Aide à la classification des événements remontés à un SOC

Isabelle Kraemer, Mathieu Langlais

Orange France

Introduction / Abstract

L'intelligence artificielle, et en particulier le Machine Learning apparaissent comme des innovations à forte valeur ajoutée pour la sécurité informatique, en particulier pour les équipes en charge de la détection, de la qualification et de la réaction face aux événements de sécurité : les SOC (Security Operations Center).

Nous proposons ici un retour d'expérience sur l'application de techniques de Machine Learning pour la qualification automatique d'événements de sécurité, traités par un SOC.

Nous commencerons par décrire brièvement les activités d'un SOC puis nous détaillerons les différentes étapes de conception et mise en œuvre d'un dispositif de qualification d'événements sécurité dans un environnement de production. En particulier, nous aborderons la méthode utilisée pour pouvoir facilement et rapidement mettre à jour les modèles appris, afin de s'adapter aux mutations des attaquants.

Présentation succincte de l'activité d'un SOC

Pour le MITRE [1], « *a SOC is a team primarily composed of security analysts organized to detect, analyze, respond to, report on, and prevent cybersecurity incidents* ». Ainsi, le SOC analyse et qualifie la pertinence des détections afin de prioriser correctement le traitement des événements les plus critiques. Ces traitements se déclinent en actions à chaud (éviter la propagation d'un attaquant) puis à froid (éradiquer l'attaquant des systèmes et remettre les systèmes en mode de fonctionnement nominal). L'analyse post-mortem boucle le cycle en identifiant les pistes d'amélioration.

Une partie des activités du SOC consiste donc à traiter un nombre souvent important de remontées.

Hors, les tâches de qualification, essentielles à la pertinence d'un SOC, sont régulières, chronophages, répétitives et donc pénibles : il est particulièrement intéressant d'automatiser ce qui peut l'être sur cette étape.

L'objectif est donc d'assister l'humain dès lors que le niveau de compétences pour traiter la tâche n'est pas significatif.

Un retour d'expérience

Nous avons appliqué cette démarche à la classification automatique d'alertes remontées par nos systèmes de supervision. Historiquement, ces alertes sont (re)qualifiées à la main, en temps différé, afin de prioriser la réaction et d'améliorer les systèmes de détection en ajustant les algorithmes de supervision. Au fil du temps, nous avons labellisé un corpus d'occurrences avec les labels « positifs » (autrement appelés « malveillants ») et de « faux positifs » (les cas « normaux »). Grâce à ce corpus, nous souhaitons classifier plus facilement les futures occurrences tout en gagnant du temps. Ce problème de Machine Learning est un problème d'apprentissage supervisé, plus exactement c'est un problème de classification dans lequel l'algorithme doit, en fonction des variables (ou « features ») des alertes, retourner un label « normal » ou « malveillant ».

Les acteurs nécessaires pour l'aide à la classification sont l'analyste et le data scientist. L'analyste apporte une vue « métier » : il connait la menace et formule la problématique à résoudre (automatiser la classification des alarmes) ; il sait également interpréter les remontées des systèmes de supervision, qui seront des inputs de la démarche, et pourra valider les résultats fournis. Le data scientist, lui, apporte son expertise mathématique : il élabore les analyses et algorithmes, en prenant en compte les inputs métier.

Construction du jeu de données

Nous nous appuyons sur les données collectées par le SOC, comme expliqué ci-dessus. Typiquement, pour des sondes réseau, on pourra s'appuyer sur des données comme le type de sonde, la date de la détection, les marqueurs de l'attaque à détecter (IP et port sources et destinations, taille des requêtes, présence d'une signature réseau connue, etc). Les marqueurs pertinents sont à déterminer avec les analystes et les experts métier.

On commencera par s'assurer de la cohérence des données fournies, d'autant plus si le travail d'analyse et de labellisation préliminaires sont majoritairement manuels (une erreur de labellisation est alors vite arrivée).

Par ailleurs, les données d'apprentissage doivent être régies par les mêmes lois que les données auxquelles on appliquera le modèle généré. Il pourra donc être nécessaire de restreindre les données en input à un sous-ensemble représentatif de la menace actuelle, par exemple en excluant des échantillons trop anciens s'ils correspondent à des modèles d'attaques obsolètes.

Feature engineering

Il est nécessaire de contextualiser suffisamment les alertes afin que l'algorithme de Machine Learning ait à disposition les éléments significatifs. Si les données collectées ne permettent pas à un analyste chevronné de conclure, alors ces données ne seront probablement pas suffisantes pour qu'un algorithme de Machine Learning puisse également classifier correctement les mêmes éléments.

Outre l'aide essentielle de l'analyste, l'analyse statistique des données permettra de trouver des liens entre les caractéristiques des alertes et les labels associés. Par ailleurs, on ignorera les données non fiables (par exemple, celles modifiables par l'attaquant).

Les features de départ ne sont pas toujours les plus adaptées pour caractériser un échantillon. Des librairies existent pour générer de nouvelles features, comme par exemple pour Python : sklearn.preprocessing.PolynomialFeatures [2], assez basique, et Featuretools [3] plus flexible. La librairie glmulti [4] , en R, propose quant à elle de comparer les combinaisons de features, pour un type de modèle donné, qui donnent les meilleurs résultats. Dans notre cas, nous avons choisi de nous baser sur Khiops[1] [5] [6] : cet outil de type boite noire offre une intégration simplifiée et permet l'usage d'un méta-langage de descriptions des features. Il produit des modèles prédictifs performants et sans réglages additionnels.

Nous détaillons plus cet aspect dans la deuxième partie de ce document.

Le choix de l'algorithme

Si l'on souhaite pouvoir interpréter aisément les résultats, par exemple pour comprendre quelles features sont structurantes dans la classification des alertes, on préfèrera la mise en œuvre d'algorithmes explicatifs, telles que les forêts aléatoires : les réseaux neuronaux sont dans ce cas à éviter. [7]

La validation de l'algorithme

Les modèles entrainés peuvent souffrir de plusieurs défauts. Ils peuvent sous-apprendre et être peu fiables sur les données d'apprentissage, comme sur les données de test : le modèle n'est alors pas adapté aux données, ou la qualité des données est insuffisante. Ils peuvent aussi sur-apprendre, c'est-à-dire qu'ils prédisent très bien pour les données d'apprentissage, mais peinent à généraliser : le modèle doit être ajusté pour prédire correctement les labels de données encore jamais vues.

[1] Khiops est un outil d'apprentissage supervisé automatique pour la fouille de grandes bases de données multi-tables. L'importance prédictive des variables est évaluée au moyen de modèles de discrétisation dans le cas numérique et de groupement de valeurs dans le cas catégoriel. Dans le cas d'une base multi-tables, par exemple des clients avec leurs achats, une table d'analyse individus _ variables est produite par construction automatique de variables. Le modèle de classification utilisé est un classifieur Bayésien naïf avec sélection de variables et moyennage de modèles. L'outil est adapté à l'analyse des grandes bases de données, avec des millions d'individus, des dizaines de milliers de variables et des centaines de millions d'enregistrements dans les tables secondaires.

La validation croisée consiste à tester, de manière itérative, la performance d'un modèle sur des données non vues lors de l'apprentissage. C'est une méthode pour limiter le sur-apprentissage.

Les courbes d'apprentissage permettent, quant à elles, de poser le diagnostic d'un sur- ou sous-apprentissage sur le modèle. Elles représentent l'évolution de la performance du modèle, d'une part sur les données d'apprentissage, d'autre part sur les données de test, en fonction du nombre d'échantillons considérés. L'écart relatif entre ces deux courbes, et leur position par rapport à la performance cible du modèle, permettent de conclure sur l'adéquation du modèle.

La question des corpus déséquilibrés

Si les mécanismes de détection en place sont efficaces, le corpus d'apprentissage, constitué de toutes les alertes remontées avec label « normal » ou « malveillant », sera très déséquilibré : la grande majorité des alertes porteront le label « malveillant ».

Une méthode consiste à « stratifier » les sous-ensembles sur lesquels on va travailler [8] : il s'agit de de garder la même répartition entre label « normal » versus « malveillant » dans les sous-ensembles de travail afin d'éviter les biais triviaux. Cela s'applique en particulier à la séparation du jeu de données en données d'apprentissage et données de test, ainsi qu'à la constitution des sous-ensembles pour la validation croisée.

D'autres méthodes existent, que nous n'avons pas explorées dans le cadre de cette étude.

Métrique d'évaluation de la performance

Cette métrique doit permettre de comparer les performances des algorithmes entre eux, pour adopter l'algorithme le plus performant. En effet, tout modèle est une approximation, on aura donc nécessairement des erreurs : labéliser « malveillant » un comportement normal (faux positifs), labéliser « normal » un comportement malveillant (faux négatifs).

Ici, nous prendrons par exemple la courbe ROC, et plus précisément son aire, comme métrique de la performance du modèle. Si l'on représente les taux de vrais positifs et les taux de faux positifs de chaque algorithme sur une courbe ROC, on s'attachera à maximiser le taux de vrai positif et à minimiser le taux de faux positifs : on choisira l'algorithme dont la courbe ROC est au-dessus des autres. Cela se traduit par une aire sous la courbe (AUC) supérieure pour les modèles les plus performants. La Figure 1 montre ainsi les courbes ROC de différents modèles : le meilleur est celui qui se rapproche le plus du modèle qui ne se trompe jamais, appelé ici « modèle oracle ».

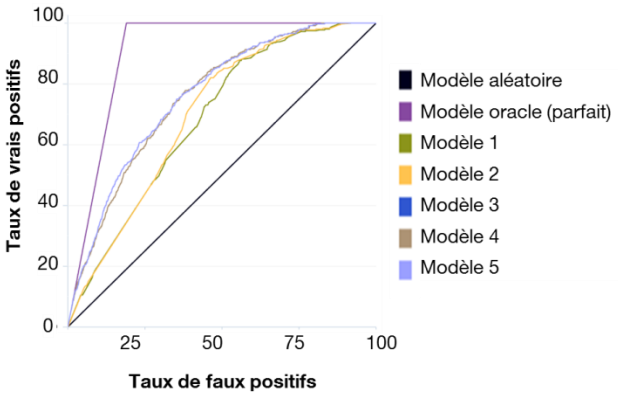


Figure 1 - Tracé de courbes ROC pour différents modèles

Par ailleurs, pour considérer que l'algorithme de classification obtenu est efficace, et comme le jeu de données est déséquilibré, l'algorithme devra être meilleur qu'un algorithme de décision trivial qui classifierait tous les échantillons comme « malveillants ». En effet, si, par exemple, 90% des alertes sont bien des usages « malveillants », alors cet algorithme trivial aurait déjà une accuracy de 90% : il faudra alors s'attacher à dépasser cette valeur pour considérer qu'un algorithme est pertinent. Cette remarque s'applique à toutes métriques d'évaluation qui pourraient être utilisées.

La nature temporelle des données

Les attaques et leurs caractéristiques évoluant au fil du temps, il est important de s'assurer que le modèle considéré se généralise correctement sur les données les plus récentes. Par exemple, lors de la séparation du jeu de données en données d'apprentissage et données de tests, on pourra mettre dans les données de tests les échantillons les plus récents.

Lors de nos travaux, nous avons commencé par constituer aléatoirement le jeu d'apprentissage et le jeu de test, comme on le fait dans les problèmes sans nature temporelle : les performances de notre algorithme sur des données inconnues étaient très bonnes. En revanche, si l'on s'entrainait sur les données les plus anciennes en validant le modèle sur les données les plus récentes, les performances étaient alors mauvaises. Après investigation, nous avons en effet constaté que certains effets étaient déterminants dans la décision de l'algorithme, alors qu'ils se manifestaient, certes, sur beaucoup d'échantillons, mais seulement sur une courte période de temps. Il a alors fallu repenser la manière dont nous avons choisi nos variables et choisir des variables relativement stables dans le temps.

La mise en œuvre opérationnelle

L'algorithme obtenu pourra être utilisé de manière purement automatique : l'analyste n'intervient plus sur la qualification des alertes ; ou comme une aide à la

décision : l'analyste a accès à une première analyse de l'algorithme et peut s'appuyer sur cette dernière pour trancher. Une approche intermédiaire consiste à utiliser un algorithme de Machine Learning dont la sortie est un label (« normal » versus « malveillant ») avec une probabilité (par exemple : « bénin avec un degré de certitude de 80% »). Le SOC peut ainsi choisir de qualifier automatiquement les alertes lorsque le degré de certitude est élevé, les autres cas devant être réévalués par un humain pour lever l'ambiguïté. Cette approche hybride permet de réduire la charge de travail de l'analyste qui peut ainsi se concentrer sur les cas les plus complexes.

L'évolution du modèle et des features

Architecture modulaire d'intégration d'algorithmes de Machine Learning

Le processus d'intégration d'un algorithme de Machine Learning suit classiquement les étapes suivantes :

1. identification des sources de données
2. construction de variables explicatives (ou features) depuis une ou plusieurs des sources disponibles et production d'un modèle prédictif avec création de KPI grâce à une partie du dataset utilisé uniquement pour le test.
3. valorisation du modèle prédictif

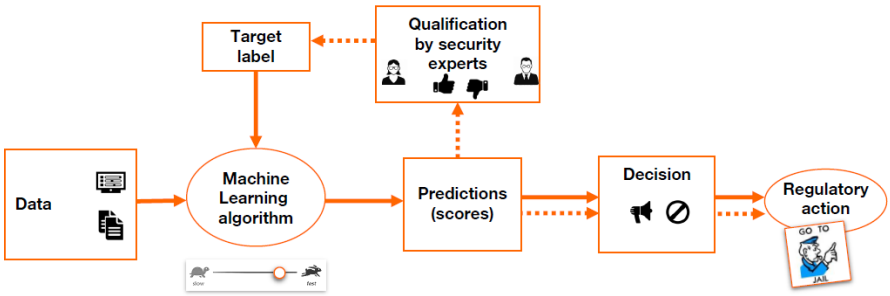


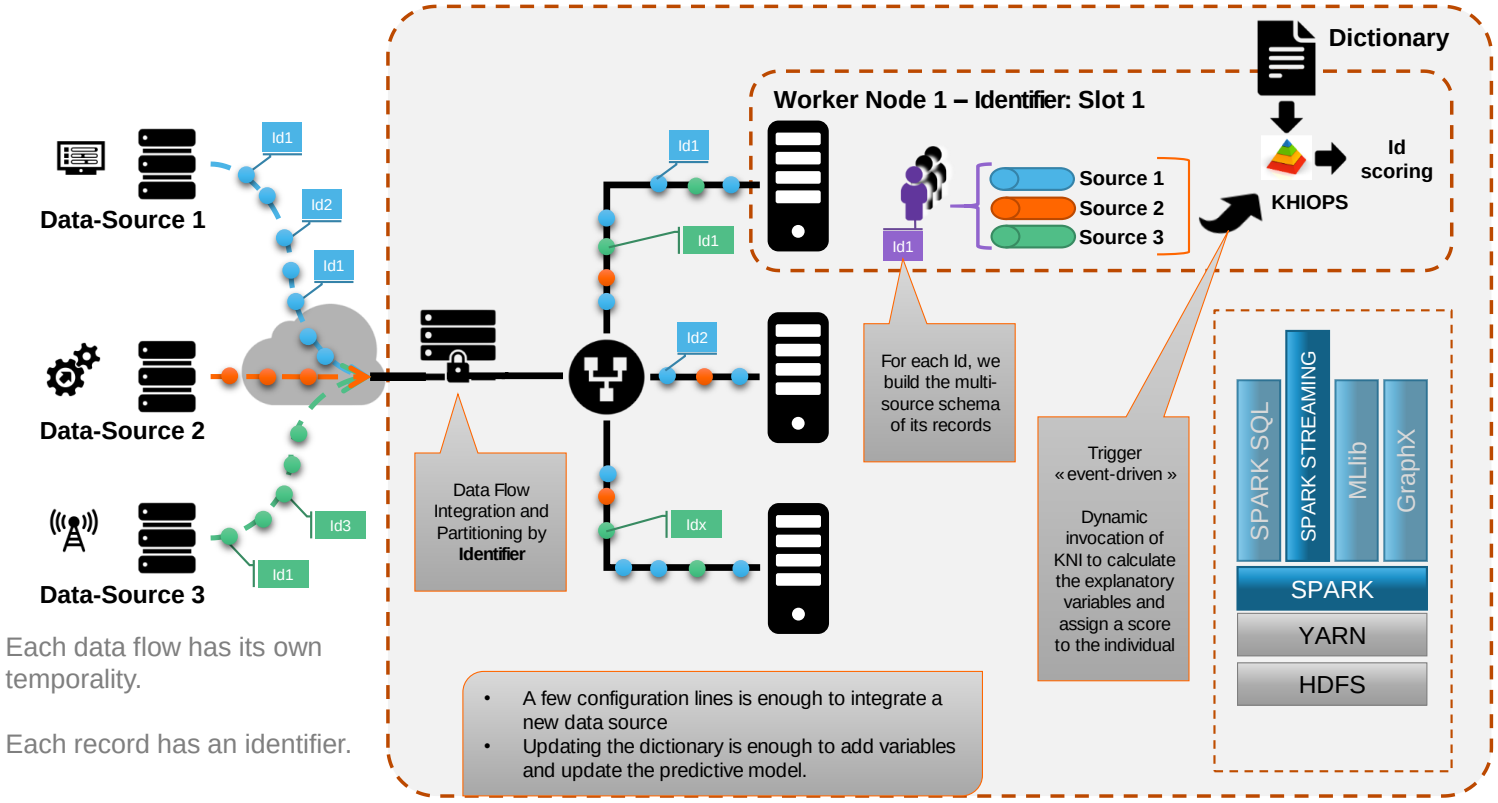
Figure 2 - processus d'intégration d'un algorithme de Machine Learning

Cette démarche est itérative : il s'agit non seulement d'améliorer l'algorithme mis en production en améliorant sa qualité, mais aussi de prendre en compte la mutation possible du comportement de l'attaquant, rendant ainsi inefficaces les algorithmes utilisés à un instant T. L'intégration et la mise à jour d'algorithmes de détection est donc une opération récurrente qui doit intégrer les contraintes opérationnelles des SOC. Cela se traduit par le cahier des charges suivant :

- Temps réel : l'algorithme de Machine Learning doit qualifier les flux en temps réel, à quelques secondes près
- Features représentatives : les features doivent capturer l'information pertinente à l'instant T, les méthodes d'attaque évoluant dans le temps

- Mise à jour aisée : on peut facilement et rapidement mettre à jour l'algorithme de détection
- Données récentes : les experts métier ont accès aux données les plus récentes pour évaluer et tester de nouveaux algorithmes.

Nous répondons à ce besoin en adoptant une architecture modulaire sur un environnement Big Data, comme le montre la Figure 3. Chacun des modules de la solution permet d'adresser une des étapes de l'intégration d'un algorithme de Machine Learning : cette correspondance est décrite dans le Tableau 1.



Each data flow has its own temporality.

Each record has an identifier.

Figure 3 - Architecture modulaire d'intégration d'algorithmes de Machine Learning

Etapes du processus	Déclinaison du processus dans notre retour d'expérience
Identification des sources de données	Collecte générique des sources d'événements et de sources de données complémentaires par un système indépendant (« *je choisis mes sources de données et la profondeur d'analyse de chacune (1 semaines glissante par exemple) que je souhaite exploiter pour construire mes variables.* ») L'ajout de nouvelles sources de données se fait grâce à un fichier de configuration : pour chaque source de données, on définit la clé de jointure avec les autres données et la profondeur que l'on souhaite valoriser
Construction de variables explicatives (ou features) depuis une ou plusieurs des sources disponibles	Construction de variables explicatives (features) grâce à un méta-langage. Par exemple : maVariable = EcartType(TableIP, DUREE_DES_SESSIONs) NombreDeSitesVus = TableCountDistinct(TableDNS) Ce langage permet de créer les variables dérivées d'éléments existants via des fonctions logiques ou mathématiques classiques (somme, moyenne, médiane, jointure, appartenance à une liste, etc) et chainables : les possibilités sont donc très étendues.
	Construction et exploration automatique de variables explicatives
Production d'un modèle prédictif avec création de KPI grâce à une partie du dataset utilisé uniquement pour le test.	Création de modèles prédictifs grâce aux variables explicatives disponibles
	Pré-validation des modèles sur les données connues grâce à la courbe ROC, l'accuracy, le rappel, la précision, par exemple.
Valorisation du modèle prédictif	Qualification des modèles sur le flux temps réel
	Retours des expert métiers par rapport aux labels produits.

Tableau 1 – Déclinaison opérationnelle du processus de mise en production d'un algorithme de Machine Learning

Amélioration de la boucle de rétroaction et gain de temps

Grâce à la mise en place de cette infrastructure technique, les phases exploratoires, telles que la phase de développement pour l'analyse descriptive et pour la création de features, sont simplifiées. Le module de Machine Learning (Khiops ici) est intégré dans la chaine de delivery ce qui permet de construire et de choisir automatiquement les features pertinentes. Ce mode est utile autant pour les phases de compréhension de la problématique que pour la phase de production de modèles « autonomes ».

Cela se traduit par un gain de temps. Auparavant, l'ajout de nouvelles features complexes[2] induisait un délai parfois de plusieurs semaines lorsqu'une mise à jour logicielle était nécessaire (développement, test, intégration, déploiement). Ici, cela devient immédiat et sans impact sur la production.

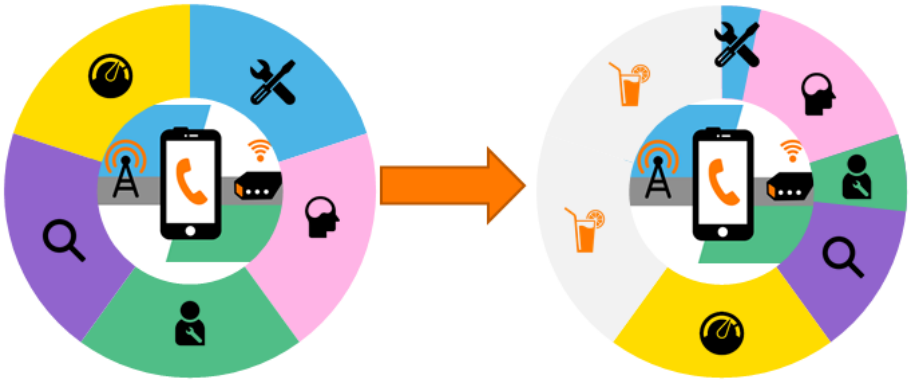


Figure 4 – L'automatisation des tâches intermédiaires (développement et intégration) libère du temps d'analyse

De manière plus générale, cette approche permet de répondre à notre cahier des charges :

- Temps réel : la production d'une prédiction statistique (comprenant le calcul des features et leurs valorisations par un modèle bayésien) s'effectue en temps réel. Pour réduire le nombre de prédictions, par exemple pour des questions de performance, on pourra l'associer à un critère interne ou externe.
- Features représentatives : l'expert métier indique les sources de données pertinentes et notre brique de Machine Learning, permet de générer facilement de multiples variables dérivées, et d'en retenir les plus représentatives uniquement

[2] Nécessitant de nouvelles sources de données et/ou des calculs intermédiaires ou complexes.

- Mise à jour aisée : la mise à jour de l'algorithme ne nécessite pas de modification logicielle lourde
- Données récentes : les flux temps réels sont utilisés pour la phase de validation du modèle

Ce mode permet de démontrer facilement l'intérêt des nouvelles features créées automatiquement. En revanche, il ne permet pas d'identifier toutes les informations pertinentes pour la résolution de la question métier : seul l'expert métier, ici l'analyste SOC, pourra apporter cet éclairage et proposer de manière pertinente de nouvelles sources de données et des variables en entrée.

En conclusion, cette flexibilité d'intégration permet des évolutions plus aisées de nos modèles avec des gains immédiats. Le SOC réduit les délais de mise en œuvre des nouveaux modèles et devient donc plus réactif face à de nouvelles attaques.

Les limites

On note que cette approche ne s'intéresse qu'aux alertes remontées jusqu'alors, et ne permet donc pas d'améliorer la détection de cas qui nous sont inconnus. Sont plus adaptés à ce cas d'usage des algorithmes de détection d'anomalies ou des algorithmes de clustering, qui mettent en avant des points « différents » des autres (en fonction de caractéristiques à identifier), points révélateurs d'un événement statistiquement anormal : potentiellement une attaque.

Par ailleurs, la démarche exposée suppose qu'un travail conséquent a été réalisé en amont par le SOC afin de classer les vrais et faux positifs générés par les systèmes de détection en place. En particulier, cela suppose que l'analyste saura finalement conclure sur la nature malveillante d'un cas qui lui est présenté : cela n'est malheureusement pas toujours possible (cas d'informations parcellaires).

Conclusion

En conclusion, le Machine Learning permet d'alléger la charge de certaines opérations du SOC. Pour accroître notre efficacité, il est nécessaire d'être au plus près des données terrains. Pour ce faire, des infrastructures adaptables permettant d'intégrer facilement les fonctionnalités recherchées sont essentielles. La simplification de la construction de variables explicatives, la construction systématique de variables explicatives à partir des variables initialement déclarées, et l'entrainement automatique de multiples modèles sur ce jeu de données rendent plus accessibles et accélèrent des étapes chronophages de la conception d'algorithmes de Machine Learning. L'expert métier peut alors se concentrer sur les prédictions réalisées par les modèles sur le flux temps réel et statuer sur la maturité du modèle avant mise en production éventuelle. Le découpage fonctionnel apporte ici les premières briques pour plus d'agilité et de réactivité.

Lexique

AUC	Area Under Curve
AUROC	Area Under ROC Curve – cf AUC
ROC	Receiver Pperating Characteristic
SOC	Security Operations Center

Travaux cités

[1] MITRE, Ten Strategies of a World-Class Cybersecurity Operations Center, 2014.

[2] «PolynomialFeatures,» [En ligne]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html.

[3] «Featuretools,» [En ligne]. Available: https://docs.featuretools.com/index.html.

[4] «glmulti,» [En ligne]. Available: https://www.rdocumentation.org/packages/glmulti/versions/0.5-1/topics/glmulti.

[5] M. Boullé, «Towards Automatic Feature Construction or Supervised Classification,» *ECML/PKDD 2014*, p. pp. 181–196, 2014.

[6] M. Boullé, C. Charnay et N. Lachiche, «A scalable robust and automatic propositionalization approach for Bayesian classification of large mixed numerical and categorical data,» *Machine Learning*, 2018.

[7] A. Bonneton et A. Husson, «Le Machine Learning confronté aux contraintes opérationnelles des systèmes de détection,» chez *SSTIC*, 2017.

[8] «Cross-validation: evaluating estimator performance,» [En ligne]. Available: http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation-iterators-with-stratification-based-on-class-labels.