UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

TIAGO DE CARVALHO MAGNUS

Impacto da criptografia da camada de transporte em análises de fluxos com aprendizado de máquina

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Jéferson de Campos Nobre

Porto Alegre
2023

“Tudo, tudo, tudo, tudo
que nóis tem é nóis.”
— EMICIDA, PRINCIPIA (2019)

AGRADECIMENTOS

Agradeço à toda minha família, por todo o amor, ensinamentos e suporte que me deram em todos os momentos da minha vida; à Januário e Jorel, pelo amor, conforto e carinho em momentos que só vocês poderiam proporcionar; à Universidade Federal do Rio Grande do Sul, pelo acesso à educação pública, gratuita e de qualidade e pelas políticas de permanência e assistência estudantil, promovidas pela Pró-Reitoria de Assuntos Estudantis, que foram fundamentais para a conclusão da minha graduação. Também agradeço às professoras e equipe do Colégio Estadual Henrique Emílio Meyer, pelo incentivo e apoio na busca por um ensino superior de qualidade; e ao Núcleo de Apoio a Eventos e Comunicação da FACED, pelo carinho e acolhimento que recebi nos meus primeiros semestres na universidade.

Agradeço à todas as amizades que se fazem presentes das mais diversas formas e nos mais memoráveis momentos. À Cristina Barros pela companhia e apoio em todos os momentos, incluindo a passagem pela educação pública em seus mais diferentes níveis e, também, por sempre me fazer acreditar que tudo é alcançável, não importa o quão distante pareça; à Maria Otilia, Tita, pelo seu acolhimento, conselhos e amizade em todos os anos que passaram e pelos que virão.

Por fim, agradeço à toda a comunidade de *software* livre e código aberto, bem como aos movimentos que lutam pela disponibilização gratuita de informações, por advogarem pela igualdade de oportunidades; e à todos os grandes músicos e poetas que, através das suas obras, me formaram enquanto pessoa política, em especial ao grupo Racionais MC's e a Leandro Roque de Oliveira, na figura de Emicida.

RESUMO

A criptografia é uma técnica amplamente utilizada na proteção das comunicações entre dois pontos de uma rede. Entretanto, o uso dessa técnica pode afetar a análise de fluxos de rede, uma vez que a criptografia torna inacessível alguns dados dos pacotes de rede. Com o aumento do uso da criptografia como forma de garantir a privacidade e proteção dos dados dos usuários, é possível que, no futuro, seja feita a de criptografia dos protocolos da camada de transporte existentes ou que se crie novos protocolos já criptografados, de forma que as portas de origem e destino, por exemplo, não estejam disponíveis nas capturas de pacotes. Este trabalho propõe e implementa uma análise sobre o impacto dessa possível criptografia em análises de fluxos com aprendizado de máquina. Os resultados mostraram que a criptografia da camada de transporte poderia afetar, ainda que em pequena escala, a análise de fluxos de rede baseada em técnicas de aprendizado de máquina. A diferença de desempenho foi observada ao comparar as métricas de teste de dois algoritmos diferentes, o GBM e o XGBoost. Para avaliar a qualidade dos resultados, foram utilizados os conceitos de explicabilidade e interpretabilidade. Através dessa avaliação, foi possível observar quais atributos são mais utilizados pelos algoritmos no conjunto de dados completo e quais atributos passam a ter mais importância quando os dados da camada de transporte são removidos.

Palavras-chave: Segurança da computação. Aprendizado de máquina. Explicabilidade.

The impact of transport layer encryption on flow analysis using machine learning

ABSTRACT

Cryptography is a widely used technique to protect communications between two points in a network. However, the use of this technique can affect network flow analysis, as encryption makes some data in network packets inaccessible. With the increasing use of encryption as a means to ensure user privacy and data protection, it is possible that in the future, existing transport layer protocols will be encrypted or that new protocols will include encryption by default, so that, for example, source and destination ports are not available in packet captures. This paper proposes and implements an analysis of the impact of this possible encryption on flow analysis with machine learning. The results showed that encryption of the transport layer could affect, albeit on a small scale, network flow analysis based on machine learning techniques. Differences in performance were observed when comparing the test metrics of two different algorithms, GBM and XGBoost. To evaluate the quality of the results, the concepts of explainability and interpretability were used. Through this evaluation, it was possible to observe which features are most used by the algorithms in the complete data set and which attributes become more important when transport layer data is removed.

Keywords: Computer security. Machine learning. Explicability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AES	Advanced Encryption Standard
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DoS	Denial of Service
DDoS	Distributed Denial of Service
DNN	Deep Neural Network
DT	Decision Tree
DPI	Deep Packet Inspection
FNR	False Negative Rate
FPR	False Positive Rate
FTP	File Transfer Protocol
GBM	Gradient Boosting Machine
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
LDA	Linear Discriminant Analysis
LightGBM	Light Gradient Boosting Machine
LIME	Local Interpretable Model-Agnostic Explanations
MCC	Matthews Correlation Coefficient
ML	Machine Learning
MLP	Multi-Layer Perceptron
NIDS	Network Intrusion Detection System
NN	Neural Network

OSI	Open Systems Interconnection
OVO	One-vs-One
OVR	One-vs-Rest
QDA	Quadratic Discriminant Analysis
RF	Random Forest
PCAP	Packet Capture
SCTP	Stream Control Transmission Protocol
SHAP	SHapley Additive exPlanations
SQL	Structured Query Language
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TNR	True Negative Rate
TPR	True Positive Rate
UDP	User Datagram Protocol
XGBoost	Extreme Gradient Boosting
XSS	Cross-Site Scripting

SUMÁRIO

1 INTRODUÇÃO

A análise de tráfego de rede é o processo de capturar e examinar o tráfego que circula em uma rede de computadores. A análise de tráfego pode ser realizada de diversas formas. As formas mais comuns são a inspeção profunda de pacotes (*Deep Packet Inspection*), que analisa o conteúdo detalhado dos pacotes capturados, e a análise de tráfego de rede baseada em fluxos, que analisa o tráfego de rede a partir de fluxos de tráfego extraídos da captura de pacotes. A DPI é mais eficiente que a análise de tráfego de rede baseada em fluxos, porém, pode ser computacionalmente mais custosa. A análise baseada em fluxos, por sua vez, tende a ser mais eficiente em termos de custo computacional, porém, pode ser menos eficiente em termos de detecção de anomalias e ataques cibernéticos (BOUKHTOUTA et al., 2016).

Segundo Alias, Manickam and Kadhum (2013), a captura de pacotes (pcap), tradicionalmente utilizada no contexto da biblioteca *libpcap*, é uma das principais formas de análise passiva de redes, sendo extensivamente utilizada por administradores de rede e por sistemas de detecção de intrusão (*Intrusion Detection System*). Essas capturas podem ser analisadas em tempo real ou salvas em arquivos para serem analisadas posteriormente ou em outro dispositivo. Além disso, a partir da captura de pacotes, é possível extrair informações sobre o tráfego de rede, como o número de *bytes* transferidos, o número de pacotes enviados, o tempo de resposta, entre outros. Essa extração de informações é feita através dos cabeçalhos dos pacotes e do agrupamento dos mesmos em fluxos de tráfego, que são agrupados de acordo com alguma característica, como o endereço IP de origem e destino. So-In (2009) apresenta uma revisão sobre as principais ferramentas de agrupamento de fluxos de tráfego.

A detecção de anomalias é uma técnica de análise de dados que busca identificar padrões incomuns ou desvios em relação a um comportamento esperado. Em redes de computadores, a detecção de anomalias pode ser utilizada para identificar atividades suspeitas que possam indicar a presença de um ataque cibernético em andamento. Existem diversas técnicas de detecção de anomalias, como *clustering*, redes neurais e árvores de decisão (USAMA et al., 2019).

A criptografia de dados é uma técnica essencial para garantir a privacidade e segurança das informações que trafegam em redes de comunicação. Porém, a implementação desse tipo de técnica pode interferir na análise de tráfego de rede, que é crucial para a detecção de anomalias e ataques cibernéticos. Com o crescente volume de dados gerados

em redes, a análise manual de fluxos de tráfego torna-se impraticável, e é necessário o uso de técnicas de aprendizado de máquina para automatizar esse processo. Recentemente, cada vez mais protocolos da camada de aplicação têm sido criptografados, como a migração do protocolo HTTP para HTTPS, ou a implementação de criptografia no protocolo FTP. Porém, a criptografia de dados em protocolos da camada de transporte, como o protocolo TCP, é ainda pouco explorada, apesar de já possuir propostas de extensão, como visto em Bittau et al. (2010), Bittau et al. (2019b) e Bittau et al. (2019a), que propõem a aplicação de criptografia oportunística. Nesse contexto, é fundamental entender como a criptografia de dados na camada de transporte poderia afetar a análise automatizada de fluxos de tráfego e a detecção de anomalias e ataques cibernéticos.

Apesar da criptografia de dados em algumas camadas da rede, existem diversas metodologias para, mesmo assim, realizar a análise automatizada de fluxos de tráfego e derivar informações relevantes sobre o tráfego analisado, como qual protocolo está sendo utilizado ou qual aplicação é responsável pelo tráfego gerado. Karagiannis, Papagiannaki and Faloutsos (2005), por exemplo, aplicam técnicas de aprendizado de máquina sobre o comportamento entre servidor e cliente e o fluxo de tráfego de rede para realizar a classificação do tipo de tráfego. Em Boutaba et al. (2018), podemos observar diversos tipos de análise de tráfego de rede que já foram realizadas. No entanto, a maioria dessas análises não considera a criptografia da camada de transporte nas comunicações de rede. O *framework* BLINC, por exemplo, afirma que se os cabeçalhos da camada de transporte fossem criptografados, o tráfego de rede não poderia ser analisado (BOUTABA et al., 2018).

Esse estudo busca associar as perspectivas mencionadas para entender como a criptografia da camada de transporte nas comunicações de rede poderia afetar a análise automatizada de fluxos de tráfego e a detecção de anomalias a partir dos mesmos. Para isso, serão comparados os resultados obtidos de um modelo de aprendizado supervisionado treinado com fluxos de tráfego de rede não criptografados e de outro treinado com os mesmos fluxos de tráfego de rede, porém, sem as estatísticas obtidas a partir dos cabeçalhos da camada de transporte. Será utilizado o *framework* SHAP (LUNDBERG; LEE, 2017) para apresentar os resultados obtidos de forma mais explicativa.

O trabalho divide-se em 6 capítulos, incluindo essa introdução, descritos a seguir. No capítulo 2, são revisados alguns conceitos centrais a esse estudo. No capítulo 3 são apresentados alguns trabalhos relacionados e seus resultados são discutidos em relação ao estudo proposto. Já no capítulo 4 é proposta uma solução para realizar a análise, que

tem sua implementação e seus resultados apresentados no capítulo 5. Como conclusão, no capítulo 6 são apresentadas as considerações finais do estudo desenvolvido, além de proposições para possíveis melhorias para essa pesquisa e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, é apresentada a fundamentação teórica do trabalho proposto. A seção 2.1 apresenta em mais detalhes a análise de dados a partir do tráfego de rede, assim como a criptografia de dados na camada de transporte. Na seção 2.2, são apresentados os conceitos de aprendizado de máquina e de detecção de anomalias, com maior foco nos tipos de modelos que foram utilizados no projeto. Na seção 2.3, as práticas de detecção de anomalias e de intrusão com aprendizado de máquina são apresentadas, assim como os atuais desafios da área. Por fim, a seção 2.4 apresenta os conceitos de explicabilidade de aprendizado de máquina e o *framework* SHAP, utilizado para explicar os resultados dos modelos obtidos.

2.1 Análise de dados de tráfego de rede

Com o aumento da utilização de redes de computadores, a necessidade de monitorar e analisar o tráfego de rede tornou-se cada vez mais importante e, ao mesmo tempo, cada vez mais difícil de ser realizada manualmente. A análise de dados de tráfego de rede é a investigação dos padrões de comunicação em uma rede de computadores. As informações coletadas podem ser utilizadas para entender como a rede funciona e para identificar comportamentos anômalos e potenciais ameaças à segurança. A análise de tráfego de rede é composta de três fases principais: captura dos dados, análise e interpretação (JOSHI; HADI, 2015).

A análise de dados de tráfego de rede pode incluir a coleta e análise de dados de fluxo, bem como a análise de pacotes individuais. A análise de fluxo de dados é a análise de dados de tráfego de rede que se concentra em fluxos de dados, que são conjuntos de pacotes que compartilham as mesmas propriedades de rede, como o endereço IP de origem e destino, o protocolo de transporte e o número de porta. Com base em fluxos de tráfego, é possível realizar uma análise automatizada de tráfego de rede através de técnicas de aprendizado de máquina. Sharafaldin, Lashkari and Ghorbani (2018) apresentam um conjunto de dados de tráfego de rede, o fluxo extraído da captura de pacotes e os resultados obtidos por alguns modelos treinados com esse mesmo conjunto de dados.

A importância da análise de tráfego de rede é evidente, pois a detecção de atividades maliciosas é um dos maiores desafios enfrentados pela segurança da informação. A análise de tráfego de rede é uma das principais técnicas utilizadas para detectar atividades

maliciosas, como malware, ataques de negação de serviço (DDoS), phishing e roubo de dados. Joshi and Hadi (2015) fazem uma revisão sobre algumas técnicas já implementadas na análise de tráfego de rede para a detecção de intrusão, classificação de tráfego e detecção de ataques de DDoS.

2.1.1 Criptografia de dados de tráfego de rede

De acordo com (STREBE, 2006), a criptografia é um processo que envolve o uso de chaves para codificar informações de forma que ela se torne ilegível para pessoas não autorizadas. No contexto de redes de computadores, a criptografia é utilizada para proteger os dados transmitidos entre os *hosts*, garantindo a confidencialidade e a integridade dos dados. A criptografia é uma das principais técnicas utilizadas para garantir a segurança da informação, pois dificulta a interceptação e a modificação dos dados transmitidos.

O protocolo de segurança da camada de transporte (*Transport Layer Security*) é um protocolo de criptografia usado para criptografar os dados transmitidos por um aplicativo para outro na internet (RESCORLA, 2018). Esse protocolo é usado para garantir que as informações transmitidas não possam ser interceptadas por terceiros e, assim, aumentar a privacidade e a confidencialidade na comunicação. Além disso, as informações criptografadas não podem ser lidas pelo software que as analisa, o que dificulta significativamente a análise de tráfego de rede (PARASKEVI et al., 2020).

Para contornar esse problema, foram criadas técnicas para inferir os protocolos de aplicação utilizados em uma rede, mesmo que os dados sejam criptografados, como visto em Wright, Monrose and Masson (2006) e em Karagiannis, Papagiannaki and Faloutsos (2005). Essas técnicas utilizam metadados do tráfego de rede, como o tamanho, a quantidade e as portas de origem e destino dos pacotes para inferir o protocolo de aplicação utilizado. No entanto, o uso do TLS para criptografar os dados é implementado apenas para a camada de aplicação, enquanto que as técnicas de inferência de protocolo de aplicação utilizam os dados da camada de transporte e de seus protocolos, como o TCP e o UDP, dado que essas informações não são criptografadas.

2.2 Aprendizado de máquina

De acordo com Mohri, Rostamizadeh and Talwalkar (2018), aprendizado de máquina (*machine learning*) pode ser definido como um conjunto de técnicas computacionais que permitem que os computadores aprendam com os dados fornecidos. Essas técnicas podem ser usadas para identificar relacionamentos entre diferentes características (*features*) de um conjunto de dados, bem como para identificar padrões e tendências nesse mesmo conjunto. Em todos os casos, a qualidade e a quantidade de dados disponíveis são cruciais para o sucesso do aprendizado de máquina.

Técnicas de aprendizado de máquina são amplamente utilizadas em muitas aplicações, incluindo reconhecimento de voz, reconhecimento de imagem, diagnóstico médico, recomendações personalizadas, detecção de fraudes, detecção de anomalias, entre outras. Mohri, Rostamizadeh and Talwalkar (2018) destacam que o aprendizado de máquina é uma área de pesquisa em constante evolução, com novas técnicas sendo desenvolvidas e aplicadas em diferentes áreas do conhecimento.

A implementação de técnicas de aprendizado de máquina tem se tornado cada vez mais comum na análise de tráfego de rede e essas são frequentemente usadas em conjunto com técnicas de criptografia para proteger os dados desse tráfego. Algoritmos de criptografia, como AES (*Advanced Encryption Standard*), são usados para criptografar os dados de tráfego de rede, enquanto algoritmos de aprendizado de máquina são usados para detectar ataques de criptoanálise, como ataques de força bruta. A combinação dessas técnicas pode fornecer uma camada adicional de segurança para os dados de tráfego de rede.

O aprendizado de máquina pode ser feito de diversas formas, sendo as principais delas o aprendizado supervisionado e o aprendizado não supervisionado. Cada uma dessas técnicas tem suas vantagens e desvantagens, dependendo do problema que está sendo resolvido e dos dados que são disponibilizados. Mohri, Rostamizadeh and Talwalkar (2018) destacam que o aprendizado supervisionado é o tipo de aprendizado de máquina mais comum e é usado para resolver problemas de classificação e regressão. Já o aprendizado não supervisionado é usado para resolver problemas de agrupamento e detecção de anomalias.

2.2.1 Aprendizado supervisionado

O aprendizado supervisionado é um tipo de aprendizado de máquina que utiliza dados previamente rotulados para treinar um modelo de aprendizado de máquina. O modelo treinado pode ser usado para prever o resultado de um conjunto de dados não rotulado. Algoritmos de aprendizado supervisionado, como árvores de decisão, redes neurais e SVM (*Support Vector Machines*), podem ser usados para a classificação de tráfego de rede com base em padrões de tráfego.

Nessa subseção, serão apresentados alguns algoritmos de aprendizado supervisionado que serão usados na análise de tráfego de rede.

2.2.1.1 Gradient Boosting Machine

O GBM (*Gradient Boosting Machine*) é um algoritmo de aprendizado supervisionado que pode ser usado para resolver problemas de classificação e regressão. Esse tipo de algoritmo combina a capacidade de aprender a partir de dados complexos com a capacidade de lidar com variáveis categóricas e contínuas, permitindo a criação de modelos altamente precisos e eficazes. Segundo o criador do algoritmo, Friedman (2001), os GBMs são capazes de lidar com dados desbalanceados e de alta dimensionalidade, tornando-os uma escolha popular para uma ampla gama de problemas de aprendizado de máquina. Ele também observa que o algoritmo é altamente escalável e pode ser usado em conjuntos de dados de grande escala.

2.2.1.2 eXtreme Gradient Boosting

O XGBoost (*eXtreme Gradient Boosting*) é um algoritmo de aprendizado supervisionado altamente otimizado e escalável, utilizado para resolver problemas de classificação e regressão. Ele é um algoritmo de *boosting* de árvore de decisão que utiliza uma técnica de otimização objetiva baseada em gradientes para treinar modelos de árvore de decisão sequencialmente, e em seguida, combina os resultados para obter uma previsão final. Além disso, o XGBoost incorpora recursos adicionais como regularização e gerenciamento de pesos de amostragem para evitar *overfitting* e lidar com dados desbalanceados. Ele é capaz de lidar com conjuntos de dados extremamente grandes e de alta dimensionalidade, tornando-se uma opção viável para muitos problemas de aprendizado de máquina (CHEN; GUESTRIN, 2016).

O XGBoost é um dos algoritmos mais populares na atualidade, sendo amplamente utilizado em competições de ciência de dados. Ele também tem sido aplicado com sucesso em uma variedade de aplicações do mundo real, incluindo detecção de fraudes financeiras, previsão de churn em empresas de telecomunicações e reconhecimento de voz, além de ter grande uso na análise de *big data* na medicina (QIAONA et al., 2021).

2.2.2 Aprendizado não supervisionado

O aprendizado não supervisionado é um tipo de aprendizado de máquina que utiliza dados não rotulados para treinar um modelo de aprendizado de máquina. O modelo treinado pode ser usado para identificar padrões e tendências em grandes quantidades de dados, sem a necessidade de análise prévia ou rotulação manual. Algoritmos de aprendizado não supervisionado, como *K-means* e *DBSCAN*, podem ser usados para agrupar tráfego de rede com base em padrões de tráfego. A partir desses agrupamentos, é possível identificar tráfego de rede anômalo, como tráfego de rede que não se encaixa em nenhum dos grupos de tráfego de rede previamente encontrados.

Segundo Usama et al. (2019), com os rápidos avanços no tamanho e na complexidade de dados a serem analisados, o aprendizado não supervisionado será uma tendência no futuro. Spiekermann and Keller (2021), por exemplo, utiliza essa técnica para, através da inspeção profunda de pacotes, detectar anomalias em redes virtualizadas. Aouedi et al. (2021), por sua vez, utiliza essa metodologia para analisar e agrupar o tráfego de rede no contexto de fatiamento de rede (*network slicing*).

2.3 Detecção de anomalias

A detecção de anomalias é uma técnica de análise de segurança utilizada para identificar atividades suspeitas ou anômalas na rede. Isso inclui a detecção de ataques de intrusão, atividades mal-intencionadas, violações de política de segurança ou comportamento inadequado dos usuários. A detecção de anomalias é uma área importante na análise de dados de tráfego de rede, especialmente porque o tráfego normal pode variar significativamente entre redes, horários e aplicativos. Ela é frequentemente utilizada em conjunto com técnicas de criptografia de dados e aprendizado de máquina para identificar e mitigar ameaças de segurança em tempo real (BHUYAN; BHATTACHARYYA;

KALITA, 2013).

Os sistemas que implementam essas técnicas de detecção de anomalias são chamados de sistemas de detecção de intrusão de rede (*Network Intrusion Detection System*). Existem diversos tipos desses sistemas, incluindo os baseados na detecção baseada em regras e os baseados na detecção de anomalias. A detecção baseada em regras envolve a definição de regras pelos especialistas em segurança com base em seu conhecimento da rede e das ameaças atuais, mas tem limitações, pois não consegue identificar novos métodos de ataque, por exemplo. Por fim, a detecção baseada em anomalias cria um modelo do tráfego normal da rede e classifica tráfego significativamente diferente desse modelo como anomalia (THOTTAN; JI, 2003).

Em resumo, a detecção de anomalias é uma parte crítica da análise de dados de tráfego de rede e pode ser feita de diversas formas. Segundo Bhuyan, Bhattacharyya and Kalita (2013), um NIDS é um sistema de software que monitora o tráfego de rede e detecta anomalias no tráfego. Essas anomalias podem ser atividades maliciosas, como ataques de intrusão, ou comportamento inadequado dos usuários, como o envio de e-mails não solicitados. Khraisat et al. (2019) apresentam uma revisão sobre as principais técnicas e desafios na detecção de anomalias em redes de computadores.

2.4 Explicabilidade

Com o avanço nos desenvolvimentos de algoritmos de aprendizado de máquina, a interpretabilidade dos resultados desses algoritmos tem se tornado um problema cada vez mais importante. Isso porque, embora os algoritmos de aprendizado de máquina sejam capazes de produzir resultados precisos, eles podem se tornar difíceis de entender e explicar, dada a complexidade dos dados analisados e da relação interna entre eles. A falta de transparência em modelos de ML é um desafio significativo, pois pode dificultar a identificação de erros e aprimoramento do modelo. Diversas técnicas têm sido desenvolvidas para explicar os resultados de modelos de aprendizado de máquina, como o *Local Interpretable Model-agnostic Explanations* (LIME) (RIBEIRO; SINGH; GUESTRIN, 2016) ou o *SHapley Additive exPlanations* (SHAP) (LINARDATOS; PAPASTEFANOPOULOS; KOTSIANTIS, 2020).

Lundberg and Lee (2017) apresentam uma revisão sobre a interpretabilidade de modelos de aprendizado de máquina a partir do SHAP. De acordo com os autores, a interpretabilidade de modelos de aprendizado de máquina é uma área de pesquisa em de-

senvolvimento, que tem como objetivo explicar os resultados de modelos de aprendizado de máquina.

A metodologia presente no SHAP é baseada na teoria dos jogos, e tenta aumentar a interpretabilidade do modelo de máquina ao calcular os valores de importância de cada *feature* para previsões individuais. Os autores propõem os valores SHAP como uma medida unificada da importância das *features*.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados alguns trabalhos relacionados, destacando suas principais contribuições. Foram selecionados artigos que focam no emprego de técnicas de aprendizado de máquina para detecção de anomalias em fluxos de rede. Os trabalhos foram separados em duas seções, uma para trabalhos que utilizam aprendizado de máquina supervisionado na detecção de anomalias e outra para trabalhos que focam na análise de *features* de fluxos de rede.

3.1 Detecção de anomalias em fluxos de rede

Utilizando diversos *datasets*, incluindo o CIC-IDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018), Gupta, Jindal and Bedi (2022) apresentam algoritmos de *deep learning ensemble* que lidam com o desbalanceamento de classes para o uso em NIDS. Os autores utilizam um sistema multicamadas, com *deep neural networks* (DNN) na primeira, XGBoost na segunda e *random forest* na terceira camada. Além disso, os autores utilizam técnicas de *oversampling*, que consiste em duplicar amostras da classe minoritária, para lidar com o desbalanceamento de classes. No CIC-IDS2017, o CSE-IDS proposto pelos autores apresenta acurácia de 92% e taxa de detecção de ataques (*Attack Detection Rate*) de 98%.

Al-Essa and Appice (2022) utilizam o XGBoost com os *datasets* NSL-KDD (TAVALLAEE et al., 2009) e CIC-IDS2017 para avaliar quais *features* dos conjuntos de dados são mais relevantes para a classificação de tráfego, através das metodologias de *feature selection* e *oversampling*. Os autores comparam diferentes metodologias de *oversampling*, como o *SMOTE* (CHAWLA et al., 2002), e de estratégias de combinação de *features*, como *OneVsOne (OVO)* e *OneVsRest (OVR)*. Os autores relatam que o uso de estratégias de OVR com o XGBoost apresenta melhores resultados, com acurácia acima de 99% e *f1 score* de 87% no CIC-IDS2017.

Utilizando o CIC-IDS2018 (SHARAFALDIN; LASHKARI; GHORBANI, 2018), Leevy et al. (2020) comparam implementações de algoritmos de *boosting*, como XGBoost e LightGBM, assim como técnicas de *ensemble*, na detecção de anomalias em fluxos de rede. Os autores também verificam se a inclusão da *feature destination_port* é significativa para a detecção de anomalias.

Com esses artigos, é possível observar que o uso do XGBoost e de técnicas de

ensemble, como o *stacking*, são comuns na literatura de detecção de anomalias em fluxos de rede. Além disso, é possível observar que o uso de técnicas de *oversampling* é comum para lidar com o desbalanceamento de classes, principalmente no CIC-IDS2017. Também notamos que esse *dataset* é bastante utilizado na literatura, o que pode ser devido ao fato de ser um *dataset* aberto, com dados de tráfego de rede simulando os de uma rede corporativa.

3.2 Análise de features de fluxos de rede

Lu et al. (2012) comparam diferentes *features* de fluxos de rede para a detecção de anomalias, utilizando o algoritmo C4.5 (QUINLAN, 2014) para a geração de árvores de decisão. Os autores analisam como combinações de *features*, como o tamanho dos pacotes combinado com a direção no fluxo, o tamanho do pacote combinado com o intervalo entre pacotes, e o *firgerprint* do protocolo, derivado de outras informações dos pacotes, afetam a detecção de anomalias. As conclusões dos autores são que a informação mútua entre o *fingerprint* do protocolo e as classes da aplicação é a *feature* que mais contribui na acurácia em cada conjunto de dados. No entanto, a estabilidade do *fingerprint* do protocolo é pior quando o ambiente de teste é diferente do ambiente de treinamento. Os autores também analisam que a combinação entre o tamanho do pacote e o intervalo entre pacotes contribui mais quando o pacote está na direção do cliente para o servidor *(PSIZE_IAT_CS)* do que quando o pacote está na direção do servidor para o cliente *(PSIZE_IAT_SC)*. Por fim, os autores concluem que os dois primeiros pacotes em uma janela de observação dão a maior contribuição na discriminação dos protocolos de aplicação.

Em Le et al. (2022), é feita uma análise sobre como as *features* dos fluxos de rede afetam o desempenho de algoritmos de aprendizado de máquina para detecção de anomalias, através da explicabilidade. Os autores utilizam os *datasets* IotIDS20 (ULLAH; MAHMOUD, 2020) e NetFlow IoT V2 (SARHAN; LAYEGHY; PORTMANN, 2022), que contém fluxos de rede de dispositivos IoT. Os autores utilizam a técnica de *ensemble* com algoritmos de DT e RF e, em seguida, utilizam explicações locais e explicações globais do SHAP para analisar a importância das *features* para a classificação.

Analisando os diagramas SHAP gerados pelos autores, o algoritmo de DT considera que a porta de destino, o *timestamp* e o identificador do fluxo (*Flow_ID*) impactam positivamente para classificar um tráfego como anomalia, com até cerca de 60 instâncias. Após isso, essas mesmas *features* passam a impactar negativamente. O inverso acontece

na classificação de tráfego como benigno. Além disso, na classificação de tráfego de ataque DoS, os autores observam que, com muitas instâncias, o algoritmo de RF depende do protocolo de aplicação (*L7_PROTO*) para classificar o tráfego como ataque.

Por fim, Sarhan, Layeghy and Portmann (2022) argumentam que o fato dos conjuntos de dados de NIDS não terem uma lista definida de *features* padronizada torna a comparação dos resultados dos modelos para classificação de tráfego baseada em fluxos de rede uma tarefa praticamente impossível. Para resolver esse problema, os autores propõem e comparam dois conjuntos diferentes de *features*. Os *datasets* utilizados são o UNSW-NB15, o BoT-IoT, o ToN-IoT e o CSE-CIC-IDS2018. Dessa forma, os autores criaram cinco novos *datasets*, com 43 *features* cada.

Com base nesses trabalhos, podemos concluir que a análise de *features* de fluxos de rede é um tópico importante e ainda em desenvolvimento na literatura de detecção de anomalias em fluxos de rede, com esforços para encontrar *features* que sejam mais eficazes para a detecção de anomalias. Também podemos obeservar que o uso da explicabilidade dos modelos de ML para a análise de *features* é uma tendência na literatura desses estudos.

4 ANÁLISE PROPOSTA

Neste capítulo é apresentada a análise proposta para criar os modelos de aprendizado de máquina e avaliar a eficácia desses algoritmos na detecção de anomalias a partir de fluxos de rede, com base na comparação de resultados obtidos pelos modelos de ML com e sem a criptografia dos dados da camada de transporte.

4.1 Conjunto de dados utilizados

A eficiência de sistemas de detecção de intrusão e da segurança de redes dependem da velocidade em que ataques e ameaças são detectados. Por isso, conjuntos de tráfego de rede são extremamente necessários para o melhor desempenho das ferramentas utilizadas (DIJKHUIZEN; HAM, 2018). Porém, apesar dessa demanda, encontrar conjuntos de dados de tráfego de rede publicamente disponíveis ainda é um desafio, dada a dificuldade de anonimizar os dados efetivamente e a possibilidade de que os mesmos contenham informações sensíveis (COULL et al., 2009). No entanto, alguns conjuntos de dados de tráfego de rede já foram publicados, como pode ser visto em Ring et al. (2019).

Dentre os conjuntos de dados disponíveis, o CIC-IDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018), da *University of New Brunswick* (UNB), no Canadá, é o que parece ser mais completo, pois, além da captura de pacotes, possui também o fluxo de tráfego da rede, gerado através da ferramenta CICFlowMeter, que é o que será utilizado nessa análise. Os autores também focaram na geração de tráfico de fundo benigno, para simular o comportamento de usuários em uma rede comum. Os fluxos de rede anômalos foram gerados a partir de um conjunto de ataques de intrusão, como *Brute Force*, *Heartbleed*, *Botnet*, *Denial of Service (DoS)*, *Distributed Denial of Service (DDoS)*, *PortScan* e um conjunto de ataques *web*, como *SQL Injection* e *Cross-site scripting (XSS)*. Rosay et al. (2022) discutem em mais detalhes as *features* e os ataques disponibilizados no *dataset* CIC-IDS2017.

Os fluxos de rede do CIC-IDS2017 possuem informações detalhadas sobre a direção dos pacotes enviados. Portanto, algumas *features* do *dataset* possuem os prefixos *bwd* e *fwd*, que indicam, respectivamente, pacotes que foram enviados de um cliente para um servidor (uma requisição) ou do servidor para o cliente, a resposta a uma requisição. Por exemplo, a *feature fwd_pkt_len_max* indica o tamanho máximo dos pacotes enviados do cliente para o servidor, enquanto a *feature bwd_pkt_len_max* indica o tamanho

máximo dos pacotes enviados do servidor para o cliente. Por exemplo, a *feature* de tamanho máximo dos pacotes da origem (*fwd_pkt_len_max*) indica o tamanho máximo dos pacotes enviados do cliente para o servidor, enquanto a *feature* de tamanho máximo dos pacotes do destino (*bwd_pkt_len_max*) indica o tamanho máximo dos pacotes enviados do servidor para o cliente.

4.1.1 Tratamento dos dados

Apesar de ser um conjunto de dados bastante completo, o CIC-IDS2017 possui alguns problemas, como dados inválidos ou duplicados e incoerência nos rótulos (*labelling*) de alguns fluxos. Para corrigir esses problemas, foi criado o *dataset* LYCOS2017 (ROSAY et al., 2021), que é uma versão corrigida do CIC-IDS2017. Dentre as correções feitas, estão a remoção de *features* duplicadas, a correção nos cálculos de algumas *features*, como a razão entre as taxas de download e upload, a contagem dos tamanhos dos pacotes e o uso das *flags* TCP, e a remoção de dados inválidos, como *'NaN'/'Infinity'*. Além disso, foram identificados falhas na detecção dos protocolos IP utilizados, assim como a definição de término de fluxos TCP através da *flag rst*.

Rosay et al. (2022) descrevem a metodologia para tratar e separar os dados em três conjuntos, um para treinamento, que corresponde a 50% dos fluxos, um para validação e outro para teste, que correspondem a 25% dos fluxos de cada classe. Os dados de treinamento são utilizados para treinar o modelo de aprendizado de máquina, enquanto que os dados de validação são usados para ajustar os hiperparâmetros do modelo. Os dados de teste são utilizados para avaliar a performance do modelo gerado, no final do processo de treinamento. É a partir desses dados que serão calculadas as métricas de desempenho, como a acurácia, a precisão e o *recall*.

Após a correção dos dados, o *dataset* LYCOS2017 possui quatorze classificações possíveis para os fluxos, que são: *benign*, *bot*, *ddos*, *dos_goldeneye*, *dos_hulk*, *dos_slowhttptest*, *dos_slowloris*, *ftp_patator*, *heartbleed*, *portscan*, *ssh_patator*, *webattack_bruteforce*, *webattack_sql_injection* e *webattack_xss*. Cada fluxo de rede possui 84 atributos, sendo que 83 deles são numéricos. O atributo restante é a classificação do fluxo, que é o que será utilizado para treinar o modelo de aprendizado de máquina de forma supervisionada nessa análise.

4.2 Criptografia da camada de transporte

A camada de transporte é responsável por estabelecer a conexão entre os *hosts*, além de prover a entrega confiável dos dados, através de protocolos como o *Transmission Control Protocol* (TCP) e o *User Datagram Protocol* (UDP). Os dados da camada de transporte são frequentemente encontrados nos campos *source port*, *destination port*, *sequence number*, *acknowledgement number*, *flags*, *window size*, *urgent pointer*, *options* e *payload*, que é a parte do pacote relativa à camada de aplicação. Os dados relativos a esses campos serão ofuscados, através da remoção das estatísticas de fluxo geradas a partir deles.

Para simular a criptografia dos dados relativos à camada de transporte, foi feita uma análise das *features* estatísticas que foram obtidas através desses dados. Os dados e estatísticas removidas do *dataset* original são os seguintes:

- *src_port* e *dst_port*: os números de portas de origem e destino dos fluxos de rede. Esses números são utilizados para identificar os processos que estão enviando e recebendo os dados;
- *fwd_tcp_init_win_bytes* e *bwd_tcp_init_win_bytes*: o tamanho da janela TCP inicial do fluxo de rede.
- *flag_rst*, *flag_urg*, *flag_ack*, *flag_psh*, *flag_syn*, *flag_fin*, *flag_cwr* e *flag_ece*: a quantidade de pacotes que possuem cada uma das *flags* de controle de fluxo do TCP, independente da direção do fluxo de comunicação.
- *fwd_flag_urg* e *bwd_flag_urg*: a quantidade de pacotes que possuem a *flag* de urgência do TCP, na direção do fluxo de comunicação.
- *fwd_flag_psh* e *bwd_flag_psh*: a quantidade de pacotes que possuem a *flag* de *push* do TCP, na direção do fluxo de comunicação.

Dessa forma, os conjuntos de dados gerados a partir da remoção das estatísticas de fluxo possuirão 76 atributos, incluindo o rótulo de classificação.

Embora os dados relativos à camada de transporte sejam criptografados, camadas inferiores do modelo OSI também possuem informações sobre o tamanho do pacote, portanto, não é possível remover estatísticas como a quantidade de *bytes* e de pacotes em um fluxo de rede. Vale ressaltar que, até a versão responsável por gerar o CIC-IDS2017, o CICFlowMeter não gerava estatísticas com os dados de camadas superiores à de transporte, como a camada de aplicação, por exemplo.

4.3 Definição do modelo

Por se tratar de um conjunto de dados rotulado, o modelo de aprendizado de máquina utilizado deve ser um modelo de aprendizado supervisionado. Para definir qual o melhor algoritmo e quais os melhores parâmetros para ele, foi utilizado o *framework* H2O (H2O.AI, 2023), que é uma ferramenta de aprendizado de máquina que possibilita a criação de modelos de forma automatizada, a partir de uma interface gráfica que também permite validar e testar modelos com base em um conjunto de dados, sem a necessidade de escrever código.

Para automatizar o processo de escolha de parâmetros e de algoritmo, o H2O possui uma ferramenta chamada *AutoML*, que é capaz de testar diversos algoritmos de aprendizado de máquina com um conjunto de dados fornecido, além de testar e otimizar os parâmetros para cada um dos algoritmos. Após a execução do *AutoML*, o H2O retorna uma tabela com os *scores* de cada algoritmo testado. O modelo a ser utilizado será o que tiver o melhor *score* na métrica *mean_per_class_error* (erro médio por classe), que é a métrica utilizada para comparar o desempenho de modelos de classificação multinomial no H2O (H2O.AI, 2023).

Após obter o melhor algoritmo e parâmetros para os conjuntos de dados de treinamento e validação, foi possível obter os *scores* de cada modelo a partir dos dados de teste, que foram utilizados para comparação com os resultados obtidos pelos autores do *dataset* LYCOS2017 (ROSAY et al., 2021).

4.4 Comparação de resultados

Para avaliar modelos de classificação multinomial, pode-se usar as métricas da matriz de confusão local, que corresponde a uma matriz de confusão por classe, ou da matriz de confusão global, que é a matriz de confusão que representa os resultados somados de todas as classes. Essa última é a utilizada pelos autores do *dataset* LYCOS2017 e, para fins de comparação, também será a utilizada nesse trabalho. Existem diversas métricas estatísticas que podem ser obtidas a partir das matrizes de confusão. Dentre elas, estão a taxa de verdadeiros positivos (TPR), a taxa de falsos positivos (FPR), a taxa de verdadeiros negativos (TNR), a taxa de falsos negativos (FNR), a acurácia (*accuracy*), a precisão (*precision*), a revocação (*recall*), a medida F1 (*f1 score*) e o coeficiente de correlação de Matthews (MCC) (H2O.AI, 2023).

Rosay et al. (2021) testaram o desempenho de diversos modelos de aprendizado de máquina, gerados com o *framework* scikit-learn (PEDREGOSA et al., 2011). A tabela 4.1 mostra os resultados obtidos por eles. Os resultados dos modelos obtidos por Rosay et al. (2021) servirão como base para avaliar se a *performance* dos algoritmos e parâmetros selecionados pelo H2O são, pelo menos, equivalentes aos resultados obtidos pelos autores do *dataset* LYCOS2017.

Tabela 4.1 – Resultados obtidos pelos autores do dataset LYCOS2017

Algorithm	***Accuracy (%)***	***Precision (%)***	***Recall (%)***	***f1 score (%)***	***False Positive Rate (%)***	***MCC***
LDA	96,59	94,00	99,54	96,69	6,35	0.9335
QDA	99,83	99,93	99,73	99,83	0,08	0.9966
SVM	99,50	99,61	99,40	99,51	0,38	0.9901
k-NN	99,95	99,91	99,96	99,93	0,10	0.9986
DT	99,92	99,92	99,92	99,85	0,08	0.9985
RF	99,96	99,95	99,98	99,96	0,06	0.9996
MLP	99,72	99,88	99,55	99,72	0,12	0.9943

Fonte: Rosay et al. (2021)

Através da comparação dos *scores* dos modelos criados nesse trabalho, é possível verificar se a criptografia dos dados da camada de transporte afeta, de forma significativa, o desempenho do modelo na detecção de atividades anômalas de rede.

5 IMPLEMENTAÇÃO E AVALIAÇÃO

Neste capítulo, é apresentada a implementação e avaliação da análise proposta. Na seção 5.1 é descrito o processo de criação dos modelos de aprendizado de máquina, enquanto que a seção 5.2 apresenta os resultados obtidos com os modelos criados. A seção 5.3 faz uma análise comparativa dos resultados e apresenta a explicabilidade dos modelos.

5.1 Criação dos modelos

Os conjuntos de dados do LYCOS2017 já haviam sido separados previamente em conjuntos de treinamento, validação e teste por Rosay et al. (2021), que os separou em proporções de 50%, 25% e 25%, respectivamente. Após o tratamento e criação dos dados criptografados, conforme descrito nas seções 4.1.1 e 4.2, foi utilizado o *framework* H2O para identificar os melhores modelos de aprendizado de máquina para o *dataset* analisado, assim como seus respectivos *scores*. Ordenando o desempenho pela métrica *mean_per_class_error*, o modelo de XGBoost obteve o melhor desempenho, seguido do modelo de GBM, ambos são modelos de *ensemble* de árvores de decisão. Nessa análise, o modelo de XGBoost será considerado o principal modelo a ser utilizado, enquanto que o modelo de GBM será utilizado apenas para efeitos de comparação.

Os parâmetros usados com o *AutoML* do H2O são apresentados abaixo:

- *max_runtime_secs=600*: limita a execução do *AutoML* a 10 minutos.
- *max_models=3*: limita o número de modelos a serem treinados para focar em desenvolver modelos melhores.
- *sort_metric=mean_per_class_error*: ordena os modelos de acordo com a métrica de avaliação.
- *seed=6898*: define uma semente para a geração de números aleatórios para garantir a reprodutibilidade.

Os demais parâmetros foram definidos com os valores padrão do *AutoML*. Maiores detalhes sobre os parâmetros do *AutoML* podem ser encontrados na documentação oficial do H2O (H2O.AI, 2023).

O modelo de XGBoost foi treinado em 74 segundos e possui os seguintes parâmetros finais:

- *early_stopping_rounds=3*: número de rodadas sem melhora para parar o treinamento.
- *max_depth=15*: profundidade máxima da árvore.
- *min_child_weight=10*: peso mínimo da soma dos gradientes para dividir um nó.
- *subsample=0.6*: taxa de amostragem de linhas por árvore (de 0.0 a 1.0).
- *colsample_bylevel=0.8*: taxa de amostragem de colunas por árvore (de 0.0 a 1.0).
- *colsample_bytree=0.8*: taxa de amostragem de colunas por árvore (de 0.0 a 1.0).
- *tree_method='exact'*: método de treinamento da árvore.
- *eta=0.3*: taxa de aprendizado.
- *objective='multi:softprob'*: função de perda.
- *nthread=8*: número de *threads*.
- *num_class=14*: número de classes.
- *reg_lambda=1*: regularização L2.
- *gamma=0*: regularização L1.
- *alpha=0*: regularização L1.
- *booster='gbtree'*: tipo de modelo a ser utilizado.
- *grow_policy='depthwise'*: política de crescimento da árvore.
- *max_delta_step=0*: limite máximo de mudança de peso.
- *seed=6898*: define uma semente para a geração de números aleatórios para garantir a reprodutibilidade.

Os demais parâmetros foram definidos com os valores padrão do XGBoost. Maiores detalhes sobre os parâmetros do XGBoost podem ser encontrados na documentação oficial do XGBoost (CHEN; GUESTRIN, 2016).

Apesar de obter resultados satisfatórios com o H2O, foi identificado posteriormente que o *framework* não possui a ferramenta de explicabilidade SHAP para classificação multinomial. Por esse motivo, os modelos de XGBoost foram traduzidos para o *framework* scikit-learn, que possui maior suporte para a ferramenta SHAP. Após a conversão, o desempenho dos modelos foi comparado e os resultados obtidos foram equivalentes.

5.1.1 Ambiente de desenvolvimento

A linguagem de programação utilizada para a criação dos modelos foi Python 3.10.9, utilizando as bibliotecas *scikit-learn* (v1.2.2) e *H2O* (v3.40.0.2), que são bibliotecas de aprendizado de máquina. Para o gerenciamento dos experimentos, foi utilizado o *Jupyter Notebook* (v6.5.3), uma ferramenta que permite a criação de documentos que contém tanto código como texto explicativo, além de permitir a execução do código em tempo real. A biblioteca *matplotlib* (v3.7.1) foi utilizada na geração de gráficos de explicabilidade.

5.2 Resultados obtidos

Ao aplicar os modelos de aprendizado de máquina nos seus respectivos *datasets* de teste, foi possível obter as matrizes de confusão resumidas, que são apresentadas na tabela 5.1 e na tabela 5.2. As matrizes de confusão completas podem ser encontradas no anexo A.

É importante ressaltar que, em Rosay et al. (2021), uma classe considerada positiva foi definida como uma classe que corresponde a um ataque ou tráfego anômalo na rede, como pode ser visto na seção 4.1, enquanto que uma classe considerada negativa foi definida como uma classe que corresponde ao tráfego benigno na rede. Os mesmos critérios foram utilizados para definir as classes positivas e negativas para os modelos apresentados neste trabalho.

Tabela 5.1 – Matriz de confusão do modelo de XGBoost com os dados completos

		Classe predita		
		Positivo	Negativo	Total
Classe real	Positivo	110034	13	110047
	Negativo	16	110142	110158
	Total	110050	110155	

Fonte: O autor

Tabela 5.2 – Matriz de confusão do modelo de XGBoost com os dados criptografados

		Classe predita		
		Positivo	Negativo	Total
Classe real	Positivo	109896	30	109926
	Negativo	80	110078	110158
	Total	109976	110108	

Fonte: O autor

Com base nas matrizes de confusão, foi possível obter os resultados estatísticos apresentados na tabela 5.3.

Tabela 5.3 – Resultados obtidos pelo autor

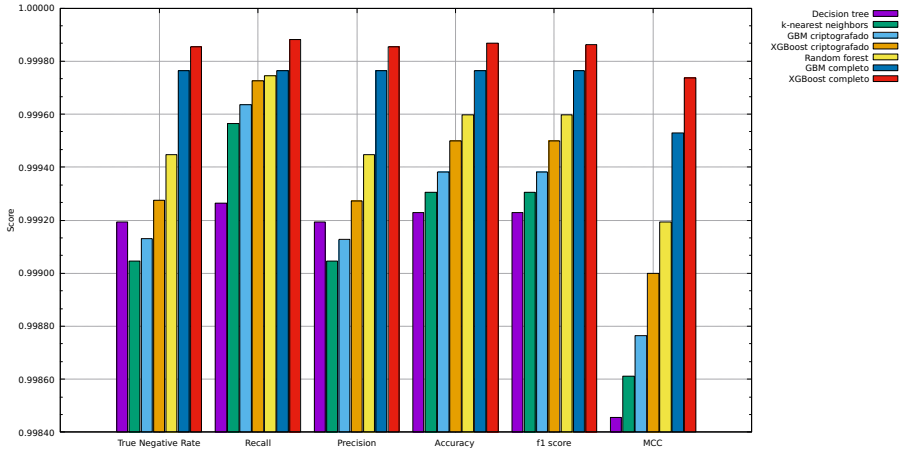
Algoritmo	***Dataset***	***Accuracy (%)***	***Precision (%)***	***Recall (%)***	***f1 score (%)***	***False Positive Rate (%)***	***MCC***
XGBoost	Completo	99,9868	99,9855	99,9882	99,9861	0,00145	0.999737
XGBoost	Criptografado	99,9500	99,9273	99,9727	99,9500	0,00726	0.999000
GBM	Completo	99,9764	99,9764	99,9764	99,9764	0,00236	0.999528
GBM	Criptografado	99,9382	99,9127	99,9636	99,9382	0,00871	0.998764

Fonte: O autor

Com esses resultados, é possível observar que, em ambos os algoritmos, os modelos de GBM e XGBoost obtiveram resultados ligeiramente divergentes entre os *datasets* completos e criptografados. No entanto, os resultados obtidos de todos os modelos foram satisfatórios, com acurácia, precisão, *recall* e *f1 score* superiores a 99,9%, e MCC superior a 99,8%.

Utilizando esses dados e os resultados obtidos pelos autores do LYCOS2017, foi possível obter uma comparação entre os resultados obtidos pelos dois *datasets*, conforme apresentado na figura 5.1.

Figura 5.1 – Comparação entre os resultados obtidos pelos autores do LYCOS2017 e o autor



Fonte: O autor

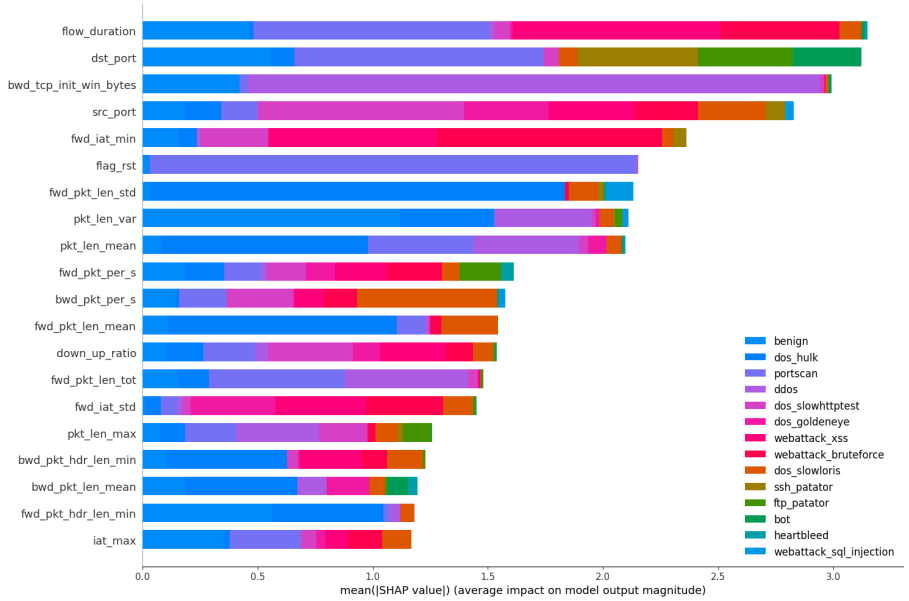
Apesar da alta performance dos modelos de GBM e XGBoost, é importante observar que o *dataset* de base utilizado por Rosay et al. (2021) é, propositalmente, um *dataset* com um enorme desbalanceamento entre as classes, principalmente entre o tráfego benigno e o tráfego anômalo, como forma de simular uma rede de computadores real (RING et al., 2019). Por esse motivo, pequenas variações nas taxas de detecção de tráfego anômalo podem representar um impacto significativo no desempenho real do modelo. Na seção 5.3, será apresentada uma análise mais detalhada dos resultados obtidos

pelos modelos de XGBoost, onde será possível observar a importância de cada *feature* no modelo.

5.3 Análise e explicação dos resultados

Através do SHAP, foi possível obter uma análise mais detalhada dos resultados dos modelos de XGBoost. As figuras 5.2 e 5.3 apresentam os gráficos de sumário SHAP, que distribuem a importância de cada *feature*, por classe, na predição do modelo.

Figura 5.2 – Importância das features por classe - dataset completo

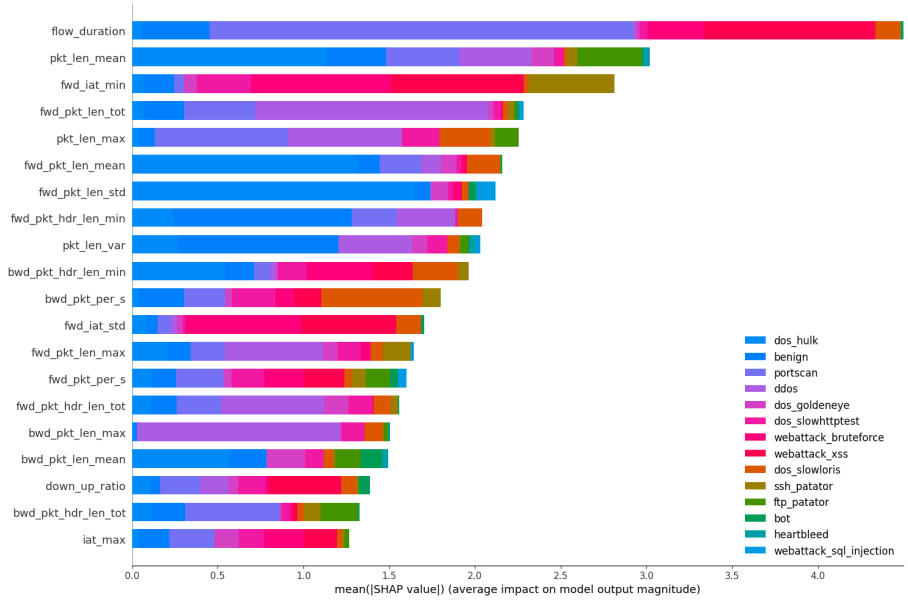


Fonte: O autor

No gráfico de sumário SHAP do modelo treinado com o *dataset* completo, é possível observar que as *features* relacionadas aos dados da camada de transporte, como porta de destino (*dst_port*) e a porta de origem (*src_port*), são bastante presentes entre as *features* mais importantes. Porém, *features* de metadados dos fluxos, como duração do fluxo e o intervalo entre pacotes são importantes para a detecção de alguns ataques, como DDoS e de ataques web.

Já no gráfico de sumário SHAP do modelo treinado com o *dataset* criptografado, podemos notar que a *feature* de duração do fluxo se mantém como a mais importante, enquanto que as *features* relacionadas à variação de tamanho de pacotes e cabeçalhos dos pacotes dominam na detecção de todos os tipos de tráfego. Um detalhe importante é que *features* que tinham pouca importância para o modelo com dados completos, como o tamanho mínimo dos headers dos pacotes da origem do fluxo (*fwd_pkt_hdr_len_min*) e

Figura 5.3 – Importância das features por classe - dataset criptografado



Fonte: O autor

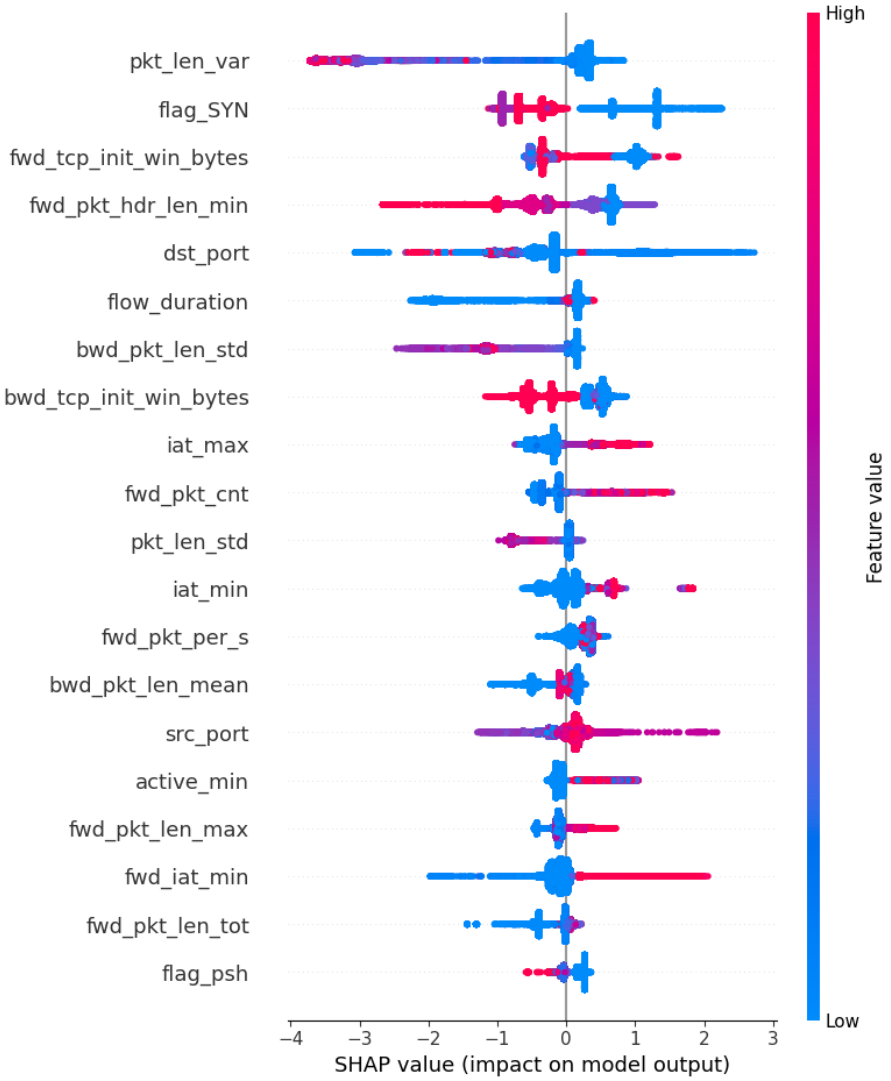
o tamanho máximo dos pacotes no fluxo (*pkt_len_max*), passam a ser importantes para o modelo com dados criptografados. Essa mudança pode indicar uma dependência entre as *features*, que pode ser explorada para melhorar a performance dos modelos.

Nas subseções a seguir, serão apresentadas análises mais detalhadas dos resultados obtidos pelos modelos de XGBoost, com base na importância de cada *feature* no modelo. Para isso, serão apresentados os gráficos de sumário SHAP para cada classe, com as *features* mais importantes para a classificação de cada fluxo como tráfego de determinado ataque ou tráfego benigno. Nesses gráficos, as *features* são ordenadas de acordo com a importância para a classificação de cada fluxo, e as cores indicam o valor de cada *feature* para o fluxo em questão. As cores mais azuis indicam valores mais baixos, enquanto as cores mais avermelhadas indicam valores mais altos. Além disso, a posição de cada ponto no gráfico indica o valor de SHAP da *feature* para aquele fluxo, sendo que valores mais altos indicam que a *feature* teve um impacto positivo na classificação do fluxo como tráfego de determinado ataque ou tráfego benigno, enquanto valores mais baixos indicam que a *feature* teve um impacto negativo na mesma classificação.

5.3.1 Detecção de tráfego benigno

Na detecção de tráfego benigno, as figuras 5.4 e 5.5 apresentam quais *features* impactam mais na classificação de um fluxo como tráfego benigno.

Figura 5.4 – Impacto das features na detecção de tráfego benigno - dataset completo

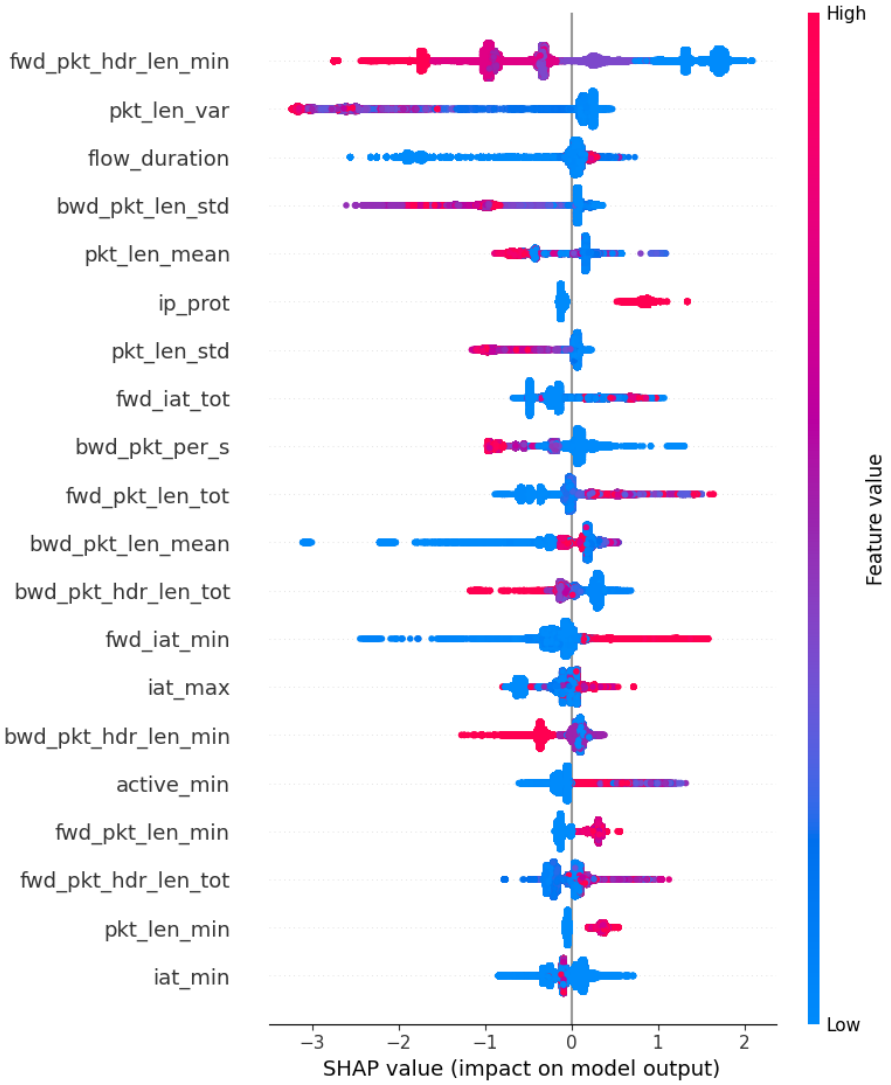


Fonte: O autor

No *dataset* completo, podemos notar como que algumas *features* da camada de transporte impactam na classificação de um tráfego como benigno. Baixas quantidades da *flag* de sincronização (*flag_SYN*) na comunicação fornecem um *score* mais alto para a predição de tráfego benigno, assim como a presença de *src_port* mais altas. É interessante notar que a simples presença de algumas *features*, independente do seu valor, impactam positivamente na classificação de um fluxo como benigno, como a quantidade de *bytes* na janela de iniciação da comunicação TCP da origem do fluxo (*fwd_tcp_init_win_bytes*).

No *dataset* criptografado, os valores das *features* passam a ser mais importantes para a classificação de um fluxo como benigno. Por exemplo, fluxos com *fwd_pkt_hdr_len_min* mais altas e mais baixas impactam de forma positiva e negativa, respectivamente, na classificação de um fluxo como benigno. Além disso, valores altos das *features* do tipo de protocolo IP (*ip_prot*), o intervalo mínimo entre a chegada de pacotes no destino

Figura 5.5 – Impacto das features na detecção de tráfego benigno - dataset criptografado



Fonte: O autor

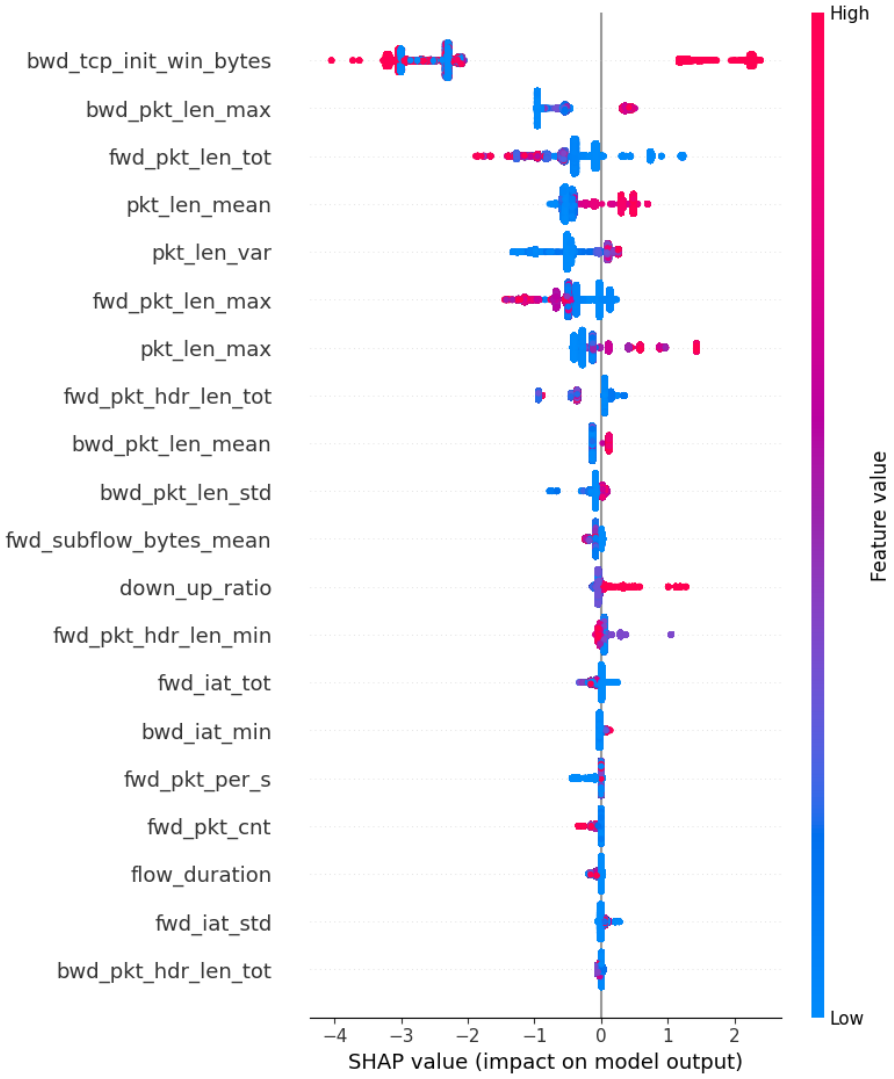
(*fwd_iat_min*) e o *pkt_len_min* impactam positivamente.

5.3.2 Detecção de ataques DDoS

Para a detecção de DDoS, as figuras 5.6 e 5.7 apresentam quais *features* impactam mais na classificação de um fluxo como DDoS. Nesse tipo de ataque, é possível notar que o tamanho dos pacotes é a *feature* mais importante para a classificação de um fluxo como DDoS, tanto no *dataset* completo quanto no *dataset* criptografado.

Com o *dataset* completo, notamos que fluxos com *bwd_pkt_len_max* mais altos impactam positivamente na classificação de um fluxo como DDoS. No entanto, o valores mais altos do tamanho máximo dos pacotes na direção oposta (*fwd_pkt_len_max*) impacta

Figura 5.6 – Impacto das features na detecção de DDoS - dataset completo



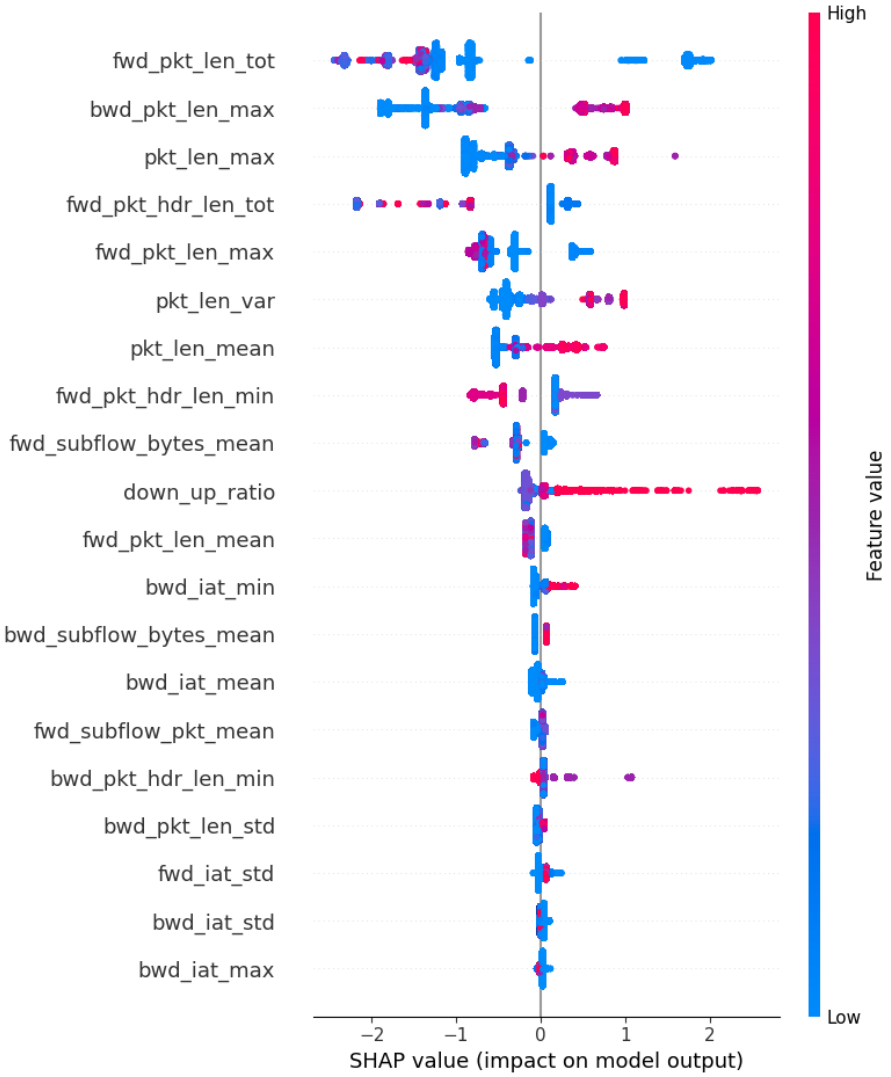
Fonte: O autor

negativamente na classificação de um fluxo como DDoS. Uma *feature* que chama atenção é a *bwd_tcp_init_win_bytes*, que fica nos extremos dos valores SHAP, indicando que valores altos da *feature* impactam positiva e negativamente, apesar de valores baixos da *feature* impactarem somente de forma negativa. Essa variação de impacto reforça a ideia de que há uma relação de dependência entre as *features*, dependendo dos seus valores.

Sem os dados da camada de transporte, as *features* direcionais dominam na classificação de um fluxo como DDoS. Por exemplo, fluxos com *fwd_pkt_len_max* mais altos impactam negativamente na classificação de um fluxo como DDoS, enquanto que fluxos com *bwd_pkt_len_max* mais altos impactam positivamente.

Em ambos os casos, altos valores da *feature* da razão entre download e upload de pacotes (*down_up_ratio*) indicam fortemente que um fluxo é do tipo DDoS.

Figura 5.7 – Impacto das features na detecção de DDoS - dataset criptografado



Fonte: O autor

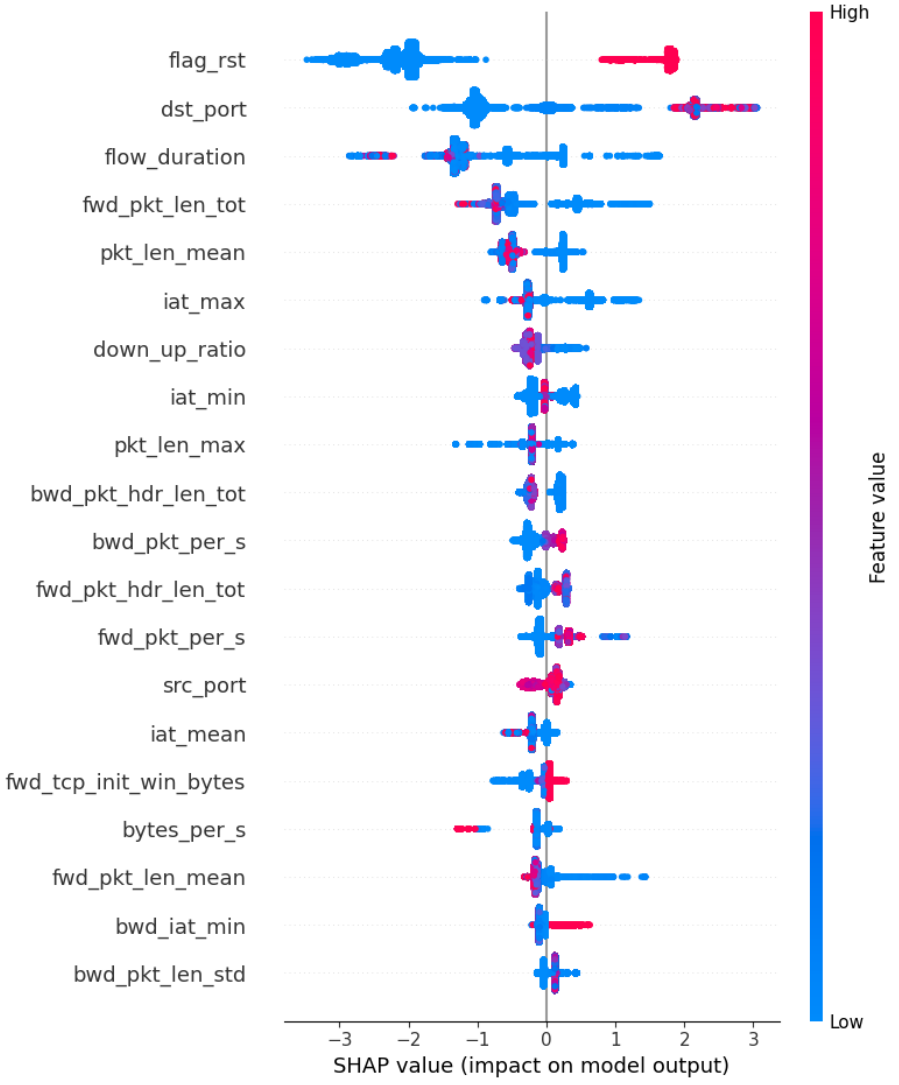
5.3.3 Detecção de ataques portscan

Para a detecção de *portscan*, as figuras 5.8 e 5.9 apresentam quais *features* impactam mais na classificação de um fluxo como *portscan*. Nesse tipo de ataque, a porta de destino, o uso da *flag rst* e a duração do fluxo (*flow_duration*) são as *features* mais importantes para classificar um fluxo como um ataque de *portscan*.

Com base no *dataset* completo, podemos observar que altas taxas de uso da *flag rst* impactam positivamente na classificação de um fluxo como *portscan*, enquanto que a falta deles indica que o fluxo pode não ser desse tipo. Por usar portas de destino raramente utilizadas, os fluxos de *portscan* são facilmente detectados. No entanto, um atacante poderia limitar a faixa de portas escaneadas para evitar ser detectado.

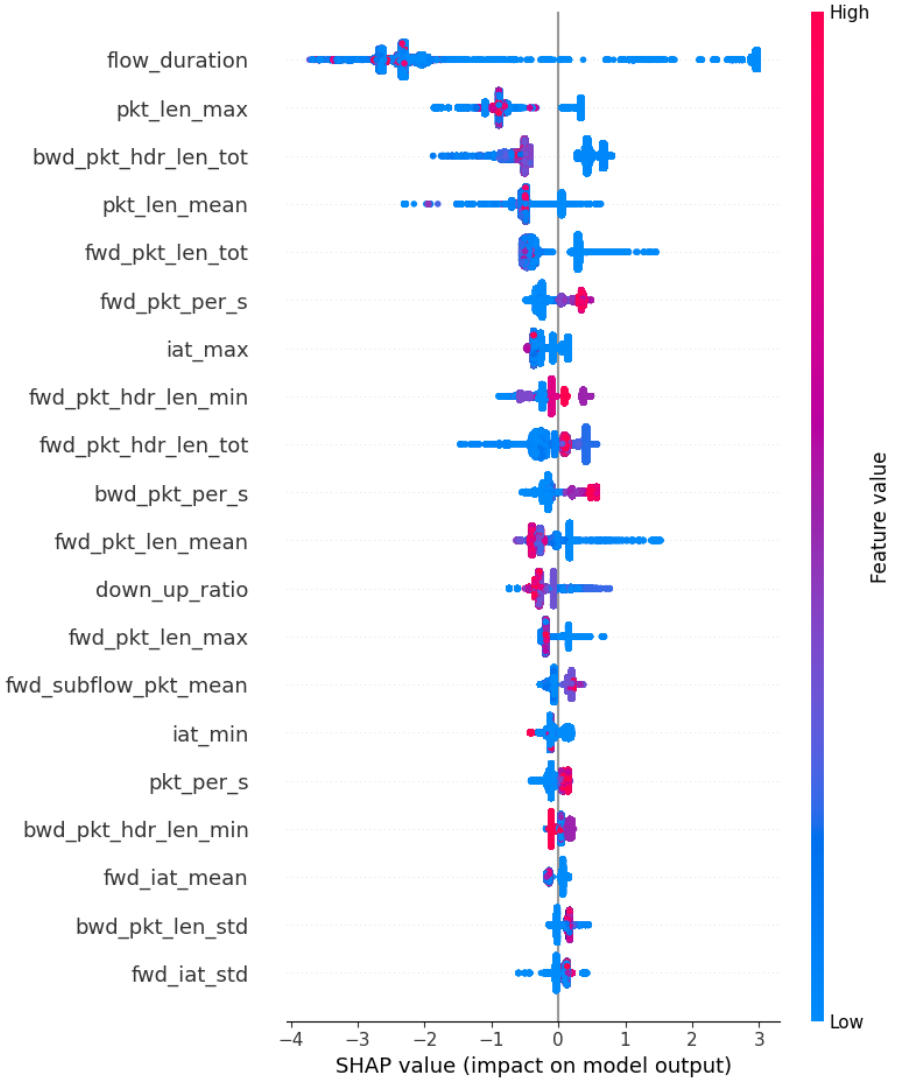
Esse tipo de ataque também é caracterizado por ter um tempo de duração mais

Figura 5.8 – Impacto das features na detecção de portscan - dataset completo



Fonte: O autor

Figura 5.9 – Impacto das features na detecção de portscan - dataset criptografado



Fonte: O autor

curto. Portanto, fluxos com *flow_duration* mais altos impactam negativamente na classificação de um fluxo como *portscan*.

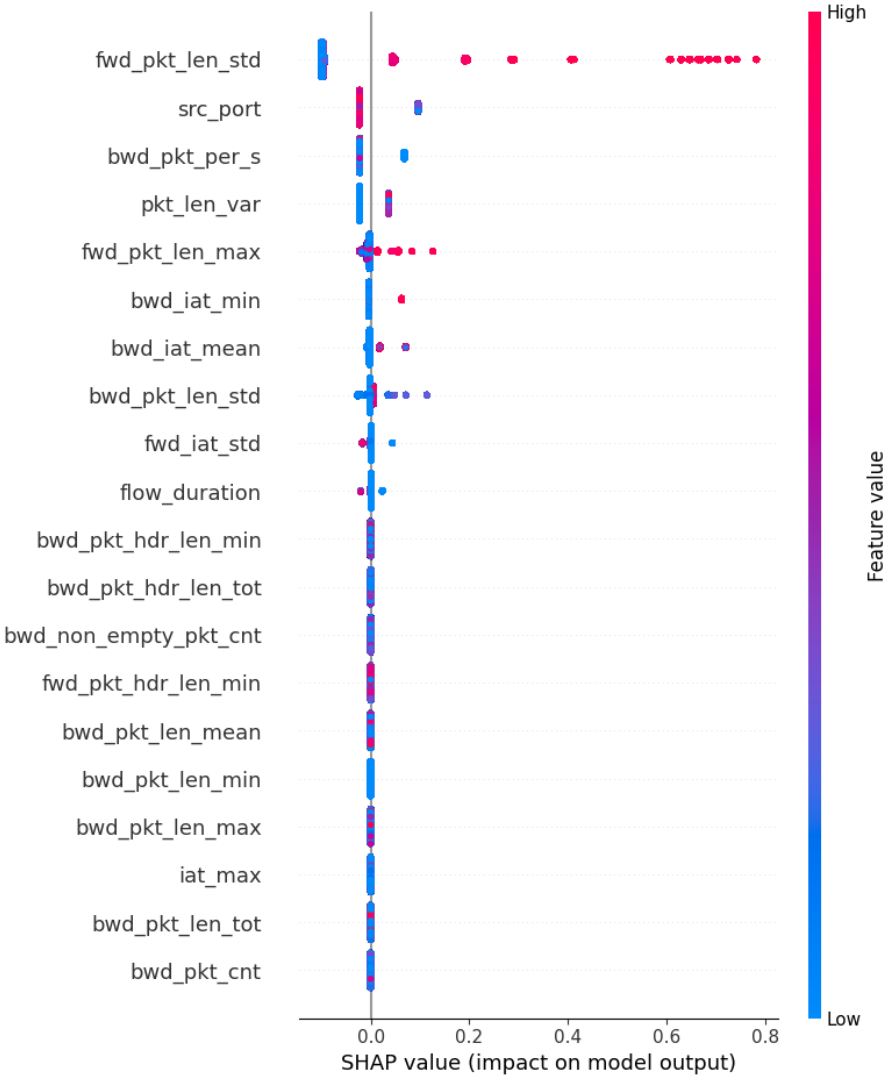
Sem os dados da camada de transporte, a *feature* mais importante para a detecção de *portscan* é a duração do fluxo. Fluxos com *flow_duration* mais altos impactam negativamente na classificação de um fluxo como *portscan*. Dessa forma, podemos ver que o modelo classifica diversos fluxos como *portscan* apenas por terem durações mais baixas.

5.3.4 Detecção de ataques XSS

Para a detecção de XSS, as figuras 5.10 e 5.11 apresentam quais *features* impactam mais na classificação de um fluxo como XSS. Nesse tipo de ataque, a *feature* da variação do tamanho dos pacotes enviados da origem (*fwd_pkt_len_std*) e a *fwd_pkt_len_max* são as informações mais relevantes para classificar um fluxo como um ataque XSS.

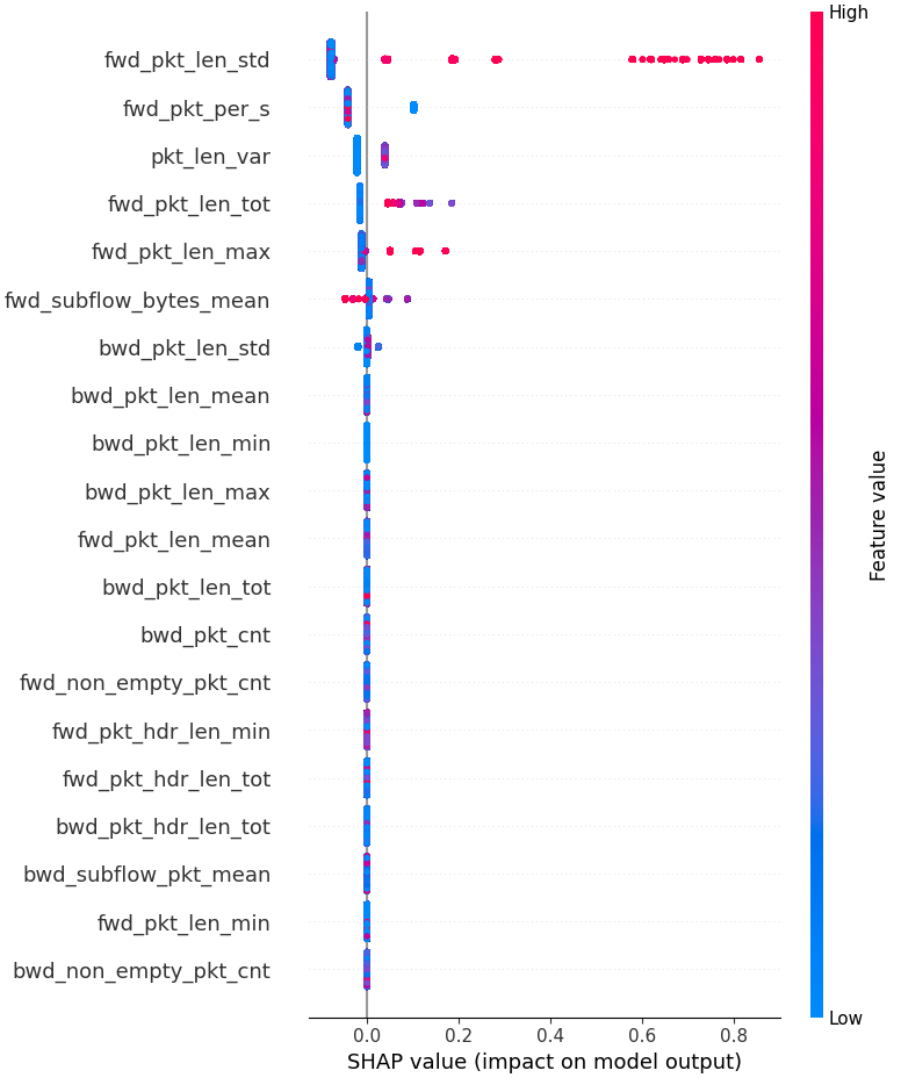
Porém, é importante lembrar que esse tipo de ataque foi o que obteve a pior precisão com ambos os *datasets*. Isso pode ser explicado pelo fato de que o *dataset* LYCOS2017 possui poucas instâncias desse tipo de ataque, o que pode ter prejudicado o modelo.

Figura 5.10 – Impacto das features na detecção de XSS - dataset completo



Fonte: O autor

Figura 5.11 – Impacto das features na detecção de XSS - dataset criptografado



Fonte: O autor

6 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, foi proposta e apresentada uma forma de avaliar a eficácia de modelos de aprendizado de máquina para a detecção de ataques em redes de computadores em situações onde os dados relativos a camada de transporte, como as portas de origem e destino e as *flags* TCP, não estariam disponíveis. Para isso, foram utilizadas técnicas de aprendizado de máquina supervisionado com os algoritmos XGBoost e GBM para a classificação de fluxos de rede com diferentes tipos de ataques. Dentre os modelos avaliados, foi possível perceber que a perda de informações da camada de transporte afetou o desempenho dos algoritmos em todas as métricas avaliadas, ainda que em pequena escala. Os resultados foram comparados com os obtidos por Rosay et al. (2021), que propuseram uma correção para o *dataset* CIC-IDS2017.

Através da comparação das métricas de *accuracy*, *precision*, *recall*, *f1 score* e MCC, foi possível perceber que os algoritmos, quando usando o conjunto de dados original, obtiveram resultados melhores que todos os modelos avaliados por Rosay et al. (2021), porém, quando usando o conjunto de dados criptografado, os resultados foram piores que os obtidos pelo melhor modelo gerado pelos autores. Através da explicabilidade dos modelos, foi possível identificar quais *features* impactam mais na classificação de um fluxo de rede como cada uma das diferentes classes de ataque, o que pode ser útil para a criação de novas técnicas de detecção de ataques e, também, para a criação de técnicas de ataque que evitem a detecção por sistemas de detecção de intrusão baseados em *machine learning*. Maior atenção foi dada às detecções de tráfego benigno e de ataques como DDoS, *portscan* e XSS, como forma de avaliar a eficácia dos modelos em diferentes contextos, como a grande quantidade de tráfego, as formas de ataque e a baixa quantidade de fluxos disponíveis, respectivamente.

Para trabalhos futuros, é possível avaliar a eficácia de outros algoritmos de aprendizado de máquina, assim como diferentes *datasets* para a detecção de ataques em redes de computadores. Além disso, pode-se avaliar a eficácia de técnicas de aprendizado de máquina não supervisionado e técnicas de *feature selection* e *stacking*, como formas de melhorar o desempenho dos modelos. Da mesma forma, seria interessante avaliar como a criptografia dos dados da camada de transporte afetaria outros protocolos, como o SCTP ou o QUIC. Também pode ser interessante avaliar o comportamento dos modelos em diferentes cenários, como a detecção de ataques em redes de computadores com diferentes topologias, redes sem fio e redes de de dispositivos IoT.

REFERÊNCIAS

AL-ESSA, M.; APPICE, A. Dealing with imbalanced data in multi-class network intrusion detection systems using xgboost. In: SPRINGER. **Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2021, Virtual Event, September 13-17, 2021, Proceedings, Part II**. [S.l.], 2022. p. 5–21.

ALIAS, S. B.; MANICKAM, S.; KADHUM, M. M. A study on packet capture mechanisms in real time network traffic. In: **2013 International Conference on Advanced Computer Science Applications and Technologies**. [S.l.: s.n.], 2013. p. 456–460.

AOUEDI, O. et al. Network traffic analysis using machine learning: an unsupervised approach to understand and slice your network. **Annals of Telecommunications**, Springer, p. 1–13, 2021.

BHUYAN, M. H.; BHATTACHARYYA, D. K.; KALITA, J. K. Network anomaly detection: methods, systems and tools. **Ieee communications surveys & tutorials**, IEEE, v. 16, n. 1, p. 303–336, 2013.

BITTAU, A. et al. **Cryptographic Protection of TCP Streams (tcpcrypt)**. RFC Editor, 2019. RFC 8548. (Request for Comments, 8548). Available from Internet: <https://www.rfc-editor.org/info/rfc8548>.

BITTAU, A. et al. **TCP-ENO: Encryption Negotiation Option**. RFC Editor, 2019. RFC 8547. (Request for Comments, 8547). Available from Internet: <https://www.rfc-editor.org/info/rfc8547>.

BITTAU, A. et al. The case for ubiquitous transport-level encryption. In: USENIX ASSOCIATION. [S.l.], 2010.

BOUKHTOUTA, A. et al. Network malware classification comparison using dpi and flow packet headers. **Journal of Computer Virology and Hacking Techniques**, Springer, v. 12, p. 69–100, 2016.

BOUTABA, R. et al. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. **Journal of Internet Services and Applications**, Springer, v. 9, n. 1, p. 1–99, 2018.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. **Journal of artificial intelligence research**, v. 16, p. 321–357, 2002.

CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2016. (KDD '16), p. 785–794. ISBN 978-1-4503-4232-2. Available from Internet: <http://doi.acm.org/10.1145/2939672.2939785>.

COULL, S. E. et al. The challenges of effectively anonymizing network data. In: IEEE. **2009 Cybersecurity Applications & Technology Conference for Homeland Security**. [S.l.], 2009. p. 230–236.

DIJKHUIZEN, N. V.; HAM, J. V. D. A survey of network traffic anonymisation techniques and implementations. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 51, n. 3, p. 1–27, 2018.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. **Annals of statistics**, JSTOR, p. 1189–1232, 2001.

GUPTA, N.; JINDAL, V.; BEDI, P. Cse-ids: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems. **Computers & Security**, Elsevier, v. 112, p. 102499, 2022.

H2O.AI. **H2O**. [S.l.], 2023. 3.40.0.2. Available from Internet: <https://github.com/h2oai/h2o-3>.

JOSHI, M.; HADI, T. H. A review of network traffic analysis and prediction techniques. **arXiv preprint arXiv:1507.05722**, 2015.

KARAGIANNIS, T.; PAPAGIANNAKI, K.; FALOUTSOS, M. Blinc: multilevel traffic classification in the dark. In: **Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications**. [S.l.: s.n.], 2005. p. 229–240.

KHRAISAT, A. et al. Survey of intrusion detection systems: techniques, datasets and challenges. **Cybersecurity**, Springer, v. 2, n. 1, p. 1–22, 2019.

LE, T.-T.-H. et al. Classification and explanation for intrusion detection system based on ensemble trees and shap method. **Sensors**, MDPI, v. 22, n. 3, p. 1154, 2022.

LEEVY, J. L. et al. Detecting cybersecurity attacks using different network features with lightgbm and xgboost learners. In: **2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)**. [S.l.: s.n.], 2020. p. 190–197.

LINARDATOS, P.; PAPASTEFANOPOULOS, V.; KOTSIANTIS, S. Explainable ai: A review of machine learning interpretability methods. **Entropy**, MDPI, v. 23, n. 1, p. 18, 2020.

LU, G. et al. Comparison and analysis of flow features at the packet level for traffic classification. In: **2012 International Conference on Connected Vehicles and Expo (ICCVE)**. [S.l.: s.n.], 2012. p. 262–267.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. In: GUYON, I. et al. (Ed.). **Advances in Neural Information Processing Systems 30**. Curran Associates, Inc., 2017. p. 4765–4774. Available from Internet: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of machine learning**. [S.l.]: MIT press, 2018.

PARASKEVI, D. et al. **Encrypted Traffic Analysis**. [S.l.], 2020. Available online: https://www.enisa.europa.eu/publications/encrypted-traffic-analysis (acesso em 30 de março de 2023). Available from Internet: <https://www.enisa.europa.eu/publications/encrypted-traffic-analysis>.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

QIAONA, Q. et al. Research progress on machine learning xgboost algorithm in medicine. 分子影像学杂志, 分子影像学杂志, v. 44, n. 5, p. 856–862, 2021.

QUINLAN, J. R. **C4. 5: programs for machine learning**. [S.l.]: Elsevier, 2014.

RESCORLA, E. **The Transport Layer Security (TLS) Protocol Version 1.3**. RFC Editor, 2018. RFC 8446. (Request for Comments, 8446). Available from Internet: <https://www.rfc-editor.org/info/rfc8446>.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "why should i trust you?"explaining the predictions of any classifier. In: **Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining**. [S.l.: s.n.], 2016. p. 1135–1144.

RING, M. et al. A survey of network-based intrusion detection data sets. **Computers & Security**, Elsevier, v. 86, p. 147–167, 2019.

ROSAY, A. et al. From cic-ids2017 to lycos-ids2017: A corrected dataset for better performance. In: **IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology**. [S.l.: s.n.], 2021. p. 570–575.

ROSAY, A. et al. Multi-layer perceptron for network intrusion detection: From a study on two recent data sets to deployment on automotive processor. **Annals of Telecommunications**, Springer, v. 77, n. 5-6, p. 371–394, 2022.

SARHAN, M.; LAYEGHY, S.; PORTMANN, M. Towards a standard feature set for network intrusion detection system datasets. **Mobile networks and applications**, Springer, p. 1–14, 2022.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. **ICISSp**, v. 1, p. 108–116, 2018.

SO-IN, C. A survey of network traffic monitoring and analysis tools. **Cse 576m computer system analysis project, Washington University in St. Louis**, Citeseer, Washington University in St. Louis, 2009.

SPIEKERMANN, D.; KELLER, J. Unsupervised packet-based anomaly detection in virtual networks. **Computer Networks**, v. 192, p. 108017, 2021. ISSN 1389-1286. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S1389128621001286>.

STREBE, M. **Network security foundations: technology fundamentals for IT success**. [S.l.]: John Wiley & Sons, 2006.

TAVALLAEE, M. et al. A detailed analysis of the kdd cup 99 data set. In: **2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications**. [S.l.: s.n.], 2009. p. 1–6.

THOTTAN, M.; JI, C. Anomaly detection in ip networks. **IEEE Transactions on Signal Processing**, v. 51, n. 8, p. 2191–2204, 2003.

ULLAH, I.; MAHMOUD, Q. H. A scheme for generating a dataset for anomalous activity detection in iot networks. In: SPRINGER. **Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33**. [S.l.], 2020. p. 508–520.

USAMA, M. et al. Unsupervised machine learning for networking: Techniques, applications and research challenges. **IEEE Access**, v. 7, p. 65579–65615, 2019.

WRIGHT, C. V.; MONROSE, F.; MASSON, G. M. On inferring application protocol behaviors in encrypted network traffic. **Journal of Machine Learning Research**, v. 7, n. 12, 2006.

APÊNDICE A — MATRIZES DE CONFUSÃO DO MODELO XGBOOST

As matrizes de confusão completas, com todas as classes identificadas no *dataset* LYCOS2017, podem ser visualizadas nas tabelas A.1 e A.2. Nas *labels* das linhas, são apresentadas as classes reais e, nas das colunas, as classes preditas pelos modelos.

Tabela A.1 – XGBoost com o dataset completo

Labels	*benign*	*bot*	*ddos*	*dos_goldeneye*	*dos_hulk*	*dos_slowhttptest*	*dos_slowloris*	*ftp_patator*	*heartbleed*	*portscan*	*ssh_patator*	*webattack_bruteforce*	*webattack_sql_injection*	*webattack_xss*	*Error*	*Rate*	*Precision*
benign	110142	0	0	1	0	8	0	0	0	5	2	0	0	0	0.0001	16 / 110,158	1.0
bot	0	183	0	0	0	0	0	0	0	0	0	0	0	0	0	0 / 183	1.0
ddos	0	0	23920	0	0	0	0	0	0	0	0	0	0	0	0	0 / 23,920	1.0
dos_goldeneye	0	0	0	1689	0	2	0	0	0	0	0	0	0	0	0.0012	2 / 1,691	1.0
dos_hulk	3	0	0	2	39736	1	0	0	0	4	0	1	0	0	0.0003	11 / 39,747	1.0
dos_slowhttptest	3	0	0	0	0	1209	3	0	0	0	0	1	0	0	0.0058	7 / 1,216	0.99
dos_slowloris	3	0	0	0	0	1	1412	0	0	1	0	0	0	1	0.0042	6 / 1,418	1.0
ftp_patator	0	0	0	0	0	0	0	1000	0	0	0	0	0	0	0	0 / 1,000	1.0
heartbleed	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0 / 2	1.0
portscan	0	0	0	0	0	1	1	0	0	39729	0	1	0	0	0.0001	3 / 39,732	1.0
ssh_patator	1	0	0	0	0	0	0	0	0	1	737	0	0	0	0.0027	2 / 739	1.0
webattack_bruteforce	1	0	0	0	0	0	0	0	0	0	0	303	0	36	0.1088	37 / 340	0.85
webattack_sql_injection	2	0	0	1	0	0	0	0	0	0	0	0	0	0	1.0	3 / 3	0.0
webattack_xss	0	0	0	0	0	0	0	0	0	0	0	49	0	114	0.3006	49 / 163	0.75
Total	110155	183	23920	1693	39736	1222	1416	1000	2	39740	739	355	0	151	0.0006	136 / 220,312	
Recall	1.0	1.0	1.0	1.0	1.0	0.99	1.0	1.0	1.0	1.0	1.0	0.89	0.0	0.70			

Fonte: O autor

Tabela A.2 – XGBoost com o dataset criptografado

Labels	*benign*	*bot*	*ddos*	*dos_goldeneye*	*dos_hulk*	*dos_slowhttptest*	*dos_slowloris*	*ftp_patator*	*heartbleed*	*portscan*	*ssh_patator*	*webattack_bruteforce*	*webattack_sql_injection*	*webattack_xss*	*Error*	*Rate*	*Precision*
benign	110078	0	0	51	0	21	3	0	0	4	1	0	0	0	0.0007	80 / 110,158	1.0
bot	0	183	0	0	0	0	0	0	0	0	0	0	0	0	0	0 / 183	1.0
ddos	0	0	23920	0	0	0	0	0	0	0	0	0	0	0	0	0 / 23,920	1.0
dos_goldeneye	13	0	0	1675	0	2	1	0	0	0	0	0	0	0	0.0095	16 / 1,691	0.97
dos_hulk	4	0	0	0	39732	1	6	0	0	4	0	0	0	0	0.0004	15 / 39,747	1.0
dos_slowhttptest	2	0	0	1	1	1201	10	0	0	0	0	1	0	0	0.0123	15 / 1,216	0.98
dos_slowloris	3	0	0	0	0	4	1403	0	0	7	0	1	0	0	0.0106	15 / 1,418	0.99
ftp_patator	1	0	0	0	0	0	0	999	0	0	0	0	0	0	0.0010	1 / 1,000	1.0
heartbleed	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1.0	2 / 2	0.0
portscan	1	0	0	0	0	0	0	0	0	39730	0	0	0	1	0.0001	2 / 39,732	1.0
ssh_patator	3	0	0	0	0	0	0	0	0	0	736	0	0	0	0.0041	3 / 739	1.0
webattack_bruteforce	1	0	0	0	0	0	0	0	0	0	0	275	0	64	0.1912	65 / 340	0.69
webattack_sql_injection	2	0	0	1	0	0	0	0	0	0	0	0	0	0	1.0	3 / 3	0.0
webattack_xss	0	0	0	0	0	0	0	0	0	0	0	121	0	42	0.7423	121 / 163	0.39
Total	110108	183	23920	1728	39733	1229	1423	999	0	39745	737	398	0	109	0.0015	338 / 220,312	
Recall	1.0	1.0	1.0	0.99	1.0	0.99	0.99	1.0	0.0	1.0	1.0	0.81	0.0	0.26			

Fonte: O autor