على أنها هجمة.
- سلبية حقيقيّة ((True Negative (TN)): و تمثّل عدد الملاحظات الطبيعية (أي التي لاتمثل هجمة) و التي صُنفت بشكل صحيح على أنها لاتمثّل هجمة.
- إيجابية خاطئة ((False Positive (FP)): عدد المُلاحظات الطبيعية لكنها صنفت بشكل خاطىء على أنها هجمة (يسمى أيضاً خطأ من النوع الأول).
- سلبية خاطئة ((False Negative (FN)): و تمثّل عدد الملاحظات التي تمثّل هجمة وصنفت بشكل خاطئ على أنها طبيعية (يُسمى أيضاً خطأ من النوع الثاني).

- الصحّة (Accuracy)؛ النسبة بين عدد المُلاحظات التي تمّ تصنيفها بشكل صحيح و العدد الكلّي للملاحظات أي:

$$Accuracy = \left(\frac{TP + TN}{TP + TN + FP + FN}\right)$$

تعدّ الصحة مقياساً جيداً في الفئات المتوازنة (أي أنّ أعداد الملاحظات في كل فئة متقاربة) وعندما تكون كلفة الخطأ الأول و الثاني متقاربة و إلا يمكن أن لايكون هذا المقياس معبراً ، فلو فرضاً تضمنت مجموعة بيانات 1000 ملاحظة من فئة 1 و 10 ملاحظات من فئة 2 عندئذ صحة مصنف يصنّف جميع الملاحظات على أنها من الفئة 1 تعادل:

accuracy = (1000 + 0)/1010 = 99%

من الواضح أنّ هذه القيمة لاتعكس أداء جيداً للمصنف في هذه الحالة وخصوصاً في الحالات الحرجة مثل اكتشاف وجود تسلل.

- الإحكام (Precision)؛ النسبة بين عدد التنبؤات الإيجابية الصحيحة ومجموع عدد التنبؤات الإيجابية الخاطئة و الإيجابية الصحيحة؛ أي مثلاً: من بين كل الاتصالات التي تنبأت بها الخوارزمية بأنها تمثل هجمة ، كم منها تمثّل هجمة فعليًا؟ كُلما كان عدد التنبؤات الإيجابية الخاطئة أقل كلّما كانت قيمة الإحكام مرتفعة.

$$Precisin = \frac{TP}{TP + FP}$$

- (الحساسية/الاستدعاء/معدل التنبؤات الإيجابية الصحيحة)(TPR/Recall/Sensitivity)؛ النسبة بين عدد التنبؤات الإيجابية الصحيحة ومجموع عدد التنبؤات الإيجابية الصحيحة و عدد التنبؤات السلبية الخاطئة. السؤال الذي تُجيب عنه الحساسية هو: من بين كل الاتصالات التي تمثل هجمة فعلياً ، كم عدد الاتصالات التي تنبأت بها الخوارزميّة على أنها هجمة؟

$$Recall = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- (النوعيّة/معدل التنبؤات السلبية الصحيحة)(TNR/Specificity)؛ النسبة بين عدد التنبؤات السلبية الصحيحة ومجموع عدد التنبؤات السلبية الصحيحة و عدد التنبؤات الإيجابية الخاطئة. السؤال الذي تُجيب عنه النوعيّة هو: من بين كل الاتصالات الطبيعية (أي ليست هجمة) ، كم عدد الاتصالات التي تنبأت بها الخوارزميّة على أنها طبيعيّة؟

$$Specificity = \frac{TN}{N} = \frac{TN}{TN + FP}$$

- معدّل التصنيف الإيجابي الخطأ أو معدّل الإنذارات الخاطئة (FPR) ويمثّل النسبة بين عدد الملاحظات الطبيعيّة التي صنّفت بشكل خاطئ ، والعدد الكلي للملاحظات الطبيعية. يُعبّر عنه بالعلاقة:

$$FPR = \frac{FP}{FP + TN}$$

- النتيجة F1 (F1- score)؛ المتوسّط الحسابي الموزون لقيمة الإحكام والحساسية.

$$F1 - score = 2\frac{Recall * Precision}{Recall + Precision}$$

- المنطقة تحت منحني خصائص تشغيل المستقبل (AUC-ROC) ؛ تقيس مدى قدرة النموذج على الفصل بين الفئات و كلما كانت قيمة AUC أكبر كلما كان النموذج أفضل. أما المنحنى ROC فيعبّر عن تغيّر العلاقة بين TPR و FPR.
- المنطقة تحت منحني الاستدعاء و الإحكام (AUC-PR) ؛ بما أنّ كلّ من الإحكام و الاستدعاء لايتضمنان في حسابهما (عدد التنبؤات السلبية الحقيقيّة TN) و لذلك في الحالات التي يكون فيها عدد المُلاحظات السلبيّة أكبر بكثير من الملاحظات الإيجابية (سمة أساسيّة للفئات الغير متوازنة العدد) يتم استخدام هذا المقياس [4].

تقدّم مكتبة تعلم الآلة في سبارك عدد مهم من هذه المقاييس و نميّز بين مقاييس تُستخدم للتصنيف الثنائي (Binary classification) و للتصنيف المتعدد الفئات (Multiclass classification) الذي يعدُّ تعميماً للتصنيف مُتعدد الفئات و يصنّف المُلاحظات بين عدة فئات ممكنة.

4. ضبط نموذج التصنيف (Classification Model Tuning):

يهدفُ ضبط النموذج إلى تدريب النموذج باستخدام المجموعة الصحيحة من الوسائط لتحقيق أفضل أداء في تحقيق الهدف المطلوب و الذي حُدد في الخطوة الأولى من مراحل تعلّم الآلة (الشكل 1). غالباً ماتكون هذه المرحلة تكرارية تتطلب جهداً و وقتاً و ربما تتضمن تدريب عدة خوارزميات تعلّم آلة أو عدة مجموعات من الوسائط. في هذا السياق نميّز بين المصطلحين الآتيين:

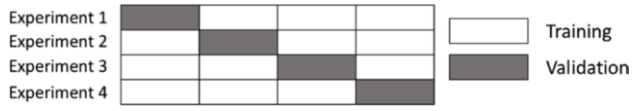
- الوسائط الفائقة للنموذج (Model hyperparameters): و تمثّل الوسائط التي تُحَدّد قبل بدء مرحلة التدريب من قبل محلّل البيانات وتستخدم للتحكم بعملية تدريب خوارزمية تعلّم الآلة. تعدّ هذه الوسائط خارجية عن النموذج و لايمكن تعلمها من بيانات التدريب كما يمكن ضبطها من خلال عملية تكرارية. كمثال على هذا النوع من الوسائط عدد الأشجار في الغابة العشوائية.
- وسائط النموذج (Model parameters): وتمثّل الوسائط التي لا تُضبط من قبل محلّل البيانات و إنّما يتمّ تعلمها و أمثلتها خلال عملية التدريب ، مثلا السمّة التي ستشكل الجذر في شجرة القرار.

تقدّم مكتبة تعلّم الآلة (ML) في سبارك صفين لضبط النموذج: الأول TrainValidationSplit و الثاني CrossValidator و كلا الصفين يتطلبان العناصر التاليّة كدخل:

- المكون المراد ضبطه: نموذج ، خوارزمية ، أو تسلسل عمليات (Pipeline).
- مجموعة الوسائط التي ستستخدم في عمليّة الضبط ، و تسمى شبكة الوسائط (grid Parameter).

- أحد مقاييس التقييم التي تستخدم لتقييم أداء النموذج اعتماداً على مجموعة بيانات اختبار (مثلا أحد المقاييس التي وضحت سابقاً: الصحة ، النوعية ، الاستدعاء...).

يقسّم الصّف الأول TrainValidationSplit مجموعة البيانات إلى مجموعتين: مجموعة تدريب (Training dataset) ومجموعة للتحقق من الصحّة (Validation Dataset) و بعد ذلك يتمّ تدريب و تقييم مجموعتي البيانات على كل توافق (Combination) ممكن من الوسطاء التي عرّفت في شبكة الوسطاء. على سبيل المثال ، لو تضمنّت شبكة الوسطاء ست توافقيات ممكنة من الوسطاء عندئذ يتم تدريب وتقييم النموذج ست مرات من أجل كل مجموعة ممكنة.
أمّا الصف الثاني CrossValidator فهو تحقيق لتقنية التحقق المتقاطع. في هذه الطريقة يتمّ تقسيم الملاحظات إلى k مجموعة أو مايسمى طيّة (Fold) لها تقريباً نفس الحجم ؛ الطيّة الأولى تستخدم لعملية التحقق من الصحة و بقيّة الطيات للتدريب و يتم تدريب النموذج k مرّة مع تدوير مجموعات التدريب و التحقق كما يوضح الشكل 2.



الشكل 2: التحقق من الصحة المتقاطع مع k=4 [3]

5. أشجار القرار (Decision Trees) و الغابات العشوائية (Random Forests) :[5]

تعدّ عائلة شجرة القرار من خوارزميات تعلّم الآلة ذات الاستخدام الواسع في تطبيقات التصنيف (Classification) كما يُمكن أن تستخدم للانحدار (Regression). يُمكن أن يتم بناء الشجرة أو عدد من الأشجار على التوازي و تقبل الخوارزمية السمات العددية (Numeric) و كذلك الفئوية (Categorical) كما أنها تقبل بيانات من أنماط مختلفة ومن مقاييس مختلفة دون أن يكون هناك حاجة لمعالجة مسبقة لهذه البيانات أو لتقييسها (Normalization). تتميزُ بأنها متينة أمام الشذوذات (Outliers) في البيانات أي أنّ وجود عدد قليل من نقاط البيانات ذات القيم القصوى وعلى الأغلب الغير صحيحة لايؤثر على الخوارزميّة[6].

تعتمدُ الخوارزمية على قيّم السمات (Features) لتقسيم فضاء القرار (Decision space) وبشكل تكراري إلى فضاءات جزئية (Subspaces) أصغر بدءاً من اختيار العقدة الجذر ، ولتنمو تدريجياً إلى التصنيف النهائي الممثّل بالأوراق.

من حيث المبدأ يوجد عدّة طرق لإنشاء الشجرة وجميعها تؤدي إلى نفس النتائج النهائية للتصنيف ، لكن يُمكن مع ازدياد عدد الخصائص و مجال القيم العددية أو عدد القيم الفئوية أن تصبح غايّة في التعقيد و أن لا تتنبأ الأشجار بنفس النتيجة في هذه الحالة ، وخصوصاً عندما تتضمن قيم بعض الخصائص ضجيج أو قيم مفقودة [7].

لاختيار السمة التي سيتم تقسيم المُلاحظات وفقها عند كل عقدة بدءاً من العقدة الجذر يوجد معايير مثل معيار ربح المعلومات (Information Gain) المُعتمد في الخوارزمية الأساسية لبناء أشجار القرار: ID3. يتم تقسيم البيانات في خوارزمية ID3 حسب السمات التي تجعل قيمة ربح المعلومات أعظمية ، مع تكرار هذه العملية عند كل عقدة حتى نصل لأوراق نقيّة (Pure) تكون الملاحظات عندها من نفس الفئة. يُمكن أن تؤدي هذه العملية التكرارية إلى شجرة ذات عدد كبير من العقد و بالتالي إلى نموذج ذو تكيّف زائد (Overfitting) مع بيانات التدريب ؛ أي أنّ النموذج يعكس مجموعة بيانات التدريب لكن لايمكنه التنبؤ بفئة بيانات لم يدرّب عليها و لذلك نقوم بعملية تقليم (Prune) من خلال وضع حد لعمق الشجرة.

يُعرّف ربح المعلومات بالعلاقة:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

حيث أنّ: f تمثّل السمة التي سيتم التقسيم عندها، D_p و D_j تمثّلان مجموعتي البيانات للأب و للعقدة البنت j ، I مقياس عدم النقاوة أو عدم تجانس البيانات (Impurity) ، N_p العدد الكلي للعينات عند عقدة الأب و N_j عدد العينات عند العقدة البنت j. أي أن ربح المعلومات يمثّل الفرق بين عدم النقاوة عند العقدة الأب ومجموع عدم النقاوة عند العقد الابناء ، و كلما كان عدم النقاوة عند الابناء أقل كلما كانت المعلومات المكتسبة أكبر. لكن للتبسيط و لتقليل مجال البحث فإن معظم المكتبات تحقّق أشجار قرار ثنائية أي أنّ كل عُقدة أب تقسم إلى عقدتين أبناء D_{left} و D_{right}.

يوجد ثلاثة معايير شائعة لعدم النقاوة:

1. الانتروبي ((I_H) Entropy) و تُعطى بالعلاقة الآتية من أجل جميع الفئات الغير خالية:

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) log_2 p(i|t), p(i|t) \neq 0$$

حيث أن $p(i|t)$ تمثّل نسبة العينات المنتمية للفئة c من أجل عقدة محددة t، وبالتالي الانتروبي تساوي صفر إن كانت جميع العينات عند عقدة منتمية لنفس الفئة ، في حين تأخذ قيمة أعظمية عندما يكون توزع العينات منتظم (Uniform class distribution).

2. معامل جيني ((I_G) Gini impurity) و يُعطى بالعلاقة:

$$I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t)) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

يُنظر لمعامل جيني كمعيار لتقليل احتمال خطأ التصنيف ، كما تكون قيمة معامل جيني أعظمية عندما تكون العينات من فئات مختلفة وفقاً لتوزيع منتظم. مثلاً في حالة فئتين (c = 2):

$$I_G(t) = 1 - \sum_{i=1}^{c} 0.5^2 = 0.5$$

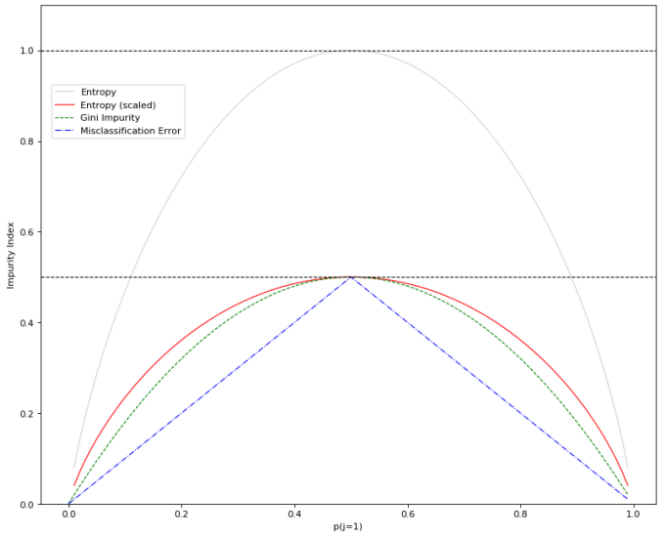
عملياً كل من المعيارين جيني و الانتروبي يعطيان نتائج متشابهة.

3. خطأ التصنيف ((I_E) Error classification):

$$I_E = 1 - max\{p(i|t)\}$$

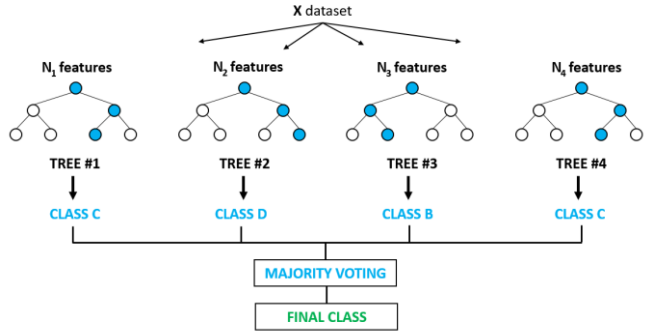
ويُنصح بهذا المعيار في حالة تقليم الشجرة لكن ليس لبناء الشجرة لأنه أقل حساسيّة لتغير احتمالات الفئات.

يوضّح الشكل 3 هذه المعايير من أجل مجال احتمال $[0, 1]$ للفئة $c = 1$:



الشكل 3: المعايير الشائعة لعدم النقاوة [5]

تعدّ الغابات العشوائية تقنية لتعلّم الآلة بالإشراف (Supervised Learning) ، من نوع تجميع (Ensemble) لمجموعة من أشجار القرار بحيث تعطي النتيجة من خلال تصويت بالأغلبية (Majority vote) على نتائج الأشجار المكونة للغابة كما يوضّح الشكل 4. يمُكن أن تُستخدم للتصنيف (Classification) أو للانحدار (Regression) و تهدفُ لتشكيل نماذج ذات أداء أفضل و أقل عُرضة للتكيّف الزائد من شجرة القرار.



الشكل 4: التصويت في الغابة العشوائية [18]

يُمكن تلخيص خوارزمية الغابات العشوائية بالخطوات الأربعة التاليّة:

1. اختيار مجموعة تمهيدية (Bootstrap) عشوائية من مجموعة التدريب (Training set) مكونّة من n ملاحظة بحيث يتم الاختيار مع استبدال (With replacement).
2. بناء شجرة قرار من المجموعة التمهيدية ، وعند كل عقدة:
   a) اختيار d سمة بشكل عشوائي بدون استبدال.
   b) تقسيم المُلاحظات عند العقدة باستخدام السمة التي تعطي أفضل قيمة لربح المعلومات (أو لتابع هدف مختلف Objective function).
3. تكرار الخطوتين الأولى و الثانيّة k مرة.
4. تجميع نتائج التنبؤ من كل شجرة لتحديد فئة من خلال تصويت أغلبية.

تكمنُ الميّزة الأساسيّة للغابات العشوائية في أنها لاتتطلب جهود كبيرة لاختيار الوسائط الفائقة (Hyperparameters) والتي تُحدّد قبل مرحلة التعلّم ، كما لاتتطلب تقليم لأن النماذج المعتمدة على تجميع نماذج بسيطة تكون مُقاومة لضجيج النماذج الجزئية. عادة الوسيط الوحيد الذي يفترض تحديده هو عدد الأشجار k (الخطوة 3) ، وكلما ازداد عدد الأشجار تحسّن أداء الغابة على حساب زيادة الكلفة الحسابية.

من الوسائط الفائقة التي يمكن تحديدها أيضاً عدد العناصر n في المجموعة التمهيدية (الخطوة الأولى) ، وكذلك عدد السمات d التي يتم اختيارها بشكل عشوائي عند كل تقسيم (الخطوتين 1، 2). يمكن من خلال حجم مجموعة البيانات التمهيدية التحكم بالموازنة بين الانحياز و التشتت (Bias-variance tradeoff) في الغابات العشوائية ، لأنّ إنقاص حجم مجموعة البيانات التمهيدية يزيد التنوع (Diversity) بين أشجار القرار المكونة للغابة نظراً لأنّ احتمال أن تكون مُلاحظة معينة مضمنّة في المجموعة التمهيدية أقل ، وبالتالي تقليص حجم المجموعة الابتدائية يمكن أن يحسّن العشوائية في الغابة العشوائية و يقلّل تأثير التكيف الزائد لكنه يعطي أداء كلي أقل ، كما أنّ زيادة حجم العينة التمهيدية يُمكن أن يزيد التكيف الزائد نظراً لأن الاشجار الجزئية تصبح أكثر تشابهاً و تتعلم من بيانات التدريب الأصلية بشكل مشابه. في معظم التحقيقات يتم اختيار عدد العناصر في المجموعة التمهيدية مساوياً لعدد الملاحظات في مجموعة التدريب الأصليّة ، أمّا بالنسبة لعدد السمات d عند كل تقسيم يتم اختيار عدد أقل من عدد السمات

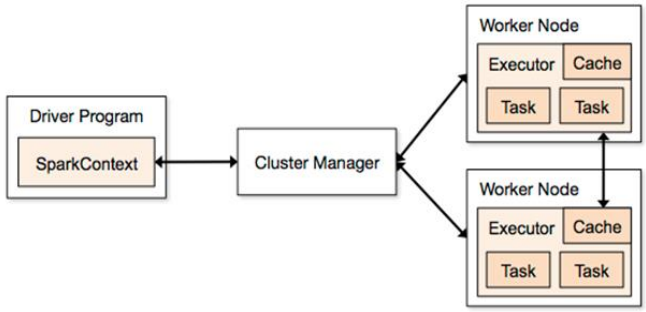
الكليّة في مجموعة بيانات التدريب و عادة بشكل افتراضي يتم اختيار $d = \sqrt[2]{m}$، حيث أن m تمثل عدد السمات في مجموعة التدريب.

6. أباتشي سبارك (Apache Spark) [3]:

نظام مفتوح المصدر لمعالجة موزعّة للبيانات الكبيرة يتميزُ بكونه متسامح مع الأخطاء ، و ذو أداء مرتفع نظراً لإمكانية معالجة البيانات في الذاكرة. حُقّق سبارك بلغة سكالا لكن يمكن كتابة تطبيقاته بأي من R ، بايثون ، جافا أو سكالا.

يتمّ عادة تشغيل سبارك على مجموعة من الأجهزة والتي تُعرف باسم عنقود سبارك (Spark cluster) ، علماً أن أكبر عنقود تمّ التصريح عنه يضمّ أكثر من 8000 جهاز.

يتولى إدارة مجموعة الأجهزة المكونّة للعنقود بشكل فعال و ذكي نظام لإدارة الموارد مثل Apache YARN أو Apache Mesos مكونٌ عادةً من مدير عنقود (Cluster manager) و عمال (Workers) كما يوضّح الشكل 5 . يعرفُ المدير مكان توضع العامل ، كميّة الذاكرة التي يمتلكها ، وعدد نوى الـ CPU ، كما أنّ واحدة من المهام الأساسيّة للمدير تنظيم العمل من خلال توزيعه على كل عامل.

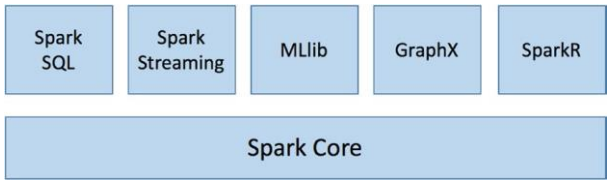


الشكل 5: التفاعل بين تطبيق سبارك و مدير العنقود [3]

تعدّ سواقة سبارك (Spark driver) المنسّق المركزي لتطبيق سبارك و تتفاعل مع مدير العنقود لمعرفة الأجهزة التي سيتم تنفيذ منطق معالجة البيانات عليها. على كلٍ من هذه الأجهزة تطلب سواقة سبارك من مدير العنقود تشغيل عملية تدعى المنفّذ (Executor) ، كما تجمع نتائج التنفيذ إن لزم من المنفذّات وتعرض النتيجة للمستخدم. تتمثلُ نقطةُ البداية في تنفيذ تطبيق سبارك في الصف SparkSession و الذي يُمكن أن يُستخدم لتحديد مجموعة من التشكيلات (Configurations) و واجهات برمجة التطبيقات (APIs) اللازمة للتطبيق.

بُني سبارك بشكل مكدس كما يوضّح الشكل 6 ، وفي طبقته الأولى المُسماة نواة سبارك (Spark Core) تمّ تضمين الوظائف الأساسيّة اللّازمة لإدارة و تنفيذ التطبيقات مثل: الجدولة ،

التنسيق ، و التسامح مع الأخطاء بالإضافة لبنى معطيات تُسمى مجموعات البيانات الموزعة و المتينة (*Resilient distributed datasets* (RDDs)) كتجريد برمجي لتمثيل ومعالجة البيانات. تتوضع فوق النواة مجموعة مكونات خُصص كلٌ منها لوظيفة محددّة مُتعلقة بمعالجة البيانات.



الشكل 6: مكدّس سبارك [3]

حالياً يقدم سبارك مكتبتين لتعلّم الآلة: Spark MLlib و Spark ML لهما واجهات APIs مختلفة لكن خوارزميات متشابهة. المكتبة الأولى و الأقدم Spark MLlib دخلت حالياً مرحلة الصيانة و التصحيحات (Bug-fix) ، أما المكتبة الثانية Spark ML و المُستخدمة في هذا البحث فهي الأحدث و مستوحاة من مكتبة تعلّم الآلة في بايثون scikit-learn ، كما أنّها تقدّم وظائف لإدارة و تبسيط العديد من مهام بناء نماذج تعلّم الآلة مثل هندسة السمات (Features Engineering)، سلسلة العمليات (Pipeline) لبناء النماذج و تقييمها و ضبط أدائها ، بالإضافة لتخزين النماذج بحيث تستخدم في التطبيقات و النظم بشكل فعلي.

7. توصيف مجموعة البيانات NSL-KDD [8]:

كانت KDD Cup مسابقة سنوية تُنظّم من قبل مجموعة مختصة من مؤسسة ACM ، و في عام 1999 خصصت المسابقة لمسألة كشف التسللات في الشبكات باستخدام تعلّم الآلة. مازالت مجموعة البيانات المتعلقة KDD'99 مُتاحة حتى الآن ، كما تُعتبر إحدى أشهر مجموعات البيانات المستخدمة لتقييم أداء نظم كشف الاختراقات.

لتكوين مجموعة بيانات التدريب (Training data) في KDD'99 تمّ تجميع حوالي 4 gigabytes من البيانات الشبكيّة خلال سبع أسابيع من شبكة محلية صُممت لتحاكي شبكة واقعية ، و تمّت معالجة تلك البيانات في خمسة ملايين سجل ، كما تمّ تجميع بيانات الاختبار (Test data) خلال أسبوعين إضافيين لتشكّل حوالي 2 مليون سجل. يختلف التوزيع الاحتمالي لمجموعة بيانات الاختبار عن التوزيع الاحتمالي لبيانات التدريب ، كما تتضمن بيانات الاختبار سجلات غير موجودة ضمن بيانات التدريب لجعل عملية التصنيف أكثر واقعيّة.

عُرّف الاتصال و المكون من 100 بايت على أنه سلسلة من الرزم التي تبدأ و تنتهي في نقاط زمنية محددة ، من عنوان IP مصدر إلى عنوان هدف وتابعة لبرتوكول محدّد و تمّت عنونته على أنه اتصال "طبيعي"، أو "هجمة" مع تحديد نوع الهجمة.

تمّ تصنيف أربع فئات (Categories) أساسيّة للهجمات:

- منع الخدمة ((Denial of Service (DoS): مثل (SYN flood).
- بعيد لمحلي ((Remote to Local (R2L) :محاولة الوصول من جهاز بعيد إلى الجهاز المحلي ، مثل (محاولة تخمين كلمة السر).
- مستخدم لجذر (User to Root) :وصول غير مسموح به لميزات (الجذر) ، مثل "Buffer Overflow".
- الجس (Probing) :مثل مسح المنافذ (Port scanning).

عرّف (Stolfo et al.) [9] مجموعة من السمات التي تسمح بتمييز الاتصالات الطبيعية من الهجمات كالآتي:

- السمات المرتبطة بنفس المضيف "Same host" وتفحص فقط الاتصالات التي لها نفس الهدف مثل الاتصال الحالي في الثانيتين السابقتين.
- السمات المتعلقة بنفس الخدمة "Same service" و تفحص فقط الاتصالات في الثانيتين السابقتين المرسلة لنفس الخدمة مثل الاتصال الحالي.

كل من سمات "نفس المضيف" و "نفس الخدمة" تدعى السمات المعتمدة على الوقت (Time-based).

- بعض هجمات الجس (Some probing) تمسح الأجهزة (أو المنافذ) باستخدام فترات زمنية أكبر من ثانيتين ، مثلاً مرّة كل دقيقة. لذلك تمّ ترتيب سجلات الاتصال وفقاً للمضيف الهدف ، وتمّ بناء السمات باستخدام نافذة من مئة اتصال إلى نفس المضيف بدلاً عن النافذة الزمنية لتنتج مجموعة من السمات المعتمدة على المضيف (Host-based).

- بعض الهجمات مثل R2L و U2R تكون متضمنة في قسم البيانات من الرزمة و تتطلب اتصال وحيد ، لذلك تمّت إضافة سمات تُعنى بالسلوك المشكوك به في أقسام البيانات مثل عدد محاولات التسجيل الفاشلة وسميّت هذه السمات: سمات المحتوى (Content features).

وبناء عليه فإن كل سجل يتكون من 41 سمة (Features) و تسمية (Label).

في هذا البحث استخدمنا مجموعة البيانات NSL-KDD [8] والتي تحل بعض مشاكل الكامنة في مجموعة البيانات KDD'99 [10] ، وبالرغم من أن مجموعة البيانات هذه مازالت تتضمن بعض المحدوديات ومن الممكن أن لاتشكّل تمثيل مثالي لشبكات فعلية إلاّ أنّها مازالت مُستخدمة بشكل كبير من قبل الباحثين كمجموعة بيانات فعّالة لتقييم ومقارنة أداء نظم كشف الاختراقات.

يوضّح السطر الآتي مثال عن سجل من مجموعة البيانات NSL-KDD:

```
0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.
00,0.00,0.00,150,25,0.17,0.03,0.17,0.00,0.00,0.00,0.05,0.00,normal,20
```

من ضمن الـ 41 سمة في السجل تتوزع السمات كالآتي: تستخدم السمات 9-1 لتمثيل السمات الأساسية للرزمة مثل مدة الاتصال ، نوع البرتوكول والخدمة ، 22-10 لسمات المحتوى، 23-31 للسمات المعتمدة على نافذة زمنية من ثانيتين ، و 41-32 للسمات المعتمدة على المضيف ، كما توضح الجداول 2 ، 3 ، 4 ، 5 الآتيّة:

رقم السمة	اسم السمة	توصيف	مثال
1	Duration	المدة الزمنية للاتصال	0
2	Protocol_type	نوع البرتوكول	TCP
3	Service	الخدمة	FTP
4	Flag	حالة الاتصال: صحيح أم خطأ	SF
5	Src_bytes	عدد البايتات المنقولة من المصدر إلى الهدف ضمن اتصال	491
6	Dst_bytes	عدد البايتات المنقولة من الهدف إلى المصدر ضمن اتصال	0
7	Land	1 إذا كان الاتصال من نفس/إلى نفس المضيف/المنفذ، و إلا تأخذ 0	0
8	Wrong_fragment	العدد الكلي للأجزاء الخاطئة في هذا الاتصال	0
9	Urgent	العدد الكلي للرزم المستعجلة التي تحمل البت urgent مفعلاً	0

الجدول 2: السمات الأساسيّة للاتصال

رقم السمة	اسم السمة	توصيف	مثال
10	Hot	عدد مرات الوصول لمجلدات النظام و انشاء و تنفيذ البرامج	0
11	Num_failed_logins	عدد مرّات فشل تسجيل الدخول	0
12	Logged_in	حالة تسجيل الدخول: 1 نجاح و إلا فشل	0
13	Num_compromised	عدد الشرط "Compromised"	0
14	Root_shell	1 إذا تمّ الوصول لـ Root shell ، و إلاّ 0	0
15	Su_attempted	1 إذا في حال محاول تنفيذ الأمر "su root" ، و إلاّ 0	0
16	Num_root	عدد مرات الوصول بصلاحيات "Root" أو عدد العمليات التي نفذت بصلاحيات "Root" في الاتصال	0
17	Num_file_creations	عدد مرات انشاء ملف في الاتصال	0
18	Num_shells	عدد الـ shell prompts	0
19	Num_access_files	عدد العمليات على ملفات التحكم بالوصول	0
20	Num_outbound_cmds	عدد الأوامر الصادرة (Outbound) في جلسة FTP	0
21	Is_hot_login	1 إن كان تسجيل الدخول ينتمي لقائمة "hot " مثل Admin أو Root	0
22	Is_guest_login	1 إن كان تسجيل الدخول كـ "guest " و إلا 0	0

الجدول 3: السمات المتعلقة بالمحتوى

رقم السمة	اسم السمة	توصيف	مثال
23	count	عدد الاتصالات إلى نفس الهدف مثل الاتصال الحالي في الثانيتين السابقتين	2
24	Srv_count	عدد الاتصالات إلى نفس الخدمة (رقم المنفذ) مثل الاتصال الحالي في الثانتيين السابقتين	2
25	Serror_rate	نسبة الاتصالات التي فعلت الراية (وسمة رقم 4) s0 ، s1 ، s2 or s3 بين الاتصالات المجمعة في count (وسمة رقم 23)	0
26	Srv_serror_rate	نسبة الاتصالات التي فعلت الراية (وسمة رقم 4) s0 ، s1 ، s2 or s3 بين الاتصالات المجمعة في srv_count (وسمة رقم 24)	0
27	Rerror_rate	نسبة الاتصالات التي فعلت الراية (وسمة رقم 4) REJ بين الاتصالات المجمعة في count (وسمة رقم 23)	0
28	Srv_rerror_rate	نسبة الاتصالات التي فعلت الراية (وسمة رقم 4) REJ بين الاتصالات المجمعة في srv_count (وسمة رقم 24)	0
29	Same_srv_rate	نسبة الاتصالات إلى نفس الخدمة من ضمن مجموعة الاتصالات المجمعة في count (وسمة رقم 23)	1
30	Diff_srv_rate	نسبة الاتصالات إلى خدمات مختلفة من ضمن مجموعة الاتصالات المجمعة في count (وسمة رقم 23)	0

31	Srv_diff_host_rate	نسبة الاتصالات إلى أجهزة مختلفة من بين الاتصالات المجمعة في srv_count (وسمة رقم 24)	0

الجدول 4: السمات المتعلقة بالزمن

رقم السمة	اسم السمة	توصيف	مثال
32	Dst_host_count	عدد الاتصالات التي لها نفس عنوان الانترنت (IP) الهدف	150
33	Dst_host_srv_count	عدد الاتصالات التي لها نفس المنفذ	25
34	Dst_host_same_srv_rate	نسبة الاتصالات لنفس الخدمة من بين الاتصالات المجمّعة في dst_host_count (الوسمة 32)	0.17
35	Dst_host_diff_srv_rate	نسبة الاتصالات إلى خدمات مختلفة من بين الاتصالات المجمّعة في dst_host_count (الوسمة 32)	0.03
36	Dst_host_same_src_port_rate	نسبة الاتصالات إلى نفس المنفذ من بين الاتصالات المجمعة في dst_host_srv_count (الوسمة 33)	0.17
37	Dst_host_srv_diff_host_rate	نسبة الاتصالات إلى أجهزة مختلفة المنفذ من بين الاتصالات المجمعة في dst_host_srv_count (الوسمة 33)	0
38	Dst_host_serror_rate	نسبة الاتصالات التي فعلت الراية (وسمة رقم 4) s0 ، s1 ، s2 or s3 بين الاتصالات المجمعة في dst_host_count (الوسمة 32)	0
39	Dst_host_srv_serror_rate	نسبة الاتصالات التي فعلت الراية (وسمة رقم 4) s0 ، s1 ، s2 or s3 بين الاتصالات المجمعة في dst_host_srv_count (الوسمة 33)	0
40	Dst_host_rerror_rate	نسبة الاتصالات التي فعلت الراية (وسمة رقم 4) REJ بين الاتصالات المجمعة في dst_host_count (الوسمة 32)	0.05
41	Dst_host_srv_rerror_rate	نسبة الاتصالات التي فعلت الراية (وسمة رقم 4) REJ بين الاتصالات المجمعة في dst_host_srv_count (الوسمة 33)	0

الجدول 5: السمات المعتمدة على المضيف

8. الأبحاث المتعلقة:

يتمحورُ عدد كبير من الأعمال البحثية أو التقنية حول موضوع تحليل مجموعة البيانات KDD'99 ونظم كشف الاختراقات ، لكن عدد قليل نسبياً ركز على موضوع تحليل البيانات الكبيرة المتعلقة بالحماية ، و من هذه الأبحاث اخترنا تلك الأقرب إلى موضوع هذا البحث.

في عام 2016 ، قدّم الباحثان (Nabila Farnaaz و M. A. Jabbar) [11] نموذج يعتمد على الغابات العشوائية لكشف الاختراقات في الشبكات الحاسوبية. تمّ تقييم أداء النموذج المُقترح في [11] باستخدام مجموعة البيانات NSL-KDD بعد اختيار السمات وفقاً لمعيار عدم الشك التناظري (Symmetrical uncertainty). استُخدم النموذج لتصنيف أنواع مختلفة من الهجمات وتم تقييمه من خلال تحديد أربعة معايير: الصحة (Accuracy) ، معدّل الكشف (Detection rate) و يمثّل النسبة بين العدد الكلّي للهجمات التي تمّ اكتشافها من قبل النظام الى العدد الكلي للملاحظات التي تم تصنيفها بشكل صحيح أي $\mathrm{Dr} = \mathrm{TP}/(\mathrm{TP} + \mathrm{TN})$ ، معدل الانذارات الخاطئة (FAR) ، و معامل ماتيو للترابط (MCC). بيّنت النتائج بأن للنموذج معدّل كشف مرتفع و معدّل إنذارات خاطئة منخفض ، كما أنّ النموذج صنّف هجمات منع الخدمة بصحة 99.67% وهذه القيمة أعلى بـ7% من خوارزمية شجرة القرار J48، كما بينت النتائج أيضا أن اختيار السمات له دور في تحسّن أداء نموذج التصنيف. جميع الاختبارات في هذا البحث تمت باستخدام الأداة WEKA.

في عام 2016، قدّم الباحثان (Tonya Fields و Jonathan Graham) [12] مُصنّف مُعتمد على الغابات العشوائية لبناء نظام كشف اختراقات شبكي (NIDS) لاكتشاف الشذوذات (Anomaly). تمّ استخدام مجموعة البيانات KDD'99 و أيضا الأداة WEKA وتمحورت الدراسة حول كيفية تغيّر الزمن اللازم لبناء النموذج مع تغير عدد الأشجار في الغابة ، وكذلك كيفية تحديد عدد الاشجار الأمثلي ، ودراسة دور عدد السمات بالنسبة لعدد الاشجار في تحقيق الدقة (Accuracy). أظهرت النتائج أنه يمكن الحصول على نتائج مناسبة عند تقليص مجموعة السمات بما يحسّن الكلفة الحسابية و يحسن صحة التنبؤ.

في عام 2017 ، قدّم الباحثان (Rifkie Primartha و Bayu Adhi Tama) [13] مقارنة لأداء نظام كشف اختراقات معتمد على مصنف الأشجار العشوائية من خلال ثلاث مجموعات بيانات متاحة على الانترنت: NSLKDD ، UNSW-NB15, and GPRS ومعيارين: الصحة (Accuracy) و معدل الانذارات الخاطئة ، وذلك باستخدام تحقيق الغابات العشوائية المضمنّة في حزمة H2O. تم تغيير أحجام الاشجار و البحث عن الوسائط الانسب باستخدام شبكة البحث (Grid search) لتظهر النتائج تفوّق الاشجار العشوائية على مصنفات مجمعّة (Ensembles) أخرى.

وفي عام 2018 ، قيّم الباحثون (Mustapha Belouch et .al) [14] أداء أربع خوارزميات تصنيف ألا وهي: شعاع الدعم الآلي ((Support Vector Machines (SVM)، بايز (Naïve Bayes)، أشجار القرار، و الغابات العشوائية باستخدام أباتشي سبارك و مجموعة البيانات UNSW-NB15 المفتوحة و المخصصة لاكتشاف الاختراقات. تمّت المقارنة من خلال تقييم خمسة معاملات للأداء: الصحة (Accuracy) ، الحساسية (Sensitivity) وتقيس نسبة الهجمات التي تم تصنيفها بشكل صحيح (TP/TP+FN) ، النوعيّة (Specificity) ، زمن بناء النموذج (Building time) و زمن التنبؤ (Prediction time). أظهرت هذه الدراسة تفوق مصنّف الغابات العشوائية على بقية المصنفات من حيث الصحة و الحساسية ، وتقارب بين الغابات العشوائية و أشجار القرار من حيث النوعية. في حين كانت خوارزمية بايز هي الأسرع من حيث التدريب، وكانت الأشجار العشوائية الاسرع باكتشاف الهجمات.

في عام 2018 ، قدّم الباحثان (Priyanka Dahiyaa و Devesh Kumar Srivastavab) [15] إطار لكشف التسللات باستخدام سبارك يحذف بداية السمات الأقل أهمية باستخدام تقنيتين:

LDA (Linear Discriminant Analysis) و CCA (Canonical Correlation Analysis (CCA)، ومن ثم تطبيق سبع تقنيات تعلم خاضع للإشراف ومنها الغابات العشوائية لكشف الاختراقات في مجموعة البيانات UNSW-NB15. تمت المقارنة بين التقنيات المطبقة من خلال مجموعة من المعايير لتظهر الدراسة أن خوارزمية الشجرة العشوائية حققت الأداء الأفضل و بأن خوارزمية LDA كانت افضل تقنية لتقليص السمات.

في عام 2018 ، قدّم الباحثان (M.Kranthi Kumar و K. Raja Kumar) [16] نظام لكشف الاختراقات في التطبيقات ذات البيانات الكبيرة باستخدام سبارك. أيضاً في هذا البحث تمت المقارنة بين عدد من المصنفات من ضمنها الغابات العشوائية ، واستخدمت مجموعتي البيانات KDD Cup99 وNSL KDD. قبل تطبيق التصنيف تم تقليص السمات باستخدام ثلاث خوارزميات: ربح المعلومات (Information Gain) ، نسبة الربح (Gain Ratio) و الترابط (Correlation). بينت النتائج أنه في حالة كل من مجموعتي البيانات حقق كل من الغابات العشوائية و مصنف الانحدار اللوجستي أفضل أداء من حيث دقة التصنيف و كانت الغابات العشوائية الأفضل من حيث زمن التنبؤ.

نحقق في هذا البحث نموذج لاكتشاف التسللات إلى الشبكات المحلية باستخدام خوارزمية الغابات العشوائية المضمنّة في مكتبة تعلم الآلة (ML) من سبارك ولغة سكالا دون

اختيار للسمات بعكس معظم الأبحاث المتعلقة ، كما نقيّم النموذج المحقّق وفقاً لعدد من المعايير بالإضافة للبحث عن مجموعة الوسائط الفائقة التي تؤدي لأداء أمثلي.

9. الدراسة العمليّة و تقييم الأداء:

تمّت هذه الدراسة باستخدام حاسب شخصي ذو مُعالج Intel Core i5-4200U مزوّد بذاكرة RAM 8.0GB و بنظام تشغيل ويندوز 8.1 64 بت ، كما استخدمنا نظام سبارك إصدار spark-2.4.1 و لغة سكالا. يوضّح الشكل 7 تشغيل البيئة.

```
C:\spark-2.4.1\bin>spark-shell.cmd --driver-memory 8g
19/09/25 16:14:13 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
Spark context Web UI available
Spark context available as 'sc' (master = local[*], app id = local-1569417268930
).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.1
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_122-e
a)
Type in expressions to have them evaluated.
Type :help for more information.
```

الشكل 7: تنفيذ سبارك

قُمنا بدراسة الحالات الثلاث التاليّة و بناء نموذج لكلّ حالة منها:

1. الحالة الأولى: تصنيف مُتعدد الفئات (Multiclass classification) بدون ترميز واحد حار (One hot encoding) حيث تمّ اعتبار جميع فئات المُلاحظات و لم تُرمّز قيّم السمات الفئوية الرقمية.
2. الحالة الثانيّة: تصنيف مُتعدد الفئات مع ترميز واحد حار.
3. الحالة الثالثة: تصنيف ثنائي (Binary classification) بدون ترميز واحد حار حيث تمّ اعتبار الاتصالات الطبيعية فئة أولى و البقية كفئة ثانية و لم يتم ترميز قيّم السمات الفئوية الرقمية.

9.1. الحالة الأولى: تصنيف مُتعدد الفئات بدون ترميز واحد حار:

حققنا هذا التصنيف وفقاً للخطوات الآتيّة:

أولاً: تحضير مجموعة البيانات

تضمنّت هذه المرحلة قراءة البيانات ومن ثم ترميز تسميات الفئات (Labels) بأدلة رقمية بالإضافة لاستكشاف البيانات مثل استعراض عدد الملاحظات ضمن كل فئة. يوضّح كل من

الشّكل 8 و الشّكل 9 على الترتيب قراءة بيانات التدريب و من ثمّ عرض سجل باستخدام التعليمة take من مجموعة البيانات الناتجة عن المرحلة الأولى:

```
scala> val KDDTrain = spark.read.option("inferSchema", true).option("header", fa
lse).csv("file:///C:/shared/NSL_KDD_Dataset/KDDTrain+.txt").toDF("duration","pro
tocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","ur
gent","hot","num_failed_logins", "logged_in","num_compromised","root_shell","su_
attempted","num_root", "num_file_creations","num_shells","num_access_files","num
_outbound_cmds", "is_host_login","is_guest_login","count","srv_count","serror_ra
te", "srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate", "diff_sr
v_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count", "dst_host_sa
me_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate", "dst_host_s
rv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_r
error_rate","dst_host_srv_rerror_rate","label","number")
[Stage 7:==============>                                          (1 + 3) / 4]

KDDTrain: org.apache.spark.sql.DataFrame = [duration: int, protocol_type: string
 ... 41 more fields]
```

الشكل 8: قراءة بيانات التدريب

```
scala> data.take(1)
[Stage 7:>                                                        (0 + 4) / 4]
[Stage 7:==============>                                          (1 + 3) / 4]
[Stage 7:=========================================>               (3 + 1) / 4]

res4: Array[org.apache.spark.sql.Row] = Array([0,icmp,eco_i,SF,8,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,1,44,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,1,95,1.0,0.0,1.0,0.51,0.0,0.
0,0.0,0.0,ipsweep])
```

الشّكل 9: تنفيذ التعليمة take

يوضّح الشكل 10 الفئات و أدلتها:

```
+------------+--------------+
|indexedlabel|         label|
+------------+--------------+
|         0.0|        normal|
|         1.0|       neptune|
|         2.0|         satan|
|         3.0|       ipsweep|
|         4.0|     portsweep|
|         5.0|         smurf|
|         6.0|          nmap|
|         7.0|          back|
|         8.0|      teardrop|
|         9.0|   warezclient|
|        10.0|           pod|
|        11.0|  guess_passwd|
|        12.0|buffer_overflow|
|        13.0|   warezmaster|
|        14.0|          land|
|        15.0|          imap|
|        16.0|       rootkit|
|        17.0|    loadmodule|
|        18.0|     ftp_write|
|        19.0|      multihop|
|        20.0|           phf|
|        21.0|          perl|
|        22.0|           spy|
+------------+--------------+
```

الشكل 10: الفئات و أدلتها

تتكون مجموعة البيانات النهائية من 125973 سجل مصنفة ضمن 23 فئة.

أعدنا بعد ذلك تقسيم مجموعة البيانات لمجموعة تدريب مكونة من 80% من السجلات و مجموعة اختبار تتضمن 20% من السجلات كما استفدنا من ميزة تخبئة للبيانات يُقدمها سبارك بهدف تسريع الحسابات (الشكل 11).

```
scala> val Array(trainingData, testData) = data.randomSplit(Array(0.8, 0.2), 11L
)
trainingData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [duration
: int, protocol_type: string ... 40 more fields]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [duration: in
t, protocol_type: string ... 40 more fields]

scala> trainingData.cache()
res5: trainingData.type = [duration: int, protocol_type: string ... 40 more fiel
ds]

scala> testData.cache()
res6: testData.type = [duration: int, protocol_type: string ... 40 more fields]
```

الشّكل 11: تقسيم مجموعة البيانات و التخبئة

يبيّن الشكل 12 عدد الملاحظات ضمن كل فئة في البيانات الكليّة.

```
scala> KDDTrainindexed.groupBy("label").count().orderBy('count).show(23)
[Stage 15:==============>                                           (1 + 3) / 4]
[Stage 16:=========================================>              (154 + 4) / 200]

+----------------+-----+
|           label|count|
+----------------+-----+
|             spy|    2|
|            perl|    3|
|             phf|    4|
|       multihop.|    7|
|      ftp_write.|    8|
|     loadmodule.|    9|
|        rootkit.|   10|
|           imap.|   11|
|           land.|   18|
|    warezmaster.|   20|
| buffer_overflow.|   30|
|   guess_passwd.|   53|
|            pod.|  201|
|    warezclient.|  890|
|       teardrop.|  892|
|           back.|  956|
|           nmap.| 1493|
|          smurf.| 2646|
|      portsweep.| 2931|
|        ipsweep.| 3599|
|          satan.| 3633|
|        neptune.|41214|
|         normal.|67343|
+----------------+-----+
```

الشكل 12: عدد الملاحظات ضمن كل فئة في البيانات الكليّة

ثانياً: تعريف تحويلات لمقابلة السمات الفئوية المحرفية (البرتوكول ، الخدمة ، الراية) بأدلة رقمية وذلك باستخدام أحد صفوف تحويل البيانات :StringIndexer والذي يرمّز القيّم بأدلة رقميّة حسب تكرار القيمة بحيث يتم ترميز القيمة الأكثر تكراراً بـ 0 و هكذا.

ثالثاً: تعريف تحويل لتشكيل أشعة السمات (Features Vectors) من خلال تجميع جميع أعمدة السمات التي ستُستخدم في نموذج تعلّم الآلة في شعاع واحد.

رابعاً: بناء النموذج باستخدام خوارزمية الغابات العشوائية (الشكل 13).

```
scala> val rf = new RandomForestClassifier().
       setSeed(5043).
       setNumTrees(50).
       setMaxDepth(10).
       setMaxBins(100).
       setFeatureSubsetStrategy("auto").
       setLabelCol("indexedLabel").
       setFeaturesCol("features").
       setPredictionCol("prediction")
rf: org.apache.spark.ml.classification.RandomForestClassifier = rfc_168917bbb0b6
```

الشكل 13: بناء نموذج الغابات العشوائية

ومن ثمّ تعريف تسلسل عمليات (Pipeline) يتضمن الخطوات الثلاث السابقة ليؤدي في النتيجة لتوليد نموذج تمّ تدريبه على مجموعة بيانات التدريب (الشكل 14).

```
scala> val model = pipeline.fit(trainingData)
[Stage 11:>                                                         (0 + 4) / 5]
[Stage 11:===========>                                              (1 + 4) / 5]
[Stage 11:=======================>                                  (2 + 3) / 5]
[Stage 11:=================================================>        (4 + 1) / 5]
```

الشكل 14: تدريب النموذج

يوضّح الشكل 13 إعدادات المصنّف حيث تمّ بداية تحديد رقم عشوائي في setSeed ، ومن ثمّ عدد المجالات التي ستُستخدم لتقطيع السمات ذات القيّم المستمرة ((setMaxBins(100)) ، أما الخاصية featureSubsetStrategy فأعطيت القيمة auto حتى تختار الخوارزميّة مجموعة من السمات الأفضل بشكل أتوماتيكي ، بالإضافة لعمود العنونة ("indexedLabel") و اسم العمود الجديد الذي يُضيفه المصنّف للقيّم التي يتنبأ بها ("prediction").

بعد بناء النموذج حددنا أهمية السمات والتي يقصد بها مدى مساهمة كل سمة في التنبؤات الصحيحة وهذه ميزة تقدمها خوارزمية أشجار القرار و الغابات العشوائية [6]. يبيّن الشكل 15 ترتيب أهميّة السمات في الحالة الأولى:

```
scala> Model.featureImportances.toArray.zip(combined).sorted.reverse.foreach(pri
ntln)
(0.11365616216748639,src_bytes)
(0.10862363062893887,diff_srv_rate)
(0.10819379056857317,same_srv_rate)
(0.056594790771059146,srv_serror_rate)
(0.056314539264287866,dst_host_serror_rate)
(0.054712752338939345,flag_indexed)
(0.049258123128110336,dst_host_srv_serror_rate)
(0.04257625997879035,dst_bytes)
(0.04209478844805279,service_indexed)
(0.039523537542342536,count)
(0.03775590236702081,protocol_type_indexed)
(0.03423642982038563,dst_host_diff_srv_rate)
(0.03378455346926617 6,serror_rate)
(0.0292125598688617,dst_host_same_srv_rate)
(0.023064329676120873,dst_host_same_src_port_rate)
(0.022093486225959 98,dst_host_srv_diff_host_rate)
(0.020939659802086816,srv_count)
(0.020075253884009835,dst_host_srv_count)
(0.017434475953492023,dst_host_srv_rerror_rate)
(0.017321388355693704,dst_host_rerror_rate)
(0.016061194584460012,dst_host_count)
(0.010911834243300939,srv_rerror_rate)
(0.010353265909217764,rerror_rate)
(0.006630880204496068,wrong_fragment)
(0.006595301677786092,logged_in)
(0.00632266164005416,num_compromised)
(0.006092532934831828,srv_diff_host_rate)
(0.00543889188836696,hot)
(0.00293031973156353,duration)
(6.3315694386135 98E-4,is_guest_login)
(2.5926693827047197E-4,num_failed_logins)
(7.522258722246172E-5,num_root)
(7.381932470789 14E-5,root_shell)
(6.268638454945078E-5,land)
(5.006819275451807E-5,num_file_creations)
(2.3635611465897796E-5,num_shells)
(1.6824382692820 28E-5,num_access_files)
(2.0225609194435472E-6,su_attempted)
(0.0,urgent)
(0.0,num_outbound_cmds)
(0.0,is_host_login)
```

الشكل 15: أهمية السمات مرتبة تنازلياً

خامساً: تقييم أداء النموذج

في هذه المرحلة طبقنا النموذج على مجموعة بيانات الاختبار(الشكل 16).

```
scala> val predictions = model.transform(testDF1)
predictions: org.apache.spark.sql.DataFrame = [duration: int, protocol_type: str
ing ... 47 more fields]
```

الشكل 16: تطبيق النموذج على بيانات الاختبار

يوضح الشكل 17 صحّة النموذج :

```
scala> val testErr =  predictionAndLabels.filter(r => r._1 != r._2).count.toDoub
le / testData.count()
[Stage 88:>                                                         (0 + 4) / 4]
[Stage 88:==============>                                           (1 + 3) / 4]
[Stage 88:==============================>                           (2 + 2) / 4]

testErr: Double = 0.0043118794256101906

scala> println("Accuracy = " + (1-testErr) * 100 + " %")
Accuracy = 99.56881205743898 %
```

الشكل 17: صحة النموذج في الحالة الأولى

ونظراً لعدم تقارب عدد الملاحظات في الفئات قيمنا الإحكام ، الاستدعاء و معدّل الانذارات الخاطئة (FPR) لكل فئة. يوضح الجدول 6 النتائج:

Precision	Recall	FPR
Precision(normal)=99.32 %	Recall(normal)=99.96 %	FPR(normal)=0.79 %
Precision(neptune)=100.00 %	Recall(neptune)=99.98 %	FPR(neptune)=0.00 %
Precision(satan)=100.00 %	Recall(satan)=97.17 %	FPR(satan)=0.00 %
Precision(ipsweep)=98.77 %	Recall(ipsweep)=99.17 %	FPR(ipsweep)=0.04 %
Precision(portsweep)=99.64 %	Recall(portsweep)=98.76 %	FPR(portsweep)=0.01 %
Precision(smurf)=100.00 %	Recall(smurf)=99.63 %	FPR(smurf)=0.00 %
Precision(nmap)=99.18 %	Recall(nmap)=93.08 %	FPR(nmap)=0.01 %
Precision(back)=100.00 %	Recall(back)=100.00 %	FPR(back)=0.00 %
Precision(teardrop)=100.00 %	Recall(teardrop)=100.00 %	FPR(teardrop)=0.00 %
Precision(warezclient)=100.00 %	Recall(warezclient)=81.67 %	FPR(warezclient)=0.00 %
Precision(pod)=100.00 %	Recall(pod)=100.00 %	FPR(pod)=0.00 %
Precision(guess_passwd)=100.00 %	Recall(guess_passwd)=100.00 %	FPR(guess_passwd)=0.00 %
Precision(buffer_overflow)=100.00 %	Recall(buffer_overflow)=50.00 %	FPR(buffer_overflow)=0.00 %
Precision(warezmaster)=100.00 %	Recall(warezmaster)=50.00 %	FPR(warezmaster)=0.00 %
Precision(land)=50.00 %	Recall(land)=100.00 %	FPR(land)=0.01 %
Precision(imap)=100.00 %	Recall(imap)=100.00 %	FPR(imap)=0.00 %
Precision(rootkit)=0.00 %	Recall(rootkit)=0.00 %	FPR(rootkit)=0.00 %
Precision(loadmodule)=0.00 %	Recall(loadmodule)=0.00 %	FPR(loadmodule)=0.00 %
Precision(ftp_write)=0.00 %	Recall(ftp_write)=0.00 %	FPR(ftp_write)=0.00 %
Precision(multihop)=0.00 %	Recall(multihop)=0.00 %	FPR(multihop)=0.00 %
Precision(phf)=0.00%	Recall(phf)=0.00%	FPR(phf)=0.00 %

الجدول 6: نتائج تقييم الإحكام ، الاستدعاء و معدل الإنذارات الخاطئة للنموذج الأول

تشير النتائج إلى أن المصنف حقق أداء مرتفع بالنسبة لمعظم الفئات ، لكن يوجد فئات مثل ftp_write لم تظهر أي نتيجة لها غير "صفر" و من الممكن أن ذلك يعود لقلة عدد الملاحظات في هذه الفئات بالنسبة لبقية الفئات.

نشير إلى أنه نظراً لتثبيت رقم ضمن setSeed في المصنف فإن تكرار التجربة سيؤدي لنفس النتائج السابقة.

سادساً: ضبط الوسائط الفائقة (Hyperparameters)

في هذا الاختبار استخدمنا الصف ParamGridBuilder و تم البحث من خلال ثلاث وسائط يأخذ كل منها قيمتان مما يعني أنه سيتم تقييم 8 وسطاء (الشكل 18):

```
scala>
     | val paramGrid = new ParamGridBuilder().
     | addGrid(rf.impurity, Seq("gini", "entropy")).
     | addGrid(rf.maxDepth, Seq(1, 20)).
     | addGrid(rf.maxBins, Seq(100, 300))
paramGrid: org.apache.spark.ml.tuning.ParamGridBuilder = org.apache.spark.ml.tun
ing.ParamGridBuilder@56fe3ec2
```

الشكل 18: تحديد ثلاث وسائط فائقة للبحث عن النموذج الأفضل

ثم طبقنا التحقق المتقاطع مع تحديد أن البحث عن النموذج الأفضل سيتم بناء على معيار الصحة. نتيجة لضبط النموذج وجدنا تحسن في الصحة لتصبح (99.82%) (الشكل 19).

```
scala> val accuracy = multiclassEval.evaluate(predictionscv)
[Stage 672:>                                                        (0 + 4) / 4]
[Stage 672:==============>                                         (1 + 3) / 4]
[Stage 672:============================>                           (2 + 2) / 4]

[Stage 674:==============>                                         (1 + 3) / 4]

accuracy: Double = 0.9981803077653388
```

الشكل 19: صحة النموذج الأفضل بعد عملية البحث

كما وجدنا أن الوسائط الفائقة لهذا النموذج كما يوضّح الشكل 20:

```
scala> bestModel.asInstanceOf[PipelineModel].stages.last.extractParamMap
res7: org.apache.spark.ml.param.ParamMap =
{
        rfc_388209a1ddbe-cacheNodeIds: false,
        rfc_388209a1ddbe-checkpointInterval: 10,
        rfc_388209a1ddbe-featureSubsetStrategy: auto,
        rfc_388209a1ddbe-featuresCol: features,
        rfc_388209a1ddbe-impurity: gini,
        rfc_388209a1ddbe-labelCol: indexedLabel,
        rfc_388209a1ddbe-maxBins: 300,
        rfc_388209a1ddbe-maxDepth: 20,
        rfc_388209a1ddbe-maxMemoryInMB: 256,
        rfc_388209a1ddbe-minInfoGain: 0.0,
        rfc_388209a1ddbe-minInstancesPerNode: 1,
        rfc_388209a1ddbe-numTrees: 50,
        rfc_388209a1ddbe-predictionCol: prediction,
        rfc_388209a1ddbe-probabilityCol: probability,
        rfc_388209a1ddbe-rawPredictionCol: rawPrediction,
        rfc_388209a1ddbe-seed: 5043,
        rfc_388209a1ddbe-subsamplingRate: 1.0
}
```

الشكل 20: وسائط النموذج الأفضل في الحالة الأولى

9.2. الحالة الثانيّة: تصنيف مُتعدد الفئات مع ترميز واحد حار:

في الحالة الثانية قمنا بترميز قيم السمات الفئوية الرقمية قبل عملية التصنيف باستخدام ترميز واحد حار بهدف دراسة إن كان هذا الترميز سيؤدي لتحقيق أداء أفضل للمصنف. لكن صحة النموذج الناتج (99.26 %) لم تكن أفضل من سابقه مما يعني أن استخدام الترميز واحد حار لم يؤدي لتحقيق مصنف ذو أداء أفضل.

9.3. الحالة الثالثة: تصنيف ثنائي بدون ترميز واحد حار:

تم اعتبار الاتصالات الطبيعية فئة أولى (0) و البقية كفئة ثانية (1) و لم يتم ترميز قيّم السمات الفئوية الرقمية. يوضح الشكل 21 تابع تحويل العنونة إلى صفر أو واحد حسب إن كان الاتصال طبيعي أم غير ذلك.

```
scala> val func = udf( (i:String) => if(i == "normal") 0.0 else 1.0 )
func: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction
(<function1>,DoubleType,Some(List(StringType)))
```

الشكل 21: تابع تحويل العنونة إلى صفر أو واحد

تم اتباع نفس خطوات المرحلة الأولى لنجد بداية أن ترتيب أهميّة السمات قد تغيّر و كانت السمات العشر الأكثر أهمية كالتالي:

(0.18560993211330665,dst_bytes)
(0.1517893921246571,src_bytes)
(0.08985512205020213,same_srv_rate)
(0.0851156044180343,diff_srv_rate)
(0.06677556272995512,dst_host_srv_count)
(0.06562562869258275,dst_host_same_srv_rate)
(0.052777321343388105,service_indexed)
(0.04115571036797827,protocol_type_indexed)
(0.033677374186416066,flag_indexed)
(0.027352062422142383,logged_in)
(0.0270730898054533,dst_host_diff_srv_rate)...

وكذلك وجدنا تحسّن ملحوظ في أداء النموذج ، حيث كانت النتائج كما يوضح الجدول 7:

Area under PR	Area under ROC Area under ROC	F1	الصحة
99.99%	99.74%	99.75 %	99.75 %

الجدول 7: نتائج تقييم الأداء للنموذج الثالث

طبقنا البحث عن الوسائط في هذه الحالة أيضاً لنلاحظ تحسّن في قيمة معيار الصحة ليصبح (99.88%) ، كما أن وسائط النموذج الأفضل موضحة في الشكل 22:

```
scala> bestModel.asInstanceOf[PipelineModel].stages.last.extractParamMap
res7: org.apache.spark.ml.param.ParamMap =
{
        rfc_ea1ab87d22c7-cacheNodeIds: false,
        rfc_ea1ab87d22c7-checkpointInterval: 10,
        rfc_ea1ab87d22c7-featureSubsetStrategy: auto,
        rfc_ea1ab87d22c7-featuresCol: features,
        rfc_ea1ab87d22c7-impurity: gini,
        rfc_ea1ab87d22c7-labelCol: label,
        rfc_ea1ab87d22c7-maxBins: 100,
        rfc_ea1ab87d22c7-maxDepth: 20,
        rfc_ea1ab87d22c7-maxMemoryInMB: 256,
        rfc_ea1ab87d22c7-minInfoGain: 0.0,
        rfc_ea1ab87d22c7-minInstancesPerNode: 1,
        rfc_ea1ab87d22c7-numTrees: 50,
        rfc_ea1ab87d22c7-predictionCol: prediction,
        rfc_ea1ab87d22c7-probabilityCol: probability,
        rfc_ea1ab87d22c7-rawPredictionCol: rawPrediction,
        rfc_ea1ab87d22c7-seed: 5043,
        rfc_ea1ab87d22c7-subsamplingRate: 1.0
}
```

الشكل 22: وسائط النموذج الأفضل في الحالة الثالثة

10.الخاتمة و التوصيات المستقبلية:

حققنا في هذا البحث عدة نماذج لاكتشاف التسللاّت إلى الشبكات المحليّة باستخدام الغابات العشوائية و مكتبة تعلّم الآلة من أباتشي سبارك وهو نظام مفتوح المصدر لمعالجة موزعّة

للبيانات الكبيرة ، كما قيمنا أداء هذه النماذج اعتماداً على عدة معايير. بيّنت النتائج دقة عالية في اكتشاف التسلّلات ، كما بينت أن التصنيف الثنائي يحقق نتائج أفضل من التصنيف المتعدد الفئات و بأن ترميز السمات الفئوية الرقمية بمرمّز واحد حار لم يكن له تحسين ملحوظ على أداء المصنف.

يمكن في الأعمال المستقبلية استخدام مجموعات بيانات مختلفة لمقارنة أداء التصنيف ، كما يمكن دراسة أداء المصنف بعد انتقاء مجموعة من السمات الأكثر الأهمية.

11. المراجع:

[1]. د.زينب خلّوف ، دراسة ومقارنة تقنيات تحسين أداء نظم كشف الاختراقات في الشبكات الحاسوبية. بحث مسجل قُبل للنشر في مجلة جامعة البعث ، المجلد 32 لعام 2010.

[2]. Apache Spark Lightning-fast unified analytics engine.
URL: https://spark.apache.org/, last retrieved: 29-9-2019

[3]. H. Luu, Beginning Apache Spark 2 With Resilient Distributed Datasets, Spark SQL, Structured Streaming and Spark Machine Learning library. Apress; 1st edition (2018).

[4]. Differences between Receiver Operating Characteristic AUC (ROC AUC) and Precision Recall AUC (PR AUC), URL: http://www.chioka.in/differences-between-roc-auc-and-pr-auc/, last retrieved: 29-9-2019.

[5]. S. Raschka and V. Mirjalili, Python Machine Learning. Birmhingham, UK: Packt Publishing (2017).

[6] J. Wills, S. Ryza, S. Owen, and . Laserson, Advanced Analytics with Spark: Patterns for Learning from Data at Scale. O'Reilly Media; 2nd edition (2017).

[7]. X-She Yang, Introduction to Algorithms for Data Mining and Machine Learning. Academic Press (2019).

[8]. NSL-KDD, https://www.unb.ca/cic/datasets/nsl.html. Last retrieved: 29-9-2019.

[9]. S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, Cost based modeling for fraud and intrusion detection: Results from the jam project, discex, vol. 02, p. 1130, 2000.

[10]. KDD99, http://kdd.ics.uci.edu/databases/kddcup99/task.html.

[11]. N. Farnaaz, M. A. Jabbar, "Random forest modeling for network intrusion detection system", Procedia Comput. Sci., vol. 89, pp. 213-217, Jan. 2016.

[12] T.Fields and J.Graham. Classifying network attack data using random forest. Computers and Their Applications (CATA), Dec 2016.

[13]. R. Primartha and B. Tama. Anomaly Detection Using Random Forest: A Performance Revisited. International Conference on Data and Software Engineering (ICoDSE), Palembang Sumatra Selatan, Indonesia, pp. 1--6, IEEE, 2017.

[14]. Belouch M, El Hadaj S, Idhammad M. Performance evaluation of intrusion detection based on machine learning using Apache Spark. Procedia Computer Science. 2018.

[15]. Dahiya P, Srivastava DK. Network intrusion detection in big dataset using Spark. Procedia Comput Sci. 2018.

[16]. M. Kranthi Kumar, K. Raja Kumar, Intrusion Detection System For Big Data Driven Application Using Apache Spark, International Journal of Management, Technology And Engineering, Volume 8, Issue XI, 2018.
[17]. Ofir Shalev, Recall, Precision, F1, ROC, AUC, and everything, May 27, 2019,
URL: https://medium.com/swlh/recall-precision-f1-roc-auc-and-everything-542aedf322b9, last retrieved: 24-5-21
[18]. Random forests or random decision forests,
URL: https://www.kaggle.com/getting-started/176257, last retrieved: 24-5-21
[19]. K. Scarfone, P. Mell, Special Publication 800-94: Guide to Intrusion Detection and Prevention Systems (IDPS), National Institute of Standards and Technology (NIST) (2007).