Sistema de Detecção de Intrusão e Técnicas de Inteligência Computacional para Redes de Computadores

Pedro Henrique Linhares Oliveira

JUIZ DE FORA

MARÇO, 2021

Sistema de Detecção de Intrusão e Técnicas de Inteligência Computacional para Redes de Computadores

Pedro Henrique Linhares Oliveira

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Edelberto Franco Silva

JUIZ DE FORA

MARÇO, 2021

Sistema de Detecção de Intrusão e Técnicas de Inteligência Computacional para Redes de Computadores

Pedro Henrique Linhares Oliveira

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Edelberto Franco Silva
Doutor em Computação

Alex Borges Vieira
Doutor em Computação

Luciano Jerez Chaves
Mestre em Computação

JUIZ DE FORA
16 DE MARÇO, 2021

Aos meus pais, família e amigos pelo apoio.

Resumo

A segurança de sistemas computacionais é uma área de grande importância e relevância em redes de computadores, se destacando como parte crítica dos sistemas conectados à Internet, principalmente devido ao crescimento massivo de dispositivos conectados e o fluxo de dados nas últimas décadas. Os Sistemas de Detecção de Intrusão (IDS) são mecanismos de monitoramento e análise de sistemas para detecção de invasores e comportamento malicioso em redes de computadores. Já os algoritmos de inteligência computacional, como *Machine Learning*, estão despertando cada vez mais atenção nos diversos sistemas distribuídos com grande volume de dados. Sendo assim, a utilização de técnicas de inteligência computacional aplicada a soluções de IDS tem despertado grande interesse, tanto pela sua boa acurácia e taxa de detecção, quanto pelo seu potencial de aprendizado e velocidade de processamento. Assim, este trabalho avalia a aplicação de algoritmos de inteligência computacional em *datasets* reais para treinamento e previsão de ataques e comportamento malicioso em redes de computadores.

Palavras-chave: IDS, *Machine Learning*, segurança de rede.

Agradecimentos

A todos os meus parentes, pelo encorajamento e apoio.

Aos amigos que fiz e também, aos que já fazem parte da minha vida e que de alguma forma me ajudou ao longo dessa jornada.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"Quando uma criatura humana desperta pra um grande sonho e nele lança toda força da sua alma, todo universo conspira a seu favor."

(Johann Goethe)

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

IDS	Sistema de detecção de intrusão
MLP	*Multilayer Perceptron*
SVM	Máquina de Vetores de Suporte
RNA	Rede Neural Artificial
ML	Aprendizado de Máquina
DDOS	Ataque de negação de serviço
R2L	*Remote to User*
U2L	*User to root*
UNSW-NB15	Banco de dados UNSW-NB15
NSL-KDD	Banco de dados NSL-KDD
IOT	Internet das Coisas

1 Introdução

1.1 Apresentação do Tema

O crescimento do número de dispositivos conectados à Internet e o aumento da utilização e dependência para os diversos ambientes (*e.g.*, Internet das Coisas - IoT -, Agricultura de Precisão, IoT Industrial, Cidades Inteligentes, Casas Inteligentes) traz a necessidade de se avaliar a segurança associada na troca de mensagens e sobre a finalidade do tráfego trocado pelos elementos de rede. Desta forma, pensar em como identificar possíveis mensagens originadas por atacantes ou invasores é de suma importância (ALQAHTANI et al., 2020).

A detecção de invasores é essencial para manter um bom funcionamento de uma rede e parte importante da segurança de um sistema (LUNT, 1988; SILVA; JULIO, 2011). Este é um processo que monitora e analisa eventos que ocorrem em um computador ou em uma rede de computadores para identificar sinais de intrusos. Pode-se identificar esses sinais através de, principalmente, duas abordagens: detecção por assinatura ou detecção por anomalias (PANDA et al., 2011).

Uma das soluções adotadas para combater esses ataques são os Sistemas de Detecção de Intrusão (IDS), que têm o objetivo de identificar ataques e classificar as atividades do sistema como idôneas ou mal-intencionadas, gerando assim alertas na ocorrência de um invasor (WAGH; PACHGHARE; KOLHE, 2013). As duas formas citadas anteriormente são clássicas em IDS. A detecção por assinatura identifica um comportamento com certo padrão malicioso, já a detecção por anomalia identifica de maneira mais dinâmica a alteração de um dado perfil de usuário ou máquina na troca de mensagens.

Atualmente, soluções em segurança de redes de computadores têm se atentado para os benefícios da utilização de técnicas de inteligência computacional, mais especificamente *Machine Learning* (ML - Aprendizado de Máquina) (LI; ZHAO; LI, 2017). Essas técnicas têm permitido mais liberdade e agilidade na identificação de ameaças em tempo real e em grandes volumes de dados (MEDEIROS et al., 2020). Assim, IDS têm sido investigadas junto com técnicas de ML para classificação dos ataques como redes neurais

artificiais (RN), *Multilayer Perceptron* (MLP), *Random Forest*, Máquina de Vetores de Suporte (SVM - *Support Vector Machine*), entre outras.

1.2 Contextualização

A quantidade de dispositivos conectados na internet é cada vez maior já que o número deve chegar a 12.3 bilhões em 2022, segundo a *Cisco Systems* (FORECAST, 2019), além disso a diversidade de dispositivos e suas diferenças de capacidade e arquitetura torna o monitoramento e combate a um ataque ao dispositivo um desafio.

Há também um grande número de dados sensíveis que circulam na rede todos os dias, como dados de transações bancárias, negócios eletrônicos, informações pessoais e comerciais, com isso a segurança em ambientes de redes se torna extremamente necessária. Casos como o `ransomware WannaCry` em que sequestram os dados e pedem dinheiro para recuperá-los ou ataques de negação de serviço - DoS (*Denial of Service*), por exemplo, geram grande prejuízo a pessoas e empresas. Logo, a segurança da informação é um item imprescindível para as empresas e um desafio para desenvolver uma infraestrutura segura e confiável.

1.3 Motivação

Um dos maiores desafios em segurança é identificar invasores em tempo real, motivado pelo grande fluxo de informações e variedades dos tipos de ataque que forçaram o desenvolvimento de soluções e algoritmos.

Uma das áreas focadas em resolver esses ataques cibernéticos é a IDS que é desenvolvida para detectar atividades maliciosas, vírus e *malware* (VINAYAKUMAR et al., 2019). Para ter sucesso, uma IDS tem que ter acurácia, velocidade de detecção e confiabilidade, logo é importante identificar o maior número de ataques possível e com o menor número de falsos positivos, ou seja, de comportamentos normais, mas classificados como ataques (JUNIOR, 2004).

Algoritmos de ML, *Deep Learning* e outras técnicas de inteligência computacional vêm sendo adotados cada vez mais nas diversas áreas da computação e na área de

segurança, e no ambiente de redes não é diferente. Tais técnicas apresentam pontos favoráveis, como automatizações e melhores resultados que outros algoritmos baseados em regras fixas, por exemplo.

Desta forma, algoritmos de ML podem ser usados em IDSs para aumentar a acurácia de detecção e diminuir a taxa de falsos positivos (SULTANA et al., 2019). Assim, este trabalho foca em investigar como utilizar técnicas associadas desde o treinamento à generalização para prever ataques em redes de computadores, experimentando os algoritmos relacionados à Redes Neurais, MLP e Floresta Randômica. Para a validação desses experimentos é de suma importância a utilização de dados - *datasets* - reais.

Com isso o objetivo do trabalho é investigar e entender os conceitos de IDS, sua importância no mercado de segurança da informação e perspectivas. Além disso, buscar avaliar algoritmos de inteligência computacional em IDS e entender os motivos do crescente uso em IDS utilizando para isso *datasets* reais.

Para atingir esses objetivos foi criada nesse trabalho uma IDS utilizando algoritmos de ML a partir de dois *datasets*, o UNSW-NB15 e o NSL-KDD. Os *datasets* contém registros de pacotes de redes de atividades normais e de ataques, servindo para o treinamento dos algoritmos e após a geração do modelo foi avaliado o desempenho de cada algoritmo por métricas de desempenho gerando o modelo ideal para a IDS.

Os algoritmos utilizados foram a árvore de decisão, *Random Forest*, MLP e o SVM e foi concluído que o modelo construído pelo algoritmo *Random Forest* apresenta o melhor desempenho para os *datasets* avaliados nesse trabalho.

No próximo capítulo, Embasamento Teórico, é apresentado a descrição formal e caracterização de uma IDS e dos algoritmos de inteligência computacional, no capítulo 3 é descrito cada *dataset* e o pré-processamento feito no trabalho. No capítulo 4 é apresentado os resultados obtidos bem como a descrição das métricas utilizadas e avaliações feitas. Por fim, o capítulo 5 mostra as conclusões obtidas e os trabalhos futuros.

2 Embasamento Teórico

Neste capítulo serão abordados os conceitos e classificações de sistemas de detecção de intrusão, além de comentar especificamente das diferenças e semelhanças entre IDS de rede e *host*. Outro ponto abordado no capítulo é sobre o tópico de inteligência computacional, apresentando os conceitos e classificações na área e os algoritmos selecionados para este estudo.

2.1 Sistema de Detecção de Intrusão

Sistemas de detecção de intrusão são sistemas automatizados de segurança que monitoram eventos que ocorrem em uma rede ou computador, analisando-o e constatando se é uma atividade maliciosa ou não. Após a constatação, o IDS irá, impedir a atuação da invasão, ou reportar para o administrador da rede ou computador (SILVA; JULIO, 2011).

As principais funções de uma IDS, segundo (MITCHELL; CHEN, 2014), é coletar os dados suspeitos, analisá-los, podendo ser através de *logs*, pacotes de uma rede ou por análise do processador(CPU), e atuar na prevenção de incidentes suspeitos que interfiram na integridade, disponibilidade e confiabilidade do sistema.

É importante destacar que o IDS não tem o papel de servir como antivírus para detectar *softwares* maliciosos ou alguma uma ferramenta de verificação de vulnerabilidade de rede ou de computadores. Seu papel é de servir como uma espécie de segunda linha de defesa na camada de segurança, sendo a primeira linha de defesa aquela que limita o acesso a uma rede ou computador.

A classificação de uma IDS é devido ao tipo de intrusão que ocorre, se dividindo em detecção de uso indevido ou detecção de anomalias (ALMSEIDIN et al., 2017). Já sua infraestrutura se divide em IDS de rede e IDS local.

Para o método baseado em assinatura, busca-se encontrar uma assinatura específica para sinalizar uma invasão. Essa assinatura poderia ser algo como um padrão de comportamento de envio de pacotes, ou formato específico de um fluxo de pacotes.

Contudo, segundo (WAGH; PACHGHARE; KOLHE, 2013), o ponto fraco desse método é que, caso exista um novo tipo de ataque, esse tipo de sistema não saberá que é um ataque, uma vez que depende do conhecimento prévio de sua assinatura.

Já o método de anomalias analisa o comportamento normal do sistema, classificando-o como adequado/correto, e quando surge um comportamento anormal (ou fora do que ele conhece como o normal) ele então o detecta. Seu ponto fraco pode ser considerada a possível geração de um número alto de falsos positivos, ou seja, comportamentos normais do sistema que foram identificados como anormais (WAGH; PACHGHARE; KOLHE, 2013).

2.1.1 IDS de Rede

Network Intrusion Detection é um tipo de IDS baseada na leitura e análise de pacotes de redes principalmente, para verificar a integridade de todos os nós de um segmento de rede. Esses nós geralmente são compostos de sensores e pode ser limitados a apenas executar a IDS facilitando na proteção do mesmo. Outra vantagem é que com poucos sensores, mas bem posicionados, é possível proteger uma grande rede.

Podemos destacar como uma desvantagem a impossibilidade de inspecionar dados criptografados, visto que esses trafegam protegidos. Outra possível desvantagem é a dificuldade em analisar um tŕafego muito grande de pacotes de redes, podendo apresentar muitas falhas durante esse período.

Segundo (JAVAID et al., 2016) para se construir uma IDS de rede existem dois grandes desafios:

- Seleção de recursos de tráfego adequados para a detecção.
- Apresentação de dados de tráfegos rotulados de redes reais.

Selecionar o recurso de tráfego é de vital importância, pois dependendo dos recursos monitorados, um pacote de rede pode ser considero normal ou ser detectado como uma anomalia, e consequentemente como um ataque. Para rotular os tráfegos de dados de redes brutos é necessário um grande esforço que varia desde a necessidade de conhecimento prévio até a possível análise da entrada. Nesse contexto, fica claro que IDSs que

apresentam resposta em tempo real se tornam um grande desafio.

Geralmente, existem dois tipos de ataques em IDS de rede: os ataques que envolvem uma única conexão e os ataques que envolvem várias conexões (LAZAREVIC et al., 2003). Com isso métodos e métricas de análise serão divididos nesses dois tipos de ataques.

2.1.2 IDS Local

Host Intrusion Detection é um tipo de IDS baseada no monitoramento e análise de portas e comportamentos de aplicativos em um único *host*, observando a interação desses aplicativos com o funcionamento do sistema através de *switches*.

O IDS Local se divide na fonte de dados, análise e método para determinar comportamento anormal. Pode-se detectar uma invasão encontrando um comportamento anormal em informações baseadas em *host*, como sistema de chamadas, *logs* do sistema, ações de aplicativos e tráfego do *host* (SALEM; TAHERI; YUAN, 2018).

Uma vantagem é poder se utilizar de recursos do sistema *host* para auxiliar na construção e execução do IDS, agilizando o processamento e resposta, outro aspecto importante é ao se localizar no computador de destino da informação o IDS encontrará os dados sem criptografia podendo assim prevenir ataques que não foram possíveis identificar em uma IDS de rede.

Os IDS Local tem a capacidade de monitorar todas as entradas de um *host* e podendo ter a capacidade de interpretar as atividades da rede localmente podendo localizar e denunciar atividades suspeitas em todas as camadas do protocolo. Além disso, ao localizar uma atividade suspeita pode comunicar com outros IDS de uma rede informando a intrusão

2.2 Inteligência Computacional

Os algoritmos de inteligência computacional, especificamente algoritmos de *machine learning*, se utilizam de um conjunto específico de dados para generalizar um comportamento, pelo princípio de inferência, chamado de indução. Pode-se dividir em dois conjuntos prin-

cipais, o aprendizado supervisionado e o não-supervisionado.

O **aprendizado supervisionado** apresenta no conjunto dados de treinamento que tanto o conjunto de entrada, quanto o rótulo de saída do algoritmo. O objetivo é construir um modelo de uma função desconhecida que permite predizer as novas entradas (MAGALHÃES; OLIVEIRA, 2020). Um conjunto de dados supervisionados, é um conjunto de tuplas $\{(x_i, y_i)\}_{i=1}^{n}$, em que em uma entrada de dados x_i tem um rótulo y_i

O **aprendizado não-supervisionado** não apresenta dados rotulados para a construção do modelo, com isso os dados serão apenas um conjunto de vetores de entrada, o objetivo do modelo é na geração de dados auto-rotulados com melhor desempenho, categorizando ou agrupando em grupos para execução do algoritmo.

O aprendizado supervisionado pode ser dividido em algoritmos de classificação e regressão para tipos de dados que se encontram em um vetor de dimensão pré-definida. Sendo considerada algoritmo de classificação os tipos de valores dos rótulos y_i são discretos, já algoritmos de regressão os tipos de valores dos rótulos y_i são contínuos.

Diversos algoritmos de classificação são aplicados em IDS de acordo com (AMUDHA; KARTHIK; SIVAKUMARI, 2013), identificando se são comportamentos normais ou outros tipos de comportamento, e são, em geral, escolhidos devido à Taxa de Detecção e Taxa de Falso Positivo.

Por fim, uma etapa importante em tais propostas é a de treinamento. Nessa etapa algoritmos sofrem um processo dinâmico e iterativo a fim de utilizar o conhecimento para modificar os parâmetros da rede em relação aos estímulos que recebe de seu ambiente.

2.2.1 Árvore de Decisão

A árvore de decisão é um algoritmo de classificação que constrói um modelo a partir do conjunto de dados. Essa árvore então classifica o dado através de atributos já classificados, visando dividi-los da melhor forma. Logo, é muito importante selecionar seus atributos da melhor forma possível, podendo utilizar métodos como padrão de classificação, regra de parada e poda para formar diferentes árvores de decisões.

Tal algoritmo busca identificar os atributos mais importantes e descartar os desnecessários para melhorar na seleção dos dados (MOON et al., 2017). A vantagem da

árvore de decisão é a rapidez e precisão para aprender um grande número de dados e também na análise dos dados sendo um poderoso algoritmo em IDS.

Conforme a figura 2.1, que representa uma árvore de decisão genérica, os dados de entrada estão na profundidade 1 da árvore. Em seguida, a árvore contém os nós de decisão que verificam os atributos de uma entrada. Para cada uma dessas verificações há uma aresta ligando à uma profundidade inferior da árvore, que ao alcançar a folha tem-se a classificação da entrada indicada.

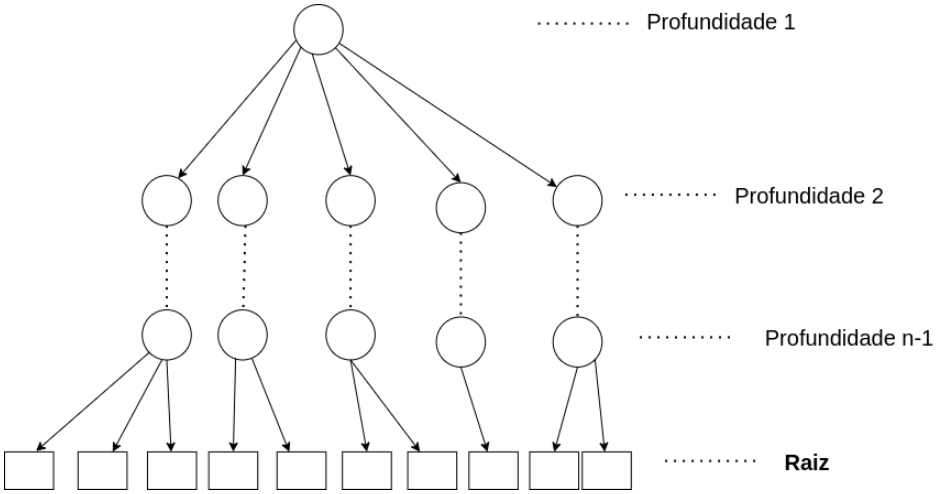

Figura 2.1: Árvore de decisão genérica.

Assim, como é importante selecionar o melhor atributo de teste em cada nó, foram desenvolvidos métodos para realizar tal seleção, como: escolher pelo menor valor do atributo ou maior valor do atributo, completamente aleatório, no maior ganho esperado, ou seja, no que resulta no menor número de sub-árvores.

Porém, a árvore de decisão pode apresentar um grande número de nós, tornando-a complexa. Para tratar essa característica, métodos foram desenvolvidos. Um deles é a poda da árvore, onde o modelo, após a construção, irá reduzir os nós de modo que as partição menos semelhantes sejam reduzidas. Outro método proposto foi o do algoritmo *Random Forest*, que será explicado em maiores detalhes mais a frente.

2.2.2 Floresta Randômica

Random Forest é um classificador que consiste na existência de muitas árvores de decisões. A sua principal característica é manter baixa a taxa de erro relacionada à classificação.

As suas vantagens em relação a uma árvore de decisão tradicional é que a floresta gerada pode ser usada em uma execução futura (FARNAAZ; JABBAR, 2016), e além disso a variação nos atributos e a sua precisão são calculadas automaticamente. Contudo sua implementação pode ser considerada mais complexa, assim como o incremento do custo computacional.

A *Random Forest* então seleciona aleatoriamente uma pequena quantidade de dados para o treinamento, se utilizando de um método chamado *bootstrap*. Outra característica é a aplicação de um método de reamostragem, utilizado para que os dados escolhidos possam ser repetidos na seleção, diferentemente da árvore de decisão que usa a totalidade de dados disponíveis.

Outro processo que compõem a *Random Forest* é o *out-of-bag*, que corresponde a média das entradas das árvores que não são usados no *bootstrap*. Esse processo fornece estimativas de erro sem precisar do conjunto de testes, fazendo uma espécie de validação cruzada durante o treinamento mas sem custo computacional.

A figura 2.2 exemplifica uma *Random Forest*, em que um dado de entrada x é fragmentado aleatoriamente usando *bootstrap*, em sequência ocorre a seleção de dados para o crescimento da árvore até o critério de parada e estimando o erro usando o *out-of-bag*. Por fim, compara com as outras árvores a predição k com menor erro é escolhida.

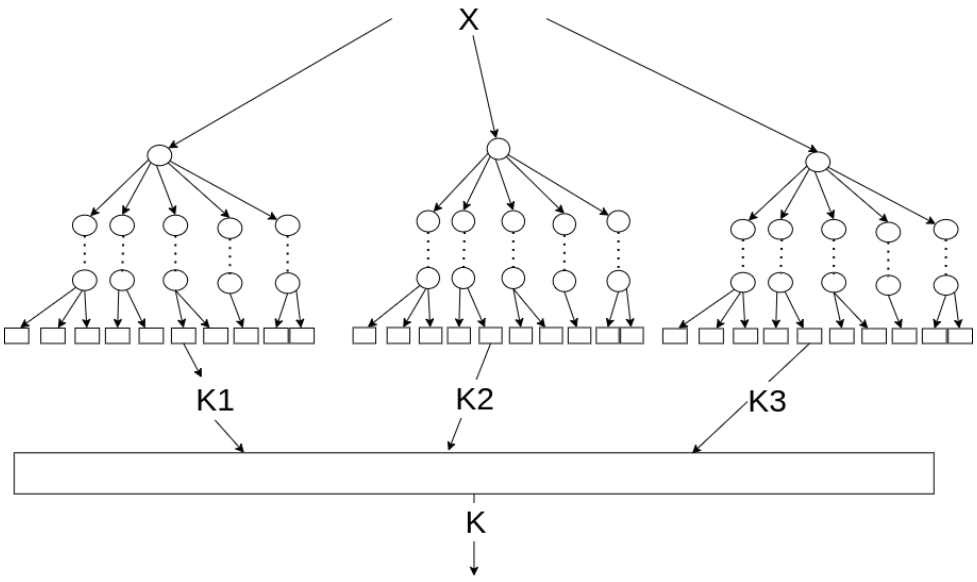

Figura 2.2: Representação de uma *Random Forest* genérica.

Com esses dois processos de aleatoriedade e a construção da árvore de maneira

independente, a árvore resultante é mais generalizada e robusta (SANTANA et al., 2020).

2.2.3 SVM

O *Support Vector Machines* (SVM) é um modelo de classificação supervisionado e tradicional para aprendizado de máquinas. Usa o conceito de planos de decisão para dividir os dados e utiliza duas técnicas principais o *One-vs-One* ou o *One-vs-All* (ARTHUR, 2018).

Os princípios do SVM são o hiperplano de divisão ideal e a função *kernel*. O objetivo do hiperplano é discriminar e dividir os conjuntos enquanto maximiza a margem de classificação, que é a menor distância entre o hiperplano e os dados de cada conjunto conforme o gráfico 2.3 indicado pelos pontos que estão com a reta tracejada. A função *kernel* tem a função de projetar os dados em um domínio de maior dimensão, pois assim aumenta a probabilidade de os dados serem linearmente separáveis do que funções com baixas dimensões, obedece a equação 2.1. O $\theta(.)$ pode ser uma expressão linear, Gaussiana ou Hiperbólica, onde deve pertencer a um domínio que seja possível o cálculo do produto interno (JUNIOR, 2010). Uma escolha adequada da função *kernel* é fundamental para obtenção de um bom desempenho no SVM.

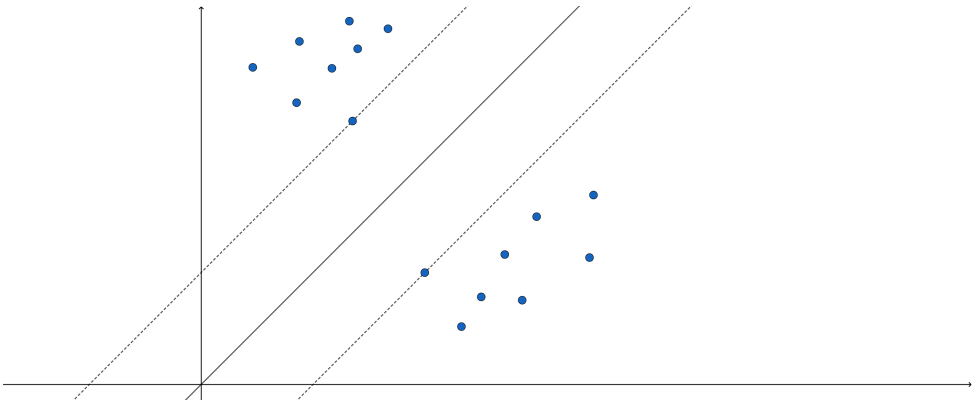

Figura 2.3: Vetor ideal dividindo as entradas em dois seguimentos.

$$K\left(x,y\right)=\theta\left(x\right)\theta(y) \tag{2.1}$$

Formalmente, o SVM tem um conjunto de treinamento x_i que contém n atributos, um rótulo de entrada y_i que está entre -1 e 1, representando a tupla de entrada conforme a expressão 2.2.

$$(x_i, y_i), \quad x \in R^n, y \in \{-1, 1\}. \tag{2.2}$$

Como o objetivo do SVM é encontrar o hiperplano ótimo, é preciso minimizar a função custo, também chamada função objetivo. Em relação ao vetor de peso w, além disso, é necessário que o conjunto de treinamento seja linearmente separável conforme a condição de restrição da equação 2.3.

$$\begin{aligned} &min \leftarrow \frac{1}{2} w^t \times w, \\ &sujeito\, a\, restrição \leftarrow y_i(w^t \times x_i + b) \geqslant 1, para\, i = 1, ..., n \end{aligned} \tag{2.3}$$

2.2.4 Redes Neurais Artificiais

Uma rede neural artificial (RNA) é um algoritmo que processa dados baseando-se em um modelo inspirado no cérebro humano. Esse modelo é composto por nós interconectados, onde cada nó é seguido de uma função de ativação somado a um elemento e gera um próximo valor que serve de entrada para a próxima camada. Esse modelo, portanto, possui um nó de entrada e um nó de saída, pelo menos (KIM et al., 2016).

A estrutura da RNA pode ser vista na figura 2.4. A figura mostra uma rede com uma entrada de n valores de x que se relacionam através de um peso w. As entradas e os pesos serão somados e aplicados a uma função de ativação $f(x)$ para prosseguir para a próxima camada da RNA. Esse processo se repete até a última camada em que gera uma saída y. O processo de treinamento de uma RNA é o reajuste dos valores do peso através de uma função de erro da rede.

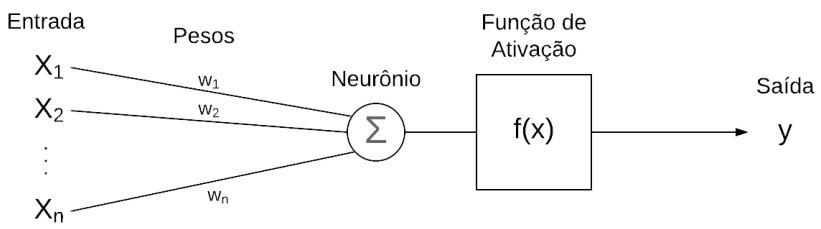

Figura 2.4: Composição da estrutura de uma RNA. Fonte (FURTADO, 2019).

Matematicamente, a estrutura da RNA é composta pela equação 2.4 , onde para cada neurônio k é realizado um somatório com todas as entradas m. A cada entrada x_i aplicam-se os pesos w_{ki}, e ao final esse valor é somado ao bias bk. O bias é um elemento utilizado para aumentar o grau de liberdade dos ajustes dos pesos, gerando por fim na rede uma saída y_k.

$$y_k = f\left(\left(\sum_{i=1}^{m} w_{ki} * x_i\right) + b_k\right) \tag{2.4}$$

De fato, também é necessário ressaltar que as RNAs atuais não apresentam uma preocupação em reproduzir o aspecto biológico fielmente, mas adaptar-se ao modelo proposto para resolução do problema e também paralelizar o processamento para otimização de desempenho (BARRETO, 2002).

Atualmente há diversas arquiteturas de redes neurais artificiais, como: MLP, Redes Neurais Convolucionais e Redes Neurais Recorrentes. Tais arquiteturas se diferem entre quantidade de camadas, função de ativação, entre outras características.

Além disso, essas arquiteturas implementam técnicas para melhorar a precisão e acurácia da rede neural, variando desde quantidade de nós e camadas, à função de ativação e peso entre camadas (FRANK et al., 2019).

2.2.5 MLP

Multi-Layer Perceptron é um classificador muito comum em diversas aplicações de problemas de classificação. O MLP se caracteriza por ser um rede neural *feedforward* (onde os neurônios seguem apenas em uma direção) e possui três ou mais camadas, sendo pelo menos uma de entrada, uma oculta e uma de saída, usando um método supervisionado chamado de *backpropagation*.

O MLP apresenta apenas conexões unidirecionais entre seus neurônios e é organizado em camadas paralela. A primeira camada é também chamada de entrada, e a última camada também é chamada de camada de saída, já as camadas entre as de entrada e saída são chamadas de camadas ocultas. A figura 2.5 mostra a estrutura de camadas do MLP.

O *backpropagation* é um método desenvolvido com base no erro da camada de

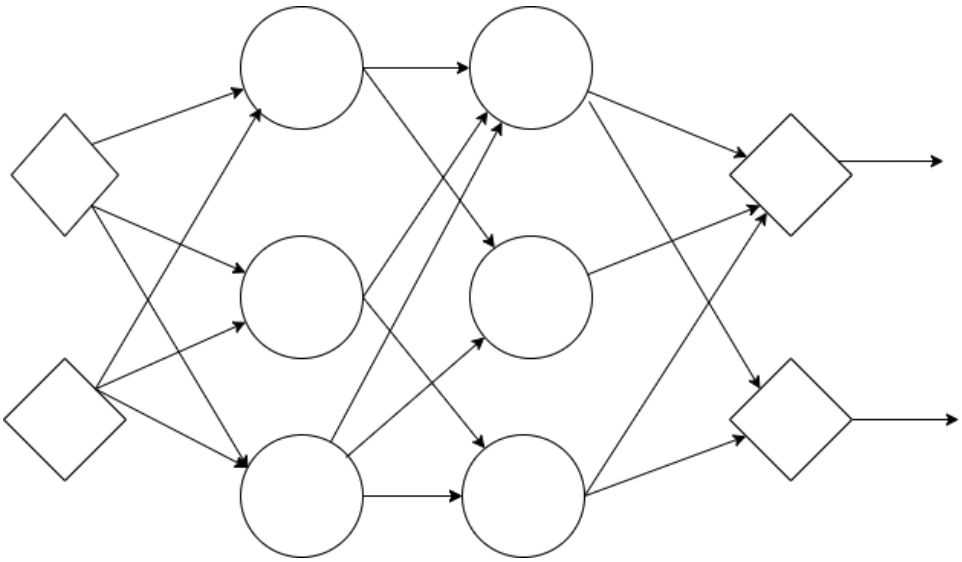

Figura 2.5: Estrutura de camadas do MLP.

saída da rede neural. A partir dela se recalcula o valor do vetor de pesos da última camada, e em seguida o propaga para as camadas anteriores, ocorrendo assim a retro propagação do erro obtido pela RNA.

Através da equação 2.5 é possível ver que o *backpropagation* atualiza o valor do peso w_i com base no valor do peso anterior w_{i-1} e do vetor gradiente de cada peso da RNA. O gradiente retorna a função de maior crescimento, e o parâmetro η representa a taxa de atualização da RNA. Quanto maior esse parâmetro, maior a correção do peso.

$$w_i \leftarrow w_{i-1} - \eta \frac{\partial E}{\partial W} \tag{2.5}$$

O algoritmo MLP pode utilizar diferentes funções de ativação, como: Lineares, Sigmoides e Hiperbólicos. Já o processo de treinamento do MLP se divide em duas etapas na seleção da arquitetura da rede neural e o ajuste de pesos (GHANEM; JANTAN, 2019). Os pesos e o fator de polarização são iniciados randomicamente no nó, e conforme o treinamento acontece, esses vão sendo ajustados. É importante destacar que o treinamento de uma MLP é o período mais longo de tempo do algoritmo, sendo classificado como um problema de otimização.

3 Datasets

Neste capítulo iremos elucidar como foram feitas as escolhas para avaliação de dois *datasets* (conjunto de dados) reais. Além disso, comentaremos as características de dados e o tratamento aplicado aos dados.

3.1 UNSW-NB15

O *dataset* UNSW-NB15 é referente ao tráfego de redes híbridas contendo tanto atividades normais de rede como também diversas atividades maliciosas, gerando um total de 100 GB de tráfego bruto para análise (MOUSTAFA; SLAY, 2015).

Essa rede híbrida, conforme mostra a figura 3.1, é composta de três servidores virtuais, sendo dois deles responsáveis pelo tráfego normal e um deles responsável pelo tráfego malicioso. O *firewall* foi configurado para permitir que passe todo o tráfego de rede e assim não interferir no tráfego malicioso, já o `tcpdump` é uma ferramenta para capturar os dados e gerar um arquivo com os dados do pacote de rede.

Combinando os dados da rede com um algoritmo é gerado o *dataset*. Além disso, ele é dividido em dois sub-*datasets*, o *training dataset* utilizado para treinamento do algoritmo e o *testing dataset* utilizado para o teste do algoritmo. No total ele contém 45 atributos, como mostrado na tabela 3.1. Foram coletados e disponibilizados também características de dados qualitativos e quantitativos e divididos em 6 categorias, são elas: a categoria de fluxo, categoria básica, categoria de conteúdo, categoria de tempo, categoria adicional e categoria de rótulos.

3.1.1 Pré-Processamento

A primeira etapa do pré-processamento foi retirar as colunas de *id* que não interessavam para a execução do algoritmo. Em sequência foram removidas todas as linhas que continham dados faltantes, Um total de 45 linhas das 82 mil linhas existentes. Como o número era pequeno em relação a quantidade de dados não foi necessário outro tipo de tratamento

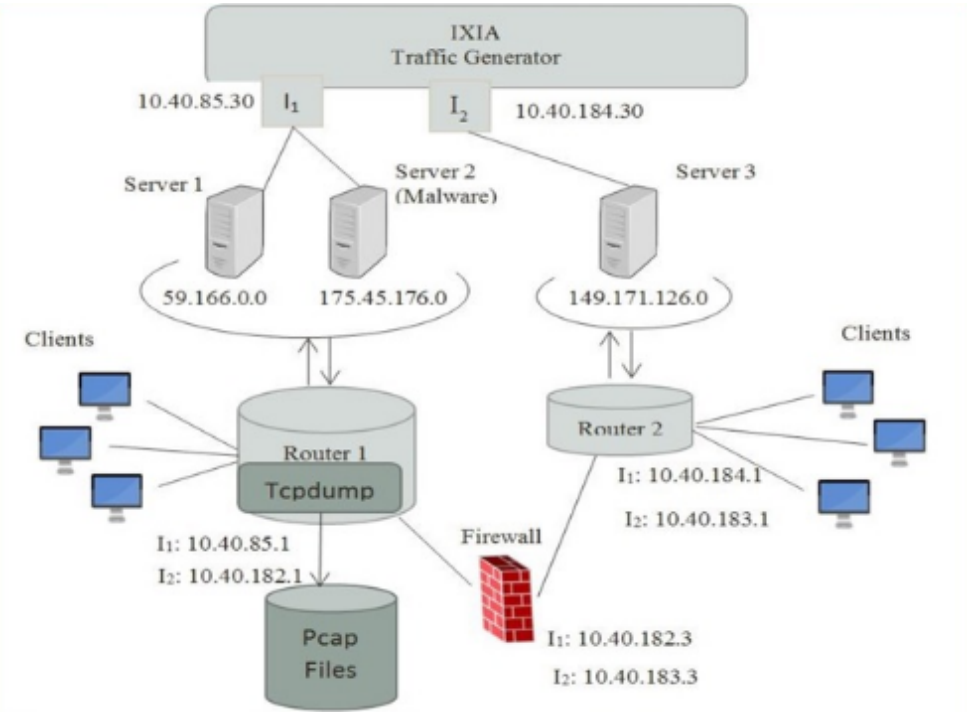

Figura 3.1: Rede construída para gerar o dataset. Fonte: (MOUSTAFA; SLAY, 2015).

devido à baixa interferência desses dados faltantes seguindo a lógica do algoritmo 1.

Algoritmo 1: Remover todas as linhas com itens nulos

```
Entrada: dataset, colunas
Saída  : datasetAtualizado
d ← dataset;
for col ← 0 até colunasTamanho do
    for lin ← 0 até linhasTamanho do
        usa uma coluna col;
        percorre lin para verificar todas as linhas da tabela;
        if d.col.lin = '−' then
            d ← d.removeLin.lin
        end if
    end for
end for
```

Não foram tratados os *outliers* – que são dados que se diferenciam muito do padrão – pois são justamente esse tipos de dados que são configurados como ataques. Mesmo podendo haver falsos positivos, a incidência é muito maior para ataques reais sobre tais dados, e, além disso, são essas discrepâncias que ajudarão no treinamento dos modelos propostos.

Por fim, foram então selecionado as colunas com atributos de tipo objeto e re-

Número	Nome	Número	Nome	Número	Nome
1	id	16	dloss	31	response_body_len
2	dur	17	sinpkt	32	ct_srv_src
3	proto	18	dinpkt	33	ct_state_ttl
4	service	19	sjit	34	ct_dst_ltm
5	state	20	djit	35	ct_src_dport_ltm
6	spkts	21	swin	36	ct_dst_sport_ltm
7	dpkts	22	sttcpb	37	ct_dst_src_ltm
8	sbytes	23	dtcpb	38	is_ftp_login
9	dbytes	24	dwin	39	ct_ftp_cmd
10	rate	25	tcprtt	40	ct_flw_http_mthd
11	sttl	26	synack	41	ct_src_ltm
12	dttl	27	ackdat	42	ct_srv_dst
13	sload	28	smean	43	is_sm_ips_ports
14	dload	29	dmean	44	attack_cat
15	sloss	30	trans_depth	45	label

Tabela 3.1: Atributos disponíveis no *dataset* UNSW-NB15.

movida a coluna de *attack-cat* dessas colunas, que indica se o pacote é um ataque ou se é uma atividade normal e através desse colunas as colunas `proto`, `state` e `service` foram identificadas como categóricas, com isso cada uma dessas categorias virou um indicador, criando um conjunto de novas colunas que auxilia os modelos a criarem grupos e predizer os ataques.

3.2 NSL-KDD

O *dataset* KDD, é uma simulação de uma rede militar de tráfego com um conjunto de seis máquinas. Nele três máquinas executam um sistema operacional e serviços, e três máquinas são usadas para falsificar endereços de IP. A rede se completa com um *sniffer* para registrar os pacotes de dados TCP (KAYACIK; ZINCIR-HEYWOOD; HEYWOOD, 2005).

Já o NSL-KDD é uma rede *offline* baseada no *dataset* KDD, com o intuito de suprir os problemas inerentes contidos no *dataset* KDD como a redundância de registros. Tal problema pode ser ilustrado pelo fato de que o KDD apresenta cerca de 75% dos registros duplicados, gerando assim uma análise discrepante da realidade e um baixo desempenho ao analisar especificamente esse KDD.

O NSL-KDD apresenta um número razoável de registros, o que não dificulta a

análise do *dataset*, e nem gera excesso de uso da memória para sua avaliação. O conjunto de treino tem aproximadamente 4 milhões de entradas e 900 mil registros, já o conjunto de testes tem 300 mil registros. Desta forma, esse *dataset* se mostra um excelente *benchmark* na área de detecção de intrusão.

Seus atributos são em um total de 41, conforme a tabela 3.2. Estão divididos em 4 categorias: categoria básica, categoria de conteúdo, categoria de tempo e categoria baseada no *host* com registros de atividades normais e de 4 tipos de ataque DDoS, Probe, R2L e U2R.

Número	Nome	Número	Nome
1	duration	23	count
2	protocol_type	24	srv_count
3	service	25	serror_rate
4	flag	26	srv_serror_rate
5	src_bytes	27	rerror_rate
6	dst_bytes	28	srv_rerror_rate
7	land	29	same_srv_rate
8	wrong_fragment	30	diff_srv_rate
9	urgent	31	srv_diff_host_rate
10	hot	32	dst_host_count
11	num_failed_logins	33	dst_host_srv_count
12	logged_in	34	dst_host_same_srv_rate
13	num_compromised	35	dst_host_diff_srv_rate
14	root_shell	36	dst_host_same_src_port_rate
15	su_attempted	37	dst_host_srv_diff_host_rate
16	num_root	38	dst_host_serror_rate
17	num_file_creations	39	dst_host_srv_serror_rate
18	num_shells	40	dst_host_rerror_rate
19	num_access_files	41	dst_host_srv_rerror_rate
20	num_outbound_cmds	42	attack
21	is_host_login	43	level
22	is_guest_login		

Tabela 3.2: Atributos disponíveis no banco de dados NSL-KDD.

3.2.1 Pré Processamento

A primeira etapa do pré-processamento é a criação de uma coluna de rotulação de ataque, pois para otimizar o treinamento do algoritmo é necessário indicar no registro se é um ataque ou não.

O segundo passo foi a identificação das colunas categóricas como ocorre também

com o UNSW-NB15, transformando os atributos *protocol_type*, *service* e *flag* em indicadores categóricos.

O terceiro passo foi a identificação de colunas numéricas categóricas, que foram: *duration*, *src_bytes* e *dst_bytes*. Por sua vez, tais colunas também foram transformadas em indicadores.

Com isso o pré-processamento do NSL-KDD se diferencia levemente do UNSW-NB15, uma vez que não foi preciso a remoção das colunas desnecessárias. Isso porque a quantidade de dados deste *dataset* é menor, e assim ocupa menos recurso computacional (*i.e.*, memória).

4 Resultados

Nesse capítulo serão apresentados e discutidos os resultados obtidos pelos algoritmos de *machine learning* aplicados sobre os *datasets* UNSW-NB15 e NSL-KDD.

Os algoritmos propostos foram executados em computador Intel Core i3-8100 CPU Intel Core i5-7400 CPU @ 3.60GHz no sistema operacional Linux Mint na linguagem *Python* na versão 3.6.9[1]. Os algoritmos de *machine Learning* foram executados com auxílio da biblioteca *scikit-learn*[2] se utilizando também para gerar as métricas de avaliação, os gráficos foram utilizados a biblioteca *matplotlib*[3], no pré-processamento foi usado algumas outras bibliotecas para leitura e tratamento de dados.

4.1 Métricas

Cada algoritmo apresenta suas vantagens e desvantagens para implementação e execução do programa. Além disso, é preciso levar em conta as características de uma IDS para se obter uma avaliação precisa dos resultados obtidos.

Para a medição do desempenho dos algoritmos usamos duas métricas, a acurácia e o *f1-score*. A acurácia é a medida que indica a porcentagem de acerto do modelo, ou seja, qual a porcentagem em que o modelo julga que é ataque ou normal e se confirma sendo verdade. O *f1-score* é uma medida de precisão do modelo, é a média harmônica entre a precisão e o *recall*, sendo o *recall* a porcentagem de dados classificados como ataque que são realmente ataques pelo modelo sobre o total de ataques contidos no banco de dados.

$$F1 = 2 \times \frac{precisao \times recall}{precisao + recall} \tag{4.1}$$

Outra métrica utilizada foi tempo de execução, que mede o tempo de execução de cada modelo de algoritmo e foi limitado a implementação das soluções com no máximo 10 minutos de execução.

[1] https://www.python.org/
[2] https://github.com/scikit-learn/scikit-learn
[3] https://github.com/matplotlib/matplotlib

Para avaliação dos atributos do *dataset* foi utilizada a matriz de correlação (correlação de Pearson) que se utilização do coeficiente de correlação e mede a intensidade e a direção da associação entre dois atributos. O coeficiente varia entre -1 e 1, sendo 0 quando os atributos não se associam. Quanto maior o coeficiente, maior a associação do atributo.

4.2 Avaliação dos modelos

4.2.1 Árvore de Decisão

O primeiro algoritmo implementado foi a árvore de decisão. Alguns fatores influenciam o desempenho de uma árvore de decisão, sendo necessário avaliar e testar esses fatores relevantes. Os seguintes fatores foram avaliados:

- O primeiro fator é o fator da qualidade de escolha dos nós de decisão e se utilizando da medida de impureza.
- O segundo fator é a escolha de divisão de cada nó podendo ser a melhor escolha ou uma aleatória.

Impureza é a medida da homogeneidade dos rótulos de um nó, sendo implementado dois principais métodos, a impureza de Gini e a medida de entropia.

A impureza de Gini mede o grau de heterogeneidade dos dados buscando a obtenção do menor grau de impureza, Já a medida de entropia divide o nó de forma que se obtenha o maior ganho de informação, segundo a equação 4.2.

$$\begin{aligned} Gini \leftarrow G(x) &= 1 - \sum_{j=1}^{c} p_j^2 \\ Entropia \leftarrow E(x) &= -\sum_{j=1}^{c} \times p_j \times log p_j \end{aligned} \tag{4.2}$$

Variando o grau de impureza e a escolha de cada nó se obteve seguintes resultados de acordo com a tabela 4.1.

Árvore de Decisão	Acurácia UNSW-NB15	F1-score UNSW-NB15	Acurácia NSL-KDD	F1-score NSL-KDD
Impureza-Gini	91.44	93.34	99.27	99.22
Impureza-Entropia	91.48	93.39	99.27	99.22
Escolha Aleatória	91.08	93.07	99.26	99.20

Tabela 4.1: Resultados obtidos em uma árvore de decisão.

Conclui-se que o melhor desempenho foi quando se utiliza o fator de qualidade de Entropia, por apresentar melhor acurácia e *F1-Score*. Contudo nota-se que variar a impureza para a análise dos dois *datasets* não apresenta uma grande relevância, mas em relação a escolha aleatória apresenta um pequeno ganho em relação ao UNSW-NB15, se tornando relevante devido a quantidade de entradas de registros.

4.2.2 Random Forest

O segundo algoritmo implementado é o *Random Forest*. Os fatores citados na seção anterior são também de grande relevância no algoritmo devido ao *Random Forest* ser uma escolha de muitas árvores de decisões. Contudo, outro fator relevante é a quantidade de árvores geradas pelo algoritmo. Em geral, quanto maior a quantidade de árvores geradas, maior a precisão. Em contrapartida ao aumento do tempo treinamento do algoritmo.

Esse aspecto se confirma conforme a tabela 4.2. Dos testes realizados se conclui também que o fator de impureza de Entropia é melhor aplicado para o algoritmo de *Random Forest* em ambos *datasets*.

Random Forest	Acurácia UNSW-NB15	F1-score UNSW-NB15	Acurácia NSL-KDD	F1-score NSL-KDD
Impureza-Gini 100 árvores	92.59	94.24	99.34	99.23
Impureza-Entropia 100 árvores	92.61	94.25	99.35	99.22
Impureza-Gini 200 árvores	92.63	94.27	99.35	99.23
Impureza-Entropia 200 árvores	92.65	94.29	99.35	99.23

Tabela 4.2: Resultados obtidos no algoritmo *Random Forest*.

4.2.3 MLP

O terceiro algoritmo implementado foi o MLP, e os fatores avaliados para a construção do algoritmo foram os seguintes:

- A quantidade de camadas ocultas no MLP.
- A função de ativação da camada oculta.
- O solucionador para otimização dos pesos.

Com isso, para avaliar o desempenho foi implementado algoritmos que variam a quantidade de camadas ocultas e as funções de ativação. As funções de ativações escolhidas para a avaliação foram relu e a hiperbólica, conforme a equação 4.3.

$$\begin{aligned} relu &\leftarrow f(x) = max(0, x) \\ hiperbolica &\leftarrow f(x) = tanh(x) \end{aligned} \tag{4.3}$$

Com isso se obteve os seguintes resultados avaliando 10 e 20 camadas ocultas e as duas funções de ativação. De acordo com a tabela 4.3, observa-se que a função Relu com 20 camadas ocultas tem o melhor desempenho no *dataset* NSL-KDD e a função de ativação hiperbólica com 20 camadas tem o melhor desempenho no *dataset* UNSW-NB15.

MLP	Acurácia UNSW-NB15	F1-score UNSW-NB15	Acurácia NSL-KDD	F1-score NSL-KDD
Relu 10 camadas ocultas	89.84	92.20	92.62	92.41
Relu 20 camadas ocultas	90.89	92.96	93.90	93.66
Hiperbolica 10 camadas ocultas	91.2	93.15	92.63	93.22
Hiperbolica 20 camadas ocultas	91.65	93.50	93.32	93.80

Tabela 4.3: Resultados obtidos no algoritmo MLP.

4.2.4 SVM

O quarto algoritmo implementado foi o SVM, utilizando o SVM com a função *kernel* linear para as avaliações seguindo dois fatores. O fator de penalidade quanto as margens

suaves, ou seja, fator que permite uma determinada taxa de registros que podem violar a margem da sua classe. Outro fator levado foi a função de perda, também chamada de função custo, que mapeia um custo para o algoritmo que deve ser minimizado no SVM.

Na função de perda foram escolhidas a perda de articulação e a perda quadrática, ambas buscam maximizar a distância do hiperplano entre duas classes do SVM, o fator de penalidade C foi variado entre $0,1$ e 1, onde o 1 gera um modelo mais generalista e o 0,1 um modelo mais específico para o *dataset*.

Segundo a tabela 4.4 com os resultados obtidos, conclui-se que para os *dataset* UNSW-NB15 e NSL-KDD os fatores com melhores desempenho foram o $C = 0,1$ e com a função de perda de articulação.

SVM	Acurácia UNSW-NB15	F1-score UNSW-NB15	Acurácia NSL-KDD	F1-score NSL-KDD
C=1 Perda de articulação	89.14	92.04	87.78	86.79
C=1 Perda Quadrática	89.11	91.90	86.96	84.61
C=0,1 Perda de articulação	89.3	92.17	91.16	91.66
C=0,1 Perda Quadrática	89.0	91.83	91.92	92.33

Tabela 4.4: Resultados obtidos no algoritmo SVM.

4.3 Avaliação do UNSW-NB15

Após a avaliação dos algoritmos individualmente para a escolha dos melhores fatores e parâmetros em cada algoritmo é necessário a comparação dos algoritmos em relação a cada *dataset*. Desta forma foram analisados os *datasets* individualmente para a geração de uma IDS ideal.

É apresentado um gráfico de correlação dos atributos na figura 4.1 para se entender a dependência de cada atributo em relação ao outro. É possível perceber que muitos atributos não se relacionam com outros e poderiam ser retirados do *dataset* sem nenhuma interferência na criação dos modelos. Isso se dá pela correlação baixa em relação a todos os outros, representados pelos tons mais escuros de cor.

Para a análise dos resultados obtidos foram inseridos os melhores desempenhos de

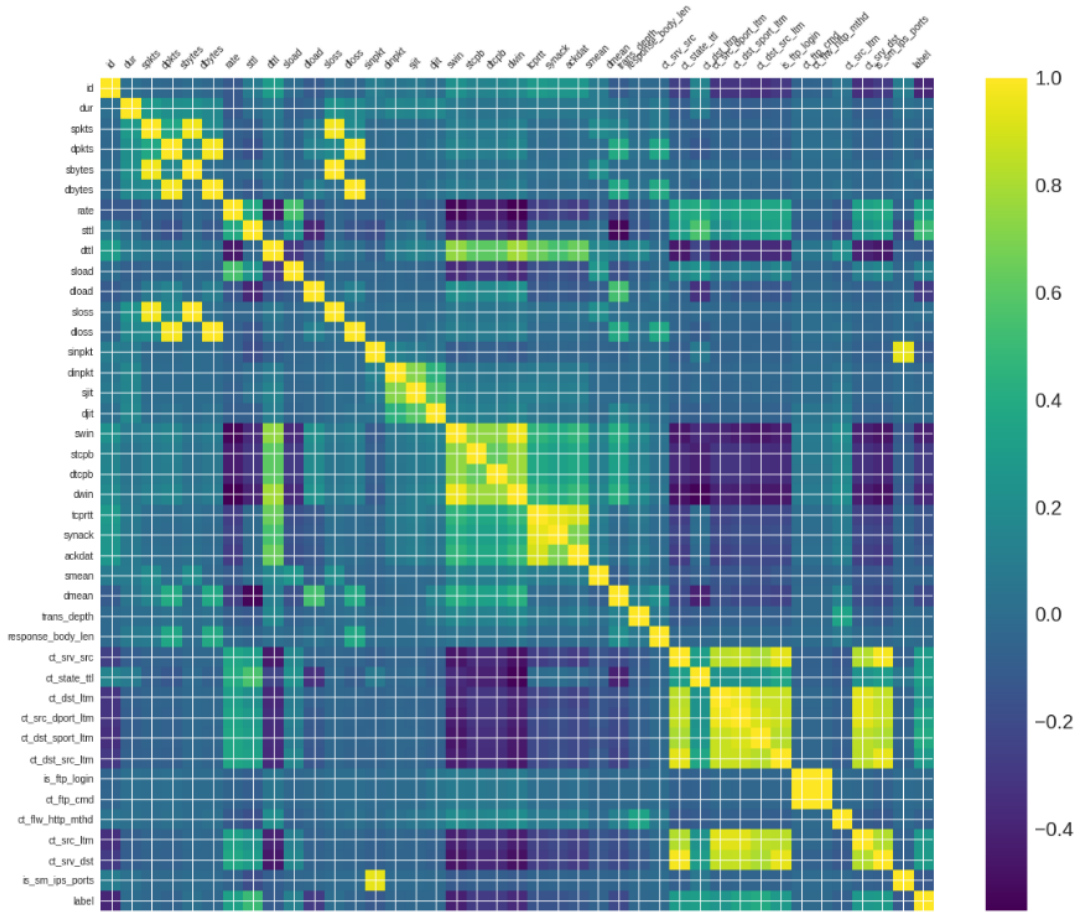

Figura 4.1: Gráfico de correlação.

cada algoritmo. No caso do MLP, visto que é o único que apresenta desempenho diferente para cada *dataset*, foi escolhida a função de ativação hiperbólica com 20 camadas que foi o algoritmo de melhor desempenho em relação ao *dataset* UNSW-NB15, gerando os gráficos de acurácia e *F1-score* nas figuras 4.2 e 4.3.

O *Random Forest* apresentou o melhor acurácia e *F1-score*, maior que a árvore de decisão como esperado, já que é uma variedade de árvores e também maior que o MLP e o SVM. O MLP tem desempenho relativamente próximo da árvore de decisão e pode-se especular que o aumento da quantidade de camadas ocultas forneceria um melhor resultado tornando-o um candidato para uma implementação da IDS junto com a *Random Forest*, e o SVM apresentou o pior desempenho dentre os algoritmos implementados.

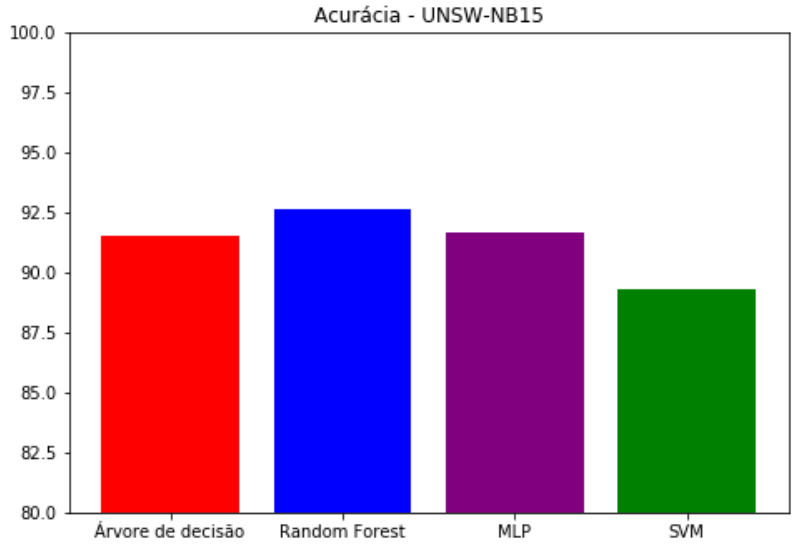

Figura 4.2: Gráfico de acurácia do UNSW-NB15.

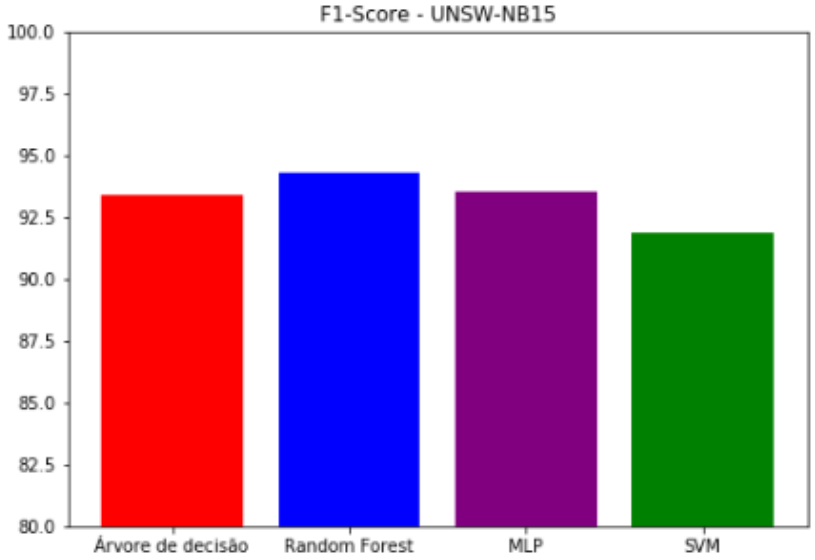

Figura 4.3: Gráfico de F1-score do UNSW-NB15.

4.4 Avaliação do NSL-KDD

Para a avaliação do NSL-KDD foi empregado também um gráfico de correlação conforme a figura 4.4.

Nota-se que a correlação entre os atributos tem uma leve melhora em relação ao UNSW-NB15. Contudo também é possível ver atributos com correlação baixa com todos os outros, conforme a barra de cores. Isso indica que para a construção de uma IDS mais rápida e ágil é desejável a remoção desses atributos nos modelos construídos.

Para a análise dos resultados obtidos, semelhantemente ao UNSW-NB15 foram utilizamos os algoritmos com melhor desempenho. Logo o MLP escolhido para a avaliação

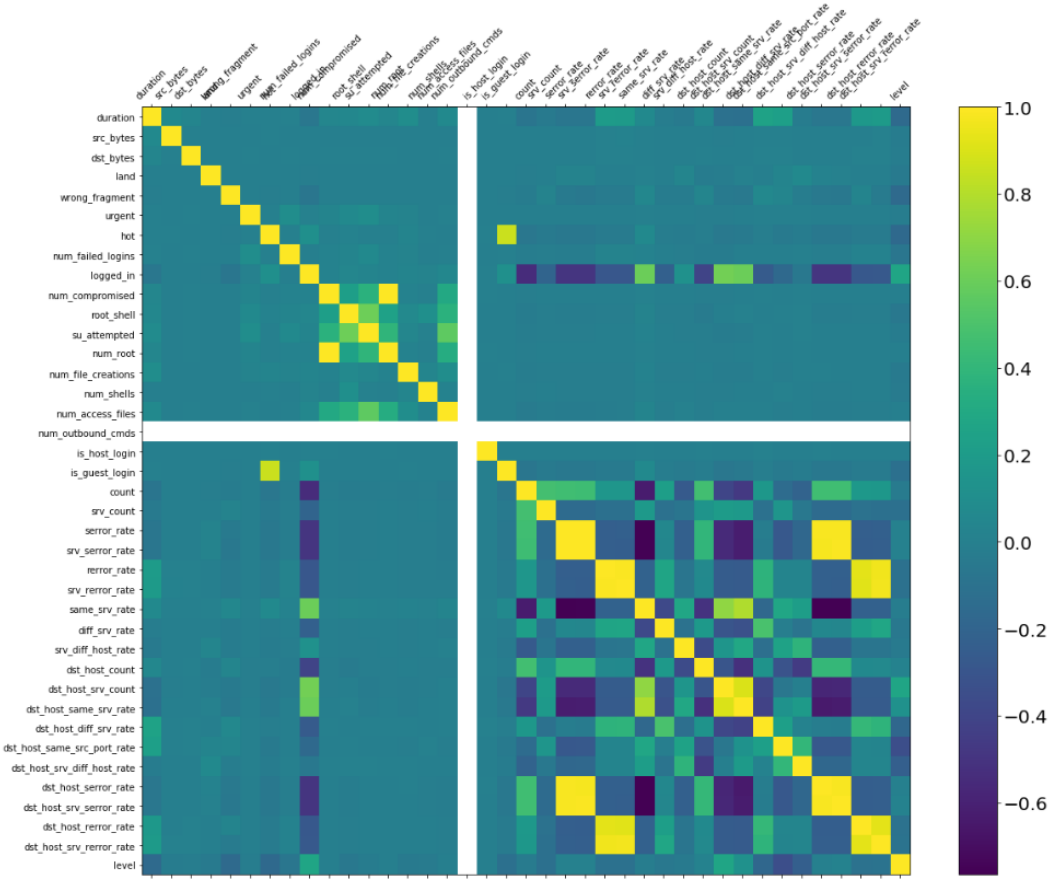

Figura 4.4: Gráfico de correlação.

no NSL-KDD é com a função de ativação do Relu e 20 camadas ocultas, diferentemente do utilizado no UNSW-NB15, os outros algoritmos permanecem com os mesmo fatores e parâmetros.

Foram obtidos os valores conforme os gráficos nas figuras 4.5 e 4.7.

O algoritmo de *Random Forest* apresentou a maior acurácia e *F1-score*. Mas devemos destacar também que a árvore de decisão apresentou resultado muito próximo, levando a inferir que não é tão relevante a utilização de diversas árvores. Além disso, a acurácia foi alta, obtendo 99.27% na árvore de decisão.

Já o MLP aplicado ao NSL-KDD não apresentou resultado próximo a árvore de decisão, como havia acontecido quando ele foi avaliado ao UNSW-NB15. Por fim, o SVM foi o algoritmo com o pior desempenho de todos os algoritmos implementados.

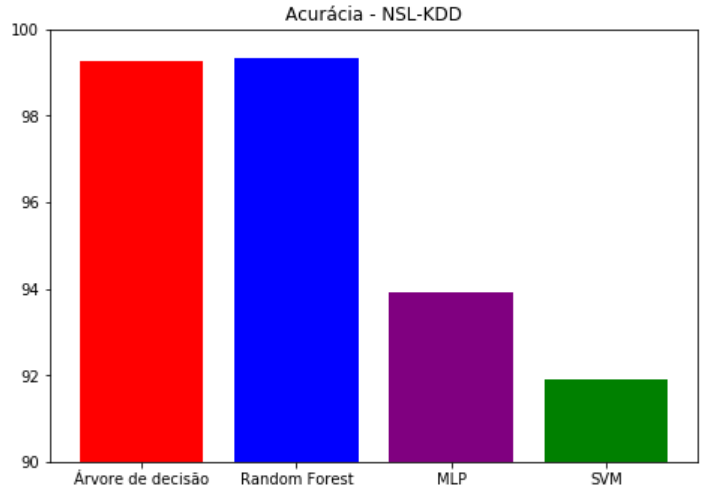

Figura 4.5: Gráfico de acurácia do NSL-KDD.

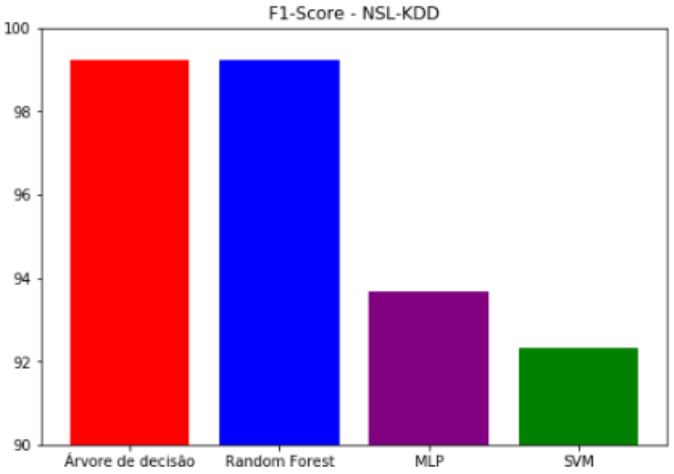

Figura 4.6: Gráfico de F1-score do NSL-KDD.

4.4.1 Comparação em tempo de execução

Esta seção demonstra a comparação em tempo de execução para cada um dos algoritmos e os respectivos *datasets*. Destacamos que o tempo de execução foi verificado apenas para as avaliações/teste, e não o treinamento.

Fica claro que em ambos *datasets* o algoritmo *Random Forest* é o que leva maior tempo de execução, com um valor até $\approx 10\times$ maior para o NSL-KDD comparado à Árvore de Decisão e $\approx 6\times$. O MLP comparado a árvore de decisão tem um tempo de $\approx 6\times$ maior de execução no UNSW-NB15 mesmo apresentando resultado de acurácia e *F1-Score* próximo. Com isso em termos de tempo de execução o algoritmo de árvore de decisão é o mais eficiente.

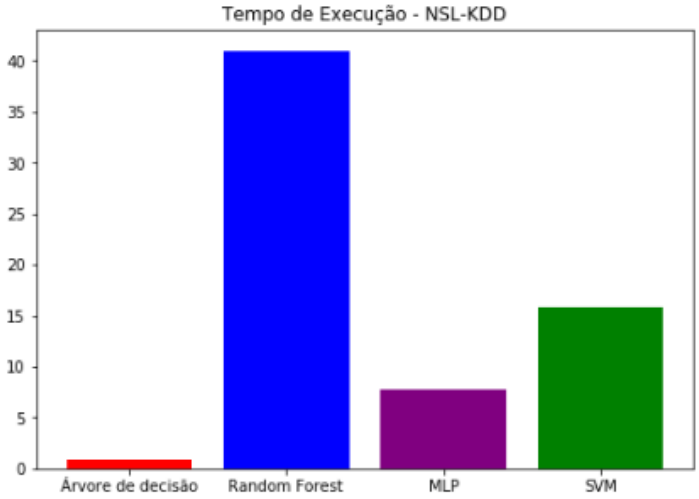

Figura 4.7: Gráfico de Tempo de Execução em segundos do NSL-KDD.

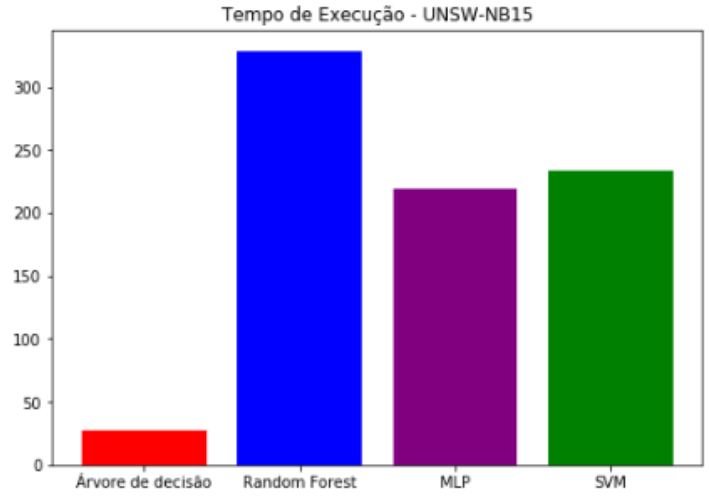

Figura 4.8: Gráfico de tempo de execução em segundos do UNSW-NB15.

5 Conclusões e Trabalhos Futuros

Este trabalho teve como propósito o estudo de IDS e técnicas de *machine learning*, avaliando em *datasets* reais os modelos quanto a sua acurácia e F1-score para se encontrar o melhor modelo com taxa de acerto e precisão para uma IDS.

O UNSW-NB15 é um *dataset* moderno conforme citado na seção 3.1, com isso apresenta desafios como o tamanho dos dados, quantidade de registros e variedade de ataques. Já o NSL-KDD é baseado um *dataset* de 1998 o KDD com isso apresenta dados mais antigos, mas que geram desafios diferentes como empregar técnicas diferentes para o pré-processamento e características de ataques mais antigas.

Ao realizar o treinamento dos algoritmos em ambos os *datasets* se obtém um modelo robusto com alta taxa de acerto e precisão para inserir em uma IDS. Além disso o *Random Forest* se mostra o modelo ideal para o uso em IDSs em que o tempo de execução não é relevante, visto que apresenta a melhor acurácia e *f1-score* e pode-se escolher uma quantidade grande de árvore para obtenção do melhor modelo.

Para IDS em que o contexto de treinamento pode-se tornar relevante o modelo escolhido será a árvore de decisão por apresentar a segunda maior acurácia e *f1-score* e ser relativamente próxima da *Random Forest* e um método com menor tempo de execução.

É importante salientar que o MLP e o SVM apresentam resultados razoáveis, sendo opções também interessantes para o emprego em IDSs contudo é preciso um estudo um pouco mais afundo em cada caso para avaliar outros parâmetros para otimização dos modelos.

Para trabalhos futuros o objetivo é a construção de uma IDS baseada na arquitetura *Lambda* (MEDEIROS et al., 2020) para processamento em tempo real e para melhorar o desempenho em relação a grandes fluxos de dados. Também pretendemos analisar modelos com novos *datasets*, como o *dataset* Bot-IoT (KORONIOTIS et al., 2019), dos mesmos criadores do UNSW-NB15 baseado em dispositivos IoT. O que possibilitaria uma maior gama de resultados e novos desafios no processamento e análise dos dados.

Outro aspecto de melhoria para trabalhos futuros é implementar métodos de

pré processamento que reduzem a quantidade e complexidade dos dados, levando em consideração a matriz de correlação, visto que algoritmos de *Machine Learning* exigem muito poder de processamento para o treinamento.

Bibliografia

ALMSEIDIN, M. et al. Evaluation of machine learning algorithms for intrusion detection system. In: IEEE. *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*. [S.l.], 2017. p. 000277–000282.

ALQAHTANI, H. et al. Cyber intrusion detection using machine learning classification techniques. In: SPRINGER. *International Conference on Computing Science, Communication and Security*. [S.l.], 2020. p. 121–131.

AMUDHA, P.; KARTHIK, S.; SIVAKUMARI, S. Classification techniques for intrusion detection-an overview. *International Journal of Computer Applications*, Citeseer, v. 76, n. 16, 2013.

ARTHUR, M. P. An svm-based multiclass ids for multicast routing attacks in mobile ad hoc networks. In: IEEE. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. [S.l.], 2018. p. 363–368.

BARRETO, J. M. Introduçaoas redes neurais artificiais. *V Escola Regional de Informática. Sociedade Brasileira de Computaçao, Regional Sul, Santa Maria, Florianópolis, Maringá*, p. 5–10, 2002.

FARNAAZ, N.; JABBAR, M. Random forest modeling for network intrusion detection system. *Procedia Computer Science*, Elsevier, v. 89, n. 1, p. 213–217, 2016.

FORECAST, G. Cisco visual networking index: Global mobile data traffic forecast update 2017–2022. *Update*, v. 2017, p. 2022, 2019.

FRANK, L. R. et al. Multilayer perceptron and particle swarm optimization applied to traffic flow prediction on smart cities. In: SPRINGER. *International Conference on Computational Science and Its Applications*. [S.l.], 2019. p. 35–47.

FURTADO, M. I. V. *Redes Neurais Artificiais: Uma Abordagem Para Sala de Aula*. [S.l.]: Atena Editora, 2019.

GHANEM, W. A.; JANTAN, A. A new approach for intrusion detection system based on training multilayer perceptron by using enhanced bat algorithm. *Neural Computing and Applications*, Springer, p. 1–34, 2019.

JAVAID, A. et al. A deep learning approach for network intrusion detection system. In: *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. [S.l.: s.n.], 2016. p. 21–26.

JUNIOR, G. Máquina de vetores suporte: estudo e análise de parâmetros para otimização de resultado. *Ciência da Computação, Universidade Federal de Pernambuco*, 2010.

JUNIOR, J. S. Análise das ferramentas de ids snort e prelude quanto à eficácia da detecção de ataque e na proteção quanto à evasões. *Revista Tecnologia e Tendências*, v. 3, n. 1, p. 19–24, 2004.

KAYACIK, H. G.; ZINCIR-HEYWOOD, A. N.; HEYWOOD, M. I. Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets. In: CITESEER. *Proceedings of the third annual conference on privacy, security and trust*. [S.l.], 2005. v. 94, p. 1723–1722.

KIM, J. et al. Long short term memory recurrent neural network classifier for intrusion detection. In: IEEE. *2016 International Conference on Platform Technology and Service (PlatCon)*. [S.l.], 2016. p. 1–5.

KORONIOTIS, N. et al. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, Elsevier, v. 100, p. 779–796, 2019.

LAZAREVIC, A. et al. A comparative study of anomaly detection schemes in network intrusion detection. In: SIAM. *Proceedings of the 2003 SIAM international conference on data mining*. [S.l.], 2003. p. 25–36.

LI, J.; ZHAO, Z.; LI, R. Machine learning-based ids for software-defined 5g network. *IET Networks*, IET, v. 7, n. 2, p. 53–60, 2017.

LUNT, T. F. Automated audit trail analysis and intrusion detection: A survey. In: *Proceedings of the 11th National Computer Security Conference*. [S.l.: s.n.], 1988. p. 65–73.

MAGALHÃES, L. M.; OLIVEIRA, F. H. L. d. Identificação de condições funcionais em pavimentos urbanos auxiliada por machine learning. 2020.

MEDEIROS, D. S. et al. A survey on data analysis on large-scale wireless networks: online stream processing, trends, and challenges. *Journal of Internet Services and Applications*, SpringerOpen, v. 11, n. 1, p. 1–48, 2020.

MITCHELL, R.; CHEN, I.-R. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 46, n. 4, p. 1–29, 2014.

MOON, D. et al. Dtb-ids: an intrusion detection system based on decision tree using behavior analysis for preventing apt attacks. *The Journal of supercomputing*, Springer, v. 73, n. 7, p. 2881–2895, 2017.

MOUSTAFA, N.; SLAY, J. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: IEEE. *2015 military communications and information systems conference (MilCIS)*. [S.l.], 2015. p. 1–6.

PANDA, M. et al. Network intrusion detection system: A machine learning approach. *Intelligent Decision Technologies*, IOS Press, v. 5, n. 4, p. 347–356, 2011.

SALEM, M.; TAHERI, S.; YUAN, J. S. Anomaly generation using generative adversarial networks in host-based intrusion detection. In: IEEE. *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. [S.l.], 2018. p. 683–687.

SANTANA, F. B. d. et al. Floresta aleatória para desenvolvimento de modelos multivariados de classificação e regressão em química analítica. [sn], 2020.

SILVA, E. F.; JULIO, E. P. Sistema de detecção de intrusão. *DevMedia. Disponível em: https://www. devmedia. com. br/sistema-de-deteccao-de-intrusao-artigo-revista-infra-magazine-1/20819. Acesso em*, v. 10, 2011.

SULTANA, N. et al. Survey on sdn based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications*, Springer, v. 12, n. 2, p. 493–501, 2019.

VINAYAKUMAR, R. et al. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, IEEE, v. 7, p. 41525–41550, 2019.

WAGH, S. K.; PACHGHARE, V. K.; KOLHE, S. R. Survey on intrusion detection system using machine learning techniques. *International Journal of Computer Applications*, Foundation of Computer Science, v. 78, n. 16, 2013.