UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Augusto Fernando de Melo Simião

Análise da Utilização de Aprendizado de Máquina na Redução do Volume de Alertas Benignos

Recife

2019

Augusto Fernando de Melo Simião

Análise da Utilização de Aprendizado de Máquina na Redução do Volume de Alertas Benignos

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Curso de Bacharelado em Sistemas de Informação

Orientador: Rodrigo F. G. Soares

Recife

2019

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Biblioteca Central, Recife-PE, Brasil

S589a Simião, Augusto Fernando de Melo
Análise da utilização de aprendizado de máquina na redução do volume de alertas benignos / Augusto Fernando de Melo Simião. – 2019.
46 f. : il.

Orientador: Rodrigo F. G. Soares.
Trabalho de Conclusão de Curso (Graduação) – Universidade Federal Rural de Pernambuco, Departamento de Estatística e Informática, Recife, BR-PE, 2019.
Inclui referências e apêndice(s).

1. Aprendizado do computador 2. Sistemas de recuperação da informação – Segurança 3. Software – Proteção 4. Cibernética I. Soares, Rodrigo F. G., orient. II. Título

CDD 004

Dedico este trabalho à aqueles que, apesar das adversidades, nunca desistem de seguir seus sonhos.

Agradecimentos

Ao meus pais, que prezaram pela minha educação e sempre estiveram comigo nos momentos mais difíceis.

Ao Professor Rodrigo Soares, que me orientou na realização do meu Trabalho de Conclusão de Curso.

À Ana Karolina, que sempre me apoiou nas decisões e momentos difíceis, nesses últimos seis anos que estamos juntos.

À Intrinium, Inc, que cedeu a base de dados utilizada nos experimentos e por servir de inspiração para compor este trabalho.

Aos meus colegas de trabalho, na Intrinium, que me deram apoio moral e cobriram os turnos que eu precisei me ausentar, durante a escrita do meu Trabalho de Conclusão de Curso.

Ao diretor do departamento de *Managed Security Services*, Sahan Fernando, que sempre reconheceu meus esforços na Intrinium, Inc e por oferecer todo o suporte possível para a conclusão do meu curso.

Ao Vice-Presidente da Intrinium, Inc, Stephen Heath, que em 2015 acreditou em mim e me ofereceu a posição de Analista de Segurança na Intrinium, onde atualmente trabalho.

“Nossa maior glória não está em jamais cair, mas em levantar a cada queda.”
(Confúcio)

Resumo

Para auxiliar no combate a ataques cibernéticos, *Managed Security Services Providers* (MSSPs) usam SIEMs (*Security Information and Event Management*). SIEMs são capazes de agregar, processar e correlacionar vastas quantidades de eventos provenientes de diferentes sistemas, alertando analistas de segurança da existência de ameças, tais como vírus de computador e ataques cibernéticos, em redes de computadores. No entanto, SIEMs são conhecidos pelas altas taxas de alertas benignos (alertas que não representam ameaça) em relação aos malignos (alertas que representam ameaça). Devido aos altos volumes e predominância de falsos alertas, o analista passa a ignorar alertas como um todo, o que inclui aqueles que representam incidentes em potencial, aumentando assim o risco da rede ser comprometida. Esse fenômeno é conhecido como fadiga de alerta e tem sido alvo frequente da aplicação de técnicas de aprendizado de máquina para a redução dos volume de alertas benignos. SIEMs modernos utilizam aprendizado de máquina, na correlação de eventos, para que apenas alertas que realmente representam possíveis ameaças sejam reportados. No entanto, essa correlação não leva em conta a deliberação do analista de segurança, permitindo assim que os SIEMs continuem gerando alertas previamente identificadas como benignos. Este trabalho investiga a utilização dos algorítimos *Naïve Bayesian Learning*, Árvore de Decisão e *Random Forest*, para a redução do volume de alertas benignos, utilizando alertas previamente identificados por analistas, ao invés da corrente de eventos que geram tais alertas. Dessa forma, foi possível mostrar, através de experimentos, que técnicas de aprendizado de máquina supervisionado podem ser aplicadas na identificação e alertas benignos previamente analisados.

Palavras-chave: SIEM, Fatiga de Alerta, Aprendizado de Máquina.

Abstract

To aid in combating cyber attacks, Managed Security Services Providers (MSSPs) use SIEMs (Security Information and Event Management). SIEMs are able to aggregate, process and correlate vast amounts of events from different systems, alerting security analysts of the existence of threats, such as computer viruses and cyber attacks, in computer networks. However, SIEMs are known for the high rates of benign alertas (non-threatening alerts) warnings relative to malign alerts (threatening alerts). Due to the high volumes and prevalence of benign alertas, the analyst ignores alerts as a whole, which includes those that represent potential threats, thereby increasing the risk of a network compromise. This phenomenon is known as alert fatigue and has been a frequent target of applying machine learning techniques to reduce the volume of benign alerts. Modern SIEMs use machine learning, in correlation of events, so that only alerts that actually represent possible threats are reported. However, this correlation does not consider the analyst's deliberation, thus allowing SIEMs to continue to generate alerts previously identified as benign. This paper investigates the use of the algorithms Naïve Bayesian Learning, Decision Tree and Random Forest, to reduce the volume of benign alerts using alerts previously identified by analysts, rather than the chain of events that generate such alerts. In this way, it was possible to show, through experiments, that supervised machine learning techniques can be applied in the identification of alerts previously identified as benign.

Keywords: SIEM, Alert Fatigue, Machine Leaning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AD	*Active Directory.*
ADT	*Advanced Threat Defense.*
CPU	*Central Processing Unit.*
MD5	*Message Digest 5.*
MSS	*Managed Security Services.*
MSSP	*Managed Security Services Provider.*
NLP	*Natural Language Processing.*
RAM	*Random Access Memory.*
SHA	*Secure Hash Algorithm.*
SIEM	*Security Information and Event Management.*
USM	*Unified Security Manager.*

Sumário

1 Introdução

A aplicação de métodos e ferramentas que garantam a segurança da informação é indispensável em um mundo onde pessoas e organizações são vulneráveis a ataques cibernéticos (VILLANO, 2018). Para auxiliar no combate a esses ataques, *Managed Security Services Providers* (MSSPs) utilizam ferramentas conhecidas como SIEMs (*Security Information and Event Management*). Os SIEMs são capazes de agregar, processar e analisar vastas quantidades de eventos provenientes de diferentes sistemas, alertando em tempo-real da existência de anomalias em redes de computadores (ZEINALI, 2016). Os alertas acionados por SIEMs são investigados por um analista de segurança que determina se um alerta é maligno (alerta que representa ameaça) ou benigno (alerta que não representa ameaça) (SINGH; NEME, 2013).

SIEMs, no entanto, são conhecidos por gerar altas quantidades de alertas benignos (SPATHOULAS; KATSIKAS, 2009). Isso acontece principalmente devido à falta de acurácia em uma grande parte dos SIEMs no mercado, que utilizam regras de detecção heurística de anomalias (LEE; ZHANG; XU, 2013). Como consequência, o analista de segurança dedica a maior parte de seu tempo analisando alertas que não representam uma ameaça. Além da perda de tempo com alertas irrelevantes, o analista atinge um estado de exaustão, conhecido como fadiga de alerta, que afeta sua capacidade de investigar alertas corretamente (HASSAN; AL., 2009). Por conta disso, ameaças cibernéticas têm maiores chances de penetrar as defesas de uma rede.

Alertas de segurança, no entanto, contém indicadores provenientes do processo de captura de pacotes de rede (NARAYAN; PARVAT, 2015), bem como a assinatura do próprio alerta (TREINEN, 2009). A frequência desses indicadores e a deliberação do analista para com a postura (maligno ou benigno) de um alerta serviram de base para um processo de detecção automática de alertas benignos, utilizando aprendizado de máquina supervisionado. Neste trabalho, investiga-se a utilização dos algorítimos de aprendizado de máquina supervisionado *Naïve Bayesian Learning*, Árvore de Decisão e *Random Forest*, na detecção de alertas benignos recorrentes.

1.1 Motivação e Relevância

Um dos maiores impactos da fadiga de alerta é o aumento do risco de uma empresa ser comprometida por uma ameaça cibernética, o que pode custar milhões de dólares em danos com reparos de infraestrutura e multas (MASTERS, 2017). Uma pesquisa realizada pelo FireEye Institute mostra que, devido à fadiga de alerta, apenas 4% dos alertas são propriamente analisados (FIREEYE INSTITUTE, 2019). Isso significa

que pelo menos 96% dos alertas são ignorados ou não são devidamente investigados. Outra pesquisa, realizada pela Skyhigh Networks, indica de aproximadamente 32% dos analistas de segurança ignoram alertas regularmente (NETWORKS, 2017). Isso indica que, um analista sofrendo com a fadiga de alerta não dará a devida atenção à investigação de alertas que possam representar uma ameaça em potencial, rendendo a empresa vulnerável a ataques e infecções. Portanto, a redução do volume de alertas benignos é essencial na manutenção da segurança digital de uma empresa.

Algorítimos de aprendizado de máquina são aplicadas em vários setores de indústria na identificação de padrões de maneira autônoma. Algorítimos tais como o *Naïve Bayesian Learning* e Árvores de Decisão, por exemplo, permitem que filtros de *e-mail* "aprendam"a bloquear *e-mails* maliciosos (*spams*), através da identificação de palavras e expressões comuns contidas nesses *e-mails* (CHAKRABORTY; MONDAL, 2012). De maneira similar, alertas benignos também apresentam indicadores em comum que o analista de segurança considera durante o processo investigativo, como endereço de IP e nome de usuário, por exemplo. A recorrente necessidade de investigar os mesmos alertas benignos é a causa a fadiga de alerta.

A alta frequência de indicadores comuns em alertas benignos sugere que, como na identificação de *spams*, técnicas de aprendizado de máquina supervisionado podem ser utilizadas para identificar alertas benignos automaticamente. A existência de um modelo aprendizado de máquina, tornaria possível a detecção automática e redução do volume de alertas benignos. Além disso, o analista economizaria o tempo gasto em alertas considerados irrelevantes, permitindo o foco na investigação de ameaças reais. Para isso, deve-se comparar diferentes algorítimos de aprendizado de máquina, a fim de encontrar o modelo que ofereça a maior acurácia na detecção de alertas benignos.

1.2 Objetivos

Este trabalho tem como objetivo comparar os algorítimos de aprendizado de máquina supervisionado *Naïve Bayesian Learning*, Árvore de Decisão e *Random Forest*, para encontrar o modelo que ofereça a melhor precisão na identificação automática de alertas benignos, provenientes do SIEM AlienVault, na Intrinium. A existência de tal algorítimo teria a capacidade de reduzir o alto volume de alertas benignos, na Intrinium. Além disso, uma redução significativa dos alertas benignos tornaria relevante o estudo da aplicação dos métodos deste trabalho em outros SIEMs e empresas.

2 Identificando Alertas Benignos

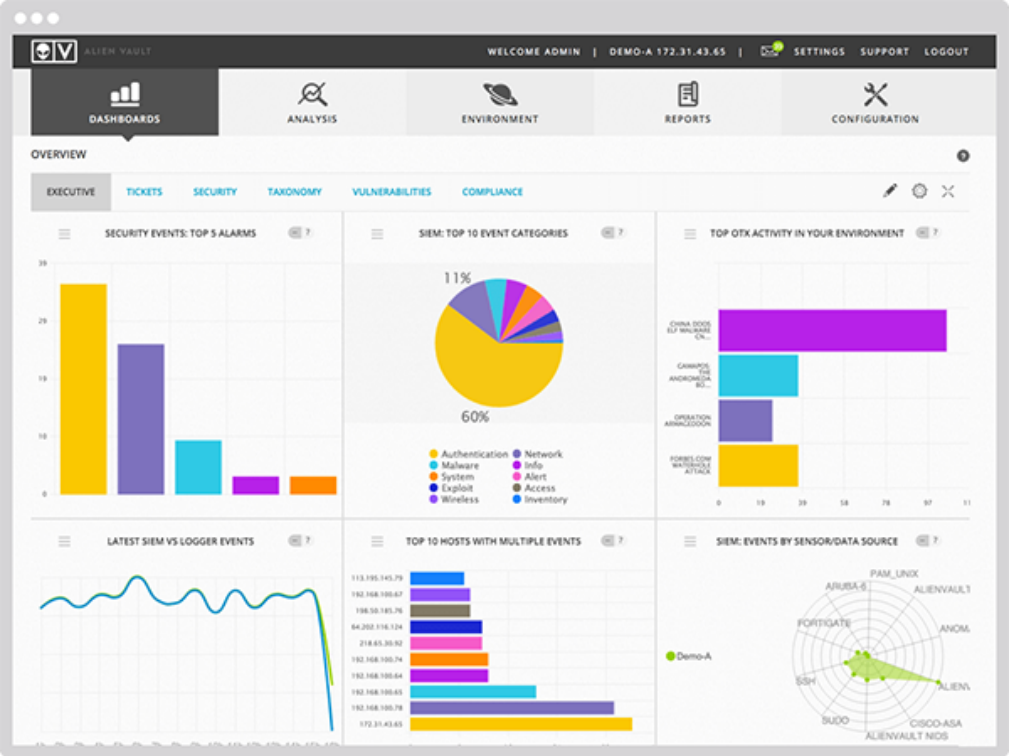
A empresa Intrinium é um MSSP que opera no mercado estadunidense de segurança da informação. Atualmente, a Intrinium utiliza a plataforma SIEM AlienVault USM (ALIENVAULT INC, 2019), para monitorar sua rede interna e as redes de clientes. A Figura 1 mostra a tela principal da plataforma. Assim como muitos SIEMs do mercado, o AlienVault USM gera um grande volume de alertas benignos. O AlienVault USM utiliza uma abordagem de detecção heurística, onde eventos são correlacionados com base em suas frequências dentro em um dado período de tempo.

De forma geral, o método heurístico de detecção é uma das causas conhecidas do excesso de alertas benignos e fadiga de alerta (LEE; ZHANG; XU, 2013). Além disso, as regras de detecção do AlienVault são estáticas, criadas pela empresa que distribui a plataforma. Essas regras são imutáveis e nem sempre permitem que se especifique quando uma série de eventos representa uma ameaça. Essa impossibilidade de adaptar as regras existentes gera uma torrente de alertas benignos que não podem ser suprimidos. Nesse caso, a supressão de um certo tipo de alerta significaria eliminar a possibilidade de capturar ameaças. Dessa forma, para garantir que ameças em potencial não passem despercebidas, a Intrinium permite que altos volumes de alertas benignos sejam gerados, levando os analistas a experienciarem a fadiga de alerta.

SIEMs avançados utilizam aprendizado de máquina na detecção de anomalias (RAJBANSHI; BHIMRAJKA; RAINA, 2017). No entanto, os alertas provenientes dessas anomalias ainda necessitam de uma revisão manual, tornando a figura do analista indispensável na detecção de alertas. (HUBBARD; SEIERSEN, 2016). Os efeitos negativos do alto volume de alertas benignos, no entanto, dificultam a investigação dos alertas malignos, rendendo a Intrinium e seus clientes vulneráveis a ataques.

Após o analista identificar uma série de alertas benignos repetidamente, é possível eles possam ser identificados e suprimidos automaticamente, com base em padrões compartilhados por esses alertas. Isso porque alertas benignos apresentam combinações de atributos mais frequentes que aqueles encontrados em alertas malignos. Diferenciar entre alertas malignos e benignos é a função do analista, o qual possui conhecimento prévio de quando certos eventos representam ou não ameaças. Na Tabela 1, é possível observar um exemplo prático da diferença entre um alerta maligno e benigno. Na Tabela 1, um alerta benigno ocorre quando o usuário *Jason Tots* (*username_source*) apresenta várias falhas de autenticação (*Multiple Authentication Failures*) em seu próprio computador (*source_ip_hostname* = *destination_ip_hostname*), durante o período da manhã (*isAfterhours* = *No*). Essa falhas de autenticação ocor-

Figura 1 – AlienVault - Página Inicial



(ALIENVAULT INC, 2019)

Tabela 1 – Exemplos de Alertas

Atributo	Falso Alerta	Incidente
plugin_sid	4735	4735
plugin_id	1505	1505
src_ip_hostname	192.168.22.22	192.168.22.22
dst_ip_hostname	192.168.22.22	192.168.22.25
target_ad_domain	Nulo	Nulo
source_ad_domain	INTRINIUM	INTRINIUM
target_ad_group	HRUSERS	HRUSERS
username_target	Nulo	Nulo
username_source	Jason Tots	Jason Tots
process_name	Nulo	Nulo
file_name	Nulo	Nulo
isAfterhours	*No*	*Yes*
postura/classe	**Falso Alerta**	**Incidente**

rem com frequência e não representam atividade maliciosa. Ainda na Tabela 1, o exemplo de alerta maligno mostra o mesmo usuário utilizando o próprio sistema para acessar remotamente o computador de outro usuário *source_ip_hostname* $\neq$ *destination_ip_hostname*) e no período da noite (*isAfternours* = *Yes*). Nesse caso, tais falhas de autenticação são normalmente provenientes de um vírus utilizando o computador de um usuário (nesse caso Jason Tots) para invadir computadores alheios.

Alertas malignos, no entanto, tendem a ser menos frequentes e apresentam valores de parâmetros geralmente diferentes dos encontradas em alertas benignos. Nos exemplos da Tabela 1, a recorrência das falhas de autenticação pelo usuário *Jason Tots*, em seu próprio computador e no período da manhã, constituem um padrão de alertas benignos. Esses poderiam ser detectados e suprimidos automaticamente, uma vez que não constituem atividade maliciosa. Para automatizar o processo de detecção automática de alertas benignos, portanto, sugere-se a utilização de técnicas de aprendizado de máquina supervisionado, devido a capacidade que possuem em identificar padrões a partir de um conjunto de treinamento (histórico de alertas benignos).

Assim, pergunta-se: qual modelo de aprendizado de máquina possui a melhor precisão na detecção de alertas benignos? Devido à vasta quantidade de algorítimos existentes, este trabalho limitou-se ao algorítimos de aprendizado de máquina supervisionado *Naïve Bayes Learning*, Árvore de Decisão e *Random Forest*, a fim de encontrar o algorítimos que ofereça a maior precisão na detecção de alertas benignos.

2.1 Desafios e Limitações

Um dos desafios de se aplicar técnicas de aprendizado de máquina, neste trabalho, diz respeito à fadiga de alertas. Por conta desse fenômeno, os analistas não investigam alertas corretamente, causando inconsistências na base de dados. Tais inconsistências se revelam na forma de alertas malignos erroneamente identificados como benignos, além de alertas malignos e benignos que compartilham os mesmos valores de atributos. Além disso, o número de alertas benignos é consideravelmente superior ao de alertas malignos, o que torna a base de dados desbalanceada. Tal discrepância, aliada às inconsistências na base de dados, têm o potencial de afetar a acurácia dos modelos de aprendizado de máquina supervisionados investigados.

Uma forma de resolver esses problemas seria comparar os resultados dos algorítimos com a investigação dos analistas, em tempo-real. Assim, os erros de detecção de um algorítimo, poderiam ser corrigidos a fim de aumentar sua acurácia. Outra maneira de solucionar esses problemas seria realizar um processo de mineração de dados, a fim de coletar amostras de alertas investigados corretamente. Dessa forma, seria possível o treinamento de modelos com uma base de dados mais balanceada e consistente, o que melhoraria a acurácia das detecções automáticas.

Finalmente, modelos de aprendizado de máquina, como o *Random Forest*, requerem um alto tempo computacional e memória, durante a fase de treinamento. Para os experimentos propostos, o hardware utilizado foi limitado, fazendo com que os passos necessários para o treinamento do modelo utilizando o algorítimo *Random Forest* levasse no mínimo 18 horas. A utilização de um hardware mais robusto, como um

servidor dedicado, seria uma solução.

3 Trabalhos Relacionados

A correlação de eventos (*logs*) para a geração de alertas é uma aplicação comum do aprendizado do máquina, na industria de segurança da informação. SIEMs correlacionam eventos para aumentar a precisão das detecções de ameaças em potencial. Villano (2018) demonstra como árvores de decisão podem ser utilizadas para correlacionar eventos e identificar ataques. Porém, a existência de novos ataques impõem desafios na criação de modelos de aprendizado de máquina confiáveis (VILLANO, 2018). Isso sugere que novos modelos devam ser desenvolvidos, a medida que novos ataques se tornam proeminentes. Além disso, assim como SIEMs em geral, a técnica sugerida pelo autor não leva em consideração a deliberação do analista, dando prioridade a pressuposição de que certos eventos, em conjunto, são mais ou menos prováveis de coincidirem com alertas malignos.

Outro método aplicado pela indústria para reduzir o volume de alertas benignos, assim como na redução da fadiga de alerta, é conhecido como *Provenance Triage*. Esse método consistem identificar toda a corrente de eventos que levaram a um ataque, através do processo de *backtracking* (HASSAN; AL., 2009). Dessa forma o analista pode atestar a relevância de um alerta baseado na lista completa de eventos que levaram o alerta. Embora eficiente, este método é exaustivo, pois necessita da análise manual de grandes quantidades de eventos (LEE; ZHANG; XU, 2013).

O NoDoze é uma *software* desenvolvido por Lee, Zang e Xu (2013), proposto como uma maneira de automatizar o processo de *Provenance Triage*, de forma a torná-lo menos exaustivo e mais eficiente. No entanto, o NoDoze é altamente dependente do método de detecção utilizado pelo SIEM que origina os alertas. A vantagem do NoDoze, esta em possibilitar que o analista veja toda a corrente de eventos que gerou o alerta de forma automática. Ainda assim o NoDoze também pressupõe a probabilidade de um ou mais eventos representarem um alerta maligno em potencial. Isso ocorre sem que o analista realize uma investigação prévia desses eventos.

Para aumentar a precisão nas classificações, é possível utilizar uma combinação de técnicas de aprendizado de máquina (KITTLER; HATED; DUIN, 1998). Essa abordagem é utilizada por Narayan e Parvat (2015) que utiliza um modelo hierárquico composto de Árvore de Decisão e *Naïve Bayesian Learning* para classificar *logs* e determinar se estes representam incidentes de segurança. Assim como nos trabalhos citados anteriormente, assume-se que certos eventos são mais ou menos prováveis de indicar uma intrusão ou infecção, sem que haja uma análise prévia do analista.

Além das estratégias baseadas em aprendizado de máquina, é possível reduzir

os volumes de alertas benignos através métodos heurísticos de decção, tais como o *Misuse Detection*. Esse método é sugerido por Beng et al. (2014) e consiste em gerar alertas com base em correntes de eventos específicos, que juntos, têm uma alta probabilidade de representar uma ameça. Nesse caso há conhecimento prévio de quando um conjunto de eventos representa uma ameaça. O resultado dessa abordagem é uma redução significativa do volume de alertas benignos. (BENG et al., 2014).

No entanto, o *Misuse Detection* é fortemente limitado a um conjunto de regras de correlação pre-determinadas. Isso significa que novos ataques cibernéticos, também conhecidos como *0-days*, podem não reportar alertas (BENG et al., 2014), devido à falta de regras de correlação específicas. SIEMs baseados em *Misuse Detection*, portanto, precisam ser frequentemente atualizados, uma vez que estes dependem especificamente das regras de detecção pre-definidas.

Os trabalhos citados têm em comum o foco na análise dos eventos correlacionados por SIEMs. Essa abordagem focada predominantemente em eventos não considera o julgamento do analista, ao invés disso, assume-se que certos alertas são intrinsecamente incidentes em potencial, sem que haja um análise mais contextual, que apenas o analista pode prover (HUBBARD; SEIERSEN, 2016). Tal análise leva em conta fatores particulares das empresas, tais como quais computadores contém dados sigilosos, quais usuários são permitidos realizar certas operações e quais programas podem ou não ser instalados em computadores.

Por não considerarem a determinação do analista quanto à postura (benigno ou maligno) dos alertas, os métodos dos autores citados não possuem a capacidade adaptar os modelos de aprendizado de máquina a medida que os alertas são identificados como malignos ou benignos. Como consequência, tais métodos ainda permitem a geração de alertas benignos frequentes. Dessa forma, uma abordagem que considere a deliberação do analista seria de grande valor na redução dos números de alertas benignos, servindo inclusive como complemento aos métodos sugeridos pelos autores.

4 Métodos

Os conceitos, terminologias e algorítimos de aprendizado de máquina utilizados são descritos nas sessões a seguir em detalhes. Inicialmente explicam-se os conceitos básicos envolvendo aprendizado de máquina e como eles se aplicam neste trabalho. Em seguida, são expostos os respectivos algorítimos utilizados nos experimentos.

4.1 Conceitos e Terminologia

Aprendizado de máquina pode ser definido um processo complexo de processamento computacional aplicado ao reconhecimento de padrões e tomada de decisão (DUA; XIAN, 2011). De maneira geral, o aprendizado de máquina pode ser supervisionado ou não-supervisionado. No caso do aprendizado supervisionado, o algorítimo de aprendizado de máquina aprende como classificar objetos, baseado em um base de dados contendo objetos pré-classificados, através de uma fase de treinamento. Em contraste, no aprendizado não-supervisionado, o algorítimo não passa pela fase de treinamento e consequentemente não tem conhecimento prévio dos objeto a serem classificados (VILLANO, 2018). A classificação de objetos tem como objetivo a atribuição de rótulos, também conhecidos como classes, as quais representam o conjunto de rótulos atribuídos às instâncias, por um algorítimo classificador (GOOGLE LLC, 2019).

Uma instância pode ser definida como sinônimo de exemplo. Neste trabalho, as instâncias representam alertas de segurança, os quais algorítimos de aprendizado de máquina são aplicados para identificar alertas benignos. As instâncias descritas neste trabalho são divididas em duas classes: benigno e maligno. Em segurança da informação, um alerta benigno pode ser definido como uma falha de detecção de incidentes, quando SIEMs, ou qualquer outra plataforma de detecção de anomalias, interpretam como anomalia uma corrente de eventos que não representa ameaça. Quando a quantidade de alertas benignos supera consideravelmente o numero de alertas malignos, o analista de segurança corre o risco de experienciar a fadiga de alertas. Em contrapartida, alertas malignos são eventos causados por atividades potencialmente maliciosas. Durante a investigação de alertas, analistas de segurança identificam alertas malignos quando estes causam danos ou representam uma ameaça real. Alertas que não resultaram em danos, mas demandam uma investigação mais profunda, também são considerados malignos, até que a devida investigação seja realizada. O objetivo principal dos SIEMs é garantir que incidentes sejam detectados em tempo hábil. No entanto, para isso os SIEMs tendem a sacrificar a acurácia na detecção de ameaças, em prol de uma maior cobertura na correlação de eventos. O resultado dessa abordagem é a

geração de altas quantidades de alertas benignos.

Algorítimos de aprendizado de máquina requerem atributos. Estes podem ser definidos como aspectos de uma instancia (CENTRE FOR ADVANCED INTERNET ARCHITECTURE, 2011), ou o conjunto de valores que definem uma instancia. O tipo de atributo (nominal ou numérico) (WINTTEN; FRANK, 2005) define o algorítimo de aprendizado de máquina utilizado, uma vez que nem todos os algorítimos de aprendizado de máquina não dão suporte a dados nominais. Os atributos desse experimento, por exemplo, são todos nominais, por serem provenientes de um sistema de análise de anomalias (AlienVault USM) de rede baseado em indicadores nominais, tais como endereço de IP e nome de usuário. Trabalhar com tais atributos, portanto, requer a aplicação de algorítimos que suportem valores nominais.

Finalmente, as instâncias utilizadas no experimento foram submetidos a uma amostragem estratificada. Tal amostragem consiste em dividir a base de dados em conjuntos de treinamento, teste e validação, garantindo que as classes fossem proporcionalmente representadas entre conjuntos (WINTTEN; FRANK, 2005) A estratificação é útil em bases de dados pequenas ou que uma amostragem randômica teria o risco de omitir uma ou mais classes nos conjuntos de teste a validação. No caso da base de dados utilizada neste trabalho, o número de alertas benignos é consideravelmente superior aos malignos, de forma que uma amostragem randômica, por exemplo, traria o risco de omitir alertas malignos, durante treinamento, validação ou teste.

4.2 Algorítimos de Aprendizado de Máquina

Neste trabalho são aplicadas os algorítimos de aprendizado supervisionado *Naïve Bayesian Learning*, *Random Forest* e Árvore de Decisão, os quais os conceitos correspondentes são explicado em mais detalhes adiante. Esses algorítimos foram escolhidos para compor os experimentos por darem suporte a valores nominais, tais como os que compõem as instâncias utilizadas nos experimentos desse trabalho.

4.2.1 *Naïve Bayesian Learning*

O *Naïve Bayesian Learning* é um algorítimo de aprendizado de máquina supervisionado, que emprega o Teorema de Bayes e o princípio da alta dependência de atributos para calcular a probabilidade *a posteriori* de um conjunto de instâncias (MITCHEL, 1997)). O primeiro passo do algorítimo *Naïve Bayesian Learning* é determinar o número total de classes e calcular as probabilidades condicionais de cada atributo (RASCHKA, 2018). Formalmente, o teorema de Bayes pode ser definido pela Figura 2, onde *P(h|D)* se refere à probabilidade *a posteriori* de uma instância pertencer a uma classe *h*, dado que um certo atributo *D* ocorreu (MITCHEL, 1997). Usando a aborda-

gem probabilística do *Naïve Bayesian Learning* é possível identificar novos alertas, com base na frequência dos valores de certos atributos.

Figura 2 – Equação da Probabilidade Condicional

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

(MITCHEL, 1997)

Onde:

- *P(h)* = Probabilidade da classe *h*.
- *P(D)* = Probabilidade do atributo *D*.
- *P(h|D)* = Probabilidade *a priori* do atributo *D*, dado que uma classe *h* ocorreu.
- *P(D|h)* = Probabilidade *a posteriori* da classe *h*, dado que o atributo *D* ocorreu.

A Tabela 2 mostra exemplos de frequências com que alertas foram identificados, por analistas, como malignos ou benignos com base na ocorrência de certos endereços de IP. A Tabela 3 mostra os valores de *P(D)*, *P(h)* e *P(D|h)*, com base nas frequências da Tabela 2, necessário para calcular as probabilidades *a posteriori P(h|D)* de cada endereço de IP ser maligno ou benigno. Por exemplo, para calcular *P(h|D)* do IP 192.168.100.5 ser maligno, basta seguir os seguintes passos:

1. Calcular a probabilidade *P(h)* da classe maligno. Esse valor é calculado através da divisão da soma frequências das classes, pela frequência da classe maligno. Nesse caso, de acordo com as frequências da Tabela 2: (18+30)/18 = 0,375.
2. Calcular a*P(D)* do IP 192.168.100.5. Para isso, dividi-se a frequência do IP, pelo número de alertas. Utilizado as frequências da Tabela 2: (8+2)/(18+30) = 0,208.
3. Calcular a probabilidade *a priori P(D|h)* do IP 192.168.100.5, dado que a classe maligno ocorreu. Nesse caso, dividi-se a frequência do IP pela frequência de classe, apenas no caso em ambos coincidem de serem malignos: 8/18 = 0,444.
4. Finalmente, basta aplicar, na Equação da Probabilidade Condicional, os valores encontrados nos passos anteriores (como mostra a Figura 2), para encontrar a probabilidade *a posteriori P(h|D)* de um alerta ser maligno, dado que o IP 192.168.100.5 ocorreu: (0,444*0,375)/0,208 = 0,8 (80%).

Tabela 2 – Exemplos de Frequências de IPs em Alertas

Endereço de IP	**Maligno**	**Benigno**
192.168.100.5	8	2
192.168.100.6	2	16
192.168.100.7	4	0
192.168.100.3	4	12
Total	**18**	**30**

Tabela 3 – Exemplos de Probabilidades de IPs em Alertas

		Maligno		**Benigno**	
Endereço de IP	***P(D)***	***P(D|h)***	***P(h|D)***	***P(D|h)***	***P(h|D)***
192.168.100.5	0,208	0,444	0,800	0,066	0,200
192.168.100.6	0,375	0,111	0,111	0,533	0,889
192.168.100.7	0,083	0,222	1,000	0,000	0,000
192.168.100.3	0,333	0,222	0,250	0,400	0.750
P(h)		0,375		0,625	

No exemplo acima, um alerta que apresenta o IP 192.168.100.5 tem 80% de chance de ser maligno e consequentemente 20% de ser benigno. Dessa forma, o algorítimos *Naïve Bayesian Learning* classificaria esse alerta como maligno. De forma semelhante, dadas as frequências dos atributos de um alerta de segurança real e sua postura, é possível que o *Naïve Bayesian Learning* possa calcular as probabilidades *a posteriori* de um novo alerta ser benigno ou maligno. Outro ponto a favor do *Naïve Bayesian Learning* é a sua capacidade de trabalhar com os atributos nominais (RASCHKA, 2018) que compõem a base de dados do experimento. Finalmente, o *Naïve Bayesian Learning* é computacionalmente simples (DOMINGOS; PAZANI, 1996), permitindo que se realize classificações com recursos computacionais limitados.

4.2.2 Árvores de Decisão

Assim como o *Naïve Bayesian Learning*, árvores de decisão são utilizadas para a detecção de anomalias (SINGH; NEME, 2013). Diferente do *Naïve Bayesian Learning*, que utiliza uma abordagem probabilística, árvores de decisão são baseadas no conceito de dividir para conquistar (NARAYAN; PARVAT, 2015). Uma árvore (Figura 5) é obtida através da divisão do conjunto de dados original em subconjuntos. Essa divisão considera os conceitos de entropia e ganho de informação (NARAYAN; PARVAT, 2015), tal que cada subconjunto é desdobrado em ordem crescente de sua entropia. Esse processo ocorre recursivamente até que cada atributo resultante possua o maior ganho de informação (VILLANO, 2018) e menor entropia. A relação entre entropia e ganho de informação é descrita nas figuras 3 e 4 respectivamente, onde *Ganho(S,A)* é

o ganho de informação no conjunto *S*, após o desdobramento do atributo *A*, enquanto *Entropia(S)* é a entropia do subconjunto de atributos *S*(NARAYAN; PARVAT, 2015).

A Figura 5 mostra um exemplo de árvore de decisão, que prevê se um jogo de futebol irá ou não acontecer (Jogar), com base em atributos do clima (Tempo, Temperatura, Umidade e Vento). A Tabela 4 descreve a distribuição de frequências dos atributos necessários para os cálculos da *Entropia(S)* e *Ganho(S,A)*. A árvore de decisão da Figura 5 é construída respeitando os seguintes passos:

1. Primeiro calcula-se a entropia da classe Jogar, de acordo com as probabilidades dos eventos positivos e negativos. Nesse caso, as probabilidades de Sim e Não, respectivamente, são 0,64 e 0,36. Aplicando os valores das probabilidades na equação da *Entropia(S)*, obtêm-se uma entropia de 0,94.

2. Em seguida, calcula-se o atributo com maior ganho de informação. Isto é, o atributo que a diferença de sua *Entropia(S)*, menos a entropia calculada no passo 1 da classe Jogar é a menor dentre todos os atributos. Aplicando a equação do *Ganho(S,A)* para cada atributo, é possível notar que o atributo Clima é o que oferece o maior Ganho (0,693). Portanto, Clima será a raiz da árvore.

3. Agora, é necessário decidir, dentre os valores de Clima, quais oferecem o maior *Ganho(S,A)*, quando comparado a outros atributos. Assim como no passo anterior, deve-se aplicar a equação do ganho de informação, porém limitando-se às probabilidades dos atributos que abrangem o valor de Clima alvo. Por exemplo, para desdobrar a arvore no valor Ensolarado (Figura 5), deve-se considerar as probabilidades que envolvem Clima Ensolarado. A Tabela 5 mostra valores de atributos válidos, nesse caso. Nesse caso, o maior *Ganho(S,A)* é o do atributo Vento. Portanto, a árvore irá se desdobrar nesse atributo.

4. Os passos 1 ao 3 são então repetidos recursivamente até que os últimos nodos possuam *Entropia(S)* = 0. No exemplo da Figura 5, os valores Sim e Não da classe Jogar são os valores que oferecem *Entropia(S)* = 0.

Figura 3 – Equação da Entropia

$$Entropia(S) = -a \log_2 a - b \log_2 b$$

(VILLANO, 2018)

Onde:

- *S* = Um conjunto de exemplos.

- *a*= Proporção de exemplos positivos.
- *b* = Proporção de eventos negativos.

Figura 4 – Equação do Ganho

$$Ganho(S,A) = Entropia(S) - \sum_{i \in Valores(A)} \frac{|Si|}{|S|} X\ Entropia(S)$$

(VILLANO, 2018)

Onde:

- |*S*| = Número de elementos em *S*.
- |*Si*| = Número de elementos em *Si*.
- *Valores(A)*= Conjunto dos valores possíveis de um atributo *A*.

Figura 5 – Exemplo de Árvore de Decisão

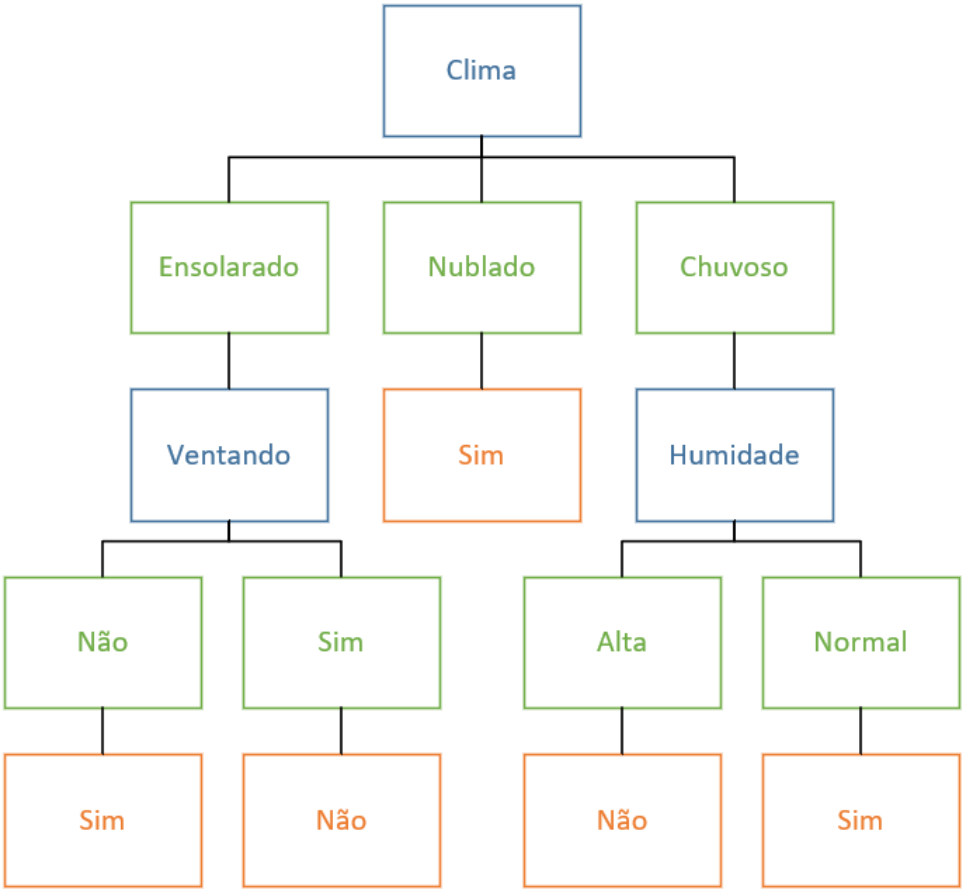


Tabela 4 – Exemplos de Frequência de Árvore de Decisão

Tempo	**Temperatura**	**Umidade**	**Vento**	**Jogar**
Chuvoso	Quente	Alta	Não	Não
Chuvoso	Quente	Alta	Sim	Não
Nublado	Quente	Alta	Não	Sim
Ensolarado	Normal	Alta	Não	Sim
Ensolarado	Frio	Normal	Não	Sim
Ensolarado	Frio	Normal	Sim	Não
Nublado	Frio	Normal	Sim	Sim
Chuvoso	Normal	Alta	Não	Não
Chuvoso	Frio	Normal	Não	Sim
Ensolarado	Normal	Normal	Não	Sim
Chuvoso	Normal	Normal	Sim	Sim
Nublado	Normal	Alta	Sim	Sim
Nublado	Quente	Normal	Não	Sim
Ensolarado	Normal	Alta	Sim	Não

Tabela 5 – Exemplos de Frequência de Árvore de Decisão Após Desdobramento

Clima	**Temperatura**	**Umidade**	**Ventando**	**Jogar**
Ensolarado	Normal	Alta	Não	Sim
Ensolarado	Frio	Normal	Não	Sim
Ensolarado	Frio	Normal	Sim	Não
Ensolarado	Normal	Normal	Não	Sim
Ensolarado	Normal	Alta	Sim	Não

4.2.3 *Random Forest*

Random Forests são uma variação das árvores de decisão, onde árvores menores sem são construídas com base em atributos escolhidos aleatoriamente (THEODORIDIS; KOUTROUM, 2009). A classe de um objeto será então definida pela classe identificada pela maioria das arvores individuais (SHALEV-SHWARTZ; BEN-DAVID, 2014). O resultado dessa abordagem tende a ser uma maior acurácia na classificação, em relação a maioria dos algorítimos de aprendizado de máquina (TROESCH; WALSH, 2014), incluindo o *Naïve Bayesian Learning* e Árvore de Decisão.

A Figura 6 mostra um exemplo de *Random Forest*, onde um conjunto *x* de instâncias é dividido em árvores menores, contendo atributos escolhidos aleatoriamente. A classe identificada pela maioria das sub-árvores será o resultado *y* da *Random Forest*. Uma *Random Forest* é obtida, de forma genérica, através dos seguintes passos:

1. Inicialmente deve-se selecionar, de forma aleatória, uma série de instâncias do conjunto total de amostras, para formar um subconjunto *Si* de instâncias utilizadas nos passos 2 a 4.

2. Em seguida, escolhe-se aleatoriamente *n* atributos. Dentre esses atributos, o atributo com maior entropia (Figura 3) será a raiz da primeira árvore de decisão.
3. Dentre os *n* atributos não escolhidos como raiz, sortear *m* atributos e escolher, dentre os *m* atributos, o que oferece o maior ganho de informação (Figura 4).
4. Os passos 2 e 3 são repetidos recursivamente, com atributos que ainda não foram selecionados, até que a primeira sub-árvore de decisão seja construída.
5. O algorítimo então irá voltar ao passo 1. Novas árvores será então geradas, até que a floresta esteja preparada para classificar instâncias.

Random Forests, no entanto, tendem a ser computacionalmente intensivas durante a fase de treinamento (SOLE; RAMISA; TORRAS, 2014), o que requer a utilização de um hardware suficientemente robusto. Apesar do hardware utilizado durante os experimentos não ter grande poder computacional,*Random Forests* merecem atenção, por serem capazes de lidar com grandes números de variáveis, sem sofrer com o *overfitting* comum das árvores de decisão (BIAU, 2012).

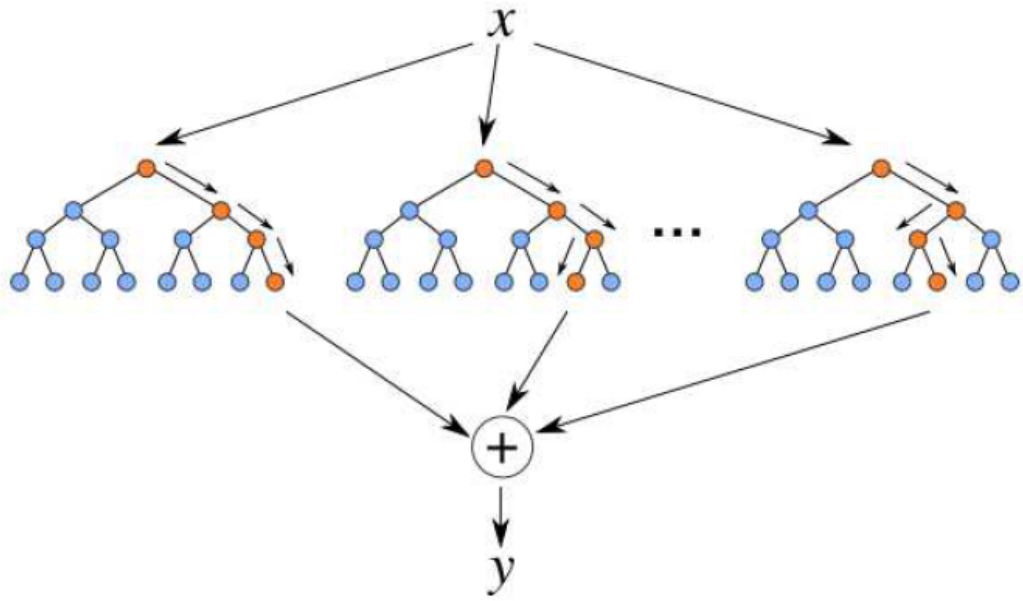


Figura 6 – Exemplo de Random Forest

Onde:

- *x* = Um conjunto de instâncias.
- *y* = A classe que obteve a maior quantidade de votos das árvores individuais.

4.3 Aplicação

A utilização do aprendizado a diversos setores da indústria. Na medicina, por exemplo, o aprendizado de máquina oferece métodos e técnicas para auxiliar no diagnóstico e prognóstico em diversas áreas da médicas, como na biomedicina (MAGOULAS; PRENTZA, 2001). Neste último, um exemplo mais específico seria o aprendizado de máquina na análise das propriedades químicas de um medicamento, com intuito de prever seus efeitos adversos em pacientes (WINTTEN; FRANK, 2005).

Empresas como Google e Amazon também são notórias por usar aprendizado de máquina em suas atividades. No caso da Google, o exemplo mais aparente talvez seja em seu buscador, que aplica aprendizado de máquina no reconhecimento de padrões que auxiliam da identificação de conteúdo duplicado e irrelevante (Kevin Rewe, 2018). Já a Amazon, desenvolve a assistente virtual Alexa, que utiliza o aprendizado de máquina, computação em nuvem e *Natural Language Processing* (NLP), para entender comandos de voz emitidos pelo usuário (Bernard Marr, 2018).

Finalmente, na indústria de segurança da informação, o aprendizado de máquina é frequentemente aplicado em sistemas de detecção de anomalias de rede. (SINGH; NEME, 2013). O sistema *Advanced Threat Defense* (ATD) desenvolvido pela McAfee, por exemplo, utiliza aprendizado de máquina, em conjunto com computação em nuvem, para analisar o impacto associado a certos indicadores, como a existência de endereços de IP e programas considerados maliciosos (MCAFEE, LLC, 2018)). Dessa forma o *Advanced Threat Defense* consegue rapidamente identificar e neutralizar ameaças cibernéticas, antes que essas causem danos a uma empresa.

5 Experimentos

Para a realização dos experimentos, foi utilizado um computador portátil, equipado com 16GB de memória RAM e uma CPU Intel Core i7-76000U (2,80 GHz). Nessa configuração o algorítimo *Random Forest* necessitou de aproximadamente 18 horas para o treinamento do modelo, após a busca aleatória de hiperparâmetros. Isso sugere que, em um ambiente de produção contendo dezenas de milhares de amostras, o *hardware* utilizado neste trabalho pode não ser suficientemente robusto.

Os algorítimos de aprendizado de máquina analisados são provenientes da API Java do Weka (versão 3.8). O Weka é uma conjunto de aplicações com o objetivo de auxiliar no estudo e aplicação de técnicas de aprendizado de máquina (UNIVERSITY OF WAIKATO, 2008), para problemas de classificação e mineração de dados.

As probabilidades das classes são expostas na Tabela 9, onde percebe-se uma discrepância considerável nos números de alertas benignos (94,84% da base de dados) e malignos. Essa diferença torna a base de dados não balanceada, introduzindo um possível viés nas classificações, direcionado à classe mais numerosa (alerta benigno). A amostragem estratificada, no entanto, auxiliou na redução do impacto desse desbalanço. A base de dados foi, portanto, dividida nas proporções de 70, 15 e 15 porcento, para os conjuntos de treinamento, validação e teste, respectivamente.

5.1 Base de Dados

A base de dados é composta de 7702 instâncias, que representam alertas de segurança reportados pelo SIEM *AlienVault*, na Intrinium, durante um período de três anos (2015 a 2018). Esses alertas têm como característica em comum serem gerados com base em eventos do *Active Directory* (AD). A infraestrutura *Active Directory* é predominante em empresas (MICROSOFT CORPORATION, 2016), o que a torna alvo recorrente de ataques cibernéticos, assim como fonte de eventos utilizados por SIEMs na detecção de tais ataques. O *Active Directory* (AD) é uma tecnologia da *Microsoft* usada para gerenciar computadores e outros dispositivos em uma rede. O *Active Directory* permite os administradores de rede criar e gerenciar domínios, usuários e objetos em uma rede (MICROSOFT CORPORATION, 2016). Para esse experimento foram, portanto, considerados os alertas descritos na Tabela 6.

Os alertas foram escolhidos com base na frequência em que são identificados como benignos, além do impacto causado por falhas na investigação, por analistas. Por exemplo, o alerta *Multiple Authentication Failure* (múltiplas falhas de autenticação)

Tabela 6 – Lista de Alertas Analisados

Nome do Alerta	Descrição
AD User Changes	Mudanças em contas de usuário sofre qualquer mudança.
AD User Creation	Criação de um novo usuário.
AD User Deletion	Usuário e removido do *Active Directory.*
AD Group Changes	Grupo do *Active Directory* sofre mudanças.
AD Group Creation	Novo grupo do *Active Directory* foi criado.
AD Group Deletion	Grupo do *Active Directory* foi removido.
AD Account Lockout	Uma conta de Usuário foi bloqueada após falhas de autenticação.
Multiple Authentication Failures	Múltiplas falhas de autenticação ocorreram.
Software Installation	Instalação de programas.
Software Removal	Remoção de programas.

ocorre quando um usuário tenta acessar um computador, usando credenciais (nome de usuário ou senha) incorretas repetidamente. Esse tipo de alerta, embora frequente, não constitui uma ameaça, pois é típico de usuários cometendo erros ao credenciais. Eventos relacionados a mudanças no *Active Directory*, como *AD Group Creation* e *AD User Creation*, por exemplo , possuem alta relevância no monitoramento de uma empresa. Isso porque um atacante com acesso a uma conta com privilégios administrativos pode comprometer a rede inteira, através da manipulação de contas de usuários (CORPORATION, 2017). Finalmente, eventos relacionados à instalação e remoção de programas são importantes por serem associado a infecções por vírus computador e programas maliciosos em geral (MALWAREBYES, 2015).

Os atributos (Tabela 7) escolhidos para compor os experimento são no total 13 e descrevem os dados revistos por analistas, na Intrinium, durante a investigação de um alerta. Demais atributos não incluídos nesse experimento incluem *policy_id*, *context_id*, *backlog_id* e *event_id*. Esses representam metadados que descrevem como os alertas são organizados no banco de dados do SIEM AlienVault e não contribuem na investigação de alertas por analistas de segurança.

Não obstante, é importante citar a ausência de valores em certos atributos, dependendo do tipo de alerta. A ausência de valores pode ser atribuída á diferença entre alertas, que utilizam certos atributos. Alertas que não envolvem a execução de um programa, como *AD User Changes*, por exemplo, não acusam valores para file_name ou process_name, porem a maioria dos algorítimos de aprendizado de máquina deixam implícito que a existência de valores nulos não e um fator significativo na classificação, quando a absência de tais valores é esperada (WINTTEN; FRANK, 2005).

Os valores dos atributos listados foram *tokenizados* utilizando um função *hash* MD5, por conta da tamanho do valor resultante (32 caracteres), que auxilia em manter bases de dados reduzida em relação a outros *hashes* ,como SHA265 (40 caracteres) e SHA1 (64 caracteres) (MAETOUQ; DAUD; AHMAD, 2018). Finalmente, a existência de implementações da funções *hash* em diversas linguagens de programação, eliminou a

Tabela 7 – Lista de Atributos

Atributo	Descrição
plugin_sid	Código numérico que identifica o alerta.
plugin_id	Define a origem dos eventos que geraram o alerta.
src_ip_hostname	Define o nome do sistema ou endereço de IP origem dos dos eventos.
dst_ip_hostname	Define define o endereço de IP ou nome do sistema sendo atacado.
target_ad_domain	Define o domínio afetado por mudanças no *active directory.*
source_ad_domain	Define o domínio *Active Directory* de uma conta que originou alertas.
target_ad_group	Define o grupo afetado por mudanças no *Active Directory.*
username_target	Define a conta de usuário alvo de um ataque.
username_source	Define a conta do usuário que causou um ou mais alertas.
process_name	Define o nome do processo que originou um ou mais alertas.
file_name	Define o nome de um arquivo ou executável que originou alertas.
*isAfterhour*s	Diz respeito ao turno (manhã/tarde ou noite) do alerta.
postura/classe	Identifica a postura (falso alerta ou incidente) de um alerta.

necessidade de desenvolver e testar um algorítimos original. Neste trabalho, utilizou-se a implementação da função *hash* MD5 proveniente da biblioteca *hashlib.md5*, disponível na linguagem Python (versão 3.7) (PYTHON SOFTWARE FOUNDATION, 2019).

5.2 Medidas de Desempenho

A eficiência dos modelos de aprendizado de máquina é determinada em termos de falso positivo, falso negativo, positivo verdadeiro e negativo verdadeiro. A probabilidade de uma alerta ser benigno é consideravelmente maior que este ser maligno (ZEINALI, 2016). Não obstante, é comum que vários alertas malignos e benignos frequentemente compartilhem atributos com valores semelhantes (HASSAN; AL., 2009). Além disso, a fadiga de alerta contribui para que alertas sejam incorretamente investigados por analistas. Ambos alertas com valores semelhantes e investigações incorretas aumentam as chances de ocorrerem erros de classificação. O erro considerado crítico é quando o algorítimo classifica um alerta maligno como benigno, que durante o monitoramento de uma rede, pode levar ao comprometimento da rede.

Figura 7 – Equação da Precisão

$$Precisão = \frac{PV}{PV + FP}$$

Onde:

PV = Total de positivos verdadeiros.

FP = Total de falsos positivos.

Figura 8 – Equação da Cobertura

$$Cobertura = \frac{PV}{PV + FN}$$

Tabela 8 – Exemplo de Matriz de Confusão

	Benigno	Maligno
Benigno	**PV**	FN
Maligno	**FP**	NV

FN = Total de falsos negativos.

NV = Total de negativos verdadeiros.

Para calcular a *Precisão*, é útil ter a matriz de confusão de um algorítimo, a fim de entender quais classificações são consideradas falsos positivos (*FP*) e positivos verdadeiros (*PV*). De acordo com a Tabela 8, portanto, para o calculo da *Precisão* (Figura 7), atribui-se a *PV* o número de alertas corretamente classificados como benignos, enquanto *FP* representa o total de alertas malignos classificados como benignos. Isso significa que, para aumentar o valor da *Precisão*, deve-se reduzi o valor de *FP*. Dessa forma, a métrica de *Precisão* foi considerada a ideal, para medir a performance de um modelo em classificar alertas benignos. A *Precisão* captura a essência da preocupação em reduzir as taxas de falsos positivos (BRENDAN; LEE, 2017), que no contexto deste trabalho, significa ignorar uma ameaça em potencial. Como o foco deste trabalho é em reduzir o volume de alertas benignos, o calculo da *Precisão* garante que se mantenha o foco na redução desses alertas, enquanto simultaneamente procura-se reduzir as chances de um alertas maligno ser incorretamente classificado como benigno.

Finalmente, a *Cobertura* (Figura 8) foi utilizada como métrica complementar, para identificar o quanto a acurácia de um modelo foi afetada por falsos negativos. O valor da *Cobertura* é inversamente proporcional ao número de falsos negativos. Neste trabalho, falsos negativos representam alertas benignos incorretamente classificados como malignos. Tal erro não oferece os mesmo risco imposto por falsos positivos, pois alertas classificados incorretamente como malignos, diferente de um *FP*, não representa uma ameaça em potencial ignorada pelo algorítimo.

5.3 Seleção de Modelos

A seleção de modelos utilizou uma busca aleatória para otimizar os hiperparâmetros em cada algorítimo de aprendizado de máquina. Normalmente, a procura pelos

Tabela 9 – Distribuição de Probabilidades das Classes

Classe	**Quantidade**	**Probabilidade**
Benigno	7305	94,84%
Maligno	397	5,15%
Total	7702	100%

Tabela 10 – Lista de Hiperparâmetros da *Random Forest*

Hiperparâmetro	**Descrição**	**Variação**
P	Tamanho de cada *bag* como proporção do conjunto de testes.	10 a 100
I	Número de iterações.	100 a 500
K	Número *n* de atributos investigados aleatoriamente.	0 a $\log_2(n) + 1$
M	Número mínimo de instâncias por folha.	1 a 100

melhores hiperparâmetros é feita manualmente, o que tende a ser exaustivo e ineficiente (CLAESEN; MOOR, 2015). Dessa forma, utilizou-se o *Random Search* para automatizar a busca pelos hiperparâmetros de maior impacto no modelo. Esse método é altamente eficiente na busca de hiperparâmetros quando comparado a outros métodos como o *Grid Search*, especialmente em espaços multidimensionais. Isso porque o *Random Search* realiza a validação das combinações de hiperparâmetros aleatoriamente (BERGSTRA; BENGIO, 2012), permitindo que se encontrem os hiperparâmetros de maior relevância, em um menor tempo computacional.

O *Naïve Bayesian Learning* dispensou a fase de otimização de hiperparâmetros, porque esse algorítimo não possuir hiperparâmetros. Os algorítimos Árvore de Decisão e *Random Forest*, no entanto, passaram pela otimização de hiperparâmetros e contaram cada um com dez mil rodadas do *Random Search*.

5.3.1 *Random Forest*

Os hiperparâmetros e seus intervalos investigados são descritos na Tabela 10. Os intervalos levaram em consideração o tempo e memória necessários para para treinar o modelo em cada rodada do *Random Search*. No caso do *Random Forest*, o hiperparâmetro com maior consumo de tempo e memória foi o número de interações *I*, que descreve o número de sub-árvores geradas, para montar a floresta. É possível que a realização de testes, em um hardware mais robusto, permitisse a utilização de intervalos maiores e maior número de rodadas do *Random Search*.

Os melhores combinações de hiperparâmetros encontrados são expostos na Tabela 11. O desvio padrão de 0.00 entre métricas demonstra que não houve diferença no desempenho de generalização entre diferentes combinações de hiperparâmetros.

Tabela 11 – Melhores Hiperparâmetros da *Random Forest*

Hiperparâmetro	**Valores**								**Desvio Padrão**
Tamanho de cada bag (*P*)	88	96	90	89	86	**93**	96	93	—
N. de iterações (*I*)	475	375	490	245	415	**105**	130	480	—
N. de atrib. invest. aleat. (*K*)	3	3	3	3	3	**3**	3	3	—
N. mínimo de instâncias p/ folha. (*M*)	1,0	1,0	1,0	1,0	1,0	**1,0**	1,0	1,0	—
Erro	0,023	0,023	0,023	0,023	0,023	0,023	0,023	0,023	0,00
Precisão	0,977	0,977	0,977	0,977	0,977	0,977	0,977	0,977	0,00
Cobertura	0,998	0,998	0,998	0,998	0,998	0,998	0,998	0,998	0,00

Tabela 12 – Lista de Hiperparâmetros da Árvore de Decisão

Hiperparâmetro	**Descrição**	**Variação**
M	Numero mínimo de instancias por folha.	2 a 100
U	Sem poda.	Sim/Não
B	Divisão binária.	Sim/Não
N	Divisões para reduzir o erro de poda.	3 a 100
C	Confiabilidade de poda.	0,25 a 0,5
R	Reduzir erro de poda.	Sim/Não
S	Não permitir *subtree raising*.	Sim/Não

Tabela 13 – Melhores Hiperparâmetros da Árvore de Decisão

Hiperparâmetro	**Valores**					**Desvio Padrão**
Conf. de poda (*C*)	**0,34**	0,34	0,43	0,44	0,40	—
Div. binária (*B*)	**Sim**	Sim	Sim	Sim	Sim	—
N. min. instâncias p/folha (*M*)	**2**	2	2	2	2	—
Sem poda (*U*)	**Não**	Não	Não	Não	Não	—
Reduzir Erro de Poda (*R*)	**Não**	Não	Não	Não	Não	—
Não permitir *subtree raising* (*S*)	**Não**	Sim	Sim	Sim	Sim	—
Divisões p/ reduzir o erro de poda (*N*)	**Não**	Não	Não	Não	Não	—
Erro	0,025	0,025	0,025	0,025	0,025	0,00
Precisão	0,974	0,974	0,974	0,974	0,974	0,00
Cobertura	1,00	1,00	1,00	1,00	1,00	0,00

Dessa forma, foram escolhidos os hiperparâmetros em negrito, em que o menor número de iterações *I* requer um menor tempo computacional e memória para a realização do treinamento. Além disso, a discrepância entre os valores de *I* obtidos (entre 105 e 475), sugere que um alto número de de sub-árvores não influenciou na acurácia.

5.3.2 Árvore de Decisão

A Tabela 12 expõe os hiperparâmetros utilizados investigados durante o *Random Search*. As variações dos hiperparâmetros seguiram a mesma lógica do *Random Forest* e teve como foco a redução o consumo de tempo e memória. As árvores testadas foram com e sem poda (*U*), binárias (*B*) e não-binárias.

Os resultados do *Random Search* para a Árvore de Decisão (Tabela 13), assim como *Random Forest*, mostram um desvio padrão de 0,00 para todas as métricas. Po-

Tabela 14 – Erros e Acertos de Classificação da Árvore de Decisão

Atributo	Acerto	Erro
plugin_sid	705e5b6b317d091229d9699f2d616df2	705e5b6b317d091229d9699f2d616df2
plugin_id	d9443549aa55a5517299ba6efdc84d0d	d9443549aa55a5517299ba6efdc84d0d
stc_ip_hostname	88855172647ea30ab6cbaf0dca532fe9	e54a43b1dcb21cd130dcb2cc4e5ed5b0
dst_ip_hostname	0642f698e2efa5c464ceb68c741e93d5	7e0d2e5b7c74f77a59bfcba4b43486cf
target_ad_domain	Nulo	Nulo
source_ad_domain	Nulo	Nulo
target_ad_group	Nulo	Nulo
username_target	Nulo	Nulo
username_source	Nulo	Nulo
process_name	Nulo	Nulo
file_name	Nulo	Nulo
isAfterhours	Yes Yes	
postura/classe	**Benigno**	**Maligno**
Classificado Como	**Benigno**	**Benigno**

rém, para reduzir a chances de ocorrer *overfitting*, hiperparâmetros que representaram uma árvore com poda foram escolhidos para realizar os teste. Finalmente, nota-se que os melhor hiperparâmetros geralmente consideraram árvores com divisão binária. Isso porque árvores de decisão tendem a realizar divisões binárias, quando possível, para reduzir *overfitting* (EDUPRISTINE, 2015) e consequentemente melhorar a acurácia.

5.4 Resultados

Os resultados foram obtidos aplicando o melhores hiperparâmetros, para treinar os respectivos modelos. Nessa fase, utilizou-se o conjunto de testes, que representa 15% do conjunto total de dados. Os resultados dos testes são descritos a seguir.

5.4.1 Árvore de Decisão

As métricas de generalização da Árvore de Decisão são descritas na Tabela 8. A métrica de precisão da árvore com poda e divisão binária foi de 96,7%, indicando a presença de falsos positivos. A cobertura de 99,6% indica que houveram poucos falsos negativos, mas estes influenciaram negativamente a acurácia do modelo que foi de 96,79%. As métricas obtidas neste teste foram próximas das obtidas durante a validação dos hiperparâmetros, como pode ser observado no desvio padrão de cada métrica (Tabela 7), sem que houvesse variações significativas. A matriz de confusão (Tabela 8) mostra o quanto a acurácia da Árvore de Decisão foi afetada por falsos positivos, os quais representaram 2,8% (33 instâncias) do total de instâncias no conjunto de testes. Exemplos de erros e acertos de classificação são observados na Tabela 2.

Pode-se observar pela Árvore de Decisão resultante (Apêndice A) que o atributo *plugin_sid* foi escolhido como raiz da árvore, indicando que esse atributo ofere-

Tabela 15 – Matriz de Confusão do Teste da Árvore de Decisão

Benigno	**Maligno**	**← Classificado**
1087	4	Benigno
33	31	Maligno

Tabela 16 – Resultados do Teste da Árvore de Decisão

Métrica	**Valores(%)**	**Desvio Padrão**
Acurácia	96,79%	0,00433
Erro	3,20%	0,00426
Precisão	**96,7%**	**0,00546**
Cobertura	99,6%	0,00282

ceu a maior valor de entropia, seguido de *username_target* e *isAfterhours*. De fato, durante o monitoramento diário da Intrinium, é possível observar que o volume total de alertas benignos é diretamente relacionado com os tipos de alertas. Por exemplo, a predominância de alertas benignos para *Multiple authentication failure*, faz com que este ofereça um ganho de informação maior na descrição de alertas. De forma similar, o *username_target* e *isAfterhours* também são altamente descritivo, devido à sua relação que com o tipo do alerta. Por exemplo, um *Multiple Authentication failure* que ocorre durante o turno da manhã é mais provável de ser um alerta benigno, devido à maior presença de usuários na rede que no turno da noite. Além disso, alertas gerados durante a noite têm maior probabilidade de indicar uma atividade maliciosa, como um vírus de computador tentando descobrir as senhas dos usuários.

5.4.2 *Random Forest*

As métricas de generalização da *Random Forest* são descritas na Tabela 17. A métrica de precisão da *Random Forest* foi de 97,50%, indicando a presença de falsos positivos. A cobertura de 100% indica que não houve falsos negativos e que a métrica de precisão obtida indica que a acurácia de 97,57% foi afetada apenas por falsos positivos. As métricas obtidas neste teste foram próximas das obtidas durante a validação dos hiperparâmetros, como pode ser observado no desvio padrão de cada métrica (Tabela 17), sem que houvesse variações significativas. A matriz de confusão (Tabela 18) mostra o quanto a acurácia da *Random Forest* foi afetada por falsos positivos, os quais representaram 2,4% (28 instâncias) do total de instâncias no conjunto de testes. Exemplos de erros e acertos de classificação são observados na Tabela 19.

5.4.3 *Naïve Bayesian Learning*

As métricas de generalização do *Naïve Bayesian Learning* são descritas na Tabela 20. A métrica de precisão desse modelo, após testes, foi de 94,7%, indicando um

Tabela 17 – Resultados do Teste do *Random Forest*

Métrica	**Valores%**	**Desvio Padrão**
Acurácia	97,57%	0,00065293
Erro	2,42%	0,00058222
Precisão	**97,50%**	**0,00162852**
Cobertura	100,00%	0,00186375

Tabela 18 – Matriz de Confusão do Teste do *Random Forest*

Benigno	**Maligno**	**← Classificado**
1091	0	Benigno
28	36	Maligno

Tabela 19 – Erros e Acertos de Classificação do *Random Forest*

Atributo	**Acertos**	**Erros**
plugin_sid	f474573a89a8f1da580cbfd9b0fecd33	705e5b6b317d091229d9699f2d616df2
plugin_id	dcad2c3aa0abb27bb3577329ccca14d1	d9443549aa55a5517299ba6efdc84d0d
src_ip_hostname	dcad2c3aa0abb27bb3577329ccca14d1	6ae64d034ad4198753ea6e5420e8b0a5
dst_ip_hostname	f0c3c9549ff298f6e5779825ba9cca57	0642f698e2efa5c464ceb68c741e93d5
target_ad_domain	Nulo	Nulo
source_ad_domain	Nulo	Nulo
target_ad_group	Nulo	Nulo
username_target	Nulo	Nulo
username_source	9bbd45bad55cfc620803907f2d8a0217	Nulo
process_name	Nulo	Nulo
file_name	Nulo	Nulo
isAfterhours	No	Yes
postura/classe	**Benigno**	**Maligno**
Classificado Como	**Benigno**	**Benigno**

maior número de falsos positivos, quando comparado aos demais algorítimos. A cobertura de 96,3% indica a presença de falsos negativos que afetam acurácia (91,42%) do *Naïve Bayesian Learning*. A matriz de confusão (Tabela 21) mostra o quanto a acurácia desse modelo foi afetada por falsos positivos, os quais representaram 5,1% (59 instâncias) do total de instâncias no conjunto de testes. Exemplos de erros e acertos de classificação podem ser averiguados na Tabela 22.

Tabela 20 – Resultados do Teste do *Naïve Bayesian Learning*

Métrica	**Valores %**
Acurácia	91,42%
Erro	8,57%
Precisão	94,7%
Cobertura	96,3%

Tabela 21 – Matriz de Confusão do Teste do *Naïve Bayesian Learning*

Benigno	**Maligno**	← **Classificado**
1051	40	Benigno
59	5	Maligno

Tabela 22 – Erros e Acertos de Classificação do *Naïve Bayesian Learning*

Atributo	**Acertos**	**Erros**
plugin_sid	705e5b6b317d091229d9699f2d616df2	705e5b6b317d091229d9699f2d616df2
plugin_id	d9443549aa55a5517299ba6efdc84d0d	d9443549aa55a5517299ba6efdc84d0d
src_ip_hostname	97e18774943eb9fc8522393dae57cfab	94d4b35e423a1891df60acdd4953596c
dst_ip_hostname	0642f698e2efa5c464ceb68c741e93d5	0642f698e2efa5c464ceb68c741e93d5
target_ad_domain	Nulo	Nulo
source_ad_domain	Nulo	Nulo
target_ad_group	Nulo	Nulo
username_target	Nulo	Nulo
username_source	Nulo	Nulo
process_name	Nulo	Nulo
file_name	Nulo	Nulo
isAfterhours	Yes	Yes
postura/classe	**Maligno**	**Maligno**
Classificado Como	**Maligno**	**Benigno**

6 Análise dos Resultados

Dentre os modelos analisados, o *Random Forest* foi que apresentou o melhor desempenho de generalização, comparado à Árvore de Decisão e *Naïve Bayesian Learning*. Por este último usar uma abordagem probabilística, é possível que as semelhanças entre atributos de falsos positivos e alertas afetaram negativamente a precisão do modelo. A árvore de decisão obteve um desempenho superior ao *Naïve Bayesian Learning*. Isso porque árvores de decisão têm um maior poder de correlação entre variáveis (VILLANO, 2018)). De fato, em alertas de segurança, os atributos analisados não são independentes, o que explica porque o *Naïve Bayesian Learning* teve a menor performance inferior dentre todos os modelos. Finalmente, a *Random Forest* obteve um desempenho de generalização superior à Árvore de Decisão. Isso porque Random Forests, geralmente, oferecem uma precisão superior às árvores de decisão (ZHANG; ZULKERNINE; HAQUE, 2008), além de serem menos suscetíveis ao *overfitting*. Finalmente, não houve a necessidade de realizar testes de hipóteses, pois as classes foram distribuídas proporcionalmente e a base de dados foi suficientemente grande.

A Tabela 23 mostra os resultados obtidos com os melhores modelos. É possível observar que o *Random Forest* obteve a maior precisão e a menor taxa de erros de classificação (2,42%). Apesar da distribuição das precisões obtidas ser relativamente uniforme (desvio padrão de 0,0144), as diferentes técnicas de classificação foram provavelmente afetadas por inconsistências da base de dados. Tais inconsistências ocorrem principalmente devido à fadiga de alerta, que leva analista a identificar alertas como malignos ou benignos. Dessa forma, conjuntos de atributos que deveriam pertencer à classe incidente acabam compondo o conjunto de alertas benignos, o que pode afetar o treinamento dos modelos e reduz a acurácia.

O *Naïve Bayesian Learning* foi o mais afetado por possíveis inconsistências, apresentando uma acurácia de 91,42%. A cobertura de 100% indica que o *Random Forest* não gerou falsos negativos. Em contrapartida, a cobertura inferior do *Naïve Bayesian Learning* (96,3%) e Árvore de Decisão (99,6%) indicam a presença de falsos ne-

Tabela 23 – Tabela de Generalização

Modelo	Erro	Precisão	Cobertura
Naïve Bayesian Learning	0,0857	0,947	0,963
Arvore de Decisão	0,0320	0,967	0,996
Random Forest	0,0242	0,975	1,000
Desvio Padrão:	**0,0335**	**0,0144**	**0,0262**

gativos que afetam a acurácia desses dois últimos modelos. Falsos negativos (alertas benignos classificados como malignos), no entanto, são negligenciáveis e considerasse a precisão com métrica de generalização. Dessa forma, a *Random Forest* obteve a melhor métrica (97,5%) e portanto pode ser considerada a melhor técnica para reduzir o volume de alertas benignos, quando comparado aos demais algorítimos.

Os modelos foram treinados utilizando uma base de dados composta de diferentes tipos de alertas, como exposto na sessão Experimentos, de forma que as diferenças inerentes a cada tipo de alerta podem ter influenciado negativamente a acurácia dos modelos. Essa diferenças são traduzidas na ausência ou presença de atributos nulos, além valores compartilhados de atributos que, em tipos de alertas diferentes, levam a atribuição de posturas (maligno ou benigno) diferentes. Por exemplo, o mesmo valor de *src_ip_hostname* pode ser associado a alertas benignos de *Multiple Authentication Failure* e a alertas malignos de *Software Instalation*.

É provável, portanto, que melhores resultados pudessem ser obtidos, se a comparação dos três algorítimos analisados fosse realizada individualmente para cada tipo de alerta. Dessa forma, seria possível identificar alertas com maior acurácia, uma vez que as diferenças entre alertas não influenciariam os algorítimos. Em experimentos futuros, portanto deve-se considerar a aplicação das técnicas de aprendizado investigadas, para cada tipo de alerta, individualmente.

Finalmente, apesar das inconsistências na base de dados e do possível impacto dos diferentes tipos de alertas, a métrica de generalização do *Random Forest*, demostra que esse algorítimo possui uma acurácia promissora na redução do volume de alertas benignos. De fato, na industria de segurança da informação, aproximadamente 32% dos analistas regularmente ignoram alertas de segurança (NETWORKS, 2017). Além disso, apenas 4% atenção e análise próprias, devido à fadiga de alerta (HASSAN; AL., 2009). A fadiga de alerta, portanto, oferece um risco potencialmente maior à rede de uma empresa, que os erros de classificação do melhor algorítimo (*Random Forest*) e modelo encontrados neste trabalho. Dessa forma, é possível afirmar que o algorítimo *Random Forest* poderia ser aplicado para identificar alertas reportados pelo SIEM AlienVault, na Intrinium, e assim reduzir o volume de alertas benignos.

7 Conclusão

A aplicação de métodos de aprendizado de máquina, em alertas do SIEM AlienVault, demonstra potencial para reduzir os números excessivos de alertas benignos. Os resultados obtidos com esse experimento mostram que técnicas de aprendizado supervisionado podem ter resultados relevantes, mesmo quando a base de dados não é consistente. Essa inconsistência provém da fadiga de alerta, que leva o analista a identificar alertas malignos como benignos. Nesse, caso, possíveis ameaças e alertas benignos acabam compartilham atributos com valores semelhantes, introduzindo um viés (direcionado à classe mais frequente) que pode reduzir a acurácia dos modelos.

Enquanto o analista for necessário na identificação de ameaças, é improvável que o erro humano seja completamente eliminado. Esse problema pode ser minimizado em trabalhos futuros, através da aplicação de técnicas complementares de otimização heurística, como algorítimos genéticos (SINGH; NEME, 2013). Estes são capazes de encontrar modelos de aprendizado de máquina com a melhor acurácia, através da identificação da hipótese que melhor se aplica a um conjunto de atributos e instâncias (MITCHEL, 1997). A realização de um procedimento de mineração de dados, com o fim para obter uma base de dados mais consistente (com menos erros de detecção) e balanceada, também seria relevante no aumento da acurácia dos modelos.

Apesar do possível viés introduzido pelo erro humano, o *Random Forest* apresentou uma métrica de generalização promissora na identificação das amostras disponíveis. Ainda assim, técnicas complementares que reduzam o impacto do erro humano são necessárias, para o aumento da acurácia dos modelos. Com a redução do volume de alertas benignos, espera-se que os analistas da Intrinium terão mais tempo para dedicar a alertas relevantes e as investigações sejam mais precisas, reduzindo o risco da empresa ser comprometida por ataques cibernéticos.

Referências

ALIENVAULT INC. *A Unified Security Platform for Threat Detection, Incident Response Compliance*. San Mateo, California, USA: alienvault.com, 2019. <https://www.alienvault.com/products/usm-anywhere>. Citado 2 vezes nas páginas 13 e 14.

BENG, L. Y. et al. A survey of intrusion alert correlation and its design considerations. *IETE Technical Review*, Taylor Francis, v. 31, n. 3, p. 233–240, 2014. Disponível em: <https://doi.org/10.1080/02564602.2014.906864>. Citado na página 18.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, v. 13, n. 1, p. 281–305, 3 2012. Citado na página 32.

Bernard Marr. *Machine Learning In Practice: How Does Amazon's Alexa Really Work?* Jersey City, New Jersey, USA: www.forbes.com, 2018. <https://www.forbes.com/sites/bernardmarr/2018/10/05/how-does-amazons-alexareally-work/#34a303091937>. Citado na página 27.

BIAU, G. Analysis of a random forests model. *Journal of Machine Learning Research*, v. 13, n. 1, p. 1063–1095, 1 2012. Citado na página 26.

BRENDAN, J.; LEE, H. Precision-recall versus accuracy and the role of large data sets. In: NIPS WORKSHOP ON MULTI-CLASS AND MULTI-LABEL LEARNING IN EXTREMELY LARGE LABEL SPACES. Long Beach, California: Association for the Advancement of Artificial Intelligence, 2017. Citado na página 31.

CENTRE FOR ADVANCED INTERNET ARCHITECTURE. *DIFFUSE Machine Learning*. Hawthorn VIC 3122, Australia: CENTRE FOR ADVANCED INTERNET ARCHITECTURE, 2011. <http://caia.swin.edu.au/urp/diffuse/ml.html>. Citado na página 20.

CHAKRABORTY, S.; MONDAL, B. Article: Spam mail filtering technique using different decision tree classifiers through data mining approach - a comparative performance analysis. *International Journal of Computer Applications*, v. 47, n. 16, p. 26–31, June 2012. Full text available. Citado na página 12.

CLAESEN, M.; MOOR, B. de. Hyperparameter search in machine learning. In: THE XI METAHEURISTICS INTERNATIONAL CONFERENCE. Agadir, Morocco, 2015. v. 2. Citado na página 32.

CORPORATION, T. M. *Privilege Escalation*. McLean, Virginia, USA: attack.mitre.org, 2017. <https://attack.mitre.org/tactics/TA0004/>. Citado na página 29.

DOMINGOS, P.; PAZANI, M. Hyperparameter search in machine learning. In: THIRTEENTH INTERNATIONAL CONFERENCE ON INTERNATIONAL CONFERENCE ON MACHINE LEARNING. San Francisco, CA, USA, 1996. v. 1. Citado na página 22.

DUA, S.; XIAN, D. *Data Mining and Machine Learning in Cybersecurity*. 1. ed. The address: CRC Press, 2011. v. 1. ISBN 9781439839423. Citado na página 19.

EDUPRISTINE. *Decision Trees – Tree Development and Scoring*. San Francisco, California, USA: www.edupristine.com, 2015. <https://www.edupristine.com/blog/decisiontrees-development-and-scoring>. Citado na página 34.

FIREEYE INSTITUTE. *How Many Alerts is Too Many to Handle?* Milpitas, California, USA: www2.fireeye.com, 2019. <http://pages.fireeye.com/HD0031b0FyNNxDt1kWI00Ly>. Citado na página 11.

GOOGLE LLC. *Machine Learning Glossary*. Mountain View, California, USA: developers.google.com, 2019. <https://developers.google.com/machine-learning/glossary/#i>. Citado na página 19.

HASSAN, W.; AL. et. Nodoze: Combatting threat alert fatigue with automated provenance triage. In: NETWORK AND DISTRIBUTED SYSTEMS SECURITY (NDSS) SYMPOSIUM. San Diego, California, USA, 2009. Citado 4 vezes nas páginas 11, 17, 30 e 39.

HUBBARD, D.; SEIERSEN, R. *How to Measure Anything in Cybersecurity Risk*. 1. ed. Hoboken, New Jersey, USA: Willey, 2016. v. 1. ISBN 9781119085294. Citado 2 vezes nas páginas 13 e 18.

Kevin Rewe. *How Search Engines Use Machine Learning: 9 Things We Know for Sure*. Deerfield Beach, Florida, USA: www.searchenginejournal.com, 2018. <https://www.searchenginejournal.com/how-search-engines-use-machinelearning/224451>. Citado na página 27.

KITTLER, J.; HATED, M.; DUIN, R. Combining classifiers: A theoretical framework. *Pattern Analysis Applications*, v. 20, n. 3, p. 18–27, 3 1998. Citado na página 17.

LEE, K. H.; ZHANG, X.; XU, D. High accuracy attack provenance via binary-based execution partition. In: 20TH ANNUAL NETWORK DISTRIBUTED SYSTEM SECURITY SYMPOSIUM. San Diego, CA, USA, 2013. Citado 3 vezes nas páginas 11, 13 e 17.

MAETOUQ, A.; DAUD, S.; AHMAD, N. Comparison of hash function algorithms against attacks: A review. *International Journal of Advanced Computer Science and Applications*, v. 9, n. 8, 2018. Citado na página 29.

MAGOULAS, G.; PRENTZA, A. *Machine Learning in Medical Applications*. 1. ed. Springer, Berlin, Germany: Heidelberg, 2001. v. 2049. 300-307 p. Citado na página 27.

MALWAREBYES. *All About Malware*. Santa Clara, California, USA: www.malwarebytes.com, 2015. <https://www.malwarebytes.com/malware/>. Citado na página 29.

MASTERS, G. *Crying wolf: Combatting cybersecurity alert fatigue*. Santa Clara, California, USA: www.scmagazine.com, 2017. <https://www.scmagazine.com/home/security-news/in-depth/crying-wolf-combattingcybersecurity-alert-fatigue>. Citado na página 11.

MCAFEE, LLC. *Advanced analytics: Rapidly turn data into insights*. Santa Clara, California, USA: www.mcafee.com, 2018. <https://www.mcafee.com/enterprise/en-us/products/security-analyticsproducts.html>. Citado na página 27.

MICROSOFT CORPORATION. *(Cloud) Tip of the Day: Latest Azure AD Statistics*. Jersey City, New Jersey, USA: blogs.technet.microsoft.com, 2016. <https://blogs.technet.microsoft.com/tip_of_the_day/2016/04/12/cloud-tip-of-the-daylatest-azure-ad-statistics>. Citado na página 28.

MITCHEL, T. *Machine Learning*. 1. ed. Ithaca, New York, USA: McGraw-Hill Science, 1997. v. 1. 157 p. Citado 3 vezes nas páginas 20, 21 e 40.

NARAYAN, A.; PARVAT, T. An intrusion detection system, (ids) with machine learning (ml) model combining hybrid classifiers. *Journal of Multidisciplinary Engineering Science and Technology*, v. 2, n. 4, 4 2015. Citado 3 vezes nas páginas 11, 22 e 23.

NETWORKS, S. *The Problem of Security: Alert Fatigue*. Santa Clara, California, USA: skyhighnetworks.com, 2017. <https://www.skyhighnetworks.com/cloud-security-blog/alert-fatigue-31-9-of-itsecurity-professionals-ignore-alerts/>. Citado 2 vezes nas páginas 12 e 39.

PYTHON SOFTWARE FOUNDATION. *hashlib — Secure hashes and message digests*. Delaware, USA: python.org, 2019. <https://docs.python.org/3/library/hashlib.html>. Citado na página 30.

RAJBANSHI, A.; BHIMRAJKA, S.; RAINA, C. K. Artificial intelligence in cyber security. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, v. 2, n. 3, p. 132–137, 5 2017. Citado na página 13.

RASCHKA, S. Model evaluation, model selection, and algorithm selection in machine learning. CoRR abs, Trier, Germany, 2018. Citado 2 vezes nas páginas 20 e 22.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning: From Theory to Algorithms*. 1. ed. 32 Avenue of the Americas, New York, USA: Cambridge University Press, 2014. v. 1. ISBN 9781107298019. Citado na página 25.

SINGH, J.; NEME, M. A survey on machine learning techniques for intrusion detection systems. *International Journal of Advanced Research in Computer and Communication Engineering*, v. 21, n. 1, p. 53, 11 2013. Citado 4 vezes nas páginas 11, 22, 27 e 40.

SOLE, X.; RAMISA, A.; TORRAS, C. Evaluation of fandom forests on large-scale classification problems using a bag-of-visual-words representation. In: CATALAN CONFERENCE ON ARTIFICIAL INTELLIGENCE. Barcelona, Espanha, 2014. Citado na página 26.

SPATHOULAS, G. P.; KATSIKAS, S. K. Using a fuzzy inference system to reduce false positives in intrusion detection. In: XVI INTERNATIONAL CONFERENCE ON SYSTEMS, SIGNALS AND IMAGE PROCESSING. Chalkida, Greece, 2009. Citado na página 11.

THEODORIDIS, S.; KOUTROUM, K. *Pattern Recognition*. 4. ed. 32 Burlington,MA 01803,USA: Elsevier Inc, 2009. v. 1. An optional note. ISBN 9780080949123. Citado na página 25.

TREINEN, J. *HEURISTICS FOR IMPROVED ENTERPRISE INTRUSION DETECTION*. 35 p. Dissertação (Mestrado) — University of Denver Denver, Denver, USA, 2009. Citado na página 11.

TROESCH, M.; WALSH, I. *Machine Learning for Network Intrusion Detection*. 450 Serra Mall, Stanford, CA 94305, EUA: [s.n.], 2014. Citado na página 25.

UNIVERSITY OF WAIKATO. *Attribute-Relation File Format (ARFF)*. Hillcrest, New Zealand: www.cs.waikato.ac.nz, 2008. <https://www.cs.waikato.ac.nz/ml/weka/arff.html>. Citado na página 28.

VILLANO, E. *Classification of logs using Machine Learning Technique*. 1,11,12,13,17 p. Dissertação (Mestrado) — Norwegian University of Science and Technology, Høgskoleringen 1, 7491 Trondheim, Norway, 2018. Citado 7 vezes nas páginas 11, 17, 19, 22, 23, 24 e 38.

WINTTEN, I.; FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques*. 3. ed. San Francisco, California, USA: Morgan Kaufmann Publishers Inc, 2005. v. 1. 157 p. Citado 3 vezes nas páginas 20, 27 e 29.

ZEINALI, S. *Analysis Of Security Information And Event Management (SIEM) Evasion And Detection Methods*. Dissertação (Mestrado) — Tallinn University of Technology, Ehitajate tee 5, 12616 Tallinn, Estônia, 2016. Citado 2 vezes nas páginas 11 e 30.

ZHANG, J.; ZULKERNINE, M.; HAQUE, A. Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 38, n. 5, p. 649–659, 9 2008. Citado na página 38.

A Árvore de Decisão Resultante