Syrian Arab Republic

Higher Institute for Applied Science and Tecnology

Forth Year

4th year project:

Packet inspection to detect attacks using machine and deep learning

By student: Hussein Salloum

Supervised by:

Dr- Sameeh Jamoul

Dr- Mohammed Bashar Dessouki

2024-2025

Contents

[ملخص 3](#_Toc205276360)

[هدف المشروع 4](#_Toc205276361)

[المتطلبات الوظيفية 4](#_Toc205276362)

[المتطلبات غير الوظيفية 5](#_Toc205276363)

[مخططات تصميم النظام 6](#_Toc205276364)

[فصل التنفيذ والاختبارات 12](#_Toc205276365)

[معمارية النظام المتبعة 12](#_Toc205276366)

[تفصيل أجزاء النظام 12](#_Toc205276367)

[Pseudocode 12](#_Toc205276368)

[خطة الاختبارات 13](#_Toc205276369)

[Dataset IOT-23 14](#_Toc205276370)

[مقدمة وتوصيف لمجموعة البيانات 14](#_Toc205276371)

[استخراج البيانات 16](#_Toc205276372)

[المعالجة المسبقة للبيانات 16](#_Toc205276373)

[دمج البيانات 17](#_Toc205276374)

[تدريب النماذج (models) واختبارهم 17](#_Toc205276375)

[مصنف الغابة العشوائية (Random Forest) 17](#_Toc205276376)

[Random Forest Hyperparameters 18](#_Toc205276377)

[Random Forest Implementation 19](#_Toc205276378)

[XGBoost Implementation 21](#_Toc205276379)

[Bibliography 22](#_Toc205276380)

[Figure 1 System block diagram 6](#_Toc205085752)

[Figure 2 Training flow chart 7](#_Toc205085753)

[Figure 3 Attacking flow chart 8](#_Toc205085754)

[Figure 4 Random forest model diagram 9](#_Toc205085755)

[Figure 5 Xgboost model diagram 9](#_Toc205085756)

[Figure 6 Lightgbm diagram 10](#_Toc205085757)

[Figure 7 Neural Network model diagram 10](#_Toc205085758)

[Figure 8 Deployment diagram 11](#_Toc205085759)

# ملخص

لقد انتشرت شبكات انترنت الاشياء بشكل ملحوظ في الأونة الأخيره مما أدّى بالطبع الى ازدياد الهجومات عليهم وخصوصا باستخدام الروبوتات الشبكية botnets, ومن هنا أتت أهمية كشف الهجومات على شبكات انترنت الاشياء في الوقت الحقيقي واخترنا لتنفيذ ذلك تدريب نماذج باستخدام التعلم الالي والتعلم العميق لكشف هجومات الروبوتات الشبكية على شبكات انترنت الاشياء باستخدام البيانات IoT-23 dataset

واحتجنا لتنفيذ ذلك الى تحميل العديد من ملفات conn.log.labeled واستخراج البيانات منها الى صيغ .csv وتبع ذلك العديد من تقنيات تنظيف البيانات وحذف السمات الغير مهمة وتحويل التصنيف الى تصنيف ثنائي (binary classification) وهو هجوم أو لا (benign or malicious) وقد دربنا على 4 أنواع من المصنفات هم: LightGBM, XGBoost, Random Forest Neural Network .

وتم عمل محاكاة لكشف الهجوم بالوقت الحقيقي باستخدام التان افتراضيتان واحده لانشاء الهجومات واخرى لكشفها

حسث أن الة الكشف استخدمت zeek لتحليل حركة المرور التي وضعت الاتصالات في ملف log ثم تم قراءة الاسطر من ملف الlog وارسالها الى pipe التي يقرأها ملف الpredection الذي يحوي نموذج مدرب ليقوم بالتنبؤات حول ماهية الاتصالات وعرض النتائج بطريقه واضحة باستخدام dashboard.

هدف المشروع

هو تدريب نظام ذكاء صنعي قادر على كشف هجومات الروبوتات الشبكية (botnets) على شبكات انترنت الأشياء IoT مثل هجومات PartOfAHorizontalPortScan عبر خوارزميات التعلم الآلي وخوارزميات التعلم العميق.

# المتطلبات الوظيفية

1. القدرة على تحميل بيانات التدريب والاختبار (IoT-23) بصيغة .csv وهي الصيغة التي استخدمناها من أجل :
   1. تدريب النموذج على بيانات التدريب (Training Data).
   2. اختبار النموذج على بيانات الاختبار (Testing Data).
2. معالجة البيانات بشكل مسبق (preprocessing data) ومنها:
   1. معالجة القيم المفقودة (NaN) مثل اسناد قيمة الصفر اليها او اسناد قيمة الوسيط (median) اليها.
   2. ترميز (encoding) قيم السمات من قيم فئوية (categorical values) الى قيم رقمية.
   3. حذف (drop) السمات غير المفيدة.
   4. تحويل أنماط (types) السمات الى الأنماط المناسبة لطبيعة قيمها.
   5. تقسيم البيانات الى بيانات للتدريب وبيانات للاختبار باعتماد نسبة معينة للتقسيم مثل (30%-70%).
3. تدريب نموذج (classifier) أو أكثر وتقييم النماذج حيث أنه يجب أن يكون النظام قادرا على:
   1. التدريب على نماذج مثل الغابة العشوائية والشبكات العصبونية (Random Forest, Neural Network, etc).
   2. قادراً على تقييم كل نموذج باستخدام المقاييس المختلفة مثل: الدقة (Precision) والاسترجاع (Recall) وF1-score والمساحة تحت المنحني (AUC: Area Under Curve) ومصفوفة الارتباك (Confusion Matrix).
   3. القدرة على حفظ النموذج المُدرَّب من أجل الاستخدام المستقبلي.
4. القدرة على تحميل النموذج واستخدامه للكشف في الوقت الحقيقي أي يجب أن يكون النظام قادراً على :
   1. تحميل النوذج المُدرّب من أجل استحدامه لكشف الهجومات.
   2. استقبال البيانات ومعالجتها بشكل مباشر في الوقت الحقيقي أي يجب أن يكون النظام قادراً على:

4-2-1. مراقبة الشبكة وأخذ بيانات الاتصالات الواردة الى جهاز الكشف.

4-2-2. معالجة البيانات الملتقطة بحيث يتم حذف السمات واجراء معالجة للبيانات لتكون صيغتها مماثلة لصيغة البيانات التي تدرّب عليها النموذج المُحمَّل.

* 1. التنبّؤ بماهية الاتصال ان كان سليم او هجوم اعتماداً على البيانات التي التقطها وعالجها.
  2. توليد مخرجات تنبؤ النموذج المحمل بطريقة واضحة.

# المتطلبات غير الوظيفية

1. أداء جيد للنظام حيث يجب أن:
   1. معالجة اتصالات الشبكة واجراء التنبؤات بشكل سريع جدّا ليكون قريب من الوقت الفعلي مثلا اعطاء التنبّؤ المتوَقّع للاتصال في مدة أقل من 50 ميلي ثانية.
   2. يكون قادراً على التعامل مع حركة مرور كبيرة ضمن الشبكة ومعالجتها من دون تأخير أي أنه يجب أن يتحمل الضغط الناجم عن الاتصالات.
   3. يكون قادراً على استخدام العتاد بفعالية مثل استخدام محدود للذاكرة الرئيسية لتجنب فشل النظام.
2. يكون النظام موثوقا حيث أنه يجب أن:
   1. يتوافر النظام بشكل كبير (availability) ويعمل باستمرار.
   2. تكون دقّة النظام جيّدة بما يكفي من أجل اكتشاف الهجومات مثلاً أن تكون مقاييس الدقة تتجاوز ال90% عند تقييمها على بيانات الاختبار (Testing data).
   3. يتعامل بسماحية مع الأخطاء مثلا حزم مشوّهة، أي بقاء النظام شغّال حتى لو حدث أخطاء خلال التحليل والكشف.
3. أمان النظام والملفات الخاصة به حيث يجب تأمين ملفات الأكواد والنموذج المُحَمَّل من الوصول لغير المحوّلين او من التعديل والعبث بهم.
4. يكون النظام سهل الاستخدام فممكن عرض تعليمات لكيفية استخدامه خصوصاً لتشغيل خدمة الكشف.
5. تتوفر إمكانية التحديث والتغيير فمثلا يجب أن يسمح النظام بتحميل نموذج اخر اذا كان أفضل من النموذج الحالي.

# مخططات تصميم النظام

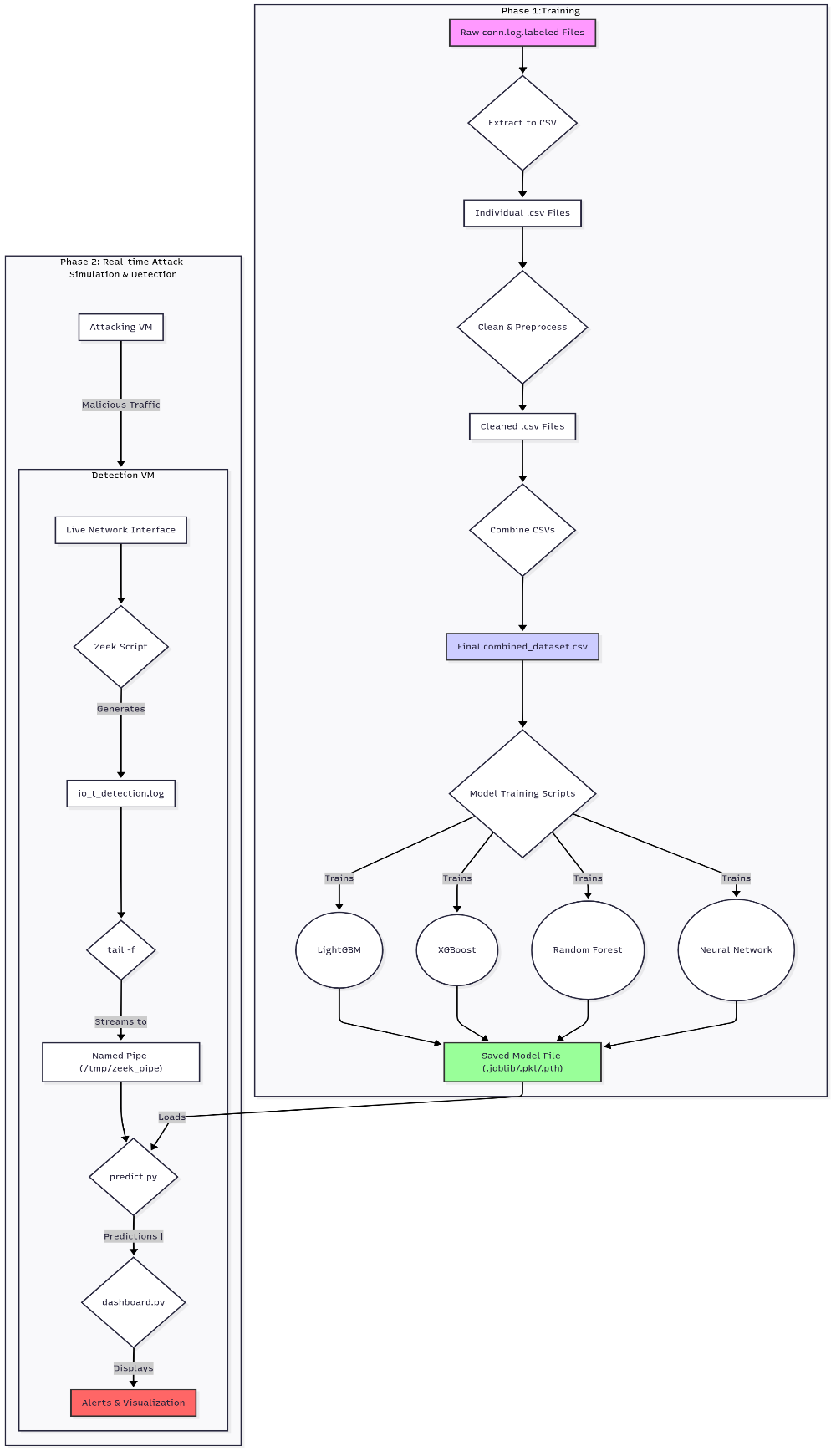


Figure 1 System block diagram

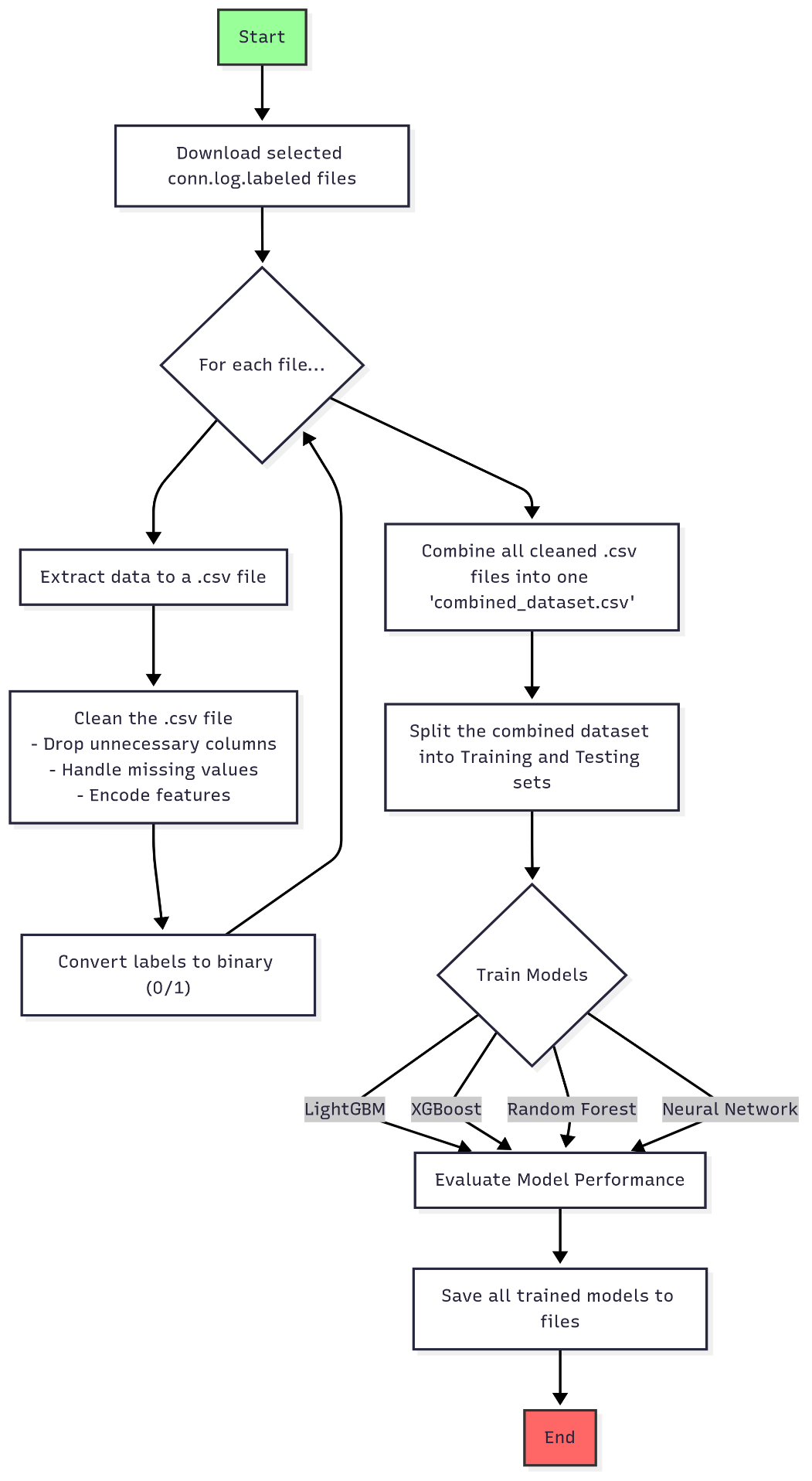


Figure 2 Training flow chart

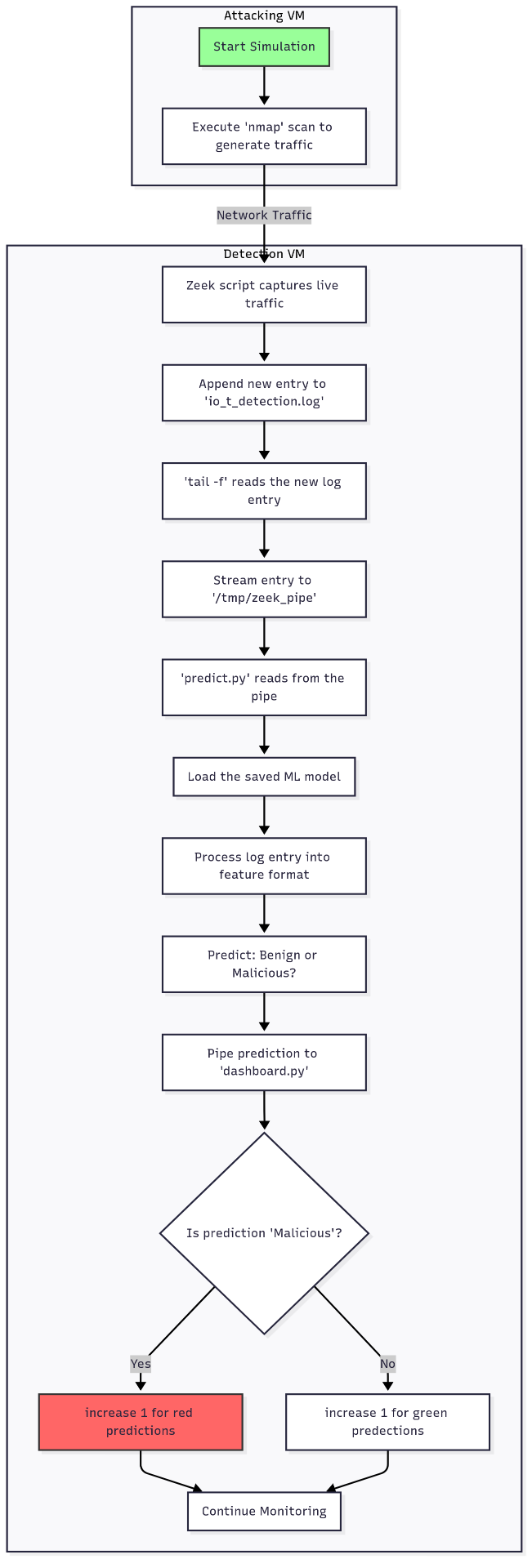


Figure 3 Attacking flow chart

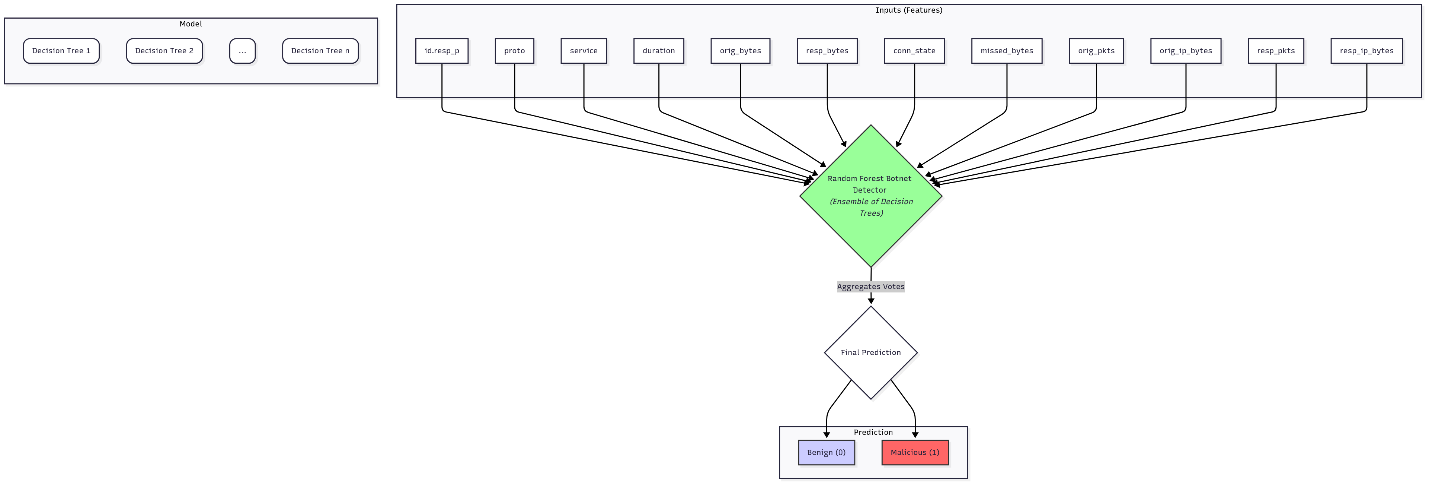


Figure 4 Random forest model diagram

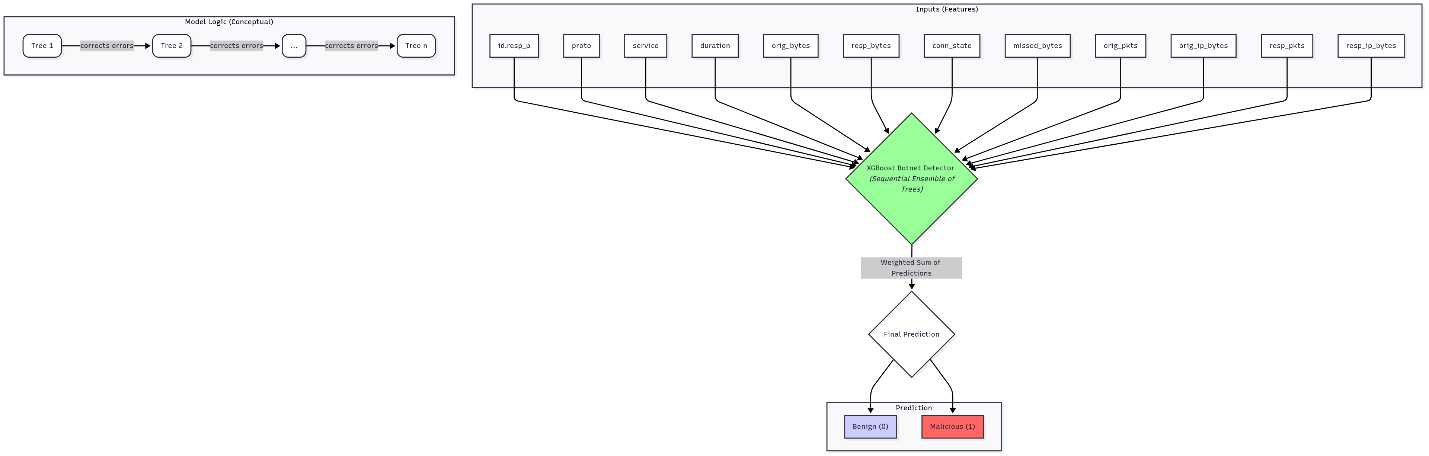


Figure 5 Xgboost model diagram

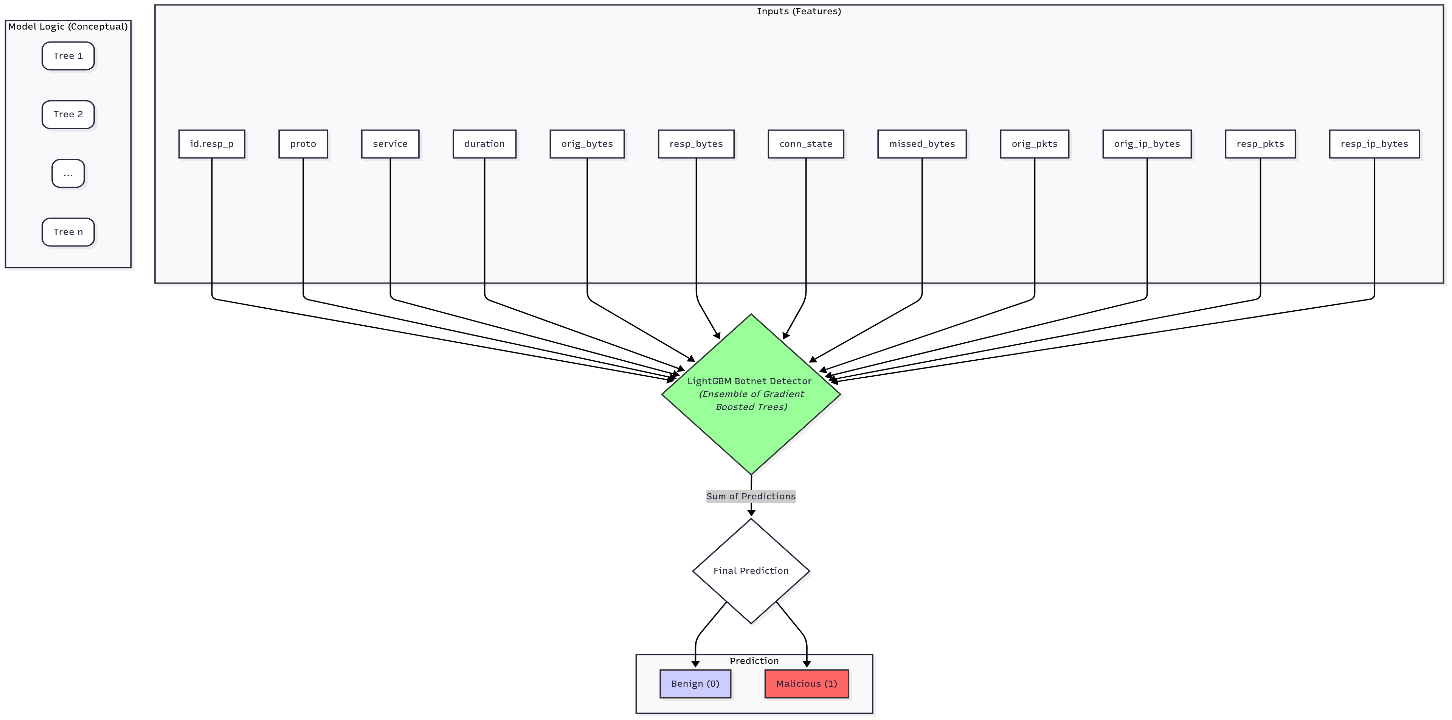


Figure 6 Lightgbm diagram

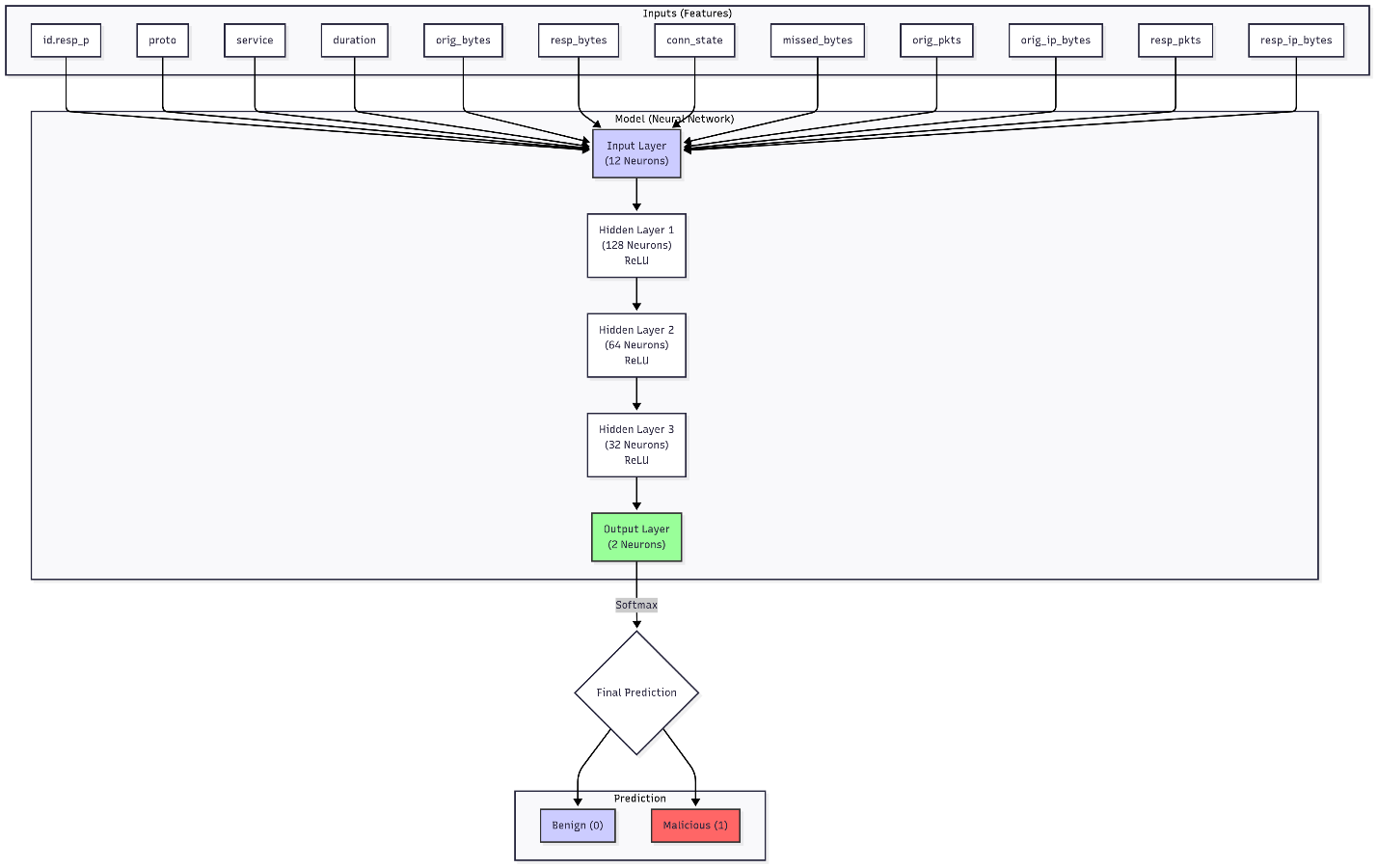


Figure 7 Neural Network model diagram

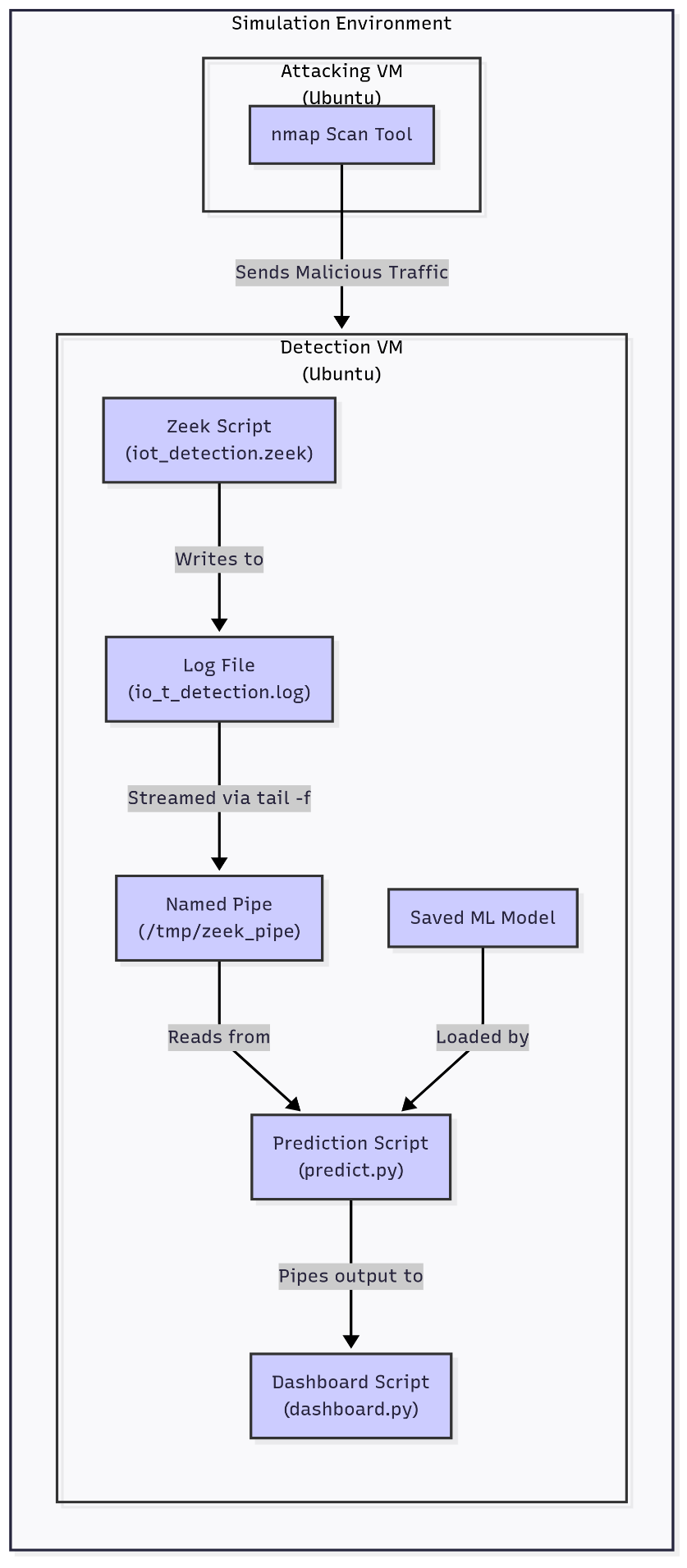


Figure 8 Deployment diagram

# فصل التنفيذ والاختبارات

معمارية النظام المتبعة: تم توضيحها في system block diagram الذي يوضح النظام كاملاً من استخراج البيانات الى محاكاة الهجومات.

تفصيل أجزاء النظام: لدينا العديد من الأجزاء مثل النماذج المدربه مثل XGBoost.pkl, Random\_forest.pkl

وملفات الاكواد مثل predict.py, iot\_detection.zeek, dashboard.py

حيث أن zeek هو لمراقبة الشبكة خيث يقوم بتجميع الحزم المتفرقه الى اتصالات ومن ثم يقوم بانشاء سطر لكل اتصال هذا السطر يمثل السمات وقيمها ويكتب هذا السطر في ملف io\_t\_detection.log

ولدينا الأمر tail -f io\_t\_detection.log > /tmp/zeek\_pipe وهو يقوم بقؤاءة الاسطر الجديده من ملف io\_t\_detection.log وارسالها الى قناة pipe

ولدينا كود التنبؤ predict.py الذي يقوم بتحميل نموذج تعلم الاله المدرب ويفوت بحلقة لقراءة الاسطر الخاصه بالقناة zeek\_pipe ومن أجل كل سطر يقوم بمعالجته المعالجة المطلوبه لبصيح الدخل للمودل مثل البيانات التي تدرب عليها، ومن ثم يتنبأ بماهية الاتصال ان كان benign أو malicious باستخدام المودل المحمّل المدرّب.

ويوجد كود dashboard.py ةهة يقةم بقراءة خرج الpredict.py ويعرض النتائج بشكل اوضح للمستخدم

تم الدمج (integration) بينهم كما التالي:

حزمة بيانات الشبكة ← أداة ←zeek ملف السجلio\_t\_detection.log ← قناة الاتصال ←(Pipe) كود التنبؤ ← طباعة النتيجة ← سكربت واجهة العرض dashboard

Pseudocode:

Start

Read conn.log.labeled files

For each file extract data from it and save it in a .csv format

Then for each .csv dataset file do:

Drop the features { 'Unnamed: 0', 'ts', 'uid', 'local\_orig', 'local\_resp', 'id.orig\_h', 'id.resp\_h', 'id.orig\_p', 'history' }

Do Label encoding: 0 for benign, 1 for malicious

convert 'duration', 'orig\_bytes', 'resp\_bytes' to numeric types

handle nan values for 'duration', 'orig\_bytes', 'resp\_bytes' features:

calculate medians for 'duration', 'orig\_bytes', 'resp\_bytes' from non-S0 connections

    FOR each row in the DataFrame:

      IF 'conn\_state' is 'S0' AND 'duration' is missing:

        'duration' <= 0

      ELSE IF 'conn\_state' is NOT 'S0' AND 'duration' is missing:

        'duration' <= calculated median

    FOR each row in the DataFrame:

      IF 'conn\_state' is 'S0' AND 'orig\_bytes' is missing:

        'orig\_bytes' <= 0

      ELSE IF 'conn\_state' is NOT 'S0' AND 'orig\_bytes ' is missing:

        'orig\_bytes ' <= calculated median

    FOR each row in the DataFrame:

      IF 'conn\_state' is 'S0' AND ‘resp\_bytes ' is missing:

        ' resp\_bytes ' <= 0

      ELSE IF 'conn\_state' is NOT 'S0' AND ' resp\_bytes ' is missing:

        ' resp\_bytes ' <= calculated median

Combine all cleaned\_datasets.csv files to one fille named combined\_dataset.csv

Select X features without the label and Y is for the Label

Split into training and testing

For each model from the models=[random\_forest, xgboost, lightgbm, neural\_network]do:

Put the corresponding hyperparameters

Train on training data

Evaluate on testing data with many metrics like [precesion, recall, f1-score, AUC]

Save the model to [.pkl or .pth pr .joblib format

End

## خطة الاختبارات

Integration tests: إنّ المحاكاة التي تم توظيف المودل بها تشمل اختبارات الدمج لأن المحاكاة تدمج المودل مع اكواد مراقبة البيانات وتحليلها zeek وكود الكشف predit.py وعرض النتائج dashboard مع الهجوم

Working tests: أيضا المحاكاة التي قمنا بها تسمح بعمل اختبارات لمعرفة اذا كان كود الكشف قادر على التنبؤ بشكل صحيح ام لا اعتمادا على النماذج المدرّبة

# Dataset IOT-23

## مقدمة وتوصيف لمجموعة البيانات

تعتمد فعالية أي نموذج من نماذج تعلم الآلة وقابليته للتطبيق في الواقع بشكل أساسي على جودة وملاءمة البيانات التي يتم تدريبه عليها. لهذا المشروع، تم اختيار مجموعة بيانات IoT-23 التي طورها مختبر الستراتوسفير (Stratosphere Laboratory) في الجامعة التقنية التشيكية. تُعد هذه المجموعة من البيانات حديثة وواسعة النطاق، وقد تم إنشاؤها خصيصًا لأغراض البحث والتطوير في مجال أنظمة كشف التسلل (Intrusion Detection Systems) لبيئة إنترنت الأشياء (Internet of Things).

نظرة عامة على مجموعة البيانات وتكوينها:

تتألف مجموعة بيانات IoT-23 من ٢٣ تسجيلاً لحركة مرور الشبكة، يُشار إلى كل منها بـ "سيناريو". تم إنشاؤها عبر تسجيل حركة المرور من أجهزة إنترنت الأشياء الحقيقية ضمن بيئة شبكة مراقبة بين عامي ٢٠١٨ و ٢٠١٩، مما يضمن أن البيانات تعكس سلوكيات الأجهزة والتهديدات المعاصرة.

تنقسم السيناريوهات الـ ٢٣ إلى فئتين رئيسيتين:

٢٠ سيناريو خبيث : (Malicious Scenarios) تم توليد حركة المرور هذه عن طريق إصابة جهاز Raspberry Pi (يعمل كجهاز إنترنت أشياء متعدد الاستخدامات) بعينات متنوعة من البرمجيات الخبيثة. شملت عائلات البرمجيات الخبيثة المستخدمة أنواعًا متنوعة وذات صلة وثيقة بمشهد التهديدات في عالم إنترنت الأشياء، مثل Mirai، وTorii، وGagfyt، وHajime. قامت هذه البرمجيات بتنفيذ مجموعة من الإجراءات الخبيثة، بما في ذلك هجمات الحرمان من الخدمة (Denial-of-Service)، ومسح المنافذ (Port Scanning)، وإنشاء اتصالات القيادة والتحكم (C&C).

٣ سيناريوهات حميدة :(Benign Scenarios) لتوفير خط أساس للنشاط الطبيعي وغير الخبيث، تم تسجيل حركة المرور من ثلاثة أجهزة إنترنت أشياء تجارية مختلفة: مصباح Philips HUE الذكي، ومكبر صوت Amazon Echo المنزلي الذكي، وقفل باب Somfy الذكي. والأهم من ذلك، أن هذه البيانات تم توليدها من أجهزة حقيقية وليست محاكاة، مما يسمح بتحليل سلوك شبكي حقيقي وموثوق.

تنسيق البيانات وعملية التصنيف (Labeling):

لكل سيناريو، تم تسجيل حركة مرور الشبكة الأولية كملف بصيغة .pcap بعد ذلك، تمت معالجة هذه التسجيلات باستخدام إطار عمل Zeek لتحليل الشبكات (المعروف سابقًا باسم Bro) لإنشاء سجلات الاتصال (conn.log) تكمن القيمة الأساسية لمجموعة بيانات IoT-23 في عملية التصنيف الدقيقة التي خضعت لها؛ حيث قام محللون بشريون بالتحقيق يدويًا في حركة المرور ووضع قواعد لتصنيف كل تدفق شبكي (network flow) داخل ملفات conn.log.

كانت النتيجة هي ملفات conn.log.labeled التي تم استخدامها في هذا المشروع، والتي لا تحتوي فقط على ميزات اتصال Zeek القياسية (مثل البروتوكول، مدة الاتصال، حجم البيانات المرسلة، حالة الاتصال)، بل تحتوي أيضًا على عمود تصنيف (label) حاسم يحدد صراحةً ما إذا كانت حركة المرور "حميدة "(Benign) أو نوعًا معينًا من النشاط "الخبيث "(Malicious)". هذا التصنيف المفصل لكل تدفق على حدة يُعد أمرًا ضروريًا لمهام تعلم الآلة الخاضع للإشراف (Supervised Machine Learning).

أسباب اختيار dataset iot23:

كانت مجموعة بيانات IoT-23 الخيار المثالي لهذا المشروع لعدة أسباب رئيسية:

١. الواقعية العالية والملاءمة: إن استخدام أجهزة إنترنت الأشياء الحقيقية لتسجيل حركة المرور الحميدة، واستخدام برمجيات خبيثة حقيقية تعمل على جهاز شبيه بأجهزة إنترنت الأشياء (Raspberry Pi)، يضمن أن البيانات تمثل تمثيلاً عالي الدقة لنشاط شبكات إنترنت الأشياء الحديثة. وهذا يتفوق بشكل كبير على البيانات المولدة عبر المحاكاة البحتة، والتي قد لا تلتقط الفروق الدقيقة في اتصالات الأجهزة في العالم الحقيقي.

٢. سيناريوهات غنية ومتنوعة: لا تقتصر مجموعة البيانات على نوع واحد من التهديدات، بل تحتوي على مجموعة واسعة من أنواع البرمجيات الخبيثة والسلوكيات الهجومية. هذا التنوع ضروري لتدريب نماذج تعلم آلة قوية وقابلة للتعميم يمكنها اكتشاف أكثر من نوع واحد من التهديدات.

٣. تصنيفات دقيقة وقابلة للتحقق: بما أنها مجموعة بيانات مصنفة (labeled)، فهي مناسبة تمامًا لمهام تعلم الآلة الخاضع للإشراف (Supervised Learning). تم إنشاء التصنيفات من خلال تحليل بشري متخصص، مما يوفر حقيقة مرجعية (ground truth) موثوقة لتدريب النماذج، والأهم من ذلك، لتقييم أدائها بثقة عالية. كما أنها توفر تصنيفات خبيثة محددة (مثل PartOfAHorizontalPortScan و DDos).

٤. توفر حركة المرور الحميدة (Benign connections) : من التحديات الشائعة في أبحاث كشف التسلل هو الحصول على بيانات حميدة عالية الجودة. من خلال تضمين تسجيلات مخصصة لحركة المرور الحميدة من أجهزة إنترنت الأشياء الاستهلاكية الشائعة، تُمكّن مجموعة بيانات IoT-23 النماذج من تعلم الفرق بين الأنماط الطبيعية والخبيثة، وهو أمر بالغ الأهمية لتقليل النتائج الإيجابية الخاطئة (false positives) في بيئة التشغيل الفعلية.

٥. تنسيق البيانات وحجمها: يتم توفير البيانات بتنسيق conn.log الخاص بـ Zeek، والذي يعد بحد ذاته شكلاً من أشكال استخلاص الميزات (feature extraction)، حيث يقدم ملخصات عالية المستوى للاتصالات بدلاً من الحزم الأولية. إضافةً الى وجود أحجام كلبرة وكافية من البيانات للتدريب.

لم يتم اختيار جميع السيناريوهات في الداتاست بل اخترنا السيناريوهات او الcaptures التالية:

CTU-IoT-Malware-Capture-1-1, CTU-IoT-Malware-Capture-3-1, CTU-IoT-Malware-Capture-4-1, CTU-IoT-Malware-Capture-5-1, CTU-IoT-Malware-Capture-7-1, CTU-IoT-Malware-Capture-8-1, CTU-IoT-Malware-Capture-9-1, CTU-IoT-Malware-Capture-20-1, CTU-IoT-Malware-Capture-21-1, CTU-IoT-Malware-Capture-34-1, CTU-IoT-Malware-Capture-35-1, CTU-part of IoT-Malware-Capture-36-1, CTU-IoT-Malware-Capture-42-1, CTU-IoT-Malware-Capture-44-1, CTU-IoT-Malware-Capture-49-1, CTU-IoT-Malware-Capture-60-1

حيث أنهم كلهم يحتوون اتصالات حميدة benign connections واتصالات خبيثة Malicious connections من عدة أنواع ولكننا لن نهتم لأنواع الهجومات لأن مسألتنا ستكون فقط التصنيف بين حميد benign وخبيث malicious.

## استخراج البيانات

بعد الحصول على ملفات conn.log.labeled الستة عشر ذهبنا الى مرحلة استخراج البيانات من صيغتها السابقة الى ملفات من صيغة .csv حيث أن الملفات ال16 التي لدينا هي عبارة عن ملفات conn.log.labeled وهو ملف نصي تكون فيه البيانات مفصولة بمسافات جدولية (tab-separated)، ويتم إنشاؤه بواسطة أداة مراقبة الشبكات Zeek ولاستخراج البيانات منه تم تصميم كود بلغة بايثون (python script) وباستخدام المكاتب pandas,numpy إن وظيفة الكود البرمجي هي قراءة ملف بصيغة conn.log.labeled ويقوم بتقسير بنية الملف وترويساته الheaders)) بشكل صحيح أي يقوم بمعرفة السمات (features) والقيم التابعة لكل سمه ويقوم بالعمليات المناسبة ليحفظ قيم كل سمة ضمن نفس العمود الخاص بالسمة ويحفظهم في اطار بيانات (dataframe) ومن ثم تصدير الاطار الى ملف .csv

إن هذه العملية تُعاد من أجل كل ملف conn.log.labeled وبالتالي ينتج لدينا 16 ملف بيانات من صيغة .csv

ومن ثم يجب تطبيف المعالجات عليهم ليكونو جاهزين لاحقاً لعملية التدريب.

## المعالجة المسبقة للبيانات

الان بعدما استخرجنا البيانات من ملفات conn.log.labeled وحصلنا عليهم بصيغة ملفات .csv يجب الان تطبيق معالجة للبيانات قبل أن يتم التدريب عليهم حيث أنه يوجد كود برمجي بلغة بايثون (python script) يقوم من أجل كل ملف بيانات .csv بتنفيذ التالي:

1. يقوم بتحويل المسألة الة مسألة من نوع تصنيف ثنائي(Binary classification) حيث يقوم بتحويلها الة تصنيف الاتصالا الى نوعين فقط هم حميدة (benign) وخبيثة (malicious) وقمنا بترميز الصفوف ب "0" للصف الحميد و "1" لجميع تصنيفات الصفوف الخبيثة.
2. تم حذف جميع السطور -التي تمثل الاتصالات- التي لا تحتوي تصنيف لأنها لن تفيدنا وهي عبارة عن ضجيج noisy data) (
3. تم حذف العديد من السمات (features) وهم:
4. { 'Unnamed: 0', 'ts', 'uid', 'local\_orig', 'local\_resp', 'id.orig\_h', 'id.resp\_h', 'id.orig\_p', 'history' }
5. تم تحويل السمات {‘duration’, ‘orig\_bytes’, ‘resp\_bytes’} من نمط (object) أي سلسلة نصية الى قيم رقمية من نمط float
6. تم معالجة القيم المفقودة داخل الأعمده {‘duration’, ‘orig\_bytes’, ‘resp\_bytes’} عبر حالتين:

6-1. بالنسبة للاتصالات ذات الحالة S0 (والتي تشير إلى محاولة اتصال لم تتم الإجابة عليها)، تم ملء القيم المفقودة بالقيمة 0 لأن الحالة لدينا هنا هي محاولة اتصال لم يتم الرد عليها وبالتالي لم يتم تبادل معطيات ومدة الاتصال ستكون صفر.

6-2. بينما في جميع حالات الاتصال الأخرى، استُخدمنا الوسيط (Median) للعمود المعني لتعويض القيم المفقودة منه، وذلك لتجنب تحييز توزيع البيانات.

حيث أنه عند تطبيق الكودالبرمجي على كل ملف بيانات غير مُعالج ينتج لدينا مجموعة بيانات معالجة يتم حفظها في ملف .csv أيضا وبالتالي سينتج لدينا 16 ملف لمجموعة بيانات معالجة.

## دمج البيانات

بعدما حصلنا على 16 ملف لمجموعات بيانات نضيفة الان يجب دمجهم جمبعا الى مجموعة بيانات واحدة، حيث أنه تم دمج ملفات .csv المعالجة الخاصة بكل سيناريو في مجموعة بيانات رئيسية واحدة تحت اسم ‘combined\_dataset.csv’ حيث أنه طورنا كود برمجي بلغة بايثون يقوم بقراءة جميع ملفات ال.csv ويدمجها الى ملف واحد بشكل عمودي وهكذا أصبح لدينا مجموعة بيانات dataset من 16 سيناريو مختلف وبالتالي ستكون مناسبة جداً لتدريب نماذج والاختبار عليها بسبب احتواءها على سيناريوهات مختلفة وبالتالي انواع مختلفة وقيم سطور connections مختلفة.

# تدريب النماذج (models) واختبارهم

لقد دربنا 4 أنواع مختلفة من المصنفات (الclassifiers) على مجموعة البيانات التي حصلنا عليها وهم:

الغابة العشوائية Random Forestو LightGBMو XGBoostو الشبكة العصبونية Neural Network

حيث أن المصنفات الثلاثة الاولى هم تعلم الي (Machine Learning) والمصنف الرابع هو تعلم الي عميق (Deep Learning)

وذلك لهدف تدريب العديد من النموذج والمقارنة بينهم.

## مصنف الغابة العشوائية (Random Forest)

يُعد مصنّف الغابة العشوائية (Random Forest) طريقة تعلّم تجميعي (Ensemble Learning) قوية وشائعة الاستخدام، حيث أنها تعمل من خلال بناء عدد كبير من أشجار القرار (Decision Trees) في مرحلة التدريب لمعالجة مشكلة الارتباط الكبير ببيانات التدريب (Overfitting) التي تعاني منها أشجار القرار الفردية(Decision tree classifier) .

ينتمي هذا المصنف إلى فئة أوسع من الخوارزميات تُعرف باسم "Bootstrap Aggregating" أو "Bagging"، ولكن مع تحسين جوهري يعزز أداءه بشكل كبير. يعتمد المبدأ الأساسي للغابة العشوائية على إدخال مصدرين مختلفين من العشوائية لتوليد مجموعة من الأشجار غير المترابطة .(Decorrelated)

أولاً، يتم تدريب كل شجرة قرار فردية على عينة فرعية عشوائية مختلفة من بيانات التدريب، يتم سحبها مع الاستبدال وهو ما يُعرف بالعينة الاستنساخية أو Bootstrap Sample))، مما يضمن أن كل شجرة قرار تتعلم من عينة من لبيانات.

ثانياً، أيضا بمصنف الغابات العشوائية يوجد عشوائية حتى على عملية اختيار السمات (features) عند كل انقسام للعقدة (Node Split) . فبدلاً من البحث عن أفضل انقسام بين جميع الميزات (الfeatures) المتاحين، تختار الخوارزمية مجموعة فرعية عشوائية من الميزات وتعتبرها فقط لتقسيم العقدة.

ومن أجل اعطاء التصنيف النهائي، يتم من خلال دمج تنبؤات جميع الأشجار الفردية - عادةً عبر تصويت الأغلبية (Majority Vote) في مهام التصنيف - يقوم النموذج النهائي بتجميع هذه المنماذج المدرّبة الأساسية المتنوعة، ذات الانحياز المنخفض (low bias) والتباين المرتفع (high variance)، في نموذج واحد قوي (Robust) يتميز بانحياز منخفض وتباين منخفض بشكل ملحوظ. هذه الاستراتيجية المزدوجة للعشوائية (للعينات وللسمات) تفصل الارتباط بين الأشجار بفعالية، مما يجعل النموذج التجميعي أقل عرضة للoverfitting وأكثر قدرة على التعميم (generalization)على البيانات الجديدة التي لم يرها من قبل أي (الunseen data).

يالاضافة الى ذلك، توفر الخوارزمية نواتج ثانوية قيمة، مثل مقياس لأهمية الميزات (Feature Importance)، والذي يتم حسابه من خلال ملاحظة مقدار الانخفاض في دقة النموذج عند تبديل قيم ميزة معينة بشكل عشوائي عبر العينات خارج الكيس (Out-of-Bag Samples).

(Breiman, 2001)

### Random Forest Hyperparameters

يعتمد الأداء التنبؤي لنموذج الغابة العشوائية بشكل كبير على إعدادات وسطائه الفائقة (Hyperparameters)، وهي وسطاء يتم تحديدها قبل البدء بعملية التدريب. يعد الضبط الصحيح لهذه المعلمات أمرًا ضروريًا للتحكم في مدى تعقيد النموذج، وإدارة المفاضلة بين الانحياز والتباين (Bias-Variance Tradeoff)، وفي النهاية تحسين قدرته على التعميم على البيانات الجديدة (Generalization to Unseen Data). ومنهم:

* **n\_estimators**: تحدد هذه المعلمة العدد الإجمالي لأشجار القرار التي سيتم إنشاؤها في الغابة. بشكل عام، يؤدي استخدام عدد أكبر من الأشجار إلى نموذج أكثر قوة واستقرارًا، حيث إن تجميع تنبؤات عدد أكبر من الأشجار غير المترابطة يقلل من تباين النموذج الكلي. ومع ذلك، فإن الأداء عادةً ما يظهر تناقصًا بعد عدد معين، بينما تزداد التكلفة الحسابية للتدريب والاختبار بشكل خطي.
* **max\_features**: يتحكم هذا الوسيط في حجم المجموعة الفرعية العشوائية من الميزات التي يتم النظر فيها عند كل انقسام للعقدة داخل الشجرة. تعد من أهم المعلمات لضبط تباين النموذج. إن استخدام قيمة أصغر للمعلمة max\_features يزيد من عشوائية كل شجرة، مما يقلل بدوره من الارتباط بين الأشجار في الغابة، ويؤدي إلى انخفاض أكبر في تباين النموذج الإجمالي. أما القيمة الأكبر فتجعل الأشجار الفردية أكثر تشابهًا، حيث يزداد احتمال اختيار الأشجار لنفس الميزات للتقسيم. غالباً يتم استخدام sqrt(# of all features)​ لمهام التصنيف.
* **max\_depth**: يحدد هذا الوسيط أقصى عمق يمكن أن تنمو إليه كل شجرة قرار فردية(Decision tree). وهي بمثابة طريقة مباشرة للتحكم في مدى تعقيد الوسطاء الأساسية. إذا لم يتم تحديدها، تستمر الأشجار في النمو حتى تصبح جميع العقد الطرفية (Leaf Nodes) نقية أو تحتوي على عدد عينات أقل من min\_samples\_split. يساعد تقييد العمق على تنظيم النموذج (Regularize)؛ فالأشجار الأقل عمقًا تكون أقل تعقيدًا وأقل عرضة للoverfitting لأنها تمنع الانحياز للضجيج أو للنقاط الشاذة الخاصة بعينة التدريب.
* **min\_samples\_split**: يحدد هذا الوسيط الحد الأدنى لعدد العينات التي يجب أن تحتويها العقدة حتى يتم النظر في تقسيمها. من خلال تعيين هذه القيمة إلى رقم أكبر من القيمة الافتراضية، يمكن منع النموذج من إنشاء انقسامات بناءً على عدد قليل جدًا من العينات التي تؤدي الى overfitting، ومنعه من زيادة تعقيده كثيرا.
* **min\_samples\_leaf**: يحدد الحد الأدنى لعدد العينات التي يجب أن تكون موجودة في العقدة الورقة (Leaf Node). لا يعتبر الانقسام صالحًا إلا إذا نتج عنه عقد تحتوي كل منها على هذا العدد من العينات أو اكثر. وبالتالي هي أيضا مهمة لمنع حدوث الoverfitting لأنه سيصبح أي تنبؤ فردي مدعوم بمجموعة كبيرة من العينات مما يجعل تنبؤات النموذج أكثرصحة.
* **criterion**: يحدد الدالة أو التابع المستخدم لقياس جودة الانقسام. بالنسبة لمهام التصنيف، الخياران الأساسيان هما **'gini'** لـ (Gini Impurity) و **'entropy'** لـ (Information Gain). تقيس gini impurity احتمالية التصنيف الخاطئ لعنصر تم اختياره عشوائيًا إذا تم تصنيفه وفقًا لتوزيع الفئات في المجموعة الفرعية. أما information gain فهو مشتق من مفهوم الإنتروبي في نظرية المعلومات. على الرغم من أن كلا المعيارين يخدمان الهدف نفسه، ولكن غالبًا ما يكون gini function هو الخيار الافتراضي لأنه أسرع قليلاً في الحساب.
* **(class weight):** صُمم هذا الوسيط لمعالجة مشكلة مجموعات البيانات غير المتوازنة، وهي حالة شائعة يتجاوز فيها عدد عينات إحدى الصفوق عدد عينات الصفوف الاخرى بشكل كبير. من خلال تحديد وزن للصفوف، تقوم الخوارزمية بتعديل دالة الخسارة (Loss Function) لإعطاء أهمية أكبر للفئة ذات التمثيل الأقل (فئة الأقلية). هذا يعني أن التصنيف الخاطئ لعينة من فئة الأقلية سيتسبب في عقوبة أكبر أثناء التدريب، مما يجبر النموذج على تعلم ميزاتها بشكل أعمق بدلاً من الانحياز للفئة ذات الأغلبية لمجرد تحقيق دقة ظاهرية عالية.
* **(n\_jobs )عدد المهام المتوازية** :هو وسيط يتعلق بالأداء الحسابي ولا يؤثر على النتائج التنبؤية النهائية للنموذج، ولكنها تتحكم في سرعة التدريب. تحدد هذه المعلمة عدد أنوية المعالج (CPU cores) التي يمكن للخوارزمية استخدامها بشكل متزامن لبناء الأشجار في الغابة. ونظرًا لأن كل شجرة يتم تدريبها بشكل مستقل، فإن هذه المهمة قابلة للموازاة (Parallelizable) بدرجة عالية. إن ضبط هذه المعلمة لاستخدام أنوية متعددة يمكن أن يؤدي إلى تقليل كبير في الوقت اللازم لتدريب النموذج، خاصة مع مجموعات البيانات الكبيرة.
* **random\_state )) حالة العشوائية**: تعد هذه المعلمة أساسية لضمان قابلية تكرار النتائج (Reproducibility). تعتمد خوارزمية الغابة العشوائية على عدة عمليات عشوائية، مثل إنشاء عينات استنساخية (Bootstrap Samples) للبيانات واختيار مجموعات فرعية عشوائية من الميزات. من خلال تعيين random\_state إلى قيمة عددية ثابتة، يتم توفير بذرة (Seed) لمولد الأرقام العشوائية. هذا يضمن استخدام نفس تسلسل الأرقام العشوائية في كل مرة يتم فيها تدريب النموذج، مما ينتج عنه نموذج متطابق بأداء متطابق. وهذا أمر حاسم لتصحيح الأخطاء، ومقارنة النماذج المختلفة بشكل عادل، والسماح للباحثين الآخرين بالتحقق من صحة النتائج.

(Breiman، 2001) (Hastie, Tibshirani, & Friedman, 2009)

### Random Forest Implementation

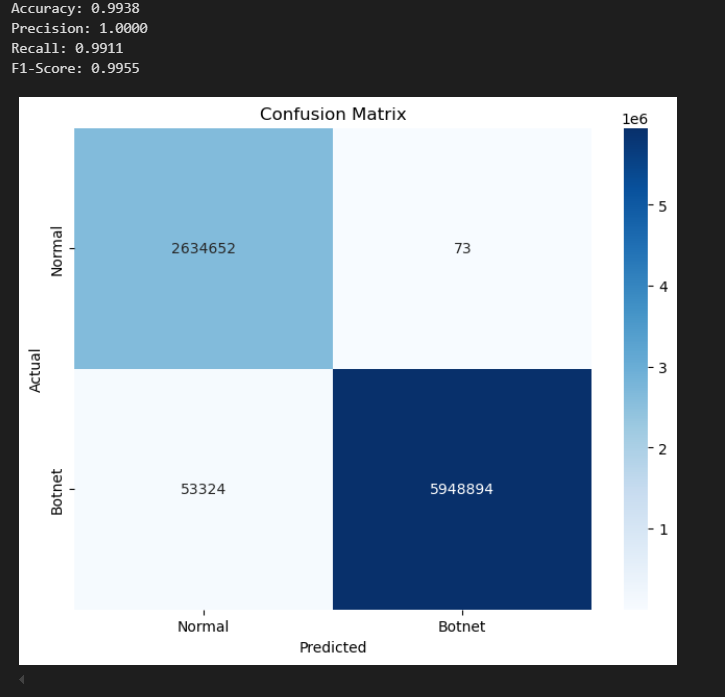
لقد تم تدريب مصنف الغابة العشوائية باستخدام كود بايثون (python code) حيث تم جلب المصنف من مكتبة sklearn عبر التالي: from sklearn.ensemble import RandomForestClassifier

وبالطبع تم قراءة ملف البيانات combined\_dataset.csv وقسمناه الى بيانات للتدريب بنسبة 70% وبيانات للاختبار بنسبة 30%.

ولقد اخترنا الhyperparameters التالية:

* **n\_estimators=200**تم اختيار 200 شجرة لأن 200 شجرة قرار (Decision tree) هو عدد جيد من الأشجار حيث أن كل شجرة تعطي تنبؤ معين وبالتالي 200 شجره عدد كافي لجدوث الاستقرار بأخذ القرار ومنع الoverfitting.
* **max\_depth=20**تم تحديد أقصى عمق للشجرة عند 20 لمنع النموذج من أن يصبح معقدًا بشكل مفرط ومن أجل الا يعاني من ال(Overfitting) على بيانات التدريب.
* **min\_samples\_split=5و min\_samples\_leaf=2** تم ضبط هذه الوسطاء على قيم صغيرة لتنظيم النموذج بشكل طفيف، مما يمنع حدوث انقسامات في العقد بناءً على عدد قليل جدًا من العينات، ويضمن أن كل تنبؤ نهائي يستند إلى عينتي تدريب على الأقل.
* **class\_weight='balanced'**أظهرت مجموعة بيانات IoT-23 عدم توازن بين الفئات، لذلك كان استخدام الوسيط 'balanced' أمر جيد لمواجهة عدم التوازن، حيث يقوم تلقائيًا بتعيين أوزان أعلى للفئة ذات الأقلية، مما يجعل النموذج على إيلاء اهتمام أكبر لها أثناء عملية التدريب وهذا يؤدي الى التوازن بين الصفين.
* **random\_state=42**تم استخدام حالة عشوائية ثابتة لضمان أن تكون عملية تدريب النموذج قابلة للتكرار بالكامل (fully reproducible)، مما يسمح بالحصول على نتائج متسقة وإجراء مقارنات عادلة مع النماذج الأخرى.

بعدما انتهى المصنف من عملية التدريب قمنا باختباره على بيانات الاختبار ةحصلنا على النتائج التالية:



## XGBoost classifier

يُعد مصنف XGBoost الذي هو اختصار ل (eXtreme Gradient Boosting)، مصنف عالي التطور ومُحسَّناً لإطار عمل **التعزيز المتدرج (Gradient Boosting)**. تم تطويره بواسطة Tianqi Chenو Carlos Guestrin، وأصبح خوارزمية أساسية للتعامل مع البيانات المهيكلة أو الجدولية(Tabular data)، حيث يشتهر بأدائه الاستثنائي في كل من الأوساط الأكاديمية والمسابقات التنافسية في مجال تعلم الآلة (machine learning). يُعتبر XGBoost طريقة تعلم تجميعي (Ensemble Learning) حيث يبني سلسلة من أشجار القرار بشكل تسلسلي، حيث يتم تدريب كل شجرة جديدة على تصحيح الأخطاء التي ارتكبتها الأشجار السابقة.

المبدأ الأساسي لـ XGBoost هو في تقنية التعزيز المتدرج. فعلى عكس خوارزمية الغابة العشوائية التي تبني أشجارًا مستقلة بشكل متوازٍ (Bagging)، فإن التعزيز (Boosting) هو عملية تجميعية إضافية (Additive Process). تبدأ الخوارزمية بتدريب مُصنف ضعيف أولي (شجرة قرار بسيطةdecision tree )، ثم تقوم بحساب الأخطاء، أو ما يُعرف بالبواقي (Residuals)، بين تنبؤات هذه الشجرة والقيم الحقيقية. لا يتم تدريب الشجرة التالية على التصنيفات الأصلية، بل على هذه البواقي. من خلال إضافة تنبؤات الشجرة الجديدة إلى تنبؤات الشجرة السابقة (مع ضربها في "معدل التعلم" أو Learning Rate)، يحسن النموذج دقته تدريجيًا. يتم حساب هذه العملية على أنها مسألة أمثلية (Optimization Problem) يستخدم فيها النموذج خوارزمية **الانحدار التدريجي (Gradient Descent)** لتقليل دالة خسارة (Loss Function) محددة مع كل شجرة جديدة تُضاف إلى النموذج التجميعي.

ان الذي يجعل مصنف XGBoost شديد (eXtreme) هو العديد من التحسينات الخوارزمية والتحسينات على مستوى النظام:

1. **هدف تعلم مُنظَّم (Regularized Learning Objective):** إن نماذج التعزيز المتدرج gradient boosting القياسية عرضة للارتباط الكبير ببيانات التدريب (Overfitting). يعالج مصنف XGBoost هذه المشكلة مباشرةً عن طريق تضمين حدي التنظيم L1 (Lasso) و L2 (Ridge) في دالة الهدف الخاصة به. تفرض هذه العملية عقوبة على مدى تعقيد النموذج (مثل عدد الأوراق الطرفية وحجم نتائجها)، مما يساعد على منع الoverfitting وتحسين قدرة النموذج على التعميم على البيانات الجديدة(generalization to unseen data).
2. **إيجاد الانقسام مع مراعاة البيانات المتفرقة (Sparsity-Aware):** في البيانات الواقعية، تعد القيم المفقودة شائعة. بدلاً من معالجتها المسبقة (preprocessing data)، يمتلك XGBoost حل مدمج للتعامل معها. عند كل عقدة في الشجرة، تتعلم الخوارزمية اتجاها افتراضيا للعينات ذات القيم المفقودة، مما يمكنها من تعلم أفضل استراتيجية لتعويض القيم المفقودة من البيانات نفسها بفعالية.
3. **تحسينات النظام من أجل قابلية التوسع (Scalability):** صُمم XGBoost ليكون فعالاً وقابلاً للتوسع. فهو يستخدم خوارزمية مبتكرة لإيجاد الأشجار تستخدم بنية بيانات تسمى الكتلة (Block) لتخزين البيانات في تنسيق عمودي مضغوط. هذه البنية مع أنماط الوصول الواعية لذاكرة التخزين المؤقت (Cache-aware)، تسمح بالحساب المتوازي والفعال لانقسامات الميزات(features splitting) أثناء بناء الشجرة.
4. **التقريب من الدرجة الثانية (Second-Order Approximation):** لتحسين دالة الخسارة، يستخدم XGBoost مفكوك تايلور من الدرجة الثانية second-order Taylor expansion، والذي يدمج معلومات من كل من المشتقة الأولى (التدرج أو Gradient) والمشتقة الثانية (Hessian). يوفر هذا معلومات أكثر حول اتجاه الأمثلية ويمكن أن يؤدي إلى تقارب أسرع.

(Chen & Guestrin, 2016)

### XGBoost Hyperparameters

تكمن قوة خوارزمية XGBoost في مرونتها، والتي يتم التحكم فيها من خلال مجموعة واسعة من المعلمات الفائقة (Hyperparameters). يعد الضبط الصحيح لهذه المعلمات أمرًا حاسمًا لتحقيق التوازن بين الانحيازbias والتباينvariance (Bias-Variance Tradeoff)، والتحكم في مدى تعقيد النموذج، والوصول إلى الأداء الأمثل. يمكن تصنيف هذه المعلمات(الhyperparameters) بشكل عام إلى وسطاء تتحكم في عملية التعزيز (Boosting) الكلية، ووسطاء تتحكم في بنية الأشجار الفردية(decision trees).

**Boosting hyperparameters:**

* **n\_estimators**: تحدد هذه المعلمة العدد الإجمالي للأشجار التسلسلية التي سيتم بناؤها، وهي تعادل عدد جولات التعزيز. يمكن أن يؤدي استخدام عدد أكبر من المُقدِّرات (Estimators) إلى أداء أفضل، ولكن فقط إلى حد معين، قد يبدأ النموذج بعده في المعاناة من فرط التخصيص (الOverfitting). غالبًا ما يتم ضبط هذه المعلمة بالتزامن مع معدل التعلم، وعادةً ما يتم إيجاد قيمتها المثلى باستخدام التوقف المبكر (Early Stopping) على مجموعة بيانات التحقق.
* **learning\_rate :** تقوم هذه المعلمة بتقليص حجم مساهمة كل شجرة جديدة في التنبؤ النهائي للنموذج التجميعي. إن استخدام معدل تعلم أصغر يقلل من تأثير كل شجرة على حدة، مما يتطلب عددًا أكبر من جولات التعزيز (n\_estimators) لبناء نموذج قوي. هذه العملية، التي تُعرف بـ "الانكماش" (Shrinkage)، تجعل عملية التعلم أكثر تحفظًا، مما يساعد على منع الoverfitting. تتراوح القيم لهذا الوسيط بين 0.01 و 0.3.

**Decision trees hyperparameters:**

* **max\_depth**: تحدد هذه المعلمة أقصى عمق يمكن أن تصل إليه شجرة القرار الفردية. يمكن للأشجار الأكثر عمقًا التقاط أنماط أكثر تعقيدًا وتحديدًا في البيانات، ولكنها أيضًا أكثر عرضة لفرط التخصيص(overfitting). يعد تقييد العمق أحد أكثر الطرق شيوعًا للتحكم في مدى تعقيد النموذج ومنع حدوث الoverfitting.
* **min\_child\_weight**: تحدد هذه المعلمة الحد الأدنى لمجموع أوزان العينات (Hessian) المطلوب في العقدة الابن. إذا أدت خطوة تقسيم الشجرة إلى عقدة طرفية ذات مجموع أوزان أقل من min\_child\_weight، فإن عملية البناء ستتوقف عن المزيد من التقسيم. بعبارة أبسط، هي تتحكم في الحد الأدنى لعدد العينات المطلوبة في العقدة الطرفية، وتعتبر معلمة تنظيم (Regularization) قوية تُستخدم لمنع فرط التخصيص على مجموعات صغيرة ومحددة من العينات.
* **gamma (أو min\_split\_loss)**: تحدد هذه المعلمة الحد الأدنى للانخفاض في الخسارة المطلوب لإجراء تقسيم إضافي على عقدة طرفية. لن يتم إجراء الانقسام إلا إذا أدى إلى انخفاض إيجابي في دالة الخسارة أكبر من قيمة gamma. إن استخدام قيمة gamma أعلى يجعل الخوارزمية أكثر تحفظاً، مما يؤدي إلى عدد أقل من الانقسامات وأشجار أبسط، وبالتالي ممكن أن تؤدي الى عدم حدوث overfitting ولكن بالطبع اختبار قيمة كبيره جدا ستؤدي الى underfitting.
* **subsample**: تتحكم هذه المعلمة في جزء بيانات التدريب (الصفوف) الذي يتم أخذ عينات منه عشوائيًا قبل بناء كل شجرة جديدة. يؤدي تعيينها إلى قيمة أقل من 1.0 إلى إدخال العشوائية في عملية التعزيز، مما يساعد على منع فرط التخصيص من خلال ضمان بناء كل شجرة على مجموعة فرعية مختلفة قليلاً من البيانات.
* **colsample\_bytree**: تشبه subsample، ولكنها خاصة بالميزات (الأعمدة). تحدد هذه المعلمة جزء الميزات الذي سيتم أخذ عينات منه عشوائيًا عند بناء كل شجرة. تعد هذه طريقة فعالة أخرى لمنع فرط التخصيص ويمكنها أيضًا تسريع عملية التدريب بسبب عدم أخذ جميع السمات في كل مرة تدريب.

**معلمات التنظيم (Regularization Parameters)**

* **reg\_alpha (تنظيم L1)** و **reg\_lambda (تنظيم L2)**: تتوافق هاتان المعلمتان مع حدي التنظيم L1 (Lasso) و L2 (Ridge) على أوزان العقد الطرفية. تؤدي زيادة هذه القيم إلى جعل النموذج أكثر تحفظًا عن طريق فرض عقوبة على قيم الأوزان الكبيرة، مما يساعد على جعل التنبؤات النهائية أكثر سلاسة ومنع فرط التخصيص.

(Chen & Guestrin, 2016)

### XGBoost Implementation

لقد تم تدريب مصنف xgbosot باستخدام كود بايثون (python code) حيث تم جلب المصنف من مكتبة xgboost عبر التالي: from xgboost import XGBClassifier

وبالطبع تم قراءة ملف البيانات combined\_dataset.csv وقسمناه الى بيانات للتدريب بنسبة60% وبيانات للاختبار بنسبة 40%.

ولقد اخترنا الhyperparameters التالية:

* **objective='binary:logistic'**

هذا الوسيط مهم لأنه يخبر خوارزمية XGBoost بأن المسألة لدينا هي مسألة تصنيف ثنائي (أي أن هناك نتيجتين محتملتين فقط وهم كما نعلم 0 أي benign و1 أي malicious). و logistic يحدد أن النموذج يجب أن يُخرج النتائج على شكل احتمالات (قيمة بين 0 و 1) بدلاً من مجرد تصنيف حاسم للتوقع (0 أو 1).

* **eval\_metric='logloss'**

تحدد هذه المعلمة المقياس الذي سيتم استخدامه لتقييم أداء النموذج أثناء التدريب (من أجل التوقف المبكر). مقياس الخسارة اللوغاريتمية (Logloss) يقيس مدى جودة التنبؤات الاحتمالية للنموذج، حيث يفرض عقوبة كبيرة على التنبؤات التي تكون واثقة وخاطئة في نفس الوقت. كلما انخفضت قيمة logloss، كان الأداء أفضل. وهو مناسب للنموذج الذي ينتج احتمالات.

* **use\_label\_encoder=False**

**وهي** معلمة تقوم بعمل ترميز للتصنيفات النهائية ووضعناخها false في مسألتنا لأننا قد قمنا بترميز الصفوف سابقا الى 0 و1 .

* **random\_state=42**

تم استخدام حالة عشوائية ثابتة لضمان أن تكون عملية تدريب النموذج قابلة للتكرار بالكامل (fully reproducible)، مما يسمح بالحصول على نتائج متسقة وإجراء مقارنات عادلة مع النماذج الأخرى.

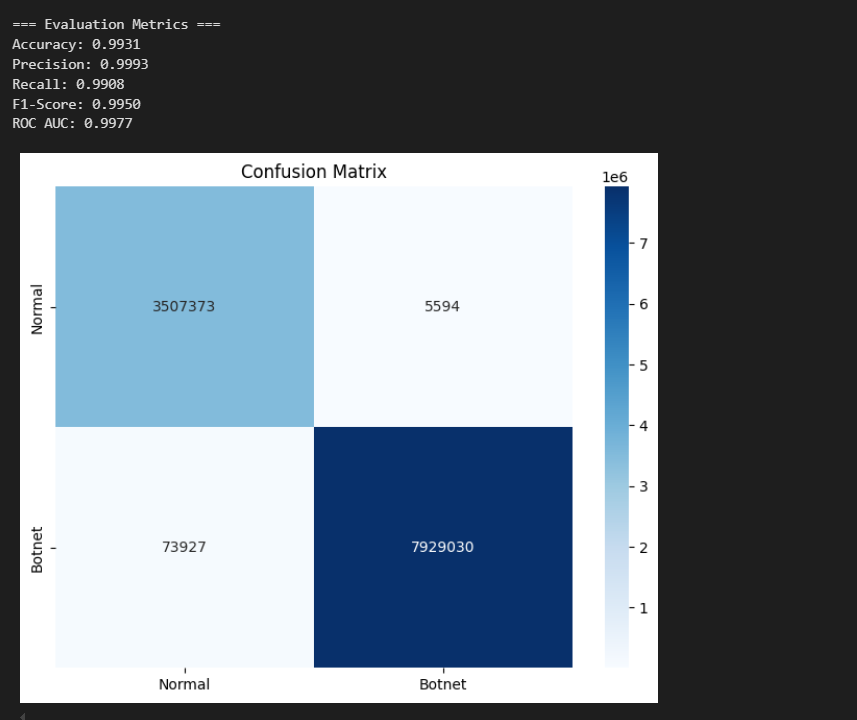
* **n\_jobs=-1**

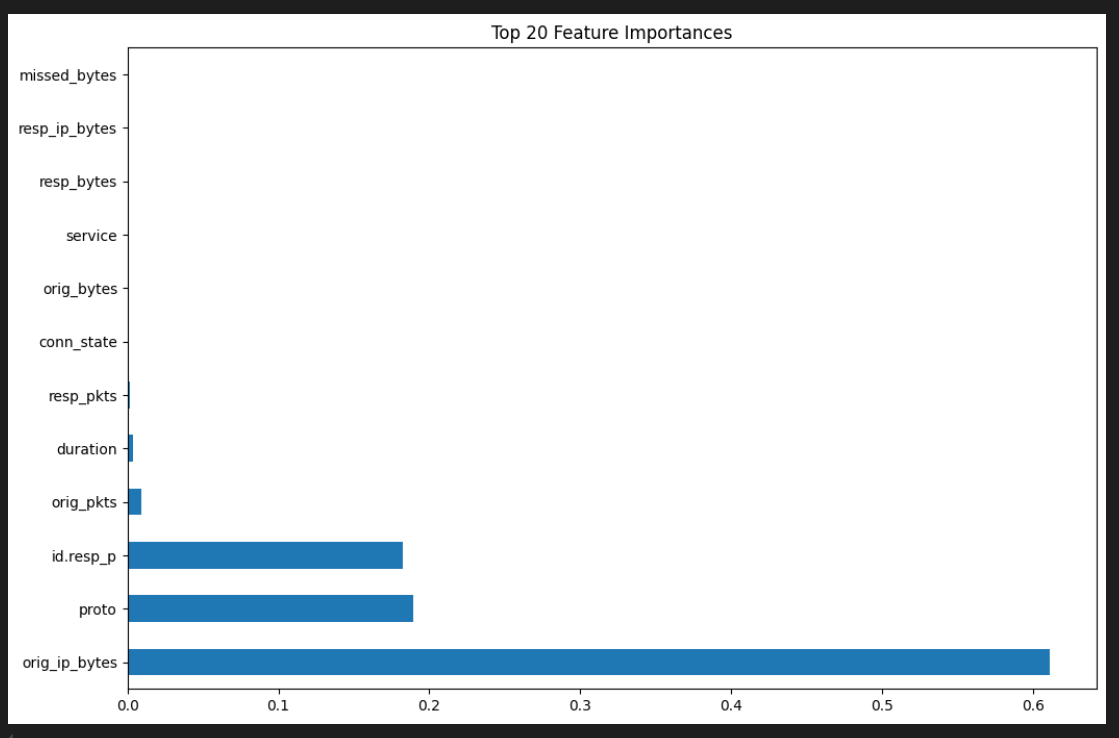
**و**هي معلمة لضبط الأداء. تخبر XGBoost باستخدام جميع أنوية المعالج (CPU) المتاحة على جهازك لتشغيل أجزاء من عملية التدريب بشكل متوازٍ، وهي ضروري لأن مجموعة بيانات IoT-23 كبيرة جدًا، وقد يكون التدريب بطيئًا. فاستخدام جميع الأنوية المتاحة سيقلل بشكل كبير من وقت التدريب، مما يجعل التدريب أسرع وأكثر كفاءة.

* **tree\_method='hist’**

تحدد الخوارزمية المستخدمة لبناء الأشجار. الطريقة الافتراضية (exact) تأخذ في الاعتبار كل نقطة انقسام محتملة للميزات. أما طريقة hist فهي خوارزمية تقريبية أسرع بكثير تقوم بتجميع الميزات في (bins) منفصلة (مثل المدرج التكراري histogram) وتجد أفضل الانقسامات بينهم.

بعدما انتهى المصنف من عملية التدريب قمنا باختباره على بيانات الاختبار حصلنا على النتائج التالية:





## Lightgbm classifier

LightGBM أو (Light Gradient Boosting Machine)، هو إطار عمل للتعزيز المتدرج (Gradient Boosting) مفتوح المصدر وعالي الأداء، تم تطويره بواسطة شركة مايكروسوفت.(Microsoft company) على الرغم من أنه يعتمد على نفس المبادئ التسلسلية لتصحيح الأخطاء الموجودة في طرق التعزيز المتدرج الأخرى مثل XGBoost، إلا أن LightGBM قد تم تصميمه خصيصًا لمعالجة اختناقات الأداء في تلك الخوارزميات، مع إعطاء الأولوية لسرعة التدريب وكفاءة استخدام الذاكرة دون انخفاض الدقة بشكل كبير. لتحقيق ذلك، يقدم LightGBM العديد من التقنيات المبتكرة التي تغير بشكل أساسي كيفية بناء أشجار القرار.

أهم ابتكارين في هذه الخوارزمية هما تقنية أخذ العينات أحادية الجانب القائمة على التدرج **(GOSS Gradient-based One-Side Sampling)** وتقنية تجميع الميزات الحصرية **(EFB Exclusive Feature Bundling)** فمن خلال تقنية GOSS، تقوم الخوارزمية بأخذ عينات من البيانات المستخدمة لبناء كل شجرة بذكاء؛ فبدلاً من استخدام جميع نقاط البيانات، تحتفظ الخوارزمية بجميع العينات ذات التدرجات الكبيرة (أي تلك التي لم يتم تدريبها بشكل جيد) وتقوم بأخذ عينات عشوائية من العينات ذات التدرجات الصغيرة. هذا النهج يركز عملية التدريب على الأخطاء الأكثر إفادة، مما يقلل بشكل كبير من حجم مجموعة البيانات في كل تكرار. أما مع تقنية EFB، فيقوم LightGBM بتقليل عدد الميزات عن طريق تجميع الميزات المتنافية — وهي تلك التي نادرًا ما تأخذ قيمًا غير صفرية في نفس الوقت (مثل الميزات الناتجة عن الترميز الأحادي "one-hot encoding") — في ميزة واحدة، مما يسرّع الحسابات بشكل كبير.

اضافة الى ذلك، يتخلى LightGBM عن استراتيجية نمو الشجرة التقليدية القائمة على المستوى (level-wise) لصالح نهج قائم على الورقة (leaf-wise)، حيث يقوم دائمًا بتقسيم الورقة التي ستحقق أكبر انخفاض في الخسارة. هذا يسمح للنموذج بالتقارب بشكل أسرع بكثير، على الرغم من أنه قد يكون أكثر عرضة لفرط التخصيص (Overfitting) على مجموعات البيانات الصغيرة. هذه التحسينات، جنبًا إلى جنب مع خوارزمية عالية الكفاءة قائمة على المدرج التكراري (histogram-based) لإيجاد نقاط الانقسام، تجعل من LightGBM خيارًا سريعًا وفعالًا للغاية من حيث استهلاك الذاكرة للمهام واسعة النطاق في تعلم الآلة.

(Ke, 2017)

### LighGBM Hyperparameters

تعتمد الكفاءة والقدرة التنبؤية لنموذج LightGBM على مجموعة من المعلمات الفائقة (Hyperparameters) التي تضبط آلياته الأساسية، خاصة استراتيجياته المتعلقة بنمو الشجرة القائم على الورقة (leaf-wise) وأخذ العينات من البيانات. وبالتالي يجب ضبط هذه الوسطاء لزيادة الدقة إلى أقصى حد مع منع فرط التخصيص (Overfitting).

**معلمات التحكم في تعقيد الشجرة(decision tree complexity)**

* num\_leaves: تعد هذه المعلمة الأكثر أهمية للتحكم في مدى تعقيد الأشجار الفردية. فهي تحدد العدد الأقصى للأوراق الطرفية في الشجرة الواحدة. على عكس max\_depth التي تقيد الشجرة عموديا، تتحكم num\_leaves في العدد الإجمالي للعقد الطرفية. ونظرًا لأن LightGBM ينمو الأشجار بطريقة leaf-wise، فإن استخدام قيمة أعلى لـ num\_leaves يسمح للنموذج بتكوين حدود قرار أكثر تعقيدًا وتفصيلاً. ومع ذلك، فإن القيمة العالية جدًا يمكن أن تؤدي بسهولة إلى فرط التخصيص. وغالبا يُنصح بإبقاء قيمة num\_leaves أقل من 2^(max\_depth).
* :max\_depthتحدد هذه المعلمة أقصى عمق يمكن أن تنمو إليه الشجرة. على الرغم من أن LightGBM ينمو بطريقة leaf-wise، إلا أن تحديد max\_depth يعمل كإجراء وقائي لمنع الأشجار من أن تصبح عميقة بشكل مفرط، وهو سبب رئيسي للoverfitting. وبالتالي هو لمنع التعقيد الكبير للشجرة.
* min\_child\_samples)**أو :(**min\_data\_in\_leaf تحدد هذه المعلمة الحد الأدنى لعدد نقاط البيانات التي يجب أن تكون موجودة في العقدة الطرفية. وهي معلمة تنظيم (Regularization) حاسمة لنمو الشجرة القائم على الورقة. إن تعيينها على قيمة كبيرة يمنع النموذج من إنشاء انقسامات مدعومة فقط بعدد قليل من نقاط البيانات، وبالتالي تجنب التقاط الضوضاء الخاصة بمجموعة التدريب وتحسين قدرة النموذج على التعميم ومنع الoverfitting.

**معلمات عملية التعزيز (Boosting)**

* :n\_estimatorsتحدد هذه المعلمة العدد الإجمالي لجولات التعزيز، أو ما يعادل عدد الأشجار التي سيتم بناؤها بشكل تسلسلي. يؤدي استخدام عدد أكبر من الأشجار بشكل عام إلى تحسين أداء النموذج، ولكن هذا التأثير يتضاءل بمرور الوقت ويزيد من خطر فرط التخصيص والتكلفة الحسابية. عادةً ما يتم تحديد قيمتها المثلى باستخدام آلية التوقف المبكر (Early Stopping).
* learning\_rate: هذه المعلمة، المعروفة أيضًا باسم "الانكماش" (Shrinkage)، تقوم بتقليص حجم مساهمة كل شجرة في التنبؤ النهائي. إن استخدام learning\_rate أصغر يجعل عملية التعزيز أكثر تحفظًا، مما يتطلب عددًا أكبر من n\_estimators لتحقيق أداء جيد، ولكنه يؤدي في النهاية إلى نموذج أكثر قوة وقابلية للتعميم.

**معلمات أخذ العينات والعشوائية**

* :feature\_fraction (or colsample\_bytree)تحدد هذه المعلمة جزء الميزات الذي سيتم النظر فيه عشوائيًا لكل شجرة. على سبيل المثال، تعني القيمة 0.8 أن LightGBM سيختار 80% من الميزات عشوائيًا قبل بناء كل شجرة. هذا يُدخل عشوائية تساعد على فك الارتباط بين الأشجار وتقليل فرط التخصيص لأن كل شجرة اصبحت مختلفة عن غيرها نوعا ما.
* :bagging\_fraction **(**subsample**)** تتحكم هذه المعلمة في جزء البيانات الذي سيتم استخدامه لتدريب كل شجرة. حيث يتم اختيار جزء من البيانات عشوائيًا بدون إرجاع. هذه التقنية، المعروفة باسم (Bagging)، يمكن أن تسرّع التدريب وهي طريقة فعالة أخرى لمنع فرط التخصيص. لكي تكون هذه المعلمة نشطة، يجب تعيين bagging\_freq إلى عدد صحيح موجب.
* bagging\_freq: تحدد هذه المعلمة عدد مرات تكرار عملية الbagging. القيمة k تعني أن عملية الbagging ستتم كل k تكرار. القيمة 0 تعطل هذه العملية.

**معلمات التنظيم (Regularization)**

* **:lambda\_l1 (L1 regularization) and lambda\_l2 (L2 regularization)** تطبق هاتان المعلمتان تنظيم L1 و L2 على أوزان الأوراق الطرفية على التوالي. تؤدي زيادة هذه القيم إلى إضافة عقوبة على تعقيد النموذج، مما يفرض أن تكون الأوزان أصغر ويجعل النموذج أكثر تحفظًا، وهي تقنية لمنع فرط التخصيص (overfitting).

(Ke، 2017) (Microsoft and LightGBM Contributors, 2025)

### lightgbm Implementation

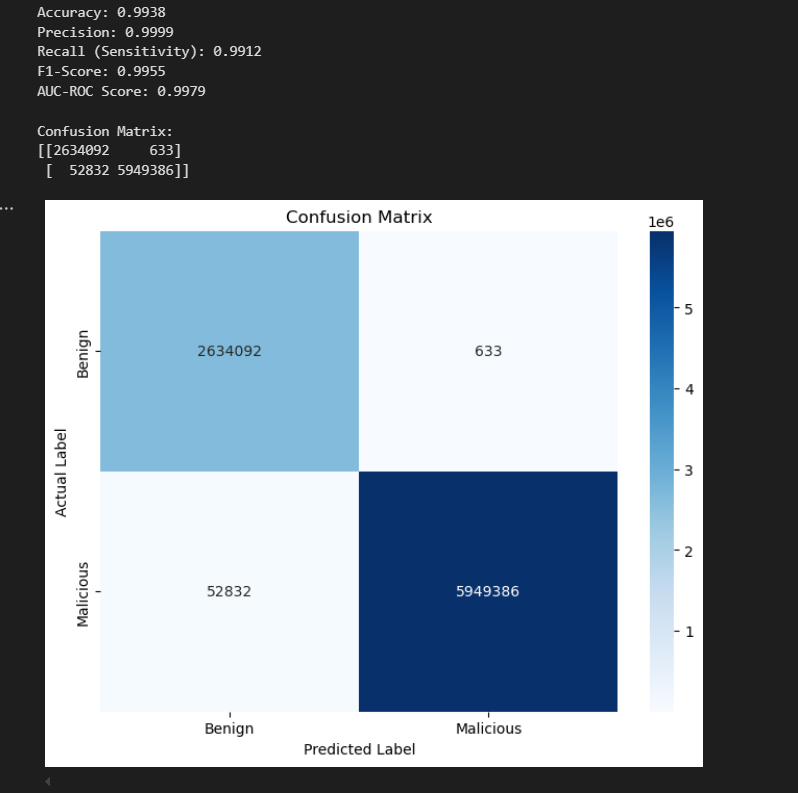
لقد تم تدريب مصنف lightgbm باستخدام كود بايثون (python code) حيث تم جلب المصنف من مكتبة lightgbm عبر التالي: import lightgbm as lgb

وبالطبع تم قراءة ملف البيانات combined\_dataset.csv وقسمناه الى بيانات للتدريب بنسبة70% وبيانات للاختبار بنسبة 30%.

ولقد اخترنا الhyperparameters التالية:س

* objective='binary’: تم تحديد هدف النموذج على أنه تصنيف ثنائي، وهو الإعداد الصحيح للمهمة الحالية التي تتطلب التمييز بين فئتين فقط: حركة المرور "الحميدة" (Benign) وحركة المرور "الخبيثة" (Malicious).
* metric='auc’: تم اختيار مقياس "المساحة تحت منحنى ROC" (Area Under the ROC Curve) لتقييم أداء النموذج. يُعد هذا المقياس خيار ممتاز لمجموعات البيانات غير المتوازنة(imbalanced data)، لأنه يقيس قدرة النموذج على الفصل والتمييز بين الفئات المختلفة، بدلاً من الاعتماد على الدقة الكلية accuracy التي قد تكون مضللة.
* n\_estimators=1000 و learning\_rate=0.05: تم ضبط هاتين المعلمتين معًا. حيث أن استخدام معدل تعلم منخفض (0.05) يجعل عملية التعلم أكثر تحفظ وتدرج، مما يقلل من خطر فرط التخصيص. وللتعويض عن بطء التعلم هذا، تم تحديد عدد كبير من الأشجار (1000) لضمان وصول النموذج إلى التقارب الأمثل وبناء نموذج قوي.
* num\_leaves=31 و max\_depth=-1: تم استخدام القيمة الافتراضية لعدد الأوراق (31)، وهي نقطة انطلاق جيدة للتحكم في تعقيد الشجرة. أما max\_depth=-1 فتعني عدم فرض أي قيود على عمق الشجرة، مما يجعل num\_leaves هي المتحكم الرئيسي في حجم الأشجار.
* random\_state=42 و n\_jobs=-1: تم تحديد حالة عشوائية ثابتة لضمان قابلية تكرار النتائج بشكل كامل، وهو أمر ضروري للمقارنات العلمية الدقيقة. كما تم استخدام جميع أنوية المعالج (n\_jobs=-1) لتسريع عملية التدريب بشكل كبير، وهو أمر حيوي نظرًا للحجم الكبير لمجموعة البيانات.
* colsample\_bytree=0.8 و subsample=0.8: تم تطبيق هاتين التقنيتين كشكل من أشكال التنظيم (Regularization) العشوائي. حيث يتم استخدام 80% من الميزات (الأعمدة) و 80% من البيانات عشوائيًا لبناء كل شجرة. هذا يُدخل العشوائية في عملية التدريب، مما يقلل من الارتباط بين الأشجار ويحسن من قدرة النموذج على التعميم.
* reg\_alpha=0.1 و reg\_lambda=0.1: تم تطبيق تنظيمي L1 و L2 بقيم صغيرة (0.1). تضيف هذه المعلمات عقوبة طفيفة على تعقيد النموذج، مما يساعد على منع فرط التخصيص دون تقييد قدرة النموذج على التعلم بشكل مفرط.
* scale\_pos\_weight=scale\_pos\_weight\_value: هذه المعلمة حاسمة لمعالجة مشكلة عدم توازن الفئات في مجموعة بيانات IoT-23. من خلال تعيين وزن أعلى للفئة الموجبة (الخبيثةmalicious )، يتم إجبار النموذج على اعطاء اهتمام أكبر للعينات الmalicious، مما يحسن بشكل كبير من قدرته على اكتشاف التهديدات وتقليل عدد السلبيات الخاطئة (False Negatives).

بعدما انتهى المصنف من عملية التدريب قمنا باختباره على بيانات الاختبار حصلنا على النتائج التالية:



## الشبكة العصبونية Neiral Network

# Bibliography

Breiman, L. (2001). Random forests. *Machine Learning*, 5-32.