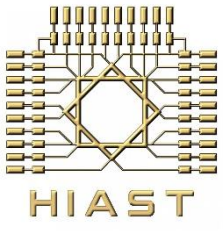
Syrian Arab Republic

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..... 4

هدف المشروع..... 5

المتطلبات الوظيفية 5

المتطلبات غير الوظيفية..... 6

فصل التنفيذ والاختبارات 7

معمارية النظام المتبعة..... 7

تفصيل أجزاء النظام 7

Pseudocode..... 7

خطة الاختبارات..... 9

النموذج المقترح 10

Dataset IOT-23..... 12

مقدمة وتوصيف لمجموعة البيانات 13

استخراج البيانات 15

المعالجة المسبقة للبيانات 15

دمج البيانات 16

معايير القياس والأدوات المستخدمة..... 16

معايير القياس (Evaluation Metrics)..... 16

مصفوفة الالتباس (Confusion Matrix) 16

الدقة (Accuracy)..... 17

الضبط (Precision)..... 17

الاستدعاء (Recall) أو الحساسية (Sensitivity) 18

النوعية (Specificity) 18

مقياس (F1-Score) 18

تدريب النماذج (models) واختبارهم 19

مصنف الغابة العشوائية (Random Forest) 19

Random Forest Hyperparameters 20

Random Forest Implementation .. 22

XGBoost classifier..... 24

25 .....XGBoost Hyperparameters
26 .....XGBoost Implementation
29 ..... Lightgbm classifier
30 .....LighGBM Hyperparameters
31 ..... lightgbm Implementation
33 ..... Neiral Network الشبكة العصبونية
34 Neural Network hyperparameters
35 . Neural Network Implementation
38 ..... (Simulation Environment for Deployment) بيئة المحاكاة والنشر
39 (Virtual Environment and Network Setup) إعداد البيئة الافتراضية والشبكة
39 (Detection Machine: System Architecture) تكوين جهاز الكشف: البنية النظامية
41 (Attacking Machine: Threat Simulation) تكوين جهاز الهجوم: محاكاة التهديدات
42 ..... تولید حركة المرور الخبيثة
43 ..... Bibliography

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 :

1-1. تدريب النموذج على بيانات التدريب (Training Data).

1-2. اختبار النموذج على بيانات الاختبار (Testing Data).

2- معالجة البيانات بشكل مسبق (preprocessing data) ومنها:

2-1. معالجة القيم المفقودة (NaN) مثل اسناد قيمة الصفر اليها او اسناد قيمة الوسيط (median) اليها.

2-2. ترميز (encoding) قيم السمات من قيم فئوية (categorical values) الى قيم رقمية.

2-3. حذف (drop) السمات غير المفيدة.

2-4. تحويل أنماط (types) السمات الى الأنماط المناسبة لطبيعة قيمها.

2-5. 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:

3-1. التدريب على نماذج مثل الغابة العشوائية والشبكات العصبونية (Random Forest, Neural Network, etc).

3-2. قادراً على تقييم كل نموذج باستخدام المقاييس المختلفة مثل: الدقة (Precision) والاسترجاع (Recall) و-F1 score والمساحة تحت المنحني (AUC: Area Under Curve) ومصفوفة الارتباك (Confusion Matrix).

3-3. 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 :

4-1. تحميل النوذج المُدرّب من أجل استحدامه لكشف الهجومات.

4-2. 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.

4-3. التنبّؤ بماهية الاتصال ان كان سليم او هجوم اعتماداً على البيانات التي التقطها وعالجها.

4-4. توليد مخرجات تنبؤ النموذج المحمل بطريقة واضحة.

المتطلبات غير الوظيفية

1- أداء جيد للنظام حيث يجب أن:

1-1. 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.

1-2. يكون قادراً على التعامل مع حركة مرور كبيرة ضمن الشبكة ومعالجتها من دون تأخير أي أنه يجب أن يتحمل الضغط الناجم عن الاتصالات.

1-3. يكون قادراً على استخدام العتاد بفعالية مثل استخدام محدود للذاكرة الرئيسية لتجنب فشل النظام.

2- يكون النظام موثوقا حيث أنه يجب أن:

2-1. يتوافر النظام بشكل كبير (availability) ويعمل باستمرار.

2-2. تكون دقّة النظام جيّدة بما يكفي من أجل اكتشاف الهجومات مثلاً أن تكون مقاييس الدقة تتجاوز ال90% عند تقييمها على بيانات الاختبار (Testing data).

2-3. 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.

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 benignأو 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:

Star

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

النموذج المقترح

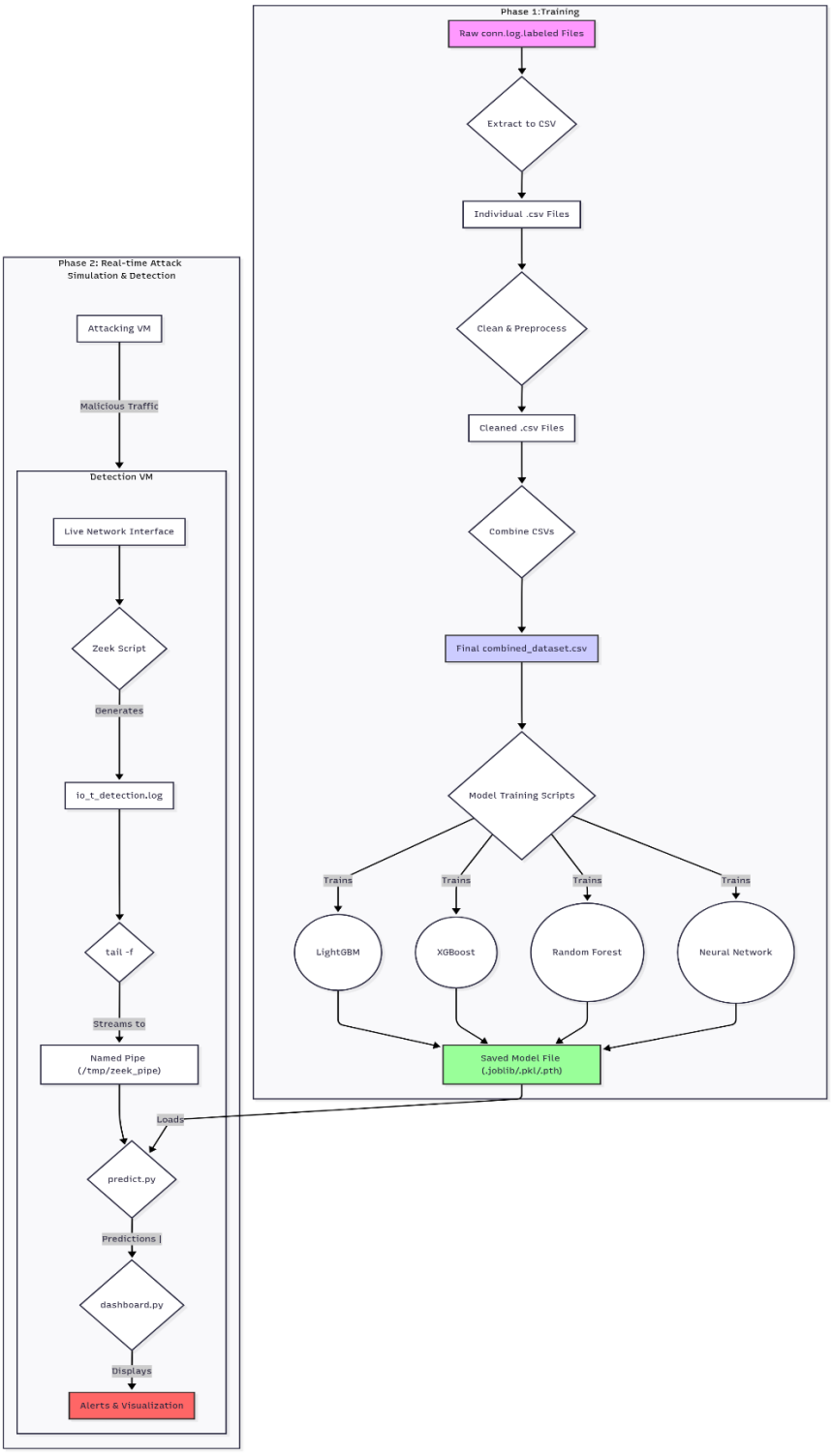


Figure 1 System block diagram

يوضح هذا المخطط سير العمل الشامل لمشروعنا هذا حيث يمكن تقسيمه الى مرحلتين أساسيتين هم مرحلة التدريب ومرحلة المحاكاة

حيث فب المرحلة الأولى: قمنا بتنزيل ملفات conn.log.labeled ثم طبقنا عليهم كود بايثةن لاستخراج ملفات الداتاست منهم بصيغة csv. ثم طبقنا كود بايثون على كل ملف csv. من أجل معالجة البيانات قبل ادخالها للتدريب والاختبار وعندها نتج لدينا 16 ملف لبيانات معالجه فقمنا بدمجهم جميعهم الى ملف بيانات كلي تحت اسم 'combined_dataset.csv' ثم قمنا بتدريب 4 أنواع من المصنفات منهم 3 تعلم الي (machine learning) ومصنف تعلم الي عميق (deep learning) وهم:

Random forest, lightgbm, xgboost, neural network

وبالطبع قمنا بحفظ المودل المدرب لكل مصنف منهم.

وفي المرحلة الثانية: شغلنا 2 ubuntu virtual machines واحده للكشف وواحدة للهجوم حيث الة الهجوم جرت هجوما على الة الكشف التي تحتوي النموذج المدرب ليتنبأ بماهية الاتصالات القادمة من الattacking machine ونرى فعاليته بكشف الهجومات في الوقت الحقيقي.

Dataset IOT-23

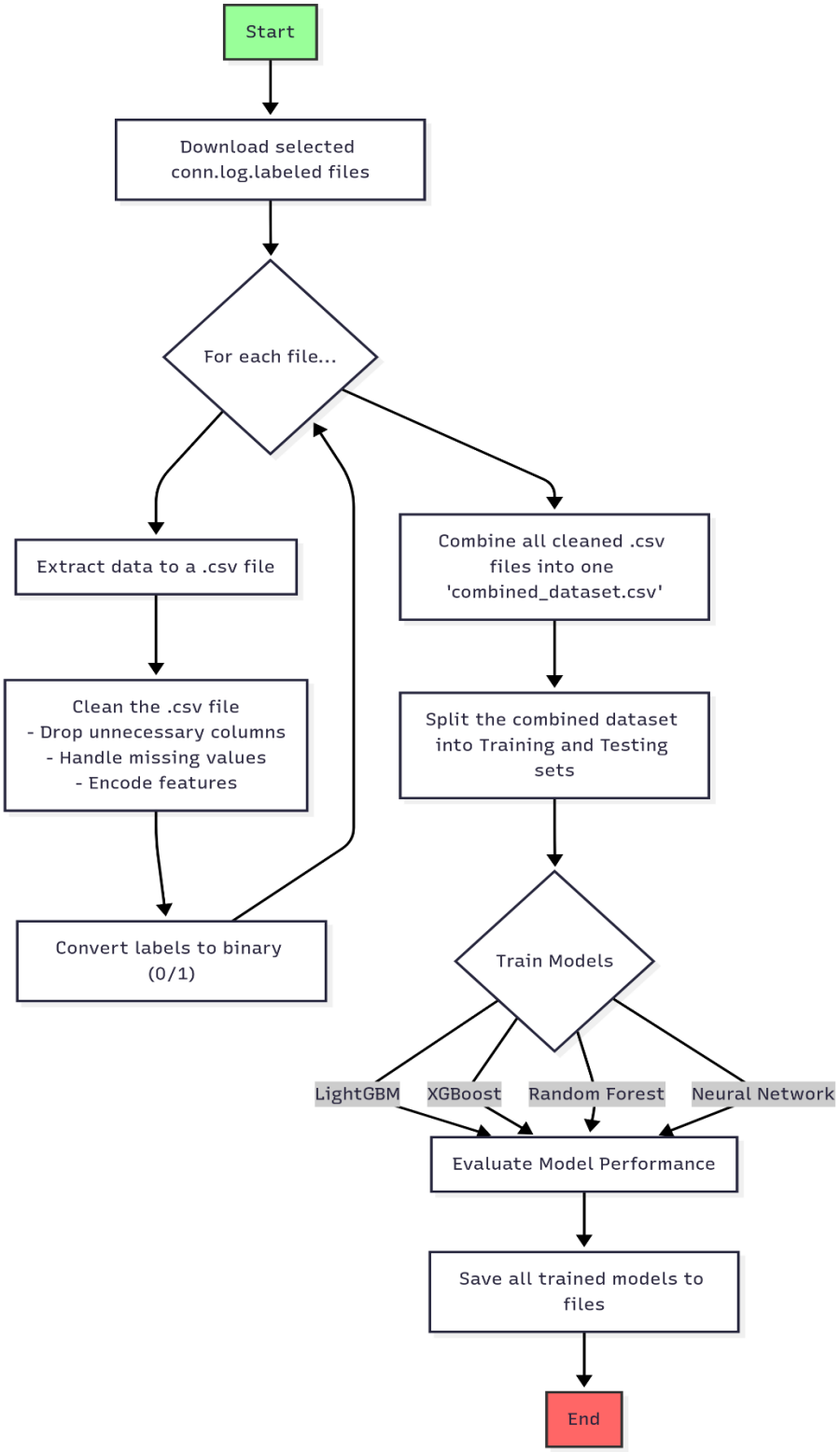


Figure 1 Training flow chart

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) (Laboratoryفي 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، مثلMirai ، وTorii، وGagfyt، و Hajime. 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 باسمBro) 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 Zeekالقياسية (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 وDDos).

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 بـZeek ، 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 ال(headers) 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 تطبيق 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.

معايير القياس والأدوات المستخدمة

معايير القياس (Evaluation Metrics)

لتقييم ومقارنة أداء نماذج التصنيف المختلفة بشكل دقيق، استخدمنا عدة مقاييس تقييم. تعد هذه المقاييس ضرورية لفهم فعالية النموذج، خاثة في المسائل الحساسة التى يكون فيها صف معين مهم مثل مسألتنا هذه (اكتشاف الMalicious).

ان أهم مقياس هو مصفوفة الإرباك (Confusion Matrix) ومنه يمككننا الحصول على باقي المقاييس.

مصفوفة الالتباس (Confusion Matrix)

	Predicted: Malicious	Predicted: Benign
Actual: Malicious	True Positive (TP)	False Negative (FN)
Actual: Benign	False Positive (FP)	True Negative (TN)

Table 1 confusion matrix structure

مصفوفة الالتباس هي عبارة عن جدول يوضح أداء خوارزمية التصنيف. فهي تقدم تفصيل دقيق لعدد التنبؤات الصحيحة وعدد التنبؤات الخاطئة وأنواع الأخطاء التي ارتكبها النموذج. بالنسبة لمشكلة تصنيف ثنائي مثل مسألتنا (حميد أو خبيث)، تحتوي المصفوفة على أربعة أنواع:

Prediction	Actual value	Type	Symbol	Explanation
1	1	True Positive	TP	Predicted Positive and was Positive
0	0	True Negative	TN	Predicted Negative and was Negative
1	0	False Positive	FP	Predicted Positive but was Negative

0	1	False Negative	FN	Predicted Negative but was Positive

criteria2 Table

- **الإيجابيات الحقيقية (True Positives - TP):** عدد الحالات التي كانت خبيثة بالفعل (malicious actual) وتنبأ بها النموذج بشكل صحيح على أنها خبيثة.
- **السلبيات الحقيقية (True Negatives - TN):** عدد الحالات التي كانت حميدة بالفعل (actual benign)وتنبأ بها النموذج بشكل صحيح على أنها حميدة.
- **الإيجابيات الخاطئة (False Positives - FP):** عدد الحالات التي كانت حميدة ولكن تنبأ بها النموذج بشكل غير صحيح على أنها خبيثة. يُعرف هذا أيضا بالخطأ من النوع الأول.
- **السلبيات الخاطئة (False Negatives - FN):** عدد الحالات التي كانت خبيثة ولكن تنبأ بها النموذج بشكل غير صحيح على أنها حميدة. هذا هو أخطر أنواع الأخطاء بحسب مسألتنا لأنه يمثل هجومًا لم يتم اكتشافه، ويُعرف أيضا بالخطأ من النوع الثاني.

الدقة (Accuracy)

وهي المقياس الأكثر انتشارا وتمثل الصحة الإجمالية للنموذج. يتم حسابها كنسبة جميع التنبؤات الصحيحة إلى العدد الإجمالي للحالات ونعطى بالعلاقة التالية:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

ولكنها بالطبع ممكن أن تكون مضللة للغاية في مجموعات البيانات غير المتوازنة. على سبيل المثال، إذا كانت 99% من حركة المرور حميدة، فإن النموذج الذي يتنبأ ببساطة بأن كل اتصال "حميد" سيحقق دقة 99% ولكنه سيكون عديم الفائدة تمامًا لأنه لن يكتشف أي هجوم. لهذا السبب، يجب استخدام مقاييس أخرى ايضا.

الضبط (Precision)

يقيس مدى موثوقية التنبؤات الإيجابية (الخبيثة). إنها تجيب على السؤال الاتي: من بين جميع الاتصالات التي صنفها (توقعها) النموذج على أنها هجوم، ما هي نسبة الهجمات الفعلية؟ تشير درجة الدقة العالية إلى انخفاض (False Positive)، ويعطى بالعلاقة الاتية:

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

الاستدعاء (Recall) أو الحساسية (Sensitivity)

وهو يقيس قدرة النموذج على تحديد جميع الحالات الإيجابية الفعلية. إنه يجيب على السؤال التالي: من بين جميع الهجمات الفعلية التي حدثت، ما هي النسبة التي نجح النموذج في اكتشافها؟ يعد الاستدعاء العالي أمر بالغ الأهمية لنظام كشف التسلل، لأنه يشير إلى انخفاض معدل السلبيات الخاطئة (FN)، أي توقع عدد قليل جدا من الاتصالات الخبيثة كاتصالات حميدة، ويعطة بالعلاقة التالية:

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

النوعية (Specificity)

وهي تقيس قدرة النموذج على تحديد جميع الحالات السلبية (الحميدة) الفعلية بشكل صحيح. إنها تجيب على السؤال التالي: من بين جميع حركة المرور الحميدة الفعلية، ما هي النسبة التي حددها النموذج بشكل صحيح على أنها حميدة؟ تعد النوعية العالية مهمة لضمان عدم تصنيف نشاط المستخدم العادي باستمرار على أنه خبيث.، وهو غير مهم في مسألتنا هذه لاننا نهتم بكشف الاصتالات الخبيثة، ويعطى بالصيغة التالية:

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

مقياس (F1-Score)

مقياس **F1** هو المتوسط التوافقي للضبط (Precision) والاستدعاء (Recall). حيث يوفرمقياس واحد يوازن بين كلا الهدفين: تقليل false positive (ضبط عالي) وتقليل false negative (استدعاء عالي). يعد مقياس F1 مفيداً بشكل خاص لمجموعات البيانات غير المتوازنة لأنه يوازن بين مقاسي الضبط والاستدعاء. تشير درجة F1 العالية إلى أن النموذج جيد جدا في الناحيتين معاً، ويعطى بالصيغة التلية:

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2\ Precision * \ Recall}{Recall + Precision}$$

حيث الضرب ب2 ليصبح مجال المقياس في [0,1] لأن من دون الضرب ب2 كان المجال هو [0,1/2].

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)

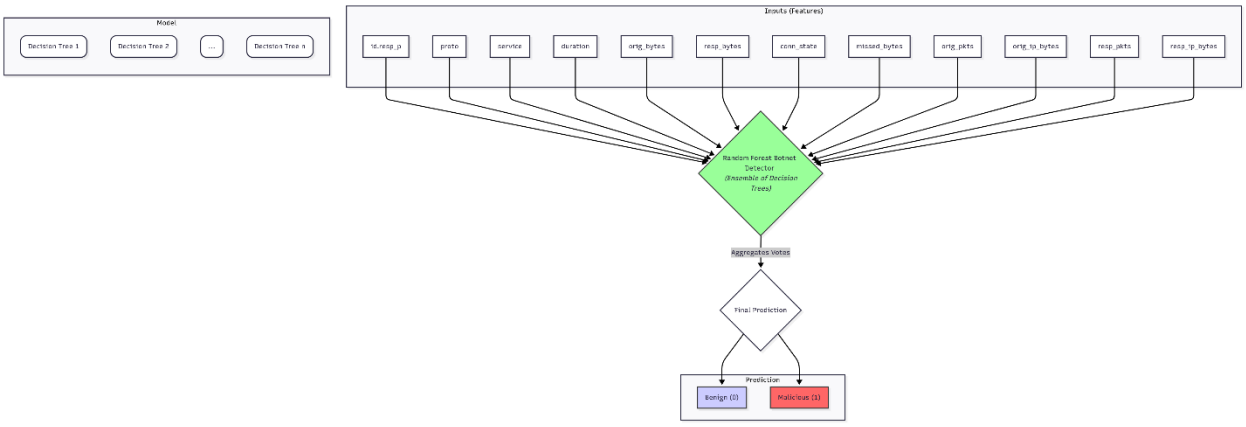


Figure 2 Random forest model diagram

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) (Samplesللبيانات واختيار مجموعات فرعية عشوائية من الميزات. من خلال تعيين random_stateإلى 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.

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

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%.

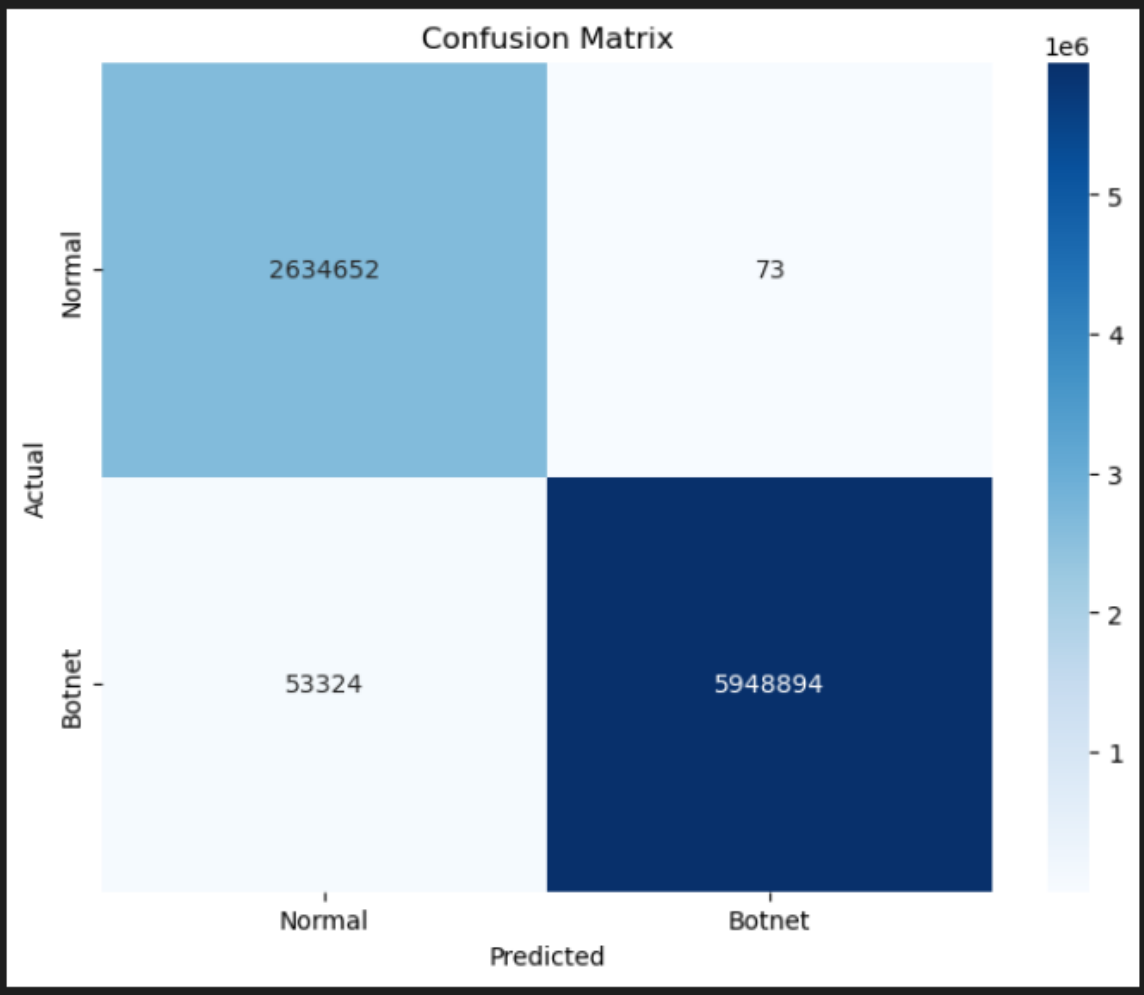
ولقد اخترنا الhyperparameters التالية:

- n_estimators=200تم 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=20تم 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='balancedأظهرت 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=42تم 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:

```
Accuracy: 0.9938
Precision: 1.0000
Recall: 0.9911
F1-Score: 0.9955
```



XGBoost classifier

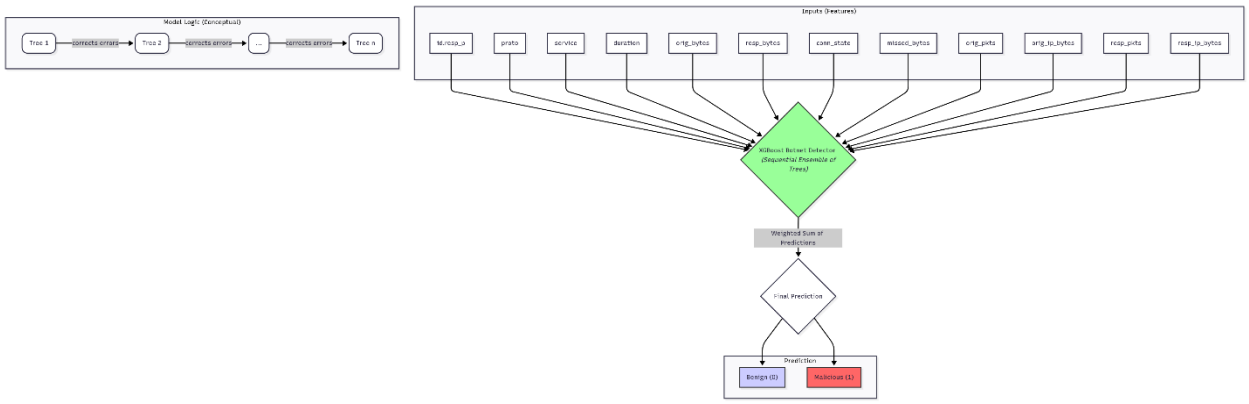


Figure 3 Xgboost model diagram

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 بسيطة decision 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 الoverfittingوتحسين 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

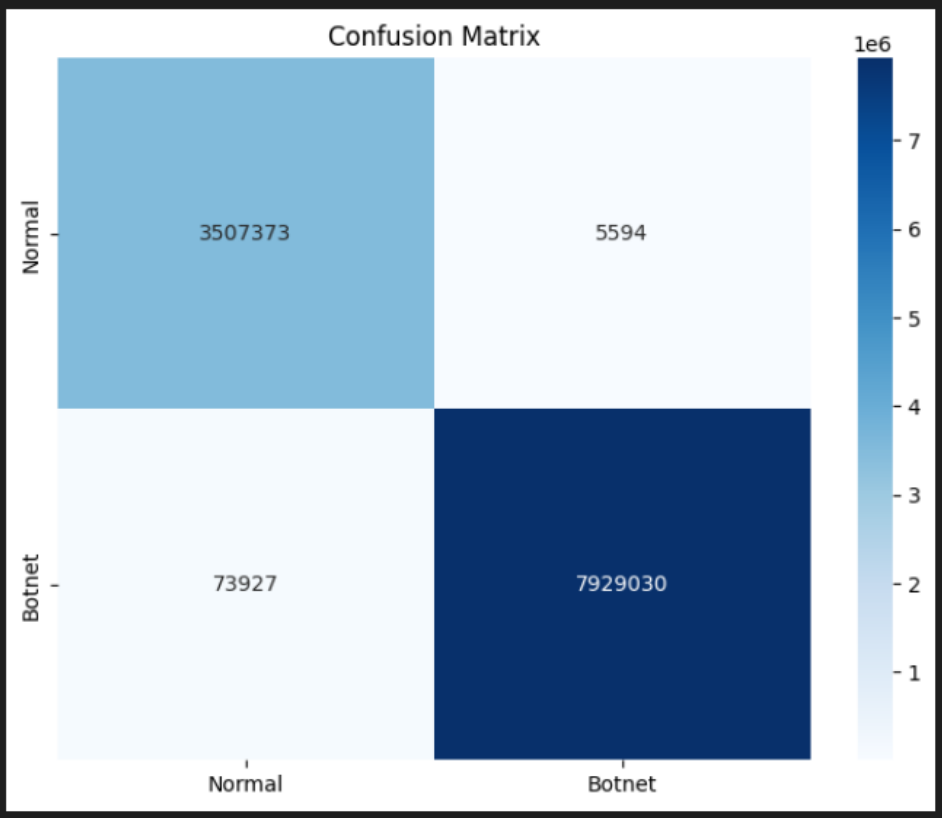
وهي معلمة لضبط الأداء. تخبر XGBoost باستخدام جميع أنوية المعالج (CPU) المتاحة على جهازك لتشغيل أجزاء من عملية التدريب بشكل متوازٍ، وهي ضروري لأن مجموعة بيانات IoT-23 كبيرة جدًا، وقد يكون التدريب بطيئًا. فاستخدام جميع الأنوية المتاحة سيقلل بشكل كبير من وقت التدريب، مما يجعل التدريب أسرع وأكثر كفاءة.

- 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:

```
=== Evaluation Metrics ===
Accuracy: 0.9931
Precision: 0.9993
Recall: 0.9908
F1-Score: 0.9950
ROC AUC: 0.9977
```



Lightgbm classifier

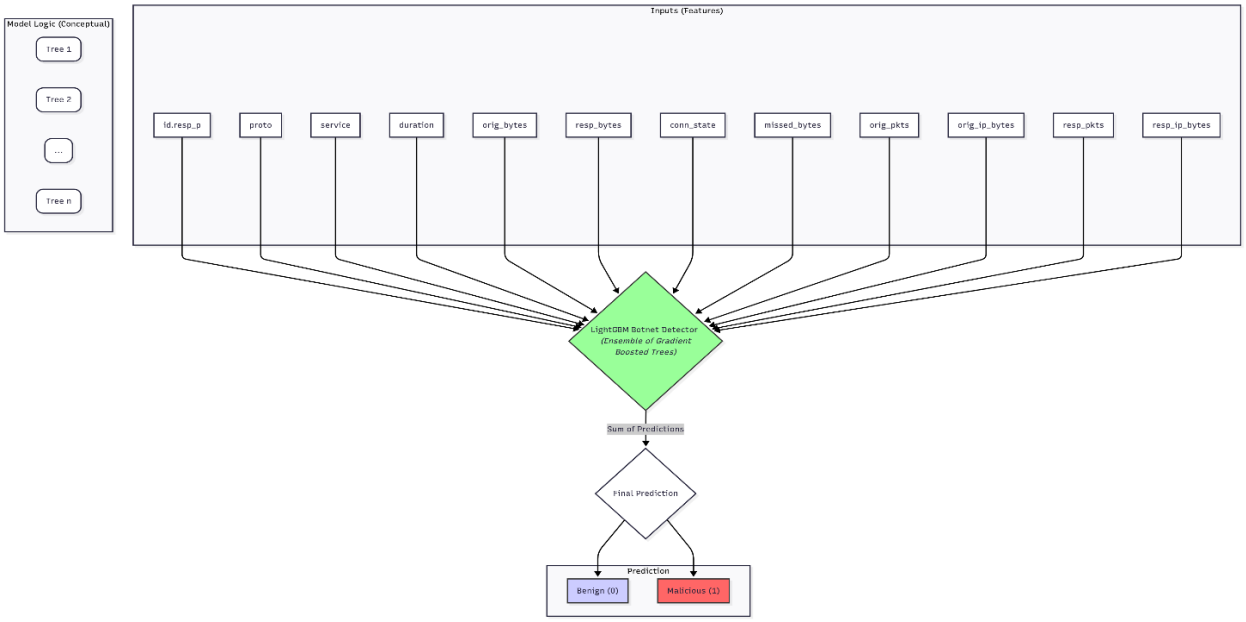


Figure 4 Lightgbm diagram

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 مثلXGBoost ، إلا أن LightGBMقد 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 تقنيةGOSS ، 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 تقنيةEFB ، 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 بطريقةleaf-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.(max_depth)^2
- max_depth: تحدد هذه المعلمة أقصى عمق يمكن أن تنمو إليه الشجرة. على الرغم من أن LightGBM ينمو بطريقةleaf-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_samplesأو 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_estimators: تحدد 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_freqإلى 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 L1و 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).

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

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

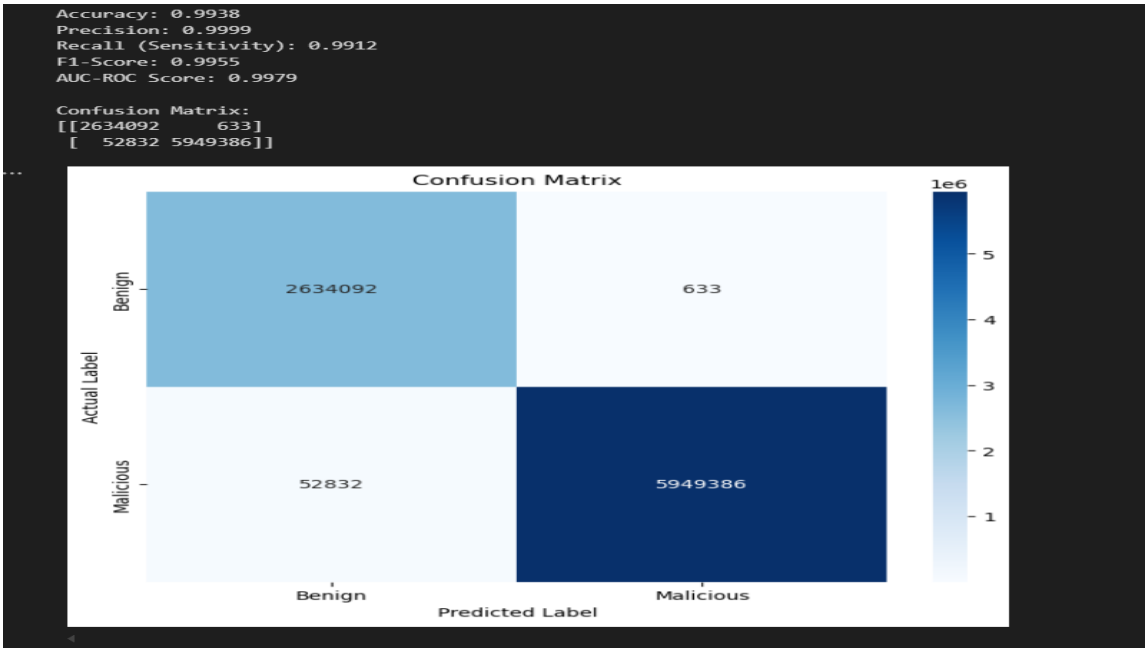
وبالطبع تم قراءة ملف البيانات combined_dataset.csv وقسمناه الى بيانات للتدريب بنسبة70% وبيانات للاختبار بنسبة 30%.

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 (Area Under the ROC Curve) "ROC 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 الكليةaccuracy 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 (الخبيثة malicious)، 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

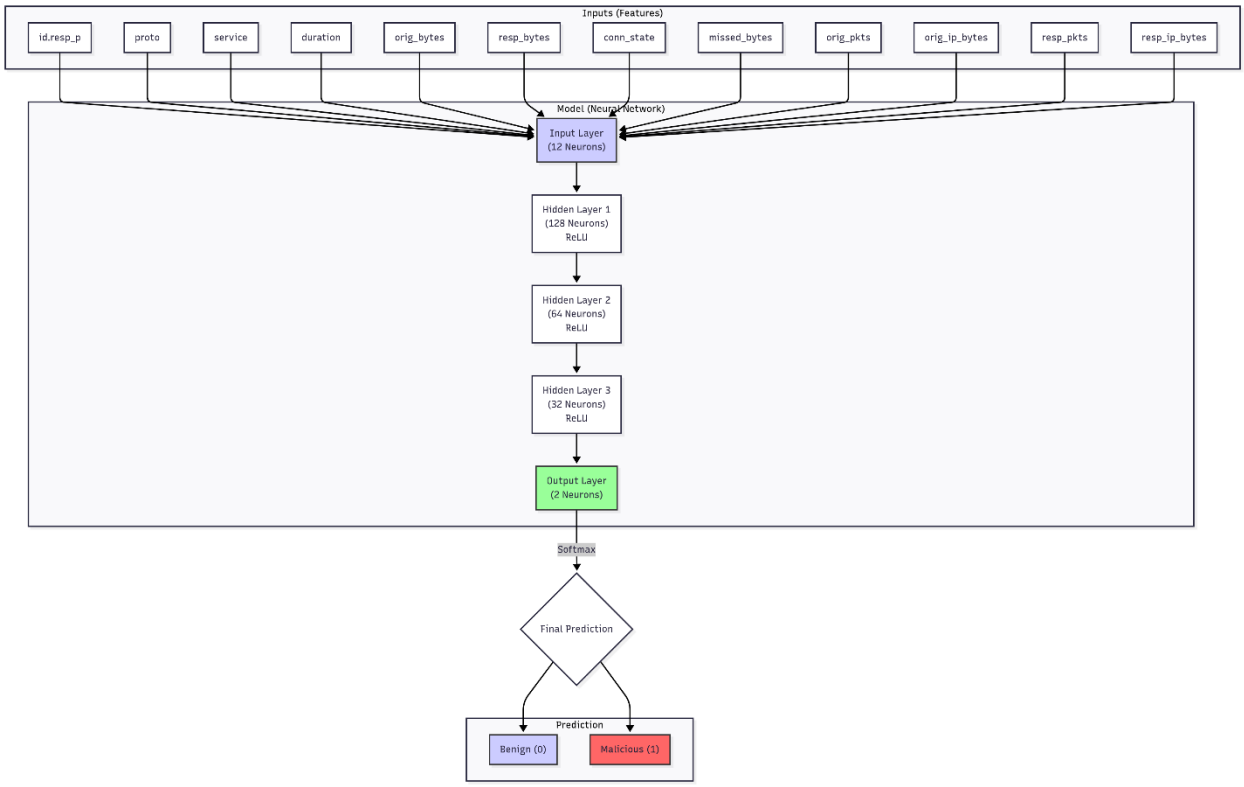


Figure 5 Neural Network model diagram

يُعد مصنف الشبكة العصبونية نموذج حسابي قوي في مجال تعلم الآلة، وهو مستوحى من بنية ووظيفة الدماغ البيولوجي. تم تصميم هذا النموذج لتعلم العلاقات المعقدة وغير الخطية(non-linear) داخل البيانات بهدف أداء مهام التصنيف(classification). الوحدة الأساسية لبناء الشبكة هي العصبون الاصطناعي

(artificial neuron) (أو العقدة)، الذي يستقبل مدخلًا واحدًا أو أكثر، ويقوم بإجراء عملية جمع مرجح (weighted sum) للأوزان، ثم يضيف قيمة انحياز(bias) ، وبعد ذلك يمرر النتيجة عبرتابع تنشيط (activation function)غير خطية لإنتاج مخرجاته.

يتم تنظيم هذه العصبونات في سلسلة من الطبقات :طبقة الإدخال (input layer) التي تستقبل بيانات الميزات الأولية(raw feature data)؛ وطبقة مخفية (hidden layer) واحدة أو أكثر، وهي المسؤولة عن تعلم أنماط وتمثيلات أكثر تجريد بشكل تدريجي من البيانات؛ وطبقة الإخراج (output layer) التي تنتج تنبؤ التصنيف النهائي .تمتلك الاتصالات بين العصبونات في الطبقات المتجاورة أوزانًا (weights) مرتبطة بها، وهي المعلمات الأساسية التي يتعلمها النموذج أثناء التدريب.

تتضمن عملية التدريب، المعروفة باسم التعلم الخاضع للإشراف(supervised learning) ، تغذية الشبكة بمجموعة بيانات كبيرة ومصنفة. لكل دخل، تُنتج الشبكة تنبؤ يتم مقارنته بالتصنيف الحقيقي باستخدام دالة خسارة (loss function) مثل الإنتروبيا المتقاطعة (cross-entropy) لتحديد مقدار الخطأ .بعد ذلك، يتم نشر هذا الخطأ بشكل عكسي عبر الشبكة باستخدام خوارزمية تسمى الانتشار الخلفي(backpropagation) ، والتي تحسب تدرج (gradient) دالة الخسارة بالنسبة لكل وزن .

ثم تستخدم خوارزمية أمثلية(optimization algorithm) ، مثل الانحدار التدريجي(Gradient Descent) ، هذه التدرجات لإجراء تعديلات طفيفة على الأوزان، مما يقلل الخطأ بشكل متكرر .في مهام التصنيف متعدد الفئات، تستخدم طبقة الإخراج عادةً دالة تنشيطSoftmax ، التي تحول النتائج الأولية للشبكة إلى توزيع احتمالي عبر جميع الفئات الممكنة(classes)، وتكون الفئة ذات الاحتمالية الأعلى هي التنبؤ النهائي.

(Goodfellow، Bengio، و Courville، 2016)

Neural Network hyperparameters

يتم التحكم في أداء وسلوك الشبكة العصبونية من خلال مجموعة متنوعة من المعلمات الفائقة (Hyperparameters) التي يجب تكوينها قبل بدء عملية التدريب. يمكن تصنيف هذه المعلمات بشكل عام إلى تلك التي تحدد بنية الشبكة (Architecture) وتلك التي تتحكم في عملية التدريب والأمثلية (Optimization).

وهم:

المعلمات الفائقة المعمارية: Architectural Hyperparameters

- عدد الطبقات المخفية (Number of Hidden Layers): يحدد هذا عمق الشبكة. الشبكة التي لا تحتوي على طبقات مخفية هي نموذج خطي بسيط، بينما تسمح إضافة طبقة مخفية واحدة أو أكثر للشبكة بتعلم وظائف غير خطية أكثر تعقيدًا وتجريدًا بشكل تدريجي. تتمتع الشبكات الأعمق بقدرة تمثيلية أكبر ولكنها أيضًا أكثر تكلفة من الناحية الحسابية وقد يكون تدريبها أكثر صعوبة.
- عدد العصبونات في كل طبقة مخفية (Number of Neurons per Hidden Layer): يحدد هذا عرض كل طبقة وقدرتها الاستيعابية. يسمح عدد أكبر من العصبونات للطبقة بتعلم تمثيلات أكثر تعقيدا من مدخلاتها. ومع ذلك، يمكن أن يؤدي وجود عدد كبير جدًا من العصبونات إلى فرط التخصيص (Overfitting)، حيث يحفظ النموذج بيانات التدريب، بينما يؤدي وجود عدد قليل جدًا إلى نقص التخصيص (Underfitting)، حيث يفتقر النموذج إلى القدرة على التقاط الأنماط الأساسية في البيانات.
- دوال التنشيط (Activation Functions): هي دوال غير خطية يتم تطبيقها على مخرجات كل عصبون، وهي ضرورية للسماح للشبكة بتعلم العلاقات غير الخطية. تشمل الخيارات الشائعة للطبقات المخفية دالة الوحدة الخطية المصححة (ReLU) ومتغيراتها (مثل Leaky ReLU)، والتي تتسم بالكفاءة الحسابية وتساعد في التخفيف من مشكلة تلاشي التدرج (Vanishing Gradient problem). بالنسبة لطبقة الإخراج في المصنف، تُستخدم دالة Sigmoid للتصنيف الثنائي لإنتاج احتمال، بينما تُستخدم دالة Softmax للتصنيف متعدد الفئات لإنتاج توزيع احتمالي عبر جميع الفئات الممكنة.

معلمات التدريب والأمثلية

- المُحسِّن (Optimizer): هي الخوارزمية المحددة المستخدمة لتحديث أوزان الشبكة أثناء الانتشار الخلفي (Backpropagation). في حين أن الانحدار التدريجي العشوائي (SGD) هو الخوارزمية التأسيسية، فإن المحسنات التكيفية الأكثر تقدمًا مثل Adam (تقدير العزم التكيفي) و RMSprop و Adagrad أصبحت الآن ممارسة قياسية. تقوم هذه المحسنات بتكييف معدل التعلم لكل وزن على حدة، مما يؤدي غالبًا إلى تقارب أسرع وأداء أفضل.
- معدل التعلم (Learning Rate): هذا هو أهم وسيط فهو يحدد حجم الخطوة التي يتخذها المحسن عند تحديث الأوزان في اتجاه التدرج السلبي. يمكن أن يؤدي معدل التعلم الصغير جدًا إلى أوقات تدريب طويلة للغاية أو التعثر

في حد أدنى محلي غير أمثل. ويمكن أن يؤدي معدل التعلم الكبير جدًا إلى تباعد التدريب أو التذبذب بعنف، مما يؤدي إلى الفشل في التقارب على الإطلاق.

- حجم الدفعة (Batch Size): يحدد هذا عدد عينات التدريب التي تتم معالجتها قبل تحديث المعلمات الداخلية للنموذج. يعد استخدام مجموعة البيانات بأكملها (Full-batch) مكلفًا من الناحية الحسابية. بدلاً من ذلك، يتم تقسيم البيانات عادةً إلى دفعات صغيرة (Mini-batches). يُدخل حجم الدفعة الأصغر مزيدا من الضوضاء في تحديثات الوزن، والتي يمكن أن يكون لها تأثير تنظيمي ولكنها قد تؤدي إلى تدريب غير مستقر. يوفر حجم الدفعة الأكبر تقديرًا أكثر دقة للتدرج ولكنه يستهلك ذاكرة أكبر.
- عدد الحقب (Number of Epochs): تمثل الحقبة (Epoch) مرور كامل واحد عبر مجموعة بيانات التدريب بأكملها. يحدد عدد الحقب عدد المرات التي سيرى فيها النموذج البيانات. سيؤدي التدريب لعدد قليل جدًا من الحقب إلى نقص التخصيص، بينما يمكن أن يؤدي التدريب لعدد كبير جدًا إلى فرط التخصيص أي الoverfitting. غالبًا ما يتم تحديد العدد الأمثل باستخدام تقنية التوقف المبكر (Early Stopping)، حيث يتم إيقاف التدريب عندما يتوقف الأداء عن التحسن.

معلمات التنظيم (Regularization)

- معدل الاسقاط (Dropout Rate): هي تقنية تنظيم قوية حيث يتم أثناء التدريب "إسقاط" أو تجاهل جزء عشوائي من العصبونات في طبقة ما مؤقتًا لكل دفعة تدريب. هذا يمنع العصبونات من التكيف المفرط مع بعضها البعض ويجبر الشبكة على تعلم تمثيلات أكثر قوة وتكرار. يحدد معدل التسرب (مثلا 0.2 إلى 0.5) احتمال إسقاط العصبون، وهي مفيدة لمنع حدوث overfitting.
- تنظيم L1/L2 (تضاؤل الوزن - Weight Decay): تضيف هذه التقنيات حد عقوبة إلى دالة الخسارة بناءً على حجم أوزان الشبكة. يعاقب تنظيم L2 (الأكثر شيوعاً) المقدار التربيعي للأوزان، مما يشجع النموذج على تعلم قيم أوزان أصغر وأكثر انتشار، مما يساعد على منع فرط التخصيص.

(Goodfellow، Bengio، و Courville، 2016) (Bishop, 2006)

Neural Network Implementation

لقد تم تدريب مصنف الشبكة العصبونية باستخدام كود بايثون (python code) حيث تم جلب المصنف من مكتبة Pytorchعبر التالي:import torch.nn as nn

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

تم بناء النموذج، المسمى BotNetDetector، كشبكة عصبونية تسلسلية متصلة بالكامل (Fully connected neural network). تتكون من طبقة إدخال(input layer)، وثلاث طبقات مخفية(3 hidden layers)، وطبقة إخراج(output layer). ولتحسين استقرار التدريب ومنع فرط التخصيص الoverfitting، تم دمج كل من (Batch Normalization)و (Dropout) في البنية.

البنية التفصيلية لكل طبقة هي التالي:

1. طبقة الإدخال :طبقة خطية تستقبل الميزات (الfeatures) الـ 12 المدخلة وتحولها إلى فضاء ذي 128 بُعداً.
2. الطبقة المخفية الأولى :طبقة خطية تحول الميزات من 128 إلى 64. يتم تمرير المخرجات عبر طبقة تطبيع الدفعات، تليها دالة تنشيط ReLUوطبقة dropout.

3. الطبقة المخفية الثانية :طبقة خطية تحول الميزات من 64 إلى 32. يتبعها تطبيع الدفعات، ودالة تنشيطReLU ، وطبقة تسرب dropout.
4. الطبقة المخفية الثالثة : طبقة تحول من 64 إلى 32 عصبونًا، متبوعة بتطبيع الدفعات، ودالةReLU ، والتسرب. تعمل هذه كآخر طبقة مخفية قبل الإخراج.
5. طبقة الإخراج :طبقة خطية أخيرة تحول الميزات الـ 32 إلى 2 من المخرجات، بما يتوافق مع الفئتين (حميد وخبيث). يتم تطبيق دالة Softmax ضمنيًا بواسطة دالة الخسارة أثناء التدريب لتوليد الاحتمالات النهائية للفئات.

اعداد الوسطاء hyperparameters configuration

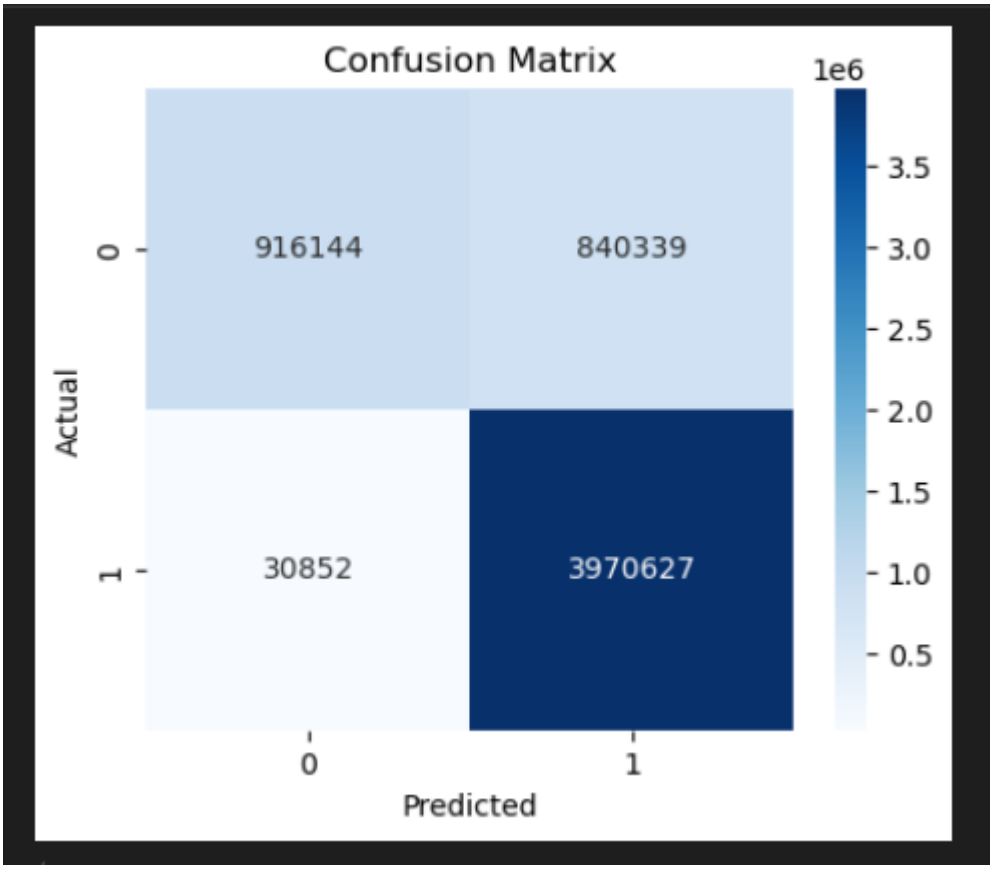
تم التحكم في عملية التدريب وسلوك النموذج من خلال مجموعة من المعلمات الفائقة:

- المُحسِّن (Optimizer) : تم استخدام مُحسِّن Adam، وهو خوارزمية أمثلية ذات معدل تعلم تكيفي مناسبة لمجموعة واسعة من المشاكل.
- معدل التعلم :(Learning Rate) تم تحديد معدل التعلم الأولي عند 0.001.
- دالة الخسارة :(Loss Function) تم اختيار خسارة الإنتروبيا المتقاطعة (nn.CrossEntropyLoss)كمعيار للخطأ. وهي دالة الخسارة القياسية لمهام التصنيف، حيث إنها فعالة في قياس أداء نموذج يُخرج احتمالات.
- حجم الدفعة :(Batch Size) تم تغذية الشبكة بالبيانات على شكل دفعات صغيرة بحجم 1024عينة. يوفر هذا الحجم توازنًا جيدًا بين الكفاءة الحسابية وتقدير التدرج المستقر.
- عدد الحقب :(Number of Epochs) تم ضبط النموذج للتدريب لمدة أقصاها 50حقبة.
- مجدول معدل التعلم :(Learning Rate Scheduler) تم تطبيق مجدول من نوع .ReduceLROnPlateauيقوم هذا المجدول بمراقبة خسارة التحقق ويقلل معدل التعلم بمقدار 0.5 إذا لم تتحسن الخسارة لمدة 3 حقب(epochs) متتالية.
- التوقف المبكر :(Early Stopping) لمنع فرط التخصيص وتوفير الوقت الحسابي، تم استخدام آلية التوقف المبكر مع patience تساوي 5 تم إيقاف التدريب عندما فشلت خسارة التحقق في التحسن لمدة خمس حقب متتالية.

معلمات التنظيم(Regularization)

- معدل التسرب :(Dropout Rate) تم تطبيق احتمال تسرب قدره 0.3 بعد كل طبقة مخفية. هذا يعني أنه خلال كل تكرار تدريبي، تم إلغاء تنشيط 30% من العصبونات عشوائياً، مما يجبر الشبكة على تعلم ميزات أكثر قوة.
- تضاؤل الوزن :(L2 Regularization) تمت إضافة قيمة تضاؤل وزن صغيرة تبلغ 1e-5 إلى مُحسّن Adam لتطبيق تنظيم L2 ، مما يساعد على منع أوزان النموذج من النمو بشكل كبير.

```
=== Test Set Performance ===
Accuracy: 0.8487
Precision: 0.8253
Recall: 0.9923
F1-Score: 0.9011
```



بيئة المحاكاة والنشر (Simulation Environment for Deployment)

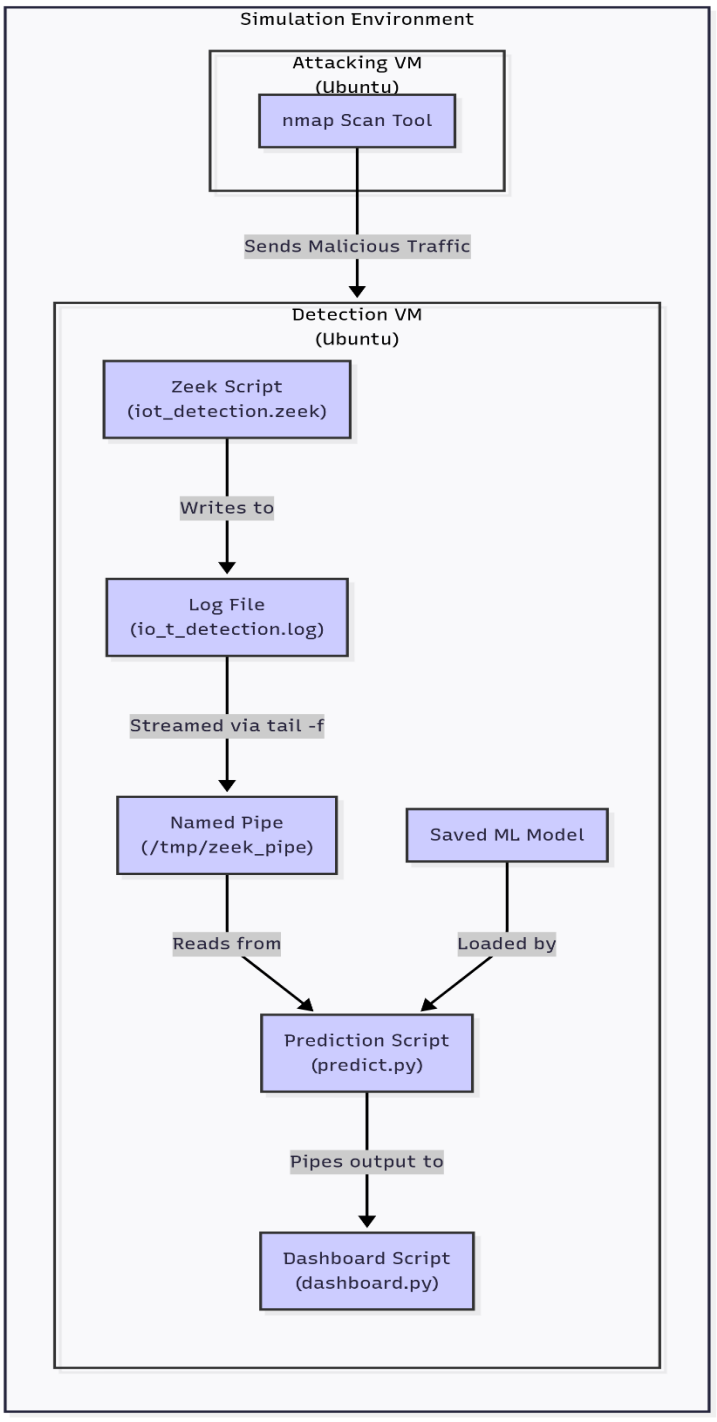


Figure 6 Deployment diagram

لغرض التحقق من الجدوى العملية والأداء في الزمن الحقيقي لنماذج تعلم الآلة المدربة، تم بناء بيئة محاكاة شاملة ومتكاملة. تم اختيار نموذج لمرحلة النشر والتوظيف، صُممت هذه البيئة لتعمل كنظام مصغر لكشف التسلل الشبكي (Network Intrusion Detection System - NIDS)، بحيث تكون قادرة على التقاط حركة مرور الشبكة وتحليلها وتصنيفها بشكل فوري.

إعداد البيئة الافتراضية والشبكة (Virtual Environment and Network Setup)

تم بناء بيئة المحاكاة باستخدام برنامج المحاكاة الافتراضية VMware Workstation، حيث قمنا باستضافة جهازين افتراضيين (Virtual Machines - VMs) منفصلين، كلاهما يعمل بنظام التشغيل Ubuntu.

- الة الكشف (NIDS, Detection Machine): عملت هذه الالة الافتراضية كنظام الكشف، وكان عنوانها ال IP هو 192.168.32.129.
- الة الهجوم (Attacking Machine): استُخدمت هذه الالة لتوليد حركة مرور شبكية خبيثة، وعنوانها الip هو 192.168.32.132.

تم اعداد كلا الجهازين الافتراضيين باستخدام محولات شبكة (Network Adapter) في وضع NAT (Network Address Translation). هذا الإعداد أنشأ شبكة افتراضية معزولة حيث يمكن للجهازين التواصل مباشرة مع بعضهما البعض، مع الحفاظ على إمكانية الوصول إلى الإنترنت عبر الجهاز المضيف (الhost). يحاكي هذا التكوين طوبولوجيا شبكة نموذجية في العالم الحقيقي، حيث تكون الأجهزة محمية خلف جهاز توجيه (Router).

تكوين جهاز الكشف: البنية النظامية (Detection Machine: System Architecture)

تم تصميم جهاز الكشف بخط أنابيب متعدد المكونات لمعالجة حركة مرور الشبكة في الزمن الحقيقي. تضمن سير العمل أربع مراحل، تم تنسيقها من خلال سلسلة من النصوص البرمجية وأدوات نظام لينكس Linux.

المكون الأول: التقاط حركة المرور واستخلاص الميزات (Zeek)

أساس نظام كشف التسلل هو أداة مراقبة أمان الشبكات Zeek. تم تكوين Zeek ليتنصت على واجهة الشبكة الرئيسية للجهاز الافتراضي (ens33) باستخدام نص برمجي مخصص باسم iot_detection.zeek. تم تصميم هذا النص خصيصًا ليتكامل مع خط أنابيب تعلم الآلة من خلال عدة تكوينات رئيسية:

- تسجيل JSON: ان النص البرمجي يقوم بتحميل سياسة تسجيل JSON في Zeek، مما يضمن أن تكون جميع المخرجات بتنسيق منظّم وقابل للقراءة آلياً، لتسهيل تحليلها بواسطة نصوص بايثون (python scripts) البرمجية اللاحقة.
- معالجة الاتصالات غير المكتملة: تم ضبط مهلة عدم نشاط بروتوكول TCP (tcp_inactivity_timeout) على 0 sec. يعد هذا تعديل ضروري يجعل Zeek يقوم بتسجيل سجلات اتصالات TCP فوراً، حتى لو لم تكتمل المصافحة الثلاثية (Three-way Handshake). تم تصميم هذا خصيصًا لالتقاط تقنيات الاستطلاع الخفية مثل هجمات مسح TCP SYN.
- هندسة الميزات في الزمن الحقيقي: قام النص البرمجي بتعريف سجل مخصص (IoTDetection::LOG) يقوم باستخلاص الميزات الـ 12 المستخدمة بالضبط أثناء تدريب النموذج. والأهم من ذلك، أنه يقوم بنفس عملية الترميز من فئوي (categorical) إلى رقمي التي تم تطبيقها على بيانات التدريب أي يقوم بعمل encoding. على سبيل المثال، يحتوي على قوائم لتحويل أسماء البروتوكولات مثل "tcp" إلى 1 وحالات الاتصال مثل "S0" إلى 0 أي أن

القوائم تربط كل قيمة لسمه معينة برقم معين. هذا يضمن أن البيانات الملتقطة يتم تنسيقها بشكل مثالي لنموذج تعلم الآلة دون الحاجة إلى خطوة معالجة مسبقة منفصلة في بايثون، مما يزيد من كفاءة النظام.

المكون الثاني: خط أنابيب البيانات في الزمن الحقيقي (Named Pipe)

لتدفق البيانات من Zeek إلى نص التنبؤ البرمجي (predict.py script)في الزمن الحقيقي، تم استخدام أنبوب pipe، وهو أسلوب معتمد في نظام لينكس.

تم تحقيق ذلك باستخدام الأمر التالي الذي يعمل في terminal مخصصه له:

```
tail -f io_t_detection.log > /tmp/zeek_pipe
```

يقوم هذا الأمر بمراقبة (-f) ملف السجل (io_t_detection.log) الذي يولده Zeek باستمرار، ويعيد توجيه أي أسطر جديدة من البيانات إلى أنبوب موجود في /tmp/zeek_pipe. يعمل هذا الأنبوب كمخزّن بيانات مؤقت فوري في الذاكرة بين مستشعر Zeek وكود التنبؤ.

المكون الثالث: محرك التنبؤ (predict.py)

يقوم كود بايثون البرمجي بالتنبؤ بماهية الاتصالات فهو يقرأ باستمرار من الأنبوب المسمى /tmp/zeek_pipe، ويعالج سجل اتصال واحد في كل مرة. حيث يعمل كالتلي:

1. تحميل النموذج: عند بدء التشغيل، يقوم النص البرمجي بتحميل نموذج lightgbm المدرب مسبقا والمحفوظ في (models/xgboost.pkl) باستخدام مكتبة joblib.
2. القراءة من الأنبوب: يقرأ النص البرمجي إدخالات السجل الجديدة بصيغة JSON التي تظهر في الأنبوب
3. التنبؤ: لكل سجل اتصال جديد، يستخرج قيم الميزات الـ 12، ويضمن أنها بالترتيب الصحيح، ثم يدخلها إلى نموذج XGBoost المحمّل. يُخرج النموذج تنبؤاً prediction (0 للحميد، 1 للخبيث) ودرجة ثقة (احتمالية أن يكون الاتصال خبيثًا)(confidence score)
4. الإخراج: يقوم النص البرمجي بعد ذلك بتجميع الميزات الأصلية، والتنبؤ، ودرجة الثقة (confidence score) في كائن JSON جديد ويطبعه على شاشة الterminal.

المكون الرابع: واجهة المستخدم (dashboard.py)

المكون النهائي هو لوحة معلومات سهلة القراءة تعمل في الزمن الحقيقي لعرض مخرجات محرك التنبؤ بطرسقة جيدة. تم ربط النصين البرمجيين باستخدام أنبوب Shell :

```
python3 scripts/predict.py | python3 scripts/dashboard.py
```

يقرأ نص dashboard.py مخرجات JSON من predict.py ويقدم واجهة terminal يتم تحديثها باستمرار.
تعرض لوحة المعلومات ما يلي:

- إحصائيات عامة: مجموع كلي للاتصالات التي تم تحليلها، مقسمة إلى حركة مرور عادية(Normal Traffic) و حركة مرور هجومية(Attack Traffic) مع أعدادهم ونسبهم المئوية.
- مخطط شريطي مرئي (Bar Chart): مخطط شريطي بسيط يمثل بصريا نسبة حركة المرور الحميدة مقابل الخبيثة.
- تفاصيل آخر هجوم: قسم مخصص يعرض الطابع الزمني، وعنوان IP المصدر والوجهة/المنفذ، وحالة الاتصال لآخر اتصال خبيث تم اكتشافه.
- ترميز لوني: لجذب انتباه المستخدم على الفور، حيث استخدمنا في الكود البرمجي ألوان لعرض جميع المعلومات المتعلقة بالهجمات باللون الأحمر وحركة المرور العادية باللون الأخضر.

تكوين جهاز الهجوم: محاكاة التهديدات (Attacking Machine: Threat Simulation)

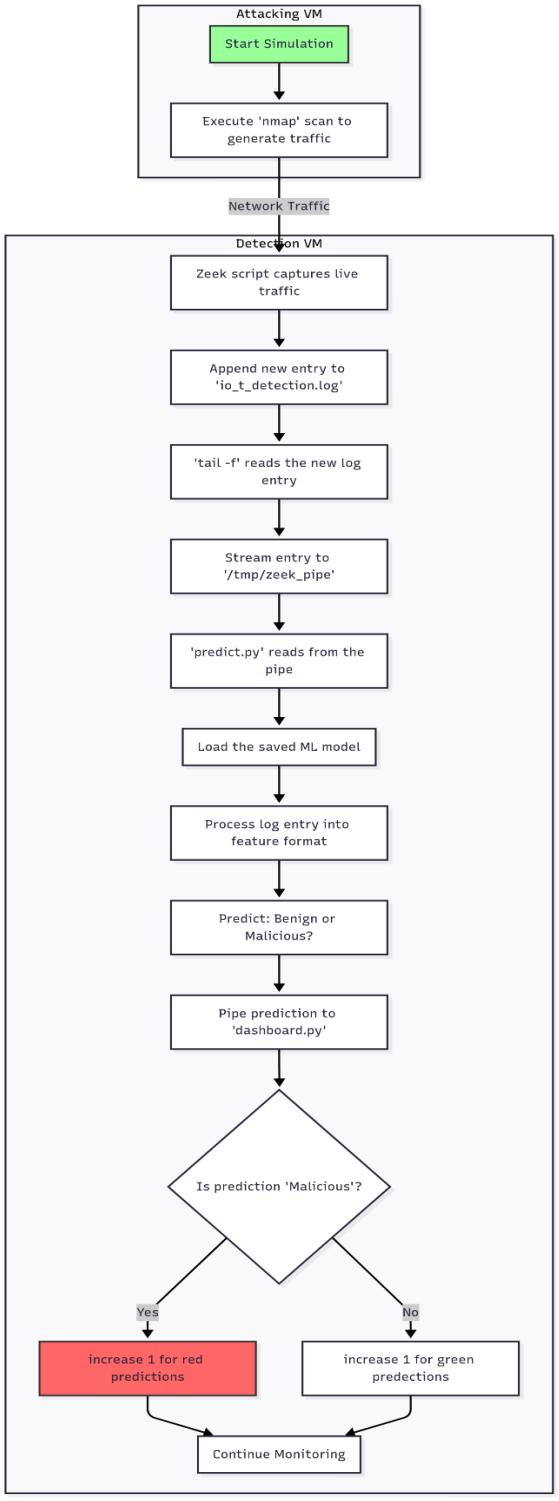


Figure 7 Attacking flow chart

تم استخدام الة الهجوم لتوليد حركة مرور شبكية نتحكم بها لاختبار نظام كشف التسلل.

توليد حركة المرور الخبيثة

تمت محاكاة النشاط الخبيث باستخدام ماسح الشبكات Nmap، وهو أداة قياسية لاستطلاع الشبكات.

حيث قمنا باستخدام الأمر التالي:

sudo nmap -sS -p 80 192.168.32.129 -n

- sS (مسح TCP SYN): تبدأ هذه العلامة مسحًا خفيًا(stealth) أو نصف مفتوح(half-open). يقوم بإرسال حزم SYN ولكنه لا يكمل الاتصال أبدا، وهي تقنية شائعة يستخدمها المهاجمون لاكتشاف المنافذ المفتوحة(open ports) دون أن يتم تسجيلهم بسهولة بواسطة الأنظمة التقليدية.
- p 80: يستهدف هذا المنفذ 80 فقط، وهو المنفذ القياسي لحركة مرور HTTP (الويب)، مما يجعل المسح يبدو وكأن المهاجم يبحث عن خوادم ويب.
- 192.168.32.129: هذا هو عنوان IP لجهاز الكشف(Detection machine)، مما يجعله الهدف المباشر للمسح.
- n: تعمل هذه العلامة على تعطيل تحليل أسماء النطاقات (DNS)، مما يسرع عملية المسح.

يولد هذا الأمر حجم كبير من الاتصالات بحالة S0 في سجلات Zeek، والتي تم تصميم نموذج XGBoost، المدرب على أنماط مماثلة من مجموعة بيانات IoT-23، لتحديدها على أنها استطلاع خبيث.

Bibliography

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* New York, NY: Springer.

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

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). San Francisco: ACM.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* Cambridge, MA: MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). New York: Springer.

Ke, G. M. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems 30 (NIPS 2017)* (pp. 3146-3154). Red Hook, NY, USA: Curran Associates, Inc.

Microsoft and LightGBM Contributors. (2025, August 4). *Parameters*. Retrieved from LightGBM Documentation: https://lightgbm.readthedocs.io/en/latest/Parameters.html