

**MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO**

**BRUNO CAMACHO BURIN
JOÃO PEDRO FRANCISCO CARUSO PEDROSO**

**PROJETO DE APLICAÇÃO DE APRENDIZADO DE MÁQUINA PARA
CRIAÇÃO AUTOMÁTICA DE REGRAS DE FIREWALL**

**RIO DE JANEIRO
2024**

BRUNO CAMACHO BURIN
JOÃO PEDRO FRANCISCO CARUSO PEDROSO

PROJETO DE APLICAÇÃO DE APRENDIZADO DE MÁQUINA PARA
CRIAÇÃO AUTOMÁTICA DE REGRAS DE FIREWALL

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador(es): Leandro de Mattos Ferreira, Maj
João Gomes, 1 Ten

Rio de Janeiro
2024

©2024

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

Camacho Burin, Bruno; Francisco Caruso Pedroso, João Pedro.

Projeto de Aplicação de Aprendizado de Máquina para Criação Automática de Regras de Firewall / Bruno Camacho Burin e João Pedro Francisco Caruso Pedroso. – Rio de Janeiro, 2024.

61 f.

Orientador(es): Leandro de Mattos Ferreira e João Gomes.

Projeto de Final de Curso (graduação) – Instituto Militar de Engenharia, Engenharia da Computação, 2024.

1. aprendizado de máquina. 2. firewall. 3. floresta aleatória. 4. gbdt. 5. svm. 6. regressão logística. 7. lightgbm. 8. mlp. 9. SHAP. i. de Mattos Ferreira, Leandro (orient.) ii. Gomes, João (orient.) iii. Título

**BRUNO CAMACHO BURIN
JOÃO PEDRO FRANCISCO CARUSO PEDROSO**

**Projeto de Aplicação de Aprendizado de Máquina para
Criação Automática de Regras de Firewall**

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador(es): Leandro de Mattos Ferreira e João Gomes.

Aprovada em 10 de outubro de 2024, pela seguinte banca examinadora:

Leandro de Mattos Ferreira

Prof Leandro de Mattos Ferreira - M.Sc. do IME

LH. cAp

Prof Luiz Henrique da Costa Araújo - D.Sc. do IME

Oscar Martins Wanderley Filho

Prof Oscar Martins - M.Sc. do IME

João Gomes e Meliws Nito

Ten João Gomes - Engenheiro da Comunicações do IME - 51 CT

Rio de Janeiro
2024

RESUMO

O presente projeto de fim de curso consiste no desenvolvimento de um IPS que tem por objetivo a proteção de uma rede interna por meio de um firewall de borda. Em sistemas tradicionais, as regras são criadas manualmente, o que dificulta a sua atualização, tornando a rede interna mais vulnerável a ataques previamente não mapeados. Considerando a crescente complexidade e frequência dos ataques cibernéticos, que possuem a capacidade de comprometer dados sensíveis e a segurança nacional, a abordagem tradicional de atualização manual das regras de firewall é ineficiente e suscetível a erros humanos. Nesse contexto, o objetivo deste trabalho consiste em desenvolver um algoritmo de aprendizado de máquina (ML) que seja responsável por criar automaticamente novas regras, tornando o sistema mais robusto contra novos tipos de ataques, de modo a aumentar a segurança da rede do Centro Integrado de Telemática do Exército (CITEEx) e do Exército Brasileiro (EB). Além disso, outras funcionalidades foram implementadas, como o desenvolvimento de um pipeline contínuo de retreinamento para garantir que as regras permaneçam atualizadas, bem como integração com o firewall e diversas outras tarefas administrativas esperadas de um IPS, como autenticação e envio de notificações, a fim de reduzir as complexidades operacionais inerentes ao processo de gestão de regras de firewall. Com relação ao algoritmo implementado, foi utilizada uma técnica de ensemble, na qual a classificação de registros é feita por meio da votação de 5 modelos (Random Forest, LightGBM, MLP, Gradient Boosting Decision Tree e Regressão Logística). Após a realização desse trabalho, o grupo conseguiu desenvolver um algoritmo que consegue identificar com f1-score de mais de 90% ataques do tipo bruteforce, botnet, DOS e DDOS, ainda que ainda possam haver melhoria no quesito da identificação de ataques infiltration e web attack.

Palavras-chave: aprendizado de máquina; firewall; floresta aleatória; gbdt; svm; regressão logística; lightgbm; mlp; SHAP.

ABSTRACT

This end-of-course project aims to develop an IPS that can protect internal networks by the use of edge firewalls. In traditional settings, firewall rules are created manually, what can easily become difficult to maintain and cause the internal network to be more vulnerable to previously unmapped attacks. Given the increasing complexity and frequency of cyberattacks that can compromise sensitive data and national security, the traditional approach of manually updating firewall rules is inefficient and quite susceptible to human error. Hence, the aim of this project is to develop a machine learning (ML) algorithm that is responsible for automatically creating new firewall rules, making the system more robust against new types of attacks as to increase the security of the network of the Army's Integrated Telematics Center (CITEEx) and the Brazilian Army (EB). Furthermore, other functionalities were implemented, such as a continuous retraining pipeline to ensure that the rules remain up-to-date and integration capabilities with external dashboards and firewalls. Also, the implemented system allows administrative control by authentication and notification sending, in order to reduce the operational complexities inherent in the process of managing firewall rules and internal networks overall. With regard to the algorithm implemented, the ensemble technique was used and the logs are classified by the voting of 5 models (Random Forest, LightGBM, MLP, Gradient Boosting Decision Tree and Logistic Regression). After carrying out the work, the group was able to develop an algorithm able to identify bruteforce, botnet, DOS and DDOS attacks with an F1-score of over 90%, even though it can still be improved regarding the identification of infiltration and web attacks.

Keywords: machine learning; firewall; random forest; gbdt; svm; logistic regression; lightgbm; mlp.

LISTA DE ILUSTRAÇÕES

Figura 1 – Fluxo descrito no experimento de Cláudio Marques , Silvestre Malta e João Magalhães.	15
Figura 2 – Pipeline descrito no experimento Roland Verbruggen.	16
Figura 3 – Representação da validação cruzada.	20
Figura 4 – Exemplo de árvore de decisão	22
Figura 5 – Representação de um nó perceptron.	24
Figura 6 – Representação de uma rede perceptron.	25
Figura 7 – Representação do mapeamento gerado pela função SHAP	25
Figura 8 – Representação de um gráfico shap para a classe salários maiores que R\$50000,00	26
Figura 9 – Representação do pipeline de tratamento de dados	29
Figura 10 – Comparação dos relatórios SHAP para diferentes classes no LightGBM	31
Figura 11 – Comparação dos relatórios SHAP para diferentes classes no MLP para a base 1	32
Figura 12 – Comparação dos relatórios SHAP para diferentes classes no <i>lightgbm</i> para a base 2	35
Figura 13 – Comparação dos relatórios SHAP para diferentes classes na regressão logística para a base 1	48
Figura 14 – Comparação dos relatórios SHAP para diferentes classes no <i>random forest</i> para a base 1	48
Figura 15 – Comparação dos relatórios SHAP para diferentes classes no <i>Gradient boosting decision tree</i>	49
Figura 16 – Comparação dos relatórios SHAP para diferentes classes no <i>lightgbm</i> para a base 2	52
Figura 17 – Comparação dos relatórios SHAP para diferentes classes no <i>gradient boost decision tree</i> para a base 2	53
Figura 18 – Comparação dos relatórios SHAP para diferentes classes na regressão logística para a base 2	54
Figura 19 – Comparação dos relatórios SHAP para diferentes classes no <i>multi layer perceptron</i> para a base 2	55
Figura 20 – Comparação dos relatórios SHAP para diferentes classes no <i>random forest</i> para a base 2	56
Figura 21 – Tela de Login	57
Figura 22 – Dashboard do cliente	57
Figura 23 – Gerenciamento de Regras Críticas	58
Figura 24 – Tela de regras de firewall	58

Figura 25 – Tela de usuários	59
Figura 26 – Swagger: usuários e regras críticas	60
Figura 27 – Swagger: outros endpoints	61
Figura 28 – Swagger: modelos	61

LISTA DE TABELAS

Tabela 1 – Descrição dos tipos de ataques e tráfego de rede	27
Tabela 2 – Descrição das variáveis presentes no <i>dataset</i> público 1	28
Tabela 3 – Descrição das variáveis mais relevantes presentes no <i>dataset</i> público 2 .	28
Tabela 4 – Modelos presentes no ensemble	29
Tabela 5 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o lightgbm para a base 1	30
Tabela 6 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o multi layer perceptron para a base 1	31
Tabela 7 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para o lightgbm na base 2	33
Tabela 8 – Ataque e seus principais indicadores	37
Tabela 9 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o gradient boost para a base 1	47
Tabela 10 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o logistic regression para a base 1	47
Tabela 11 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o random forest para a base 1	47
Tabela 12 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para o gradient boost decision tree	50
Tabela 13 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para regressão logistica na base 2	50
Tabela 14 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para Multi layer perceptron na base 2	51
Tabela 15 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para o random forest na base 2	51

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de programação de aplicação
CITEx	Centro Integrado de Telemática do Exército
DMZ	Zona desmilitarizada
EB	Exército Brasileiro
GBDT	Algoritmo de ML <i>Gradient Boosting Decision Tree</i>
IA	Inteligência artificial
IDS	Sistema de detecção de intrusão
IPS	Sistema de prevenção de intrusão
ML	Aprendizado de Máquina
MLP	Algoritmo de ML <i>Multi Layer Perceptron</i>
PFC	Projeto de Fim de Curso
SHAP	Relatório <i>Shapley Additive Explanations</i>
SVM	Algoritmo de ML <i>Support Vector Machine</i>

SUMÁRIO

1	INTRODUÇÃO	12
1.1	MOTIVAÇÃO	12
1.2	JUSTIFICATIVA	13
1.3	OBJETIVOS	13
1.4	ORGANIZAÇÃO	14
2	REVISÃO BIBLIOGRÁFICA	15
2.1	AVALIAÇÃO DOS TRABALHOS	16
3	FUNDAMENTAÇÃO TEÓRICA	17
3.1	FIREWALL	17
3.2	DMZ	17
3.2.1	MÉTODOS DE ATAQUE	17
3.3	IDS E IPS	18
3.4	MÉTRICAS DE AVALIAÇÃO	19
3.5	VALIDAÇÃO CRUZADA	20
3.6	ALGORITMOS DE APRENDIZADO DE MÁQUINA	21
3.6.1	REGRESSÃO LOGÍSTICA	21
3.6.2	ALGORITMOS EM ÁRVORES	22
3.6.2.1	ÁRVORE DE DECISÃO	22
3.6.2.2	RANDOM FOREST	22
3.6.2.3	GRADIENT BOOSTING DECISION TREE	23
3.6.2.4	LIGHTGBM	23
3.6.3	MLP	23
3.6.3.1	PERCEPTRON	23
3.6.3.2	MLP	24
3.7	RELATÓRIOS SHAP CLASSE	25
3.7.1	CONCEITO	25
3.8	EXEMPLO ILUSTRATIVO	25
4	PROJETO	27
4.1	METODOLOGIA	27
4.1.1	BASES DE DADOS UTILIZADAS	27
4.1.2	MODELOS UTILIZADOS	29
4.2	RESULTADOS	30
4.2.1	DATASET 1	30

4.2.2	DATASET 2	32
4.2.3	FUNCIONALIDADES IMPLEMENTADAS DO SISTEMA	34
5	CONCLUSÃO	37
5.1	DATASET 1	37
5.2	DATASET 2	37
5.2.1	CONCLUSÕES FINAIS	38
	REFERÊNCIAS	40
	APÊNDICE A – CASOS DE USO	42
A.1	LOGIN	42
A.2	GERENCIAMENTO DE USUÁRIOS	42
A.3	GERENCIAMENTO DO MODELO DE IA	43
A.4	GERENCIAMENTO DE REGRAS CRÍTICAS	44
A.5	ALTERAÇÃO DE CONFIGURAÇÕES DO SISTEMA	45
	APÊNDICE B – RESULTADOS PARA O DATASET 1	47
	APÊNDICE C – RESULTADOS PARA O DATASET 2	50
	APÊNDICE D – TELAS DO CLIENTE DE NAVEGADOR	57
	APÊNDICE E – API: DOCUMENTAÇÃO SWAGGER	60

1 INTRODUÇÃO

A segurança da informação busca garantir a integridade, autenticidade, irretratabilidade, confidencialidade e disponibilidade da informação. A violação de qualquer um desses princípios pode gerar grandes custos financeiros a uma empresa ou organização. Na última década, ataques cibernéticos geraram prejuízos anuais que figuram entre 300 bilhões e 1 trilhão de dólares (1). Nesse contexto, os *firewalls* surgem como componentes cruciais para garantir a proteção da rede interna, uma vez que são eles os responsáveis por filtrar quais pacotes podem ou não sair da ou ingressar na rede.

Na maioria dos casos, os atacantes possuem conhecimento das regras que estão sendo utilizadas por uma organização, de modo que se faz necessário atualizar as regras de *firewall* com regularidade para evitar que a rede se torne vulnerável (2). Entretanto, as atualizações costumam ser feitas manualmente por especialistas da área de segurança da informação, que buscam padrões nos ataques para poder gerar novas regras. Esse processo dificulta a atualização constante das regras, deixando a rede vulnerável a atacantes externos.

Uma alternativa é utilizar técnicas de aprendizado de máquina, que têm como objetivo eliminar a necessidade de conhecimento prévio sobre o assunto. Essas técnicas, através de uma abordagem interativa, permitem identificar padrões complexos em um banco de dados de treino, os quais não são facilmente detectáveis por humanos (3). Nesse contexto, a aplicação de técnicas de ML na criação de regras de *firewall* tornou-se o estado da arte.

1.1 Motivação

A segurança da informação é uma preocupação constante para qualquer organização que lide com dados sensíveis. No caso do Centro Integrado de Telemática do Exército (CITEEx) e do Exército Brasileiro (EB), essa preocupação é ainda mais acentuada devido à natureza crítica e sensível das informações que são manipuladas. As redes militares são alvos frequentes de ataques cibernéticos sofisticados que podem comprometer a segurança nacional. Portanto, é crucial para o CITEEx e o EB adotar medidas de segurança robustas e atualizadas para proteger suas infraestruturas de rede.

Os *firewalls* são a primeira linha de defesa contra esses ataques, mas a sua eficácia depende da atualização constante das regras de filtragem. Tradicionalmente, essa tarefa é executada manualmente, o que pode ser demorado e suscetível a erros humanos. Com o crescente volume e complexidade dos ataques cibernéticos, torna-se difícil depender exclusivamente da intervenção manual para manter a segurança da rede em níveis aceitáveis.

Nesse contexto, a aplicação de técnicas de aprendizado de máquina surge como uma solução promissora. A capacidade dos algoritmos de ML de identificar padrões complexos em grandes volumes de dados oferece uma maneira eficiente de atualizar as regras de *firewall*. Isso não só pode melhorar a resposta a ataques cibernéticos, como também pode agir a favor de reduzir a carga de trabalho dos especialistas em segurança, permitindo que se concentrem em ameaças mais sofisticadas e menos previsíveis. Portanto, o uso de ML para a criação automática de regras de *firewall* é uma iniciativa de grande valor para o CITEEx e o EB.

1.2 Justificativa

A elaboração deste projeto de fim de curso (PFC) contribui em relação à crescente necessidade de fortalecer as defesas cibernéticas do Exército Brasileiro ao oferecer uma solução via *software* para a automatização de um problema recorrente. Em particular, o uso de técnicas de ML possui alguns benefícios imediatos.

O primeiro deles é o aumento da segurança, uma vez que os algoritmos de ML integrados ao sistema podem detectar e responder a ameaças em tempo real. Destaca-se, também, que um *pipeline* contínuo de retreinamento dos modelos de IA garante que as regras geridas estejam atualizadas, resguardando a rede interna contra ameaças mais atuais. Em segundo lugar, é obtido um aumento da eficiência operacional: a utilização desses algoritmos reduzirá a necessidade de intervenção manual, permitindo mais agilidade e precisão nas respostas a incidentes de segurança. Além disso, a integração do algoritmo com o *firewall*, *dashboards*, e sistemas de autenticação e notificação facilita a administração e o gerenciamento das novas regras aplicadas, reduzindo a complexidade operacional.

1.3 Objetivos

O presente projeto possuiu os objetivos elencados a seguir.

1. Desenvolver algoritmos de ML capazes de gerar, automaticamente, regras de *firewall*, tanto de forma estática a partir de arquivos de registro quanto de forma dinâmica, a partir da leitura de pacotes em tempo real com treinamento prévio;
2. Construção de um pipeline que implemente um ciclo de retreinamento periódico, à medida que novos dados de tráfego são coletados;
3. Elaboração de um sistema capaz de se integrar com *dashboards* e o *firewall*, assumindo, também, tarefas administrativas, como autenticação e envio de notificações, agindo, de modo geral, como um sistema de prevenção de intrusão (IPS).

1.4 Organização

O restante do trabalho foi organizado da seguinte forma:

- O capítulo 2 analisa os trabalhos relacionados, buscando avaliar outros trabalhos acadêmicos voltados à utilização de modelos de ML para a criação de regras de *firewall*, além de avaliá-los, trazendo sua utilidade ao presente contexto.
- O capítulo 3 explica os fundamentos teóricos empregados no decorrer deste trabalho.
- O capítulo 4 demonstra as decisões de projeto que nortearam o desenvolvimento do sistema, incluindo a escolha de algoritmos de ML, a arquitetura do pipeline de retreinamento e a integração com o *firewall* e *dashboards*, além de outras tarefas administrativas, como autenticação e envio de notificações. Por fim foram apresentados os resultados para cada base de dados, e a discussões sobre deles
- Já o capítulo 5 apresenta a interpretação final dos resultados e discussões por parte do grupo além de apresentar as considerações finais.

2 REVISÃO BIBLIOGRÁFICA

A análise de trabalhos relativos ao tema já publicados na área trouxe à tona os seguintes resultados.

Em (2), os autores propuseram uma classificação multi-classes de arquivos de *log* utilizando diversas técnicas de ML, como *Random Forest*, KNN, *artificial neural networks*, NB e J48. Para isso, eles utilizaram o fluxo apresentado na Figura 1 (2). Com relação à base de dados, eles utilizaram um *dataset* com as mesmas variáveis utilizadas neste artigo, porém com mais linhas de registro, e alcançaram um f1-score próximo de 100% para todos os métodos propostos.

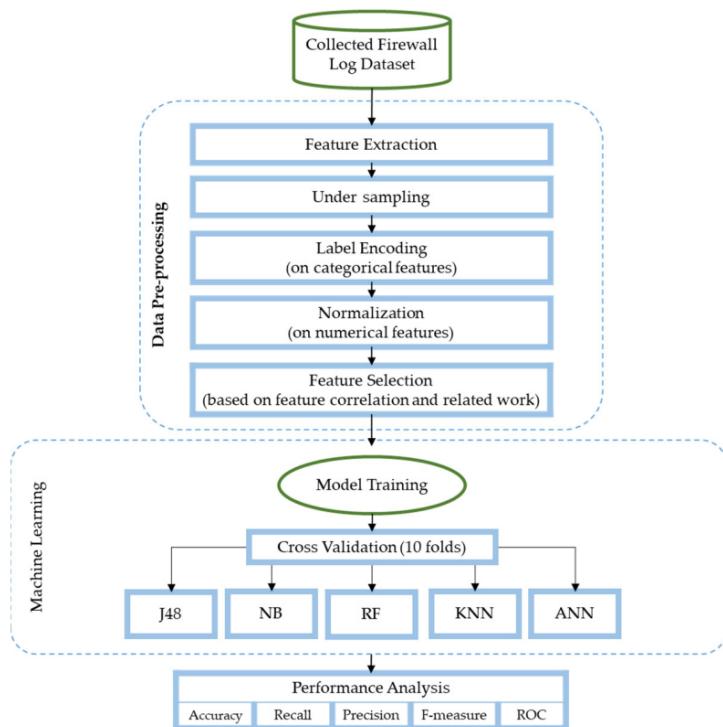


Figura 1 – Fluxo descrito no experimento de Cláudio Marques , Silvestre Malta e João Magalhães.

O trabalho de (4) também realizou classificação de *logs* para detectar pacotes maliciosos. Com essa finalidade, os autores utilizaram os algoritmos de KNN, SVM, *Random Forest*, *lightgbm* e *stacking classifier*. O *dataset* utilizado por eles foi o mesmo utilizado neste projeto (5). Esse trabalho também propôs a realização da conversão das variáveis referentes às portas em um grafo direcionado, e, a partir dele, extrair métricas como *Common neighbors*, *Jaccard Index*, *Salton Index*, *Sorensen Index* e *Adamic/Adar Index*, transformações que também foram incorporadas neste projeto. Os resultados obtidos nesse trabalho indicam valores de *F₁*-score próximo de 100% em todos os métodos.

Além disso, em (1), também utilizam-se algoritmos de ML para desenvolver regras de *firewall*. Entretanto, esse artigo se baseou em algoritmos de árvores de decisão, que apresentam maior interpretabilidade e permitem a construção de regras condicionais, a custo de uma menor eficiência. Quanto à base de dados, o autor gerou artificialmente bases de tráfego normal e malicioso utilizando o software *Kingview*. Sua metodologia está descrito no fluxo da Figura 2 (1).

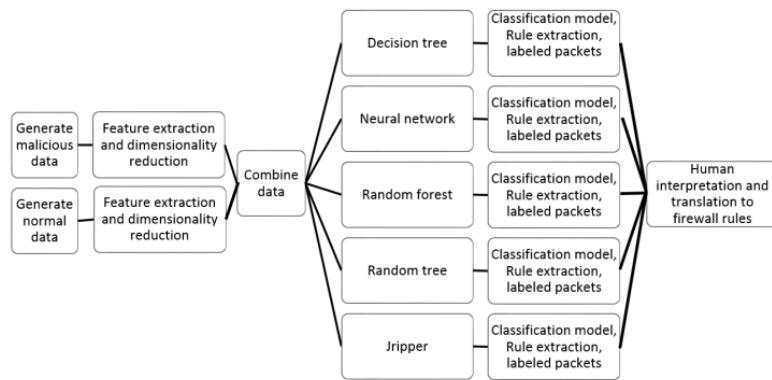


Figura 2 – Pipeline descrito no experimento Roland Verbruggen.

Em(6) o autor propõe uma multi classificação utilizando uma nova base, que será detalhada em seções subsequentes. O autor propõe a mapeamentos dos ataques em 6 categorias e a realização de um tratamento prévio dos dados por meio da remoção de outliers e seleção de colunas mais importantes, por meio da eliminação de colunas correlacionadas.

2.1 Avaliação dos trabalhos

Da avaliação dos trabalhos relacionados, chegou-se à conclusão de que seria possível desenvolver um algoritmo de ML capaz de construir regras de *firewall*. Para isso está sendo desenvolvido um algoritmo de ML conforme descrito em (2) para a base 1, a fim de conseguir gerar previsões pacote por pacote. Quanto a base 2, está sendo desenvolvido um algoritmo baseado no trabalho de (6). Desse modo, para levantar as regras para o *firewall*, será utilizado uma abordagem a qual será detalhada na metodologia desse trabalho em questão.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Firewall

Um *firewall* é uma coleção de componentes ou sistema colocado entre duas redes distintas e que satisfaz as seguintes propriedades: todo tráfego de rede, de entrada ou saída, deve passar por ele; somente tráfego autorizado, determinado pelas políticas de segurança, deve ser capaz de passar por ele; ele deve ser impenetrável (7). Um *firewall* pode ser implementado via software, hardware ou por uma combinação de ambos e atua como uma barreira entre uma rede confiável e outra não confiável, geralmente a *Internet*. Por conta dessas características, ele desempenha um papel crucial na proteção de redes contra ataques cibernéticos.

Um dos mecanismos mais importantes de um *firewall* é a filtragem de pacotes, consistindo de um grande número de regras de filtragem de baixo nível configuradas com ferramentas específicas oferecidas pelo fornecedor, sendo uma das formas mais comuns de empresas implementarem suas políticas de segurança de alto nível (8). Essas regras são fundamentais para o funcionamento de um *firewall* de filtragem e são baseadas em diversos critérios, como endereços IP de origem e destino, portas de comunicação, protocolos utilizados (TCP, UDP, ICMP, etc.) e sentido do fluxo (*inbound* ou *outbound*).

3.2 DMZ

A zona desmilitarizada (DMZ) é uma sub-rede que fornece serviços externos, como servidores web e de *email*, para uma rede não confiável, como a *Internet*, protegendo a rede interna de acessos diretos pela rede externa (9). A DMZ redireciona o tráfego destinado a esses serviços, impedindo o acesso direto à rede interna por usuários externos, enquanto mantém acesso limitado à rede interna confiável, proporcionando uma camada adicional de segurança.

3.2.1 Métodos de Ataque

Os ataques cibernéticos são ações maliciosas destinadas a comprometer a segurança de sistemas, redes e informações. Esses ataques podem assumir várias formas, cada uma com técnicas próprias e objetivos específicos. No contexto deste projeto, o modelo de treinado com a segunda base de dados utilizada (vide seção 4.1.1) categoriza as seguintes

formas de ataque principalmente: força bruta, injeção SQL, negação de serviço, negação de serviço distribuída.

Os ataques de força bruta representam tentativas sistemáticas de adivinhar senhas ou chaves criptográficas ao tentar todas as combinações possíveis. Este tipo de ataque pode ser direcionado a diferentes serviços, nos quais os atacantes tentam repetidamente descobrir credenciais de login. Neste projeto, a análise com os ataques de força bruta foi feita com os protocolos SSH e FTP e em conjunto com outras técnicas, como a exploração de vulnerabilidades de Cross-Site Scripting (XSS) em aplicações web, visando comprometer contas de usuários.

A injeção de SQL (SQL Injection) é outro método comum de ataque, que explora vulnerabilidades em aplicações web permitindo a inserção de comandos SQL maliciosos. Esses comandos podem manipular ou coletar dados do banco de dados da aplicação, comprometendo a confidencialidade e integridade das informações armazenadas.

Os ataques de negação de serviço (DoS) buscam tornar recursos ou serviços indisponíveis para seus usuários legítimos ao sobrecarregar o sistema alvo com uma quantidade excessiva de solicitações. As técnicas avaliadas no desenvolvimento do projeto incluem o envio de grandes volumes de tráfego HTTP, nos ataques Hulk e GoldenEye, e a manutenção de conexões abertas por meio de solicitações lentas, nos ataques SlowHTTPTest e Slowloris. Esses métodos esgotam os recursos do servidor, impedindo seu funcionamento normal.

Os ataques de negação de serviço distribuída (DDoS) são uma variação dos ataques DoS, onde múltiplos sistemas comprometidos são usados para lançar o ataque. Utilizando ferramentas como HOIC e LOIC, esses ataques geram um grande volume de tráfego proveniente de várias fontes, dificultando ainda mais a mitigação e resposta ao incidente.

3.3 IDS e IPS

Sistemas de detecção de intrusão (IDS, *Intrusion Detection System*) e sistemas de prevenção de intrusão (IPS, *Intrusion Prevention System*) são ferramentas importantes para garantir a segurança de redes e sistemas. Embora ambos tipos de sistema atuem em conjunto para detectar e mitigar ameaças, o intuito e escopo das funcionalidades são distintos.

O IDS é um sistema passivo que monitora o tráfego de rede ou eventos de sistema em busca de sinais de atividades suspeitas ou maliciosas. Sua principal função é detectar possíveis ataques, como tentativas de invasão ou explorações de vulnerabilidades, enviando alertas para que administradores de rede tomem as devidas providências. Esses sistemas podem ser baseados em assinaturas, detectando padrões conhecidos de ataques, ou baseados em anomalias, identificando desvios de comportamento em relação ao que

é considerado normal para um ambiente específico. Embora seja altamente eficaz para identificar tentativas de invasão, o IDS não age diretamente para bloquear os ataques, limitando-se ao papel de detecção.

Já o IPS, por outro lado, possui uma função mais proativa. Além de monitorar o tráfego como o IDS, ele é capaz de bloquear automaticamente atividades maliciosas em tempo real. O IPS é configurado para intervir diretamente, interrompendo comunicações ou bloqueando pacotes de dados que correspondam a padrões de ataque, prevenindo que a ameaça comprometa a rede ou sistema. Ao contrário do IDS, o IPS atua de maneira inline, ou seja, ele se posiciona diretamente entre o tráfego de rede e os sistemas, podendo interromper o tráfego suspeito antes mesmo que ele chegue ao destino.

3.4 Métricas de avaliação

Uma etapa importante no treinamento de algoritmos supervisionados consiste na avaliação dos resultados. Usualmente, utiliza-se a acurácia do modelo como métrica de avaliação. Ela é definida conforme a equação (3.1), em que n_a é numero de acertos e n_t é numero total de casos.

$$\text{acc} = \frac{n_a}{n_t} \quad (3.1)$$

Entretanto, a acurácia pode conduzir a conclusões equivocadas quando as classes não estão balanceadas. Por exemplo, considere uma base de dados na qual 90% dos dados pertencem a uma dada classe A . Se o modelo treinado classificar todas as entradas como sendo da classe A , a acurácia final seria de 90%. No entanto, isso não indica que o modelo é bom, pois ele simplesmente classifica todas as entradas da mesma maneira, independentemente da entrada real.

Devido a essas limitações, outras métricas, como precisão, *recall* e F_1 -score, são definidas para fornecer uma avaliação mais robusta dos resultados de um modelo, conforme apresentado nas equações (3.2), (3.3) e (3.4), utilizando os conceitos de verdadeiro e falso positivo e de verdadeiro e falso negativo nos resultados do modelo. Verdadeiro positivo refere-se à correta classificação de uma entrada como pertencente a uma determinada classe. Falso positivo, por sua vez consiste na classificação incorreta de uma entrada como pertencente a uma classe quando, na verdade, esse não é o caso. De maneira semelhante, verdadeiro negativo ocorre quando uma entrada é corretamente classificada como não pertencente a uma classe. Finalmente, falso negativo é a classificação incorreta de uma entrada como não pertencente à classe.

Nesse contexto, uma precisão alta indica poucos falsos positivos, ou seja, quando o modelo classifica uma entrada como pertencente a uma classe, na maioria das vezes está

correto. Por outro lado, um *recall* alto significa poucos falsos negativos, indicando que, quando uma entrada pertence a uma classe, o modelo geralmente a classifica corretamente. Isto é, a precisão mede a frequência com que as previsões positivas estão corretas, enquanto o *recall* avalia a capacidade do modelo de identificar todas as instâncias positivas de uma classe. Finalmente, o F_1 -score é uma média harmônica da precisão e do *recall*, utilizada para avaliar se o modelo possui simultaneamente poucos falsos positivos e falsos negativos para uma determinada classe.

$$\text{prec} = \frac{\text{verdadeiro positivo}}{\text{verdadeiro positivo} + \text{falso positivo}} \quad (3.2)$$

$$\text{recall} = \frac{\text{verdadeiro positivo}}{\text{verdadeiro positivo} + \text{falso negativo}} \quad (3.3)$$

$$F_1 = \frac{\text{prec} * \text{recall}}{\text{prec} + \text{recall}} \quad (3.4)$$

3.5 Validação cruzada

Nos problemas de aprendizado de máquina, é comum dividir os dados em dois conjuntos: um para treino e outro para validação. Entretanto, em muitos casos, a quantidade de dados disponíveis é escassa, de modo que a divisão entre tais grupos pode causar uma queda no desempenho do algoritmo. Para solucionar esse problema, utiliza-se a técnica de validação cruzada, a qual consiste em dividir a base de dados em n grupos, treinar o algoritmo com $n - 1$ grupos e validar no restante e, em seguida, selecionar um novo grupo para ser o de validação e treinar novamente nos $n - 1$ grupos restantes. Repetem-se esses passos até que cada grupo tenha sido utilizado para validação, como mostrado na Figura 3 (10).

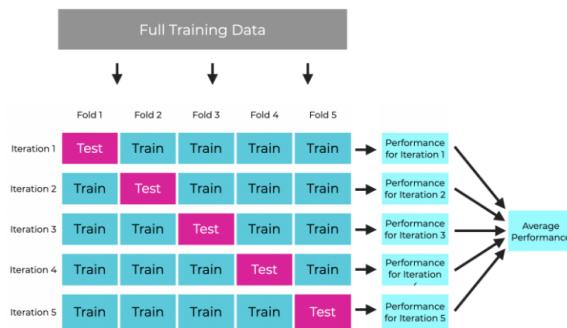


Figura 3 – Representação da validação cruzada.

A validação cruzada oferece uma estimativa mais robusta e confiável do desempenho do modelo, especialmente quando a quantidade de dados é limitada. Esta técnica permite que todos os dados sejam utilizados tanto para treino quanto para validação, garantindo que o modelo seja avaliado de maneira abrangente e reduzindo a possibilidade

de sobreajuste (*overfitting*) - ocorre quando um modelo de aprendizado de máquina se ajusta excessivamente aos dados de treino, capturando ruídos e padrões específicos que não generalizam bem para novos dados. Além disso, a validação cruzada facilita a comparação entre diferentes modelos ou hiperparâmetros, fornecendo uma base sólida para a seleção do modelo mais adequado para o problema em questão.

3.6 Algoritmos de Aprendizado de máquina

3.6.1 Regressão logística

A regressão logística é um método estatístico amplamente utilizado para modelar a probabilidade de ocorrência de um evento binário, ou seja, um evento com duas possíveis saídas (por exemplo, sucesso ou falha). Ela é particularmente útil em problemas de classificação. A probabilidade de um evento ocorrer é determinada pela equação (3.5), onde $\beta_0, \beta_1, \dots, \beta_n$ são os coeficientes que o modelo precisa estimar, e X_1, X_2, \dots, X_n são as variáveis independentes. Esta fórmula utiliza a função sigmoide, que mapeia qualquer valor real em um intervalo entre 0 e 1, permitindo a interpretação dos resultados como probabilidades.

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)}} \quad (3.5)$$

Ao aplicar a transformação *logit* na equação (3.5), obtemos a equação (3.6), que lineariza a relação entre as variáveis independentes e a variável dependente.

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n \quad (3.6)$$

Para estimar os coeficientes β , utiliza-se a técnica da máxima verossimilhança, que ajusta o modelo de forma a maximizar a probabilidade de observar os dados amostrais dados os parâmetros estimados. Em problemas de classificação binária, uma vez calculada a probabilidade de um evento ocorrer, aplica-se um limiar (*threshold*) para decidir a classificação final. Por exemplo, se a probabilidade calculada for maior que 0.5, o modelo pode classificar a entrada como um evento positivo ($Y = 1$). Para classificações multinomiais (com mais de duas classes), a mesma ideia é aplicada usando a função *softmax*, que é uma generalização da função sigmoide para múltiplas classes. A função *softmax* é expressa na equação (3.7).

$$P(Y = j|\mathbf{x}) = \frac{e^{\beta_j \cdot \mathbf{x}}}{\sum_{i=1}^k e^{\beta_i \cdot \mathbf{x}}} \quad (3.7)$$

A regressão logística, portanto, fornece uma maneira poderosa e flexível de modelar probabilidades em problemas de classificação, permitindo tanto a interpretação das relações entre variáveis quanto a tomada de decisões baseadas em probabilidades calculadas.

3.6.2 Algoritmos em árvores

3.6.2.1 Árvore de decisão

Uma árvore de decisão é um algoritmo de aprendizado supervisionado que funciona construindo uma estrutura em forma de árvore de regras decisórias. Cada nó da árvore representa uma regra envolvendo uma das variáveis, enquanto os nós folha representam os resultados finais da classificação. Durante o processo de avaliação, a árvore é percorrida de cima para baixo. Em cada nó, a condição é analisada: se a condição for verdadeira, a avaliação segue para o nó da direita; caso contrário, segue para o nó da esquerda. Esse processo se repete até que se alcance um nó folha, que fornece a classificação final. A Figura 4 mostra um exemplo de uma árvore de decisão.

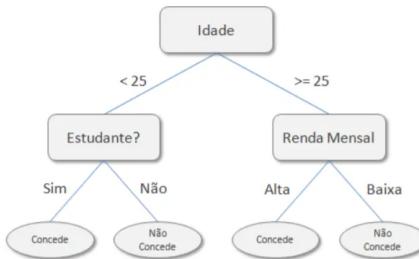


Figura 4 – Árvore de decisão para determinar fornecimento de crédito .

Embora as árvores de decisão ofereçam uma boa interpretabilidade dos resultados, permitindo que as decisões tomadas pelo modelo sejam facilmente compreendidas, elas apresentam algumas desvantagens. Uma das principais limitações é a susceptibilidade ao sobreajuste (*overfitting*), onde uma pequena variação nos dados pode resultar em uma configuração de árvore completamente diferente, prejudicando a capacidade do modelo de generalizar para novos dados (11).

3.6.2.2 Random forest

Para abordar o problema de generalização limitada das árvores de decisão, surgiu o algoritmo de *Random Forest*. Esse método combina várias árvores de decisão para melhorar a precisão e a robustez da previsão. O algoritmo de *Random Forest* segue os passos elencados abaixo.

1. Seleção, com reposição, de N elementos do conjunto de dados original, criando várias amostras de treinamento;

2. Seleção aleatória de K variáveis do conjunto de dados;
3. Geração de uma árvore de decisão para cada amostra criada.

Para determinar a classificação de uma entrada, o algoritmo avalia a entrada em todas as árvores da floresta e seleciona a classificação mais frequente entre as árvores. Essa abordagem de agregação reduz a variância e melhora a capacidade de generalização do modelo, mitigando o problema de sobreajuste típico das árvores de decisão individuais.

3.6.2.3 Gradient boosting decision tree

Semelhante ao *Random Forest*, o *Gradient Boosting Decision Tree* (GBDT) também utiliza múltiplas árvores de decisão para realizar classificações. No entanto, ao invés de fazer previsões de forma paralela, como no *Random Forest*, o GBDT opera sequencialmente. Cada árvore é construída para corrigir os erros da árvore anterior. Essa abordagem iterativa permite que o GBDT se ajuste melhor aos dados, oferecendo uma capacidade de generalização superior, especialmente em situações com classes não balanceadas e dados com valores nulos. Ao focar na correção de erros anteriores, o GBDT tende a produzir modelos mais precisos e robustos (12).

3.6.2.4 Lightgbm

O *LightGBM* é uma implementação avançada do GBDT, focada em eficiência e desempenho. Este algoritmo é otimizado para reduzir o tempo de execução e o consumo de memória, além de oferecer suporte a processamento por GPU (13). O *LightGBM* se diferencia dos outros algoritmos de árvores ao incorporar uma técnica chamada *Gradient One-Side Sampling* (GOSS). Essa técnica mantém as instâncias de dados com gradientes maiores (que indicam maiores erros) e amostra uma parcela dos dados com gradientes menores (dados bem ajustados). Ao concentrar-se nos dados mais difíceis de ajustar, o *LightGBM* melhora a eficiência do treinamento e a precisão do modelo (14).

3.6.3 MLP

3.6.3.1 Perceptron

O Perceptron é um dos modelos mais simples de rede neural, desenvolvido inicialmente para tarefas de classificação binária. Seu funcionamento pode ser descrito através das etapas descritas abaixo.

1. Entrada: O modelo recebe um vetor numérico de entradas, representado como $\mathbf{x} = (x_1, x_2, \dots, x_n)$.

2. Multiplicação por Pesos: Cada entrada x_i é multiplicada por um peso correspondente w_i . Esses pesos são ajustados durante o treinamento do modelo.
3. Soma Ponderada: Os produtos das entradas e seus pesos são somados, resultando em uma soma ponderada $u(x) = \sum_{i=1}^n w_i x_i$, conforme a equação (3.8).
4. Função de Ativação: A soma ponderada é passada por uma função de ativação $f(u)$. No caso do Perceptron clássico, utiliza-se a função degrau (ou Heaviside), que produz uma saída binária baseada em um limiar θ . Se $u(x) > \theta$, a saída é 1; caso contrário, é 0. Matematicamente, isso é representado pela equação (3.9).

$$u(x) = \sum_{i=1}^n w_i x_i \quad (3.8)$$

$$y = f(u(x)) = (u(x) - \theta > 0) \quad (3.9)$$

A Figura 5 ilustra a estrutura de um nó Perceptron, onde as entradas são combinadas através de pesos, somadas e, em seguida, processadas por uma função de ativação para produzir uma saída binária.

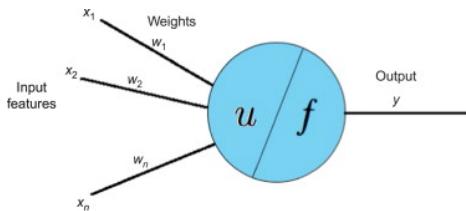


Figura 5 – Representação de um nó perceptron.

Originalmente concebido para classificação binária, o Perceptron pode ser estendido para tarefas de classificação multiclasse. Isso é feito substituindo a função de ativação sigmoide pela função *softmax*. A função *softmax* transforma o vetor de saídas do Perceptron em um vetor de probabilidades, onde cada entrada representa a probabilidade da amostra pertencer a uma das classes possíveis. Essa extensão permite ao Perceptron lidar com problemas mais complexos, mantendo a simplicidade e eficiência do modelo original.

3.6.3.2 MLP

O *Multi-Layer Perceptron* (MLP) é uma concatenação de camadas de perceptrons, conforme ilustrado pela Figura 6 de modo a permitir a aproximação de funções não lineares. Para poder ajustar seus pesos em cada ciclo de treinamento, o MLP utiliza uma técnica chamada de *back-propagation* (15). As duas fases do algoritmo estão explicitadas abaixo.

1. O passo para frente, ou *forward pass*, onde as entradas são passadas pela rede e as saídas são obtidas e comparadas com o resultado esperado

2. O passo para trás, ou *backward pass*, onde o gradiente da função de perda na camada final da rede é calculado. Após esse cálculo, o resultado é utilizado para atualizar de forma recursiva os pesos por meio da regra da cadeia (*chain rule*).

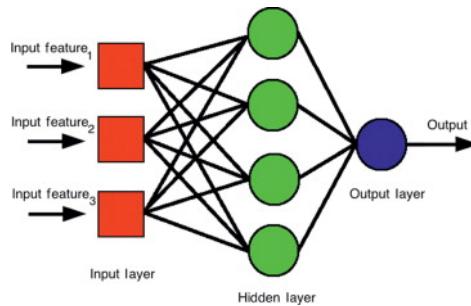


Figura 6 – Representação de uma rede perceptron.

3.7 Relatórios Shap classe

3.7.1 Conceito

O *Shapley Additive Explanations* ou SHAP é uma técnica baseada na teoria dos jogos que permite atribuir uma maior interpretabilidade a um modelo de ML mesmo que ele possua uma estrutura complexa, como no caso de redes neurais. Essa técnica calcula individualmente o impacto de cada variável, de modo a determinar seu nível de importância. Para isso, a técnica aproxima funções complexas por meio de funções lineares, de modo a definir coeficientes que estipulam o impacto de cada variável (16). Essa ideia é representada na Figura 7. O método possui como grande vantagem o fato de ser neutro frente aos algoritmos, isto é, ele se comporta da mesma forma independente da técnica de ML utilizada.

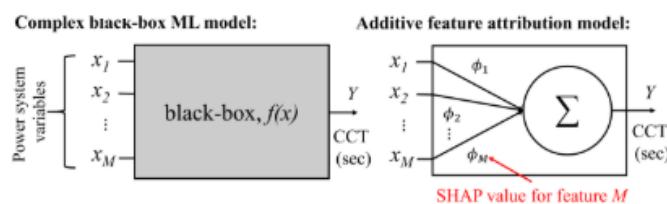


Figura 7 – Representação do mapeamento gerado pela função SHAP

3.8 Exemplo ilustrativo

Suponha que se queira desenvolver um modelo para determinar se uma pessoa recebe um salário maior que R\$50.000,00 por mês com bases em suas características. Para isso, desenvolve-se um modelo de ML qualquer e, para conseguir interpretar o impacto

de cada variável, é plotado o relatório SHAP associado à classe das pessoas que recebem mais do que R\$50.000,00, apresentado na Figura 8.

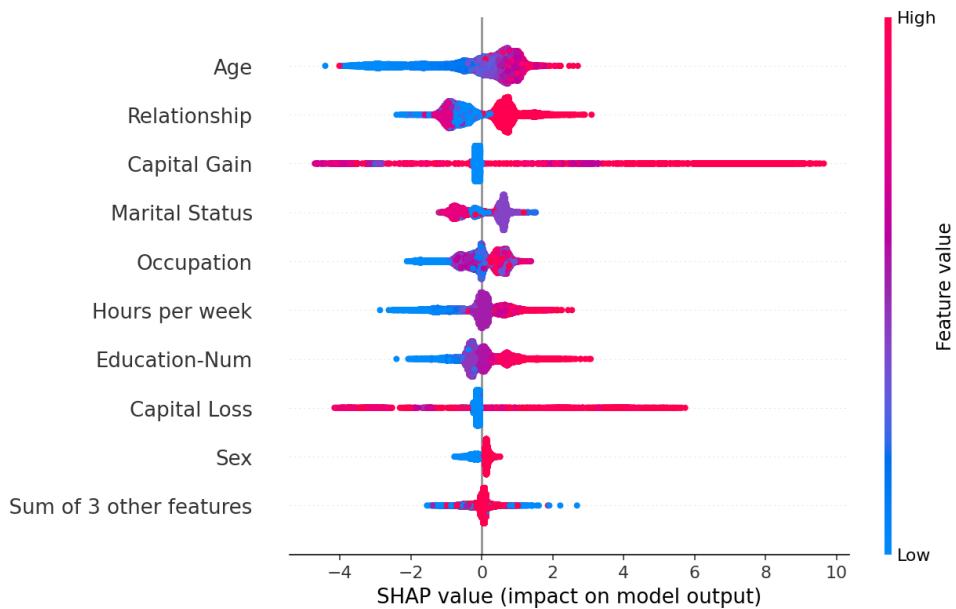


Figura 8 – Representação de um gráfico shap para a classe salários maiores que R\$50000,00

Observando o gráfico, percebe-se que há uma série de pontos variando do espectro vermelho ao azul. Cada ponto representa uma entidade avaliada: no caso deste problema, uma pessoa. Pontos mais próximos do espectro vermelho significam que a variável para aquela pessoa possuía uma valor alto, ao passo que pontos azuis significam o contrário. No eixo x, está representado o SHAP-value, que representa o impacto da variável. Valores mais à direita significam que o impacto é positivo, enquanto os mais à esquerda representam o contrário.

Assim, observando a primeira linha (idade), observa-se que valores de cor vermelha estão mais à direita e valores mais azuis à esquerda. Desse modo, pode-se concluir que idade avançada contribui positivamente para a classe mais que R\$50.000,00. Portanto, pode-se concluir que pessoas mais velhas tendem a ganhar maiores salários. No relatório SHAP, sempre se almeja que exista uma divisão clara de cores de um lado para o outro, porém nem sempre é possível. No caso da variável “ganho de capital”, por exemplo, não é possível concluir pelo gráfico como a variável impacta a classe.

4 PROJETO

Neste capítulo serão abordadas as tarefas realizadas durante o desenvolvimento deste trabalho, tanto no que diz respeito aos modelos de aprendizado de máquina quanto ao desenho do sistema em si. Em seguida, serão abordados os resultados encontrados.

4.1 Metodologia

4.1.1 Bases de dados utilizadas

Para realizar o treinamento da IA foram selecionados duas bases de dados distintas. A primeira (5) contém variáveis apresentadas na Tabela 2. Essa base possui três classes: *allow*, *deny* e *drop*. No contexto dessa base, *allow* significa permitir que o pacote seja transmitido, enquanto *deny* significa bloquear o pacote e enviar uma notificação de bloqueio e o *drop* somente bloquear, sem notificação adicional.

A segunda base (17) possui uma quantidade muito maior de colunas, totalizando 80 variáveis distintas, as quais podem ser consultadas em (18). Dentre elas, destaca-se, no contexto do presente projeto, a seleção apresentada na tabela 3. Além disso, essa base classifica os pacotes de acordo com as possíveis classes apresentadas na tabela 1, que também mostra o mapeamento utilizado neste projeto para simplificar o processo de treinamento.

Ataque	Mapeamento do Ataque
SSH-Bruteforce	bruteforce
FTP-BruteForce	bruteforce
Brute Force -XSS	web attack
Brute Force -Web	web attack
SQL Injection	web attack
DoS attacks-Hulk	DOS
DoS attacks-SlowHTTPTest	DOS
DoS attacks-Slowloris	DOS
DoS attacks-GoldenEye	DOS
DDOS attack-HOIC	DDOS
DDOS attack-LOIC-UDP	DDOS
DDoS attacks-LOIC-HTTP	DDOS
Bot	botnet
Benign	allow
Infiltration	infiltração

Tabela 1 – Descrição dos tipos de ataques e tráfego de rede

Em relação ao enriquecimento da base, desenvolveram-se funções responsáveis

por realizar o pré-tratamento dos dados. Para a primeira base de dados, foi seguido o procedimento descrito em (4). Em relação à segunda base, o pré-processamento seguiu o *pipeline* apresentado na Figura 9.

	Nome da variável	Descrição
1	Source Port	Porta de partida do cliente
2	Destination Port	Porta de destino do cliente
3	NAT Source Port	Porta de partida com tradução NAT
4	NAT Destination Port	Porta de destino com tradução NAT
5	Elapsed Time	Tempo decorrido do fluxo
6	Bytes	Total de bytes
7	Bytes Sent,	Total de bytes enviados
8	Bytes Received	Total de bytes recebidos
9	Packets	Total de pacotes
10	pkts_sents	pacotes enviados
11	pkts_received	Pacotes recebidos
12	Action	Classe (allow,drop,deny)

Tabela 2 – Descrição das variáveis presentes no *dataset* público 1

Termo	Descrição
Fwd seg size min	Tamanho mínimo do segmento observado na direção direta
Dst port	porta de destino
Init fwd win bytes	Número de bytes enviados na janela inicial na direção direta
Init bwd win bytes	Número de bytes enviados na janela inicial na direção reversa
Bwd pkts/s	Número de pacotes na direção reversa por segundo
Flow pkts/s	Taxa de pacotes do fluxo, que é o número de pacotes transferidos por segundo
ECE flag cnt	Número de pacotes com ECE (usada para indicar notificação explícita de congestionamento)
ACK flag count	Número de pacotes com ACK (usada para confirmar o recebimento de pacotes)
PSH flag cnt	Número de pacotes com PSH (usada para empurrar dados imediatamente para a aplicação receptora)
URG flag count	Número de pacotes com URG (usada para indicar que os dados contidos no pacote devem ser processados imediatamente)

Tabela 3 – Descrição das variáveis mais relevantes presentes no *dataset* público 2

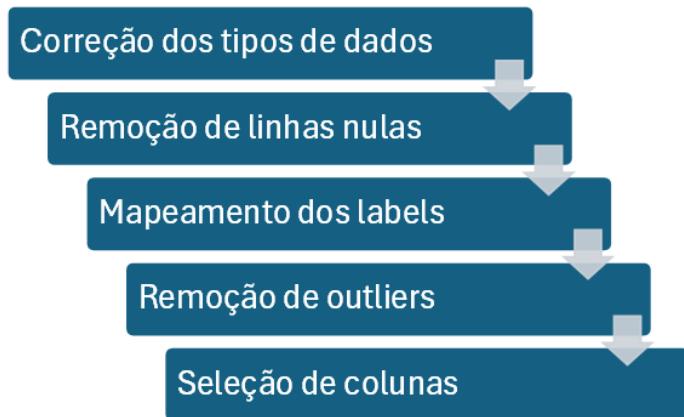


Figura 9 – Representação do pipeline de tratamento de dados

4.1.2 Modelos utilizados

Fez-se uso de um modelo de *Ensemble*, que funciona como um modelo democrático. Seu princípio consiste em realizar uma avaliação de um dado por meio da resposta mais votada dentre uma seleção de diversos modelos. Para o experimento em questão, selecionaram-se os modelos apresentados na Tabela 4.

1	Regressão logística
2	Random forest
3	lightgbm
4	MLP
5	Gradient boosting decision tree

Tabela 4 – Modelos presentes no ensemble

Quanto a avaliação do modelo, para a base 1, como há poucos logs, foi implementada uma validação cruzada, enquanto para a base 2, como há muitos logs, essa técnica não se fez necessária.

Concluída a etapa de testes e de pré-processamento, iniciou-se o processo de desenvolvimento das regras de firewall. A fim de criar as regras de firewall, foi desenvolvido o seguinte procedimento:

1. Um registro é classificado pelo modelo;
2. Caso ele seja classificado como maligno, ele será adicionado como uma possível nova regra de block;
3. Caso essa nova regra não conflita com nenhuma regra crítica e caso não haja nenhuma outra regra semelhante já criada, ela, então, será adicionada

4. Para as variáveis categóricas como porta e protocolo, foi considerado o valor exato na regra;
5. Para valores numéricos como tamanho do pacote, foi pego um *range* entre $[0.95 * \text{valor}, 1.05 * \text{valor}]$ (valores configuráveis).

4.2 Resultados

4.2.1 Dataset 1

Os resultados referentes à base de dados 1 utilizando o modelo lightgbm está descrito na matriz de confusão presente em Tabela 5.

	Allow	Deny	Drop	Precisão
Allow	9945	1	0	0.99979893
Deny	2	7182	0	0.9972
Drop	0	2762	9958	0.78286164
Recall	0.99989946	0.72217195	1.00000	
F1 Score	0.99984919	0.83857785	0.87820795	
Accuracy		0.9037		

Tabela 5 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o lightgbm para a base 1

Analizando a matriz de confusão, percebe-se que o modelo consegue identificar muito bem os casos nos quais ele deve permitir a transmissão do pacote, encontrando um f1-score de mais de 99%. Entretanto, esse resultado não se mantém para as classes deny e drop, para as quais o f1-score atinge resultados em torno de 85%. Dessa maneira, nota-se que o modelo classifica muitos pacotes como drop quando a classificação correta deveria ser deny. Para entender esse fenômeno, pode-se avaliar o relatório SHAP presente em 10. Ao analisá-lo, observa-se que muita importância é dada para a variável *NAT translation destination*. No gráfico allow, os valores altos dessa variável (marcado de rosa) indicam um impacto negativo para a classe, enquanto para as demais ele possui um impacto positivo.

Sabe-se que essa variável é booleana, ou seja, assume valor 1 quando o protocolo NAT é aplicado e 0 caso contrário. Desse modo pode-se concluir que a aplicação do protocolo NAT implica em bloquear o pacote, seja como deny ou drop, o que justifica ele identificar muito bem quando é para classificar como allow. Porém, não há uma variável que ajude a identificar com facilidade quando um pacote é deny ou drop, o que implica a classificação incorreta entre deny e drop para diversos cenários. Esse mesmo resultado é encontrado para os demais modelos em árvores como random forest e gradient boost, como pode ser visto nos resultados presentes no apêndice B.

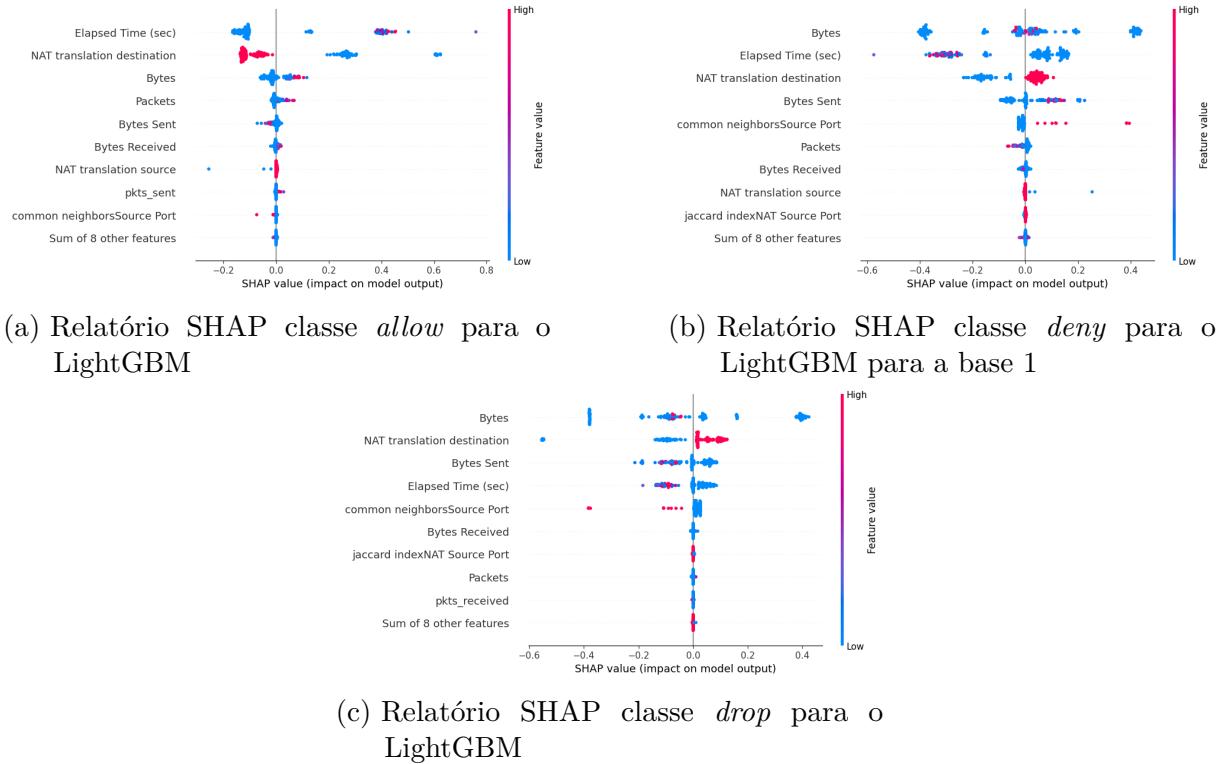


Figura 10 – Comparaçao dos relatórios SHAP para diferentes classes no LightGBM

Quanto ao MLP, seu resultado é descrito na matriz de confusão presente em 6

	Allow	Deny	Drop	Precisão
Allow	9925	5	0	0.9995
Deny	18	1108	0	0.98401421
Drop	4	8832	9958	0.52984995
Recall	0.9978	0.11141277	1.00000	
F1 Score	0.99864165	0.20016259	0.69268225	
Accuracy			0.70321608	

Tabela 6 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o multi layer perceptron para a base 1

Analizando esses valores, observa-se que, assim como nos algoritmos baseados em árvores, o modelo identifica muito bem a classe allow, porém confunde ainda mais a classificação entre deny e drop, com muitos registros deny sendo classificados como drop, o que leva a um f1-score muito baixo para a classe deny.

Analizando o relatório SHAP presente em 15, nota-se que continua sendo dado muito peso à classe NAT translation destination, o que explica porque o modelo MLP identifica relativamente bem a classe allow. Contudo, há um comportamento diferente para as demais variáveis. Algumas variáveis, como jaccard index, têm um impacto negativo para valores altos, de modo que ela, embora não tenha uma impacto tão significativo como nos

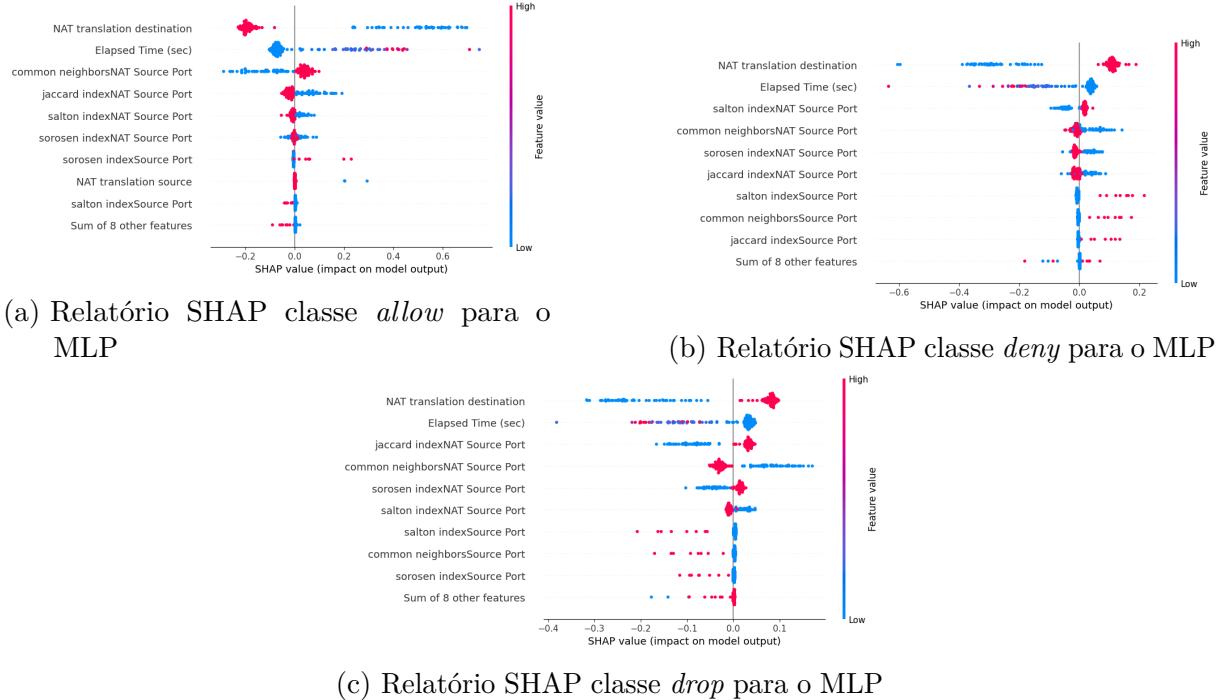


Figura 11 – Comparação dos relatórios SHAP para diferentes classes no MLP para a base 1

algoritmos em ávore, está sendo utilizada de forma incorreta pelo modelo para diferenciar deny de drop.

Quanto ao modelo de regressão logística, ele apresenta um resultado semelhante ao do MLP, como pode ser verificado pela sua matriz de confusão e seu relatório SHAP, todos presentes no anexo B.

4.2.2 Dataset 2

Em relação ao dataset 2 utilizado por este projeto, o resultado é apresentado na tabela de confusão para o modelo lightgbm na Tabela 7

Observando os resultados, percebe-se que o modelo consegue identificar muito bem os tipos de ataque, com exceção do *infiltration*, para o qual o F1-Score encontrado foi de 43%. Isso se deve a muitos registros de *infiltration* serem classificados como *allow*, além de muitos registros de *allow* serem classificados como *infiltration*, o que reduz tanto sua precisão quanto seu recall.

Para entender o porquê disso, pode-se analisar o relatório SHAP apresentado na Figura 16. Observando a variável fwd seg size min (tamanho mínimo do segmento no sentido direto), observa-se que valores altos dessa variável (marcado em rosa) têm impacto negativo em ambas classes, o que dificulta o modelo em diferenciar as duas classes.

Para a classe web, observa-se que, em casos nos quais a variável ECE é positiva, o

	allow	bruteforce	WEB	DOS	DDOS	botnet	infiltration	Precisão
allow	770685	10	34	45	6	11	19198	0.98
bruteforce	17	359413	0	71222	0	0	5	0.83
WEB	22	0	548	0	0	0	12	0.94
DOS	20	2261	0	539846	0	0	1	1.00
DDOS	0	0	0	0	651667	0	1	1.00
botnet	62	0	0	0	0	271312	3	1.00
infiltration	284353	4	52	3	0	52	111683	0.29
Recall	0.73	0.99	0.86	0.88	1.00	1.00	0.86	
F1 Score	0.84	0.91	0.90	0.94	1.00	1.00	0.43	
Accuracy						0.88		

Tabela 7 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para o lightgbm na base 2

modelo tende mais a classificar registros como ataques web quando comparados à classe allow. Para o *bruteforce*, observa-se que valores altos da variável fwd seg size min tendem a fazer que o modelo classifique registros como ataques de força bruta. Para ataques DDOS, observa-se que valores altos de init fwd win bytes indica ao modelo que o registro se trata de um ataque DDOS.

Para o caso dos ataques DOS, percebe-se que ele também possui como principal indicador a variável fwd seg size, de modo que valores altos também indicam esse tipo de ataque. Desse modo, em algumas situações, o modelo pode confundir DOS com bruteforce, já que ambos utilizam o mesmo indicador predominantemente. Assim, para diferenciar entre ambos tipos de ataques, o modelo precisa utilizar outra variável: pelo relatório SHAP, esta pode ser bwd pkts/s que, para valores altos, tende a ser bruteforce e, para valores baixos, tende a ser DOS. Analisando os ataques botnet, observa-se que valores altos de tot lenlen fwd pkts tendem a ter um efeito negativo para a classe botnet, sendo assim um dos seus principais indicadores.

Essas conclusões majoritariamente se repetem para os demais modelos, cujos resultados se encontram no apêndice C. Pontuando as diferenças observadas, a primeira reside na detecção do ataque bruteforce, pois, no caso da regressão logística, não há muita influência do fwd seg size min, reduzindo a precisão do modelo.

Quanto ao ataque de DDOS, os modelos que não são baseados em árvores detectam uma influência nas variáveis ECE flag e ACK flag. Isso, porém, não afetou a detecção dessa classe. Em relação ao DOS, os modelos que não são baseados em árvores detectam uma influência na variável e ACK flag, sem resultar, entretanto, em mudanças significativas no resultado.

Analizando a classe *infiltration*, percebe-se que para o modelo de regressão logística teve uma considerável influência sobre a flag ACK, o que explica por que ele apresentou o

pior resultado para esse tipo de ataque. O gradient boost e o MLP não tiveram um impacto negativo para valores altos de fwd seg size min, o que faz que o modelo se confunda menos com a classe allow (precisão maior), mas reduz o poder de deteção dessa classe (recall menor)

Por fim, outra diferença encontrada para o modelo de regressão linear reside no fato dele apresentar um péssimo f1-score para web attack, porém não é possível chegar em grandes conclusões pois há muitos poucos elementos na base com esse ataque, o que dificulta o treinamento.

4.2.3 Funcionalidades implementadas do sistema

Para determinar as funcionalidades a serem implementadas no sistema, foi feito, primeiramente, um estudo inicial sobre o tema, revisando a literatura acerca do funcionamento de firewalls e fazendo a revisão bibliográfica mencionada no capítulo 2. Em seguida, foi feita uma modelagem inicial, visando a atender os objetivos propostos para o projeto. Na sequência, fez-se uma reunião para levantar os requisitos e fazer uma modelagem mais detalhada da solução a ser entregue. Após isso, elaboraram-se os casos de uso apresentados no Apêndice A.

Começou-se, então, a Fase I de desenvolvimento. Nela, foram implementadas funcionalidades básicas, porém de grande importância, como sistemas de persistência em diretório de arquivos e em banco de dados, API para comunicação com usuários (tanto sistêmicos quanto clientes de navegador), sistema de notificação via e-mail, sistema de autenticação e configurações customizáveis para ajustar o comportamento do sistema, além de modelagem de regras críticas e dos modelos de IA a ser utilizados, conforme descrito anteriormente em 4.1.1.

Na Fase II de desenvolvimento, houve dois grandes avanços na implementação de um sistema funcional. O primeiro - e consideravelmente relevante no contexto do projeto - foi a integração do sistema com o firewall Iptables, permitindo a leitura e escrita de regras de firewall. A implementação foi feita de forma abstrata, com o conceito de *FirewallWriter*, sendo o *IPTablesWriter* a primeira implementação concreta. Para permitir a integração com outros firewalls comerciais, basta criar uma implementação concreta do *FirewallWriter* correspondente. Ainda em relação a essa funcionalidade, implementou-se a lógica de execução de comandos tanto em ambientes *Linux* locais, quanto em ambientes remotos via *SSH*, a fim de permitir integrações com firewalls instalados tanto local quanto remotamente.

Além disso, na Fase II, começou-se a implementação, de fato, de um cliente Web para a interação usuário-sistema conforme apresentada no Anexo A. Fez-se um cliente reativo e funcional nos principais navegadores comerciais. Em particular, implementou-se

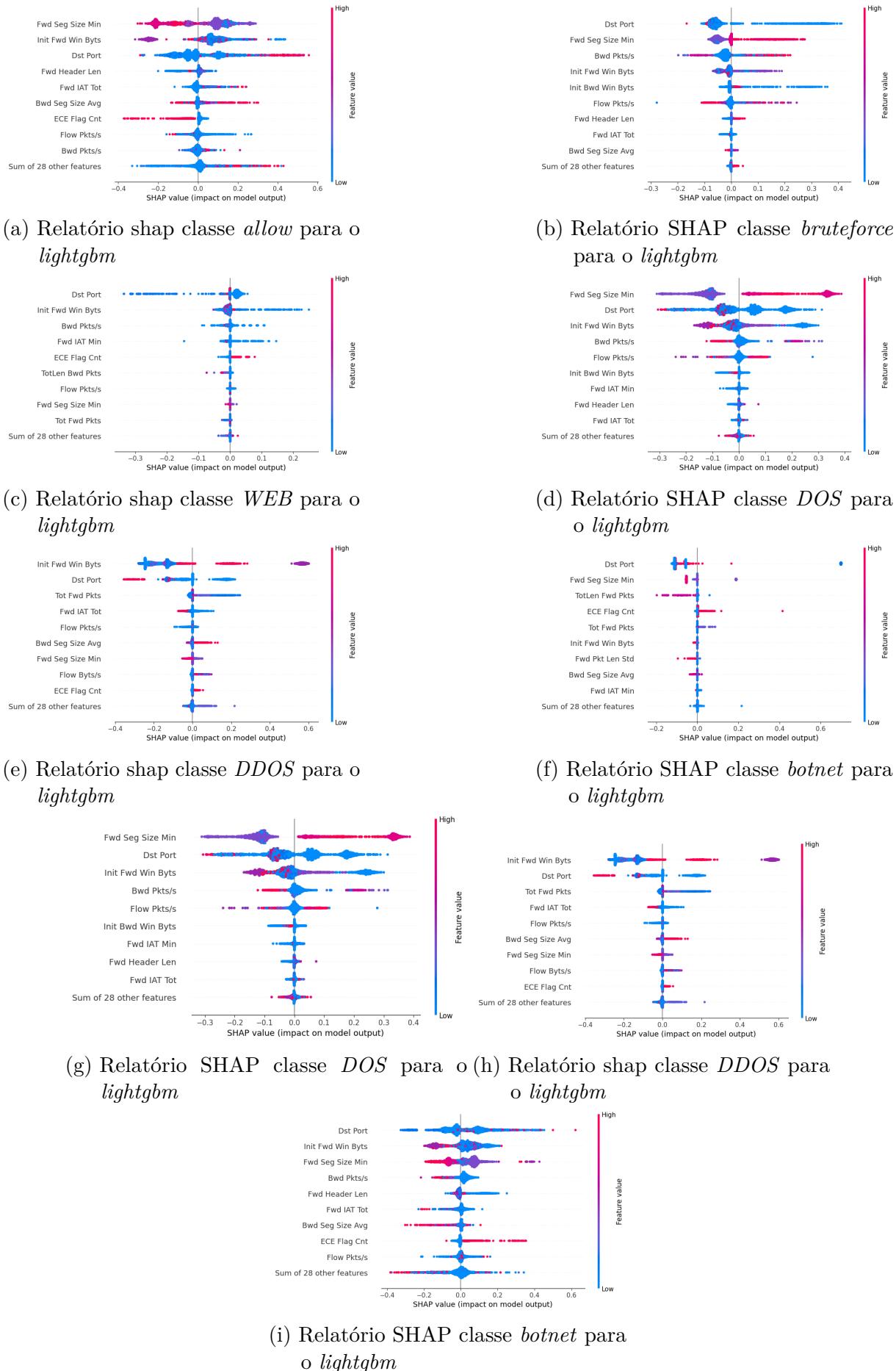


Figura 12 – Comparação dos relatórios SHAP para diferentes classes no *lightgbm* para a base 2

uma tela de login com a lógica de autenticação correspondente ao protocolo *server-side* utilizado (OAuth2), conforme apresentado no caso de uso A.1. Também foram criadas telas específicas para o gerenciamento de usuários e regras críticas, segundo as descrições explicitadas nos casos de uso A.2 e A.4.

Por fim, na fase III, trabalhou-se para finalizar tanto os serviços do lado do servidos quanto as funcionalidades restantes do lado do cliente. Primeiramente, do lado do servidor, criou-se o *PacketSniffer*, que pode ser executado como um processo independente para monitorar o tráfego de rede, e o módulo *PacketSnifferConnector* do serviço principal do lado servidor, que tem por intuito fazer a comunicação interprocesso com o *PacketSniffer* e alimentar os pacotes capturados na placa de rede monitorada ao módulo de IA, a fim de possibilitar a criação de regras dinamicamente.

Além disso, ao avaliar os pacotes de rede, implementou-se, também, a coleta de estatísticas sobre os ataques e o modelo *Ensemble* (quantidades de ataques por tipo, quantidades por horário, quantidade de ataques por porta, estatísticas do treinamento do modelo) dinamicamente, armazenando-as em cache. Em seguida, utilizamos essas informações para criar um dashboard que pode ser mostrado ao usuário no cliente de navegador, conforme mostrado na Figura 22. Imagens das demais telas do cliente podem ser encontradas no Apêndice D.

Por motivos de priorização de outras atividades no período de desenvolvimento do PFC, não foram feitas telas no cliente de navegador referentes aos casos de uso A.3 e A.5. Entretanto, ambos fluxos estão implementados por completo no lado do servidor, e podem ser executados manualmente, seja por reiniciar o serviço ou por chamar os *endpoints* correspondentes. Isso pode ser feito utilizando a documentação interativa, por exemplo, vide Apêndice E.

5 CONCLUSÃO

5.1 Dataset 1

Após a análise das métricas e dos relatórios SHAP, observa-se que o F_1 -score possui valores elevados para as três classes nos algoritmos baseados em árvores. Entretanto, para os demais, ele apresenta um F_1 -score baixo para a classe *deny*. Como dito no capítulo anterior, isso se deve à alta influência da variável *NAT translation destination*.

Portanto, embora o modelo consiga diferenciar bem entre ataque e não-ataque, percebe-se que há um overfitting da base, posto que muito peso é dado para uma única variável, o que indica que ele está bloqueando todos os pacotes que precisam da aplicação do protocolo NAT e não necessariamente é verdade.

5.2 Dataset 2

Observando os resultados descritos na seção anterior, percebe-se que os modelos para essa base são muito mais robustos, uma vez que diferentes variáveis auxiliam na detecção do tipo de ataque. Isso, por sua vez, reduz a possibilidade de overfitting. As variáveis desse caso são resumidas na tabela 8

Termo	Principais fatores de detecção do ataque
DOS	Baixo fluxo de pacotes por segundo na direção contrária alto aliadas ao um grande tamanho mínimo para o pacote no sentido direto
Botnet	Grande quantidade de flags ECE
Bruteforce	Grande fluxo de pacotes por segundo na direção contrária alto aliadas ao um grande tamanho mínimo para o pacote no sentido direto
DDOS	Grande quantidade de bytes enviados na janela inicial na direção direta

Tabela 8 – Ataque e seus principais indicadores

Para a classe *infiltration*, é difícil identificar qual é a principal razão para o ataque, devido aos seus indicadores terem comportamento semelhante ao da classe *allow*. Isso ficou evidente ao observar uma classificação errônea de uma classe na outra, conforme foi discutido na seção de resultados.

Quanto ao ataque web, não foi possível determinar com exatidão os principais fatores que caracterizam esse tipo de ataque, pois há poucos registros que podem ser utilizados para treinamento para esse tipo ataque, o que também justifica uma maior variação dos resultados obtidos pelos modelos.

Ademais, analisando comparativamente os modelos, nota-se que eles apresentam resultados semelhantes, embora alguns modelos identifiquem indicadores de maneira diferenciada para algumas classes sem alterar significativamente o resultado da classificação. A exceção desse fato ocorre para o modelo de regressão logística, no qual é aplicado muito peso para a variável *flag ACK*, reduzindo o seu poder de detecção.

5.2.1 Conclusões finais

Os objetivos propostos inicialmente para o desenvolvimento deste trabalho de fim de curso foram elencados em 1.3: elaborar algoritmos de ML capaz de gerar regras de firewall (estática e dinamicamente), criar *pipelines* automatizados de retreinamento e reavaliação de regras estáticas e implementação de um sistema integrado com firewall, assumindo diversas tarefas administrativas, como autenticação e envio de notificações. O primeiro objetivo foi atingido em sua totalidade, pois foi entregue um modelo ensemble treinado a partir de bases públicas que é capaz de, com grande acurácia, analisar pacotes e criar regras de firewall (em particular, fez-se a implementação para o firewall *open source* iptables). O segundo objetivo também foi alcançado inteiramente, pois, dentro das funções administrativas do sistema, está a capacidade de gerenciar diversos tipos distintos de tarefas de inicialização síncronas e assíncronas e tarefas periódicas. Portanto, configurando as *cronstrings* de um dado padrão de repetição, criaram-se tarefas periódicas de retreinamento e recriação das regras estáticas. O terceiro objetivo não foi totalmente completo, pois não foi feita uma integração com dashboards comerciais por motivos de priorização, nem foram implementadas as interfaces gráficas referentes às mudanças de configuração em tempo de execução e à mudança do modelo carregado para um previamente salvo, ainda que ambas funcionalidades estejam implementadas e possam ser utilizadas pela documentação interativa da API no Swagger. No geral, obteve-se uma ótima prova de conceito para um IPS baseado em técnicas de ML.

Em relação aos trabalhos que serviram como revisão bibliográfica e estudo prévio das técnicas já utilizadas, após a realização do trabalho, não foi possível replicar os resultados encontrados por (4) na base 1, uma vez que ocorreu um overfitting que levou à classificação incorreta entre as classes deny e drop. Entretanto, no que se refere à segunda base de dados utilizada neste projeto, foi possível replicar os resultados encontrados por (6), visto que foi feita a classificação com precisão de todos os tipos de ataques, à exceção de *infiltration* e *web attack*.

Por fim, percebe-se que, embora os modelos para as duas bases sejam o mesmo, encontrou-se um resultado muito superior e com menos overfitting para a segunda base, o que indica que em muitas situações uma base de dados mais bem construída, contribui mais do que um modelo super poderoso

Quanto ao onboarding do projeto foram enviados o repositório do github com

todo o código desenvolvido no período, além de 3 vídeos, o primeiro com um overview do código e dos seus principais componentes, o segundo com a demonstração do treinamento e criação de regras e por fim um video com o overview do front-end da aplicação.

REFERÊNCIAS

- 1 VERBRUGGEN, R. *Creating firewall rules with machine learning techniques*.
- 2 ALJABRI, M.; ALAHMADI, A. A.; MOHAMMAD, R. M. A.; ABOULNOUR, M.; ALOMARI, D. M.; ALMOTIRI, S. H. Classification of firewall log data using multiclass machine learning models. *Department of Computer Science, College of Computer and Information Systems, Umm Al-Qura University, Makkah 21955, Saudi Arabia*, June 2022. Correspondence: msaljabri@iau.edu.sa or mssjabri@uqu.edu.sa.
- 3 JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. *Machine learning and deep learning*. 2021.
- 4 PATIL, S. *Routing network traffic based on firewall logs using Machine Learning*. 2021. Disponível em: <https://medium.com/@shubham_patil/routing-network-traffic-based-on-firewall-logs-using-machine-learning-1234567890>.
- 5 INTERNET Firewall Data. <<https://archive.ics.uci.edu/dataset/542/internet+firewall+data>>. Último acesso em 22/05/2024.
- 6 AHMED, M. *DDQL RL Multi-Classification*. 2024. <<https://www.kaggle.com/code/mohamedahmedae/ddql-rl-multi-classfication>>. Acessado em: 29/07/2024.
- 7 CHESWICK, W. R.; BELLOVIN, S. M.; RUBIN, A. D. *Firewalls and Internet Security: Repelling the Wily Hacker*. 2. ed. USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 020163466X.
- 8 TONGAONKAR, A.; INAMDAR, N.; SEKAR, R. Inferring higher level policies from firewall rules. In: *LISA*. [S.l.: s.n.], 2007. v. 7, p. 1–10.
- 9 PATEL, M. *Demilitarized zone: An exceptional layer of network security to mitigate DDoS attack*. Dissertação (Mestrado) — University of Windsor (Canada), 2020.
- 10 EBNER, J. *Cross Validation, Explained*. 2023. Accessed: 21/05/2024. Disponível em: <<https://www.sharpsightlabs.com/blog/cross-validation-explained/>>.
- 11 SHARMA, H.; RIZVI, M. A. Prediction of heart disease using machine learning algorithms: A survey. *Department of Computer Engineering and Applications, National Institute of Technical Teachers' Training and Research*, 2018.
- 12 FEATHERS, W. *Random Forest vs Gradient Boosted Trees: A Comparison*. 2023. Medium.
- 13 LI, S.; DONG, X.; MA, D.; DANG, B.; ZANG, H.; GONG, Y. Utilizing the lightgbm algorithm for operator user credit assessment research. In: HUACONG QINGJIAO INFORMATION TECHNOLOGY (BEIJING) CO., LTD. *Proceedings of the Conference on Machine Learning and Applications*. Beijing, China: IEEE, 2023.
1. Computer Technology, Huacong Qingjiao Information Technology (Beijing) Co., Ltd., Beijing, China.
2. Management Information Systems, University of Maine at Presque Isle, Presque Isle,

US.

3. Computer Science, Stevens Institute of Technology, New Jersey, US.

4. Computer Science, San Francisco Bay University, Fremont CA, US.

5. Physics and Mathematics, Universitario Tecnologico Universitam, Tijuana, Mexico.

6. Computer and Information Technology, Northern Arizona University, Flagstaff, US.

*Corresponding author: Shaojie Li, lishaojie@tsingj.com.

14 KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In: MICROSOFT RESEARCH. *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Long Beach, CA, USA: Curran Associates, Inc., 2017. p. 3146–3154.

1. Microsoft Research: {guolin.ke, taifengw, wche, weima, qiwye, tie-yan.liu}@microsoft.com

2. Peking University: qimeng13@pku.edu.cn

3. Microsoft Redmond: tfinely@microsoft.com.

15 CILIMKOVIC, M. Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, v. 15, n. 1, 2015.

16 HAMILTON, R. I.; PAPADOPOULOS, P. N. *Using SHAP Values and Machine Learning to Understand Trends in the Transient Stability Limit*.

17 BRUNSWICK, U. of N. *IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018)*. <<https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv>>. Acessado em: 29/07/2024.

18 BRUNSWICK, U. of N. *IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018)*. <<https://www.unb.ca/cic/datasets/ids-2018.html>>. Acessado em: 29/07/2024.

APÊNDICE A – CASOS DE USO

A.1 Login

- **Autor primário:** analista de segurança da informação.
- **Descrição:** este caso descreve o processo para acessar o sistema.
- **Pré-condições:**
 1. O usuário deve estar registrado no sistema.
- **Fluxo principal:**
 1. Usuário acessa página de login do sistema.
 2. A página exibe os campos "Usuário" e "Senha" e botões "Entrar" e "Esqueci minha senha".
 3. O usuário insere seu nome de usuário ou e-mail e sua senha.
 4. O usuário clica no botão "Entrar".
 5. Se as informações estiverem corretas, o usuário é redirecionado para a página inicial.
- **Fluxos alternativos:**
 1. Se o usuário esqueceu a senha, ele pode clicar em "Esqueci minha senha" para receber instruções de recuperação.
 2. Se as informações do usuário estiverem incorretas e o número de tentativas não ultrapassar o máximo estipulado, retorna-se ao passo 2.
 3. Se as informações estiverem incorretas e o número de tentativas ultrapassar o máximo estipulado, bloqueia-se o usuário, enviando e-mails comunicando a eles e a administradores responsáveis sobre o ocorrido.
- **Pós-condições:**
 1. O usuário está autenticado e tem acesso aos recursos permitidos à sua conta.

A.2 Gerenciamento de Usuários

- **Autor primário:** administrador do sistema.

- **Descrição:** este caso descreve o cadastro de um usuário, especificando os seus acessos.
- **Pré-condições:**
 1. O usuário deve ter privilégios administrativos.
- **Fluxo principal:**
 1. O sistema exibe uma interface com um botão de "Criar" e uma tabela listando os usuários cadastrados no sistema, onde cada linha contém botões de "Atualizar" e "Remover".
 2. Se o administrador clica no botão de "Criar", um modal é exibido para que ele preencha os dados do usuário a ser cadastrado. Ele então preenche os campos e clica em "Criar" para submeter a novo usuário.
 3. Se o administrador clica no botão "Atualizar" em uma linha específica, um modal é exibido com as informações desse usuário já preenchidas. Ele então modifica as informações conforme necessário e clica em "Atualizar" para submeter as mudanças.
 4. Se o administrador clica em "Remover" em uma linha específica, um modal é exibido solicitando a confirmação da remoção. O usuário confirma a remoção do cadastro.
- **Fluxos alternativos:**
 1. Se o administrador não confirma a remoção, o cadastro não é removido e nenhuma ação adicional é realizada.
- **Pós-condições:**
 1. Um usuário é cadastrado, atualizado ou removido do sistema, conforme solicitado pelo administrador.

A.3 Gerenciamento do Modelo de IA

- **Autor primário:** analista de segurança da informação.
- **Descrição:** este caso descreve o processo para gerenciar o modelo de IA utilizado pelo sistema, incluindo a visualização de métricas do sistema carregado, o carregamento de um modelo antigo ou solicitação de treinamento ou retreinamento.
- **Pré-condições:**

1. O usuário deve ter acesso à visualização de métricas do modelo de IA carregado no sistema.
2. O usuário deve ter acesso ao carregamento de modelos antigos.
3. O usuário deve ter acesso a solicitações de treinamento do modelo de IA do sistema.

- **Fluxo principal:**

1. O sistema exibe uma interface contendo as métricas do atual modelo de IA carregado no sistema e botões de "Carregar" e "Treinar".
2. Se o usuário clicar em "Carregar", um modal é exibido mostrando os modelos treinados e salvos para serem carregados. O usuário seleciona um desses modelos e clica em "Carregar", sobrescrevendo o modelo carregado no sistema (se houver) pelo selecionado.
3. Se o usuário clicar em "Treinar", é solicitado um treinamento ou retreinamento do modelo de IA. Quando o treinamento for finalizado, ele é automaticamente salvo e carregado no sistema.

- **Fluxos alternativos:**

1. Se não há nenhum modelo de IA carregado no sistema, é mostrada uma mensagem solicitando ao usuário que ele treine um modelo ou que ele carregue um modelo treinado anteriormente.
2. Se já tiver sido solicitado um treinamento de modelo de IA, isso é explicitado na interface e não é permitido clicar em "Carregar" ou "Treinar" até esse treinamento estar finalizado.
3. Se houver um erro no treinamento, ele é mostrado na tela.
4. Se o usuário clicar em "Carregar" e não houver nenhum modelo anteriormente treinado e salvo, é mostrada uma mensagem informando isso ao usuário.

- **Pós-condições:**

1. Um modelo de IA foi treinado ou carregado no sistema, conforme solicitado pelo usuário.

A.4 Gerenciamento de regras críticas

- **Autor primário:** analista de segurança da informação.
- **Descrição:** este caso descreve o processo para gerenciar (visualizar, cadastrar, atualizar e remover) regras críticas.

- **Pré-condições:**

1. O usuário deve ter acesso de leitura e escrita às regras críticas.

- **Fluxo principal:**

1. O sistema exibe uma interface com um botão de "Criar" e uma tabela listando as regras críticas cadastradas, onde cada linha contém botões de "Atualizar" e "Remover".
2. Se o usuário clica no botão de "Criar", um modal é exibido para que ele preencha os campos da regra crítica (portas, endereços, tempos de início e fim, e prioridade). O usuário preenche os campos e clica em "Criar" para submeter a nova regra.
3. Se o usuário clica no botão "Atualizar" em uma linha específica, um modal é exibido com as informações da regra preenchidas. O usuário modifica as informações conforme necessário e clica em "Atualizar" para submeter as mudanças.
4. Se o usuário clica em "Remover" em uma linha específica, um modal é exibido solicitando a confirmação da remoção. O usuário confirma a remoção da regra.

- **Fluxos alternativos:**

1. Se as datas de início e fim são fornecidas e a data de início é maior que a data de fim, o sistema não permite a criação ou atualização da regra e solicita a correção das informações, retornando ao passo correspondente (2 ou 3).
2. Se o usuário não confirma a remoção, a regra não é removida e nenhuma ação adicional é realizada.

- **Pós-condições:**

1. Uma regra crítica é cadastrada, atualizada ou removida do sistema conforme solicitado pelo usuário.

A.5 Alteração de Configurações do Sistema

- **Autor primário:** analista de segurança da informação

- **Descrição:** modificar, em tempo de execução, configurações do sistema, sem a necessidade de reiniciá-lo.

- **Pré-condições:**

1. O usuário deve ter acesso de leitura e escrita às configurações do sistema.

- **Fluxo principal:**

1. O sistema exibe uma interface contendo as configurações atuais do sistema e um botão “Atualizar”.
2. O usuário pode modificar as configurações conforme necessário e clica em “Atualizar” para submeter as mudanças.
3. Um modal é exibido solicitando a confirmação da atualização. Além disso, o usuário é lembrado que essas alterações não serão persistidas automaticamente e que, se ele quiser que elas o sejam, ele também deve aplicá-las ao arquivo de configuração correspondente.

- **Fluxos alternativos:**

1. Se o usuário utilizar uma formatação errada na configuração, é mostrada uma mensagem de erro indicando o erro de formatação específico, e a atualização não é submetida.
2. Se o usuário não confirma a atualização, a atualização não é feita e nenhuma ação adicional é realizada.

- **Pós-condições:**

1. Configurações do sistema são alteradas em tempo de execução, conforme solicitado pelo usuário.

APÊNDICE B – RESULTADOS PARA O DATASET 1

	Allow	Deny	Drop	Precisão
Allow	9947	3	0	1.00000000
Deny	0	7248	0	1.00000
Drop	0	2694	9958	0.78706924
Recall	0.99969849	0.72880845	1.00000	
F1 Score	0.99984922	0.84313383	0.88084918	
Accuracy			0.909648241	

Tabela 9 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o gradient boost para a base 1

	Allow	Deny	Drop	Precisão
Allow	9820	8	0	0.999186
Deny	32	1093	0	0.97155556
Drop	95	8844	9958	0.52696195
Recall	0.98723233	0.10990447	1.00000	
F1 Score	0.9931732	0.19747064	0.69020967	
Accuracy			0.69919598	

Tabela 10 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o logistic regression para a base 1

	Allow	Deny	Drop	Precisão
Allow	9945	2	0	0.99979893
Deny	2	7213	0	0.99971965
Drop	0	2811	9958	0.77985747
Recall	0.99979893	0.71714429	1.0000000	
F1 Score	0.99979893	0.8351777	0.87631452	
Accuracy			0.859467825	

Tabela 11 – Matriz de Confusão com F1 Score, Recall, Precisão e Acurácia para o random forest para a base 1

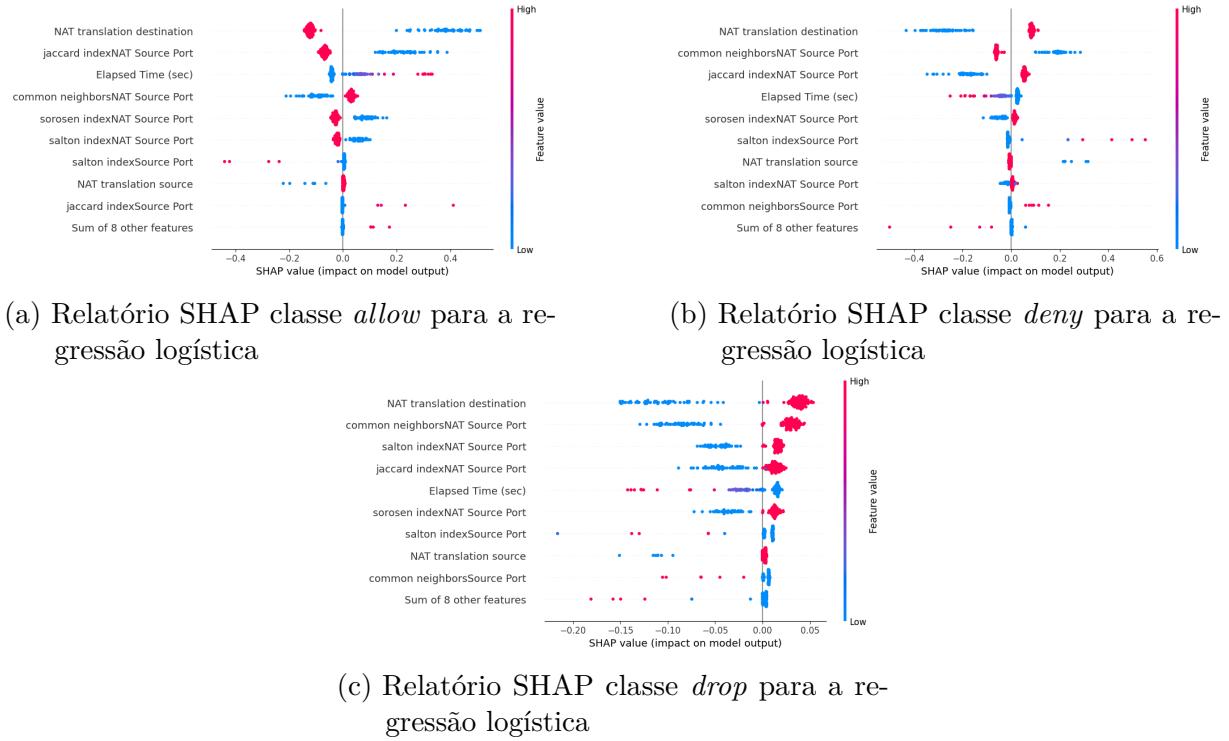


Figura 13 – Comparação dos relatórios SHAP para diferentes classes na regressão logística para a base 1

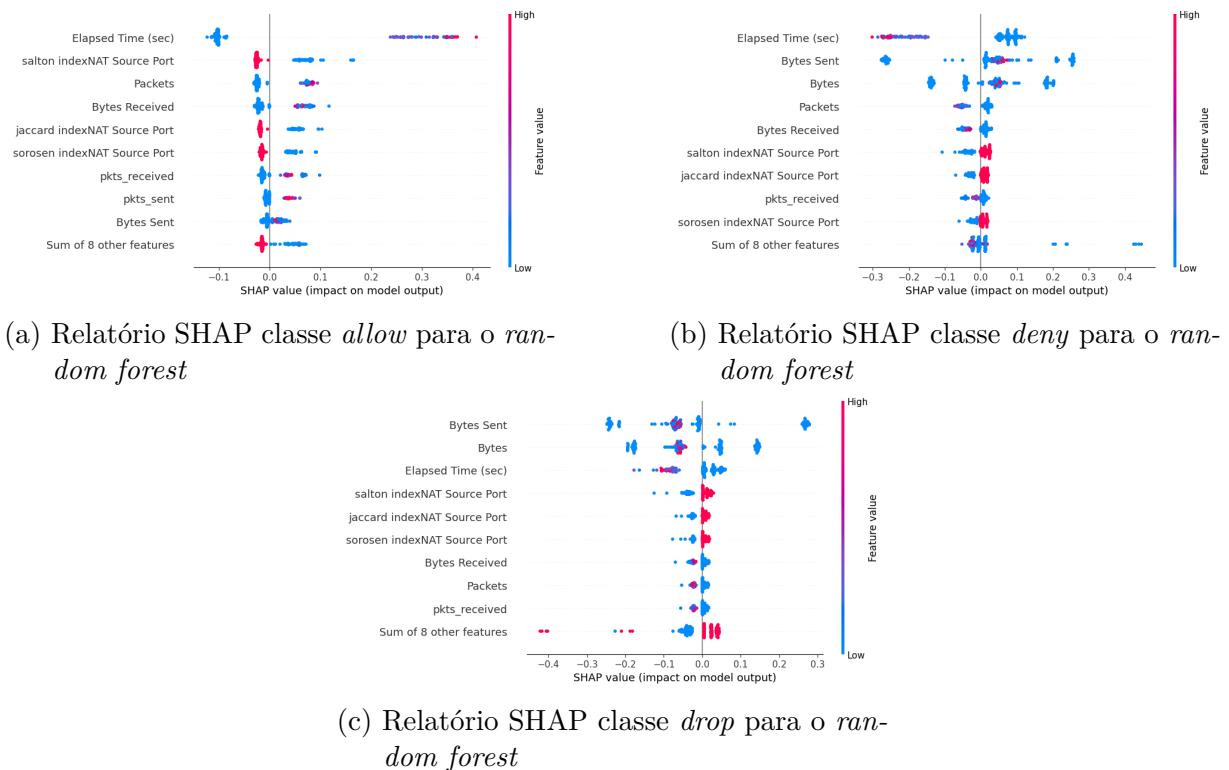


Figura 14 – Comparação dos relatórios SHAP para diferentes classes no *random forest* para a base 1

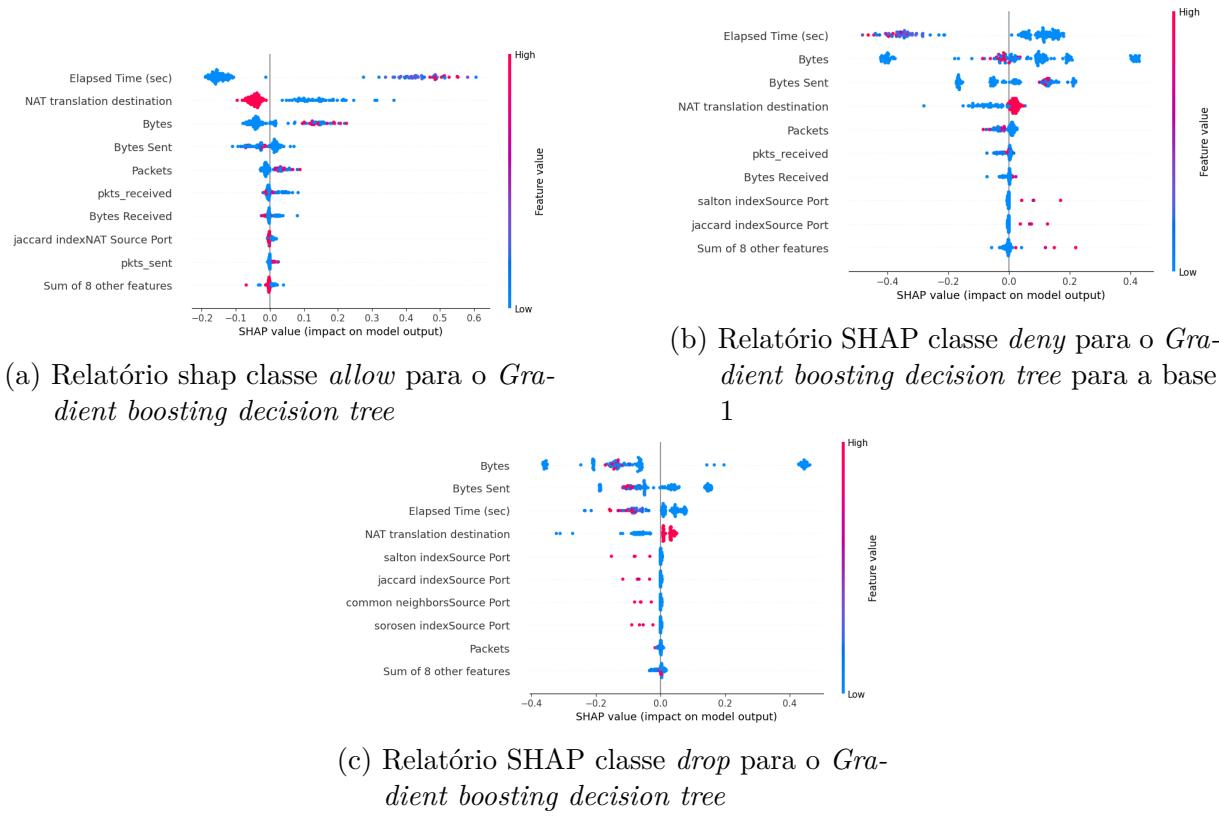


Figura 15 – Comparação dos relatórios SHAP para diferentes classes no *Gradient boosting decision tree*

APÊNDICE C – RESULTADOS PARA O DATASET 2

	allow	bruteforce	WEB	DOS	DDOS	botnet	infiltration	Precisão
allow	1039608	0	132	0	11	55	98229	0.91
bruteforce	23	341366	0	65258	0	0	4	0.84
WEB	238	0	469	44	3	7	29	0.69
DOS	299	20322	0	545821	0	0	39	0.93
DDOS	5	0	0	0	651656	0	1	1.00
botnet	52	0	0	0	0	271315	1	1.00
infiltration	15425	0	0	0	0	0	35634	0.70
Recall	0.98	0.94	0.79	0.89	1.00	1.00	0.27	
F1 Score	0.95	0.89	0.69	0.93	1.00	1.00	0.39	
Accuracy					0.94			

Tabela 12 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para o gradient boost decision tree

	allow	bruteforce	WEB	DOS	DDOS	botnet	infiltration	Precisão
allow	457617	0	0	245	19	0	17089	0.96
bruteforce	4679	340934	5	121539	0	0	372	0.73
WEB	78219	0	620	0	1	222	11431	0.01
DOS	2807	20751	1	489094	0	0	387	0.95
DDOS	4382	0	0	0	651662	0	452	0.99
botnet	22897	0	0	0	0	268283	3718	0.91
infiltration	465689	0	0	217	0	2724	100367	0.18
Recall	0.45	0.94	0.99	0.80	1.00	0.99	0.75	
F1 Score	0.61	0.82	0.01	0.87	1.00	0.95	0.29	
Accuracy					0.94			

Tabela 13 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para regressão logistica na base 2

	allow	bruteforce	WEB	DOS	DDOS	botnet	infiltration	Precisão
allow	1035456	0	355	30	14	67	19309	0.91
bruteforce	105	340507	0	69159	0	0	20	0.83
WEB	34	0	246	0	0	0	32	0.79
DOS	403	21182	0	541912	0	0	174	0.96
DDOS	26	0	0	0	651656	0	20	1.00
botnet	397	0	0	0	0	271283	10	1.00
infiltration	19309	0	33	0	0	26	37264	0.66
Recall	0.98	0.94	0.39	0.89	1.00	1.00	0.28	
F1 Score	0.95	0.88	0.52	0.92	1.00	1.00	0.39	
Accuracy						0.94		

Tabela 14 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para Multi layer perceptron na base 2

	allow	bruteforce	WEB	DOS	DDOS	botnet	infiltration	Precisão
allow	997867	5	243	3	10	110	85241	0.92
bruteforce	28	345918	0	70666	0	0	2	0.83
WEB	19	0	402	0	0	0	2	0.95
DOS	29	15763	0	539644	0	0	2	0.97
DDOS	0	0	0	0	651669	0	0	1.00
botnet	14	0	0	0	0	271123	2	1.00
infiltration	57954	0	0	0	0	0	48680	0.46
Recall	0.95	0.96	0.62	0.88	1.00	1.00	0.36	
F1 Score	0.93	0.89	0.75	0.93	1.00	1.00	0.4	
Accuracy						0.94		

Tabela 15 – Matriz de Confusão com Precisão, Recall, F1 Score e Acurácia para o random forest na base 2

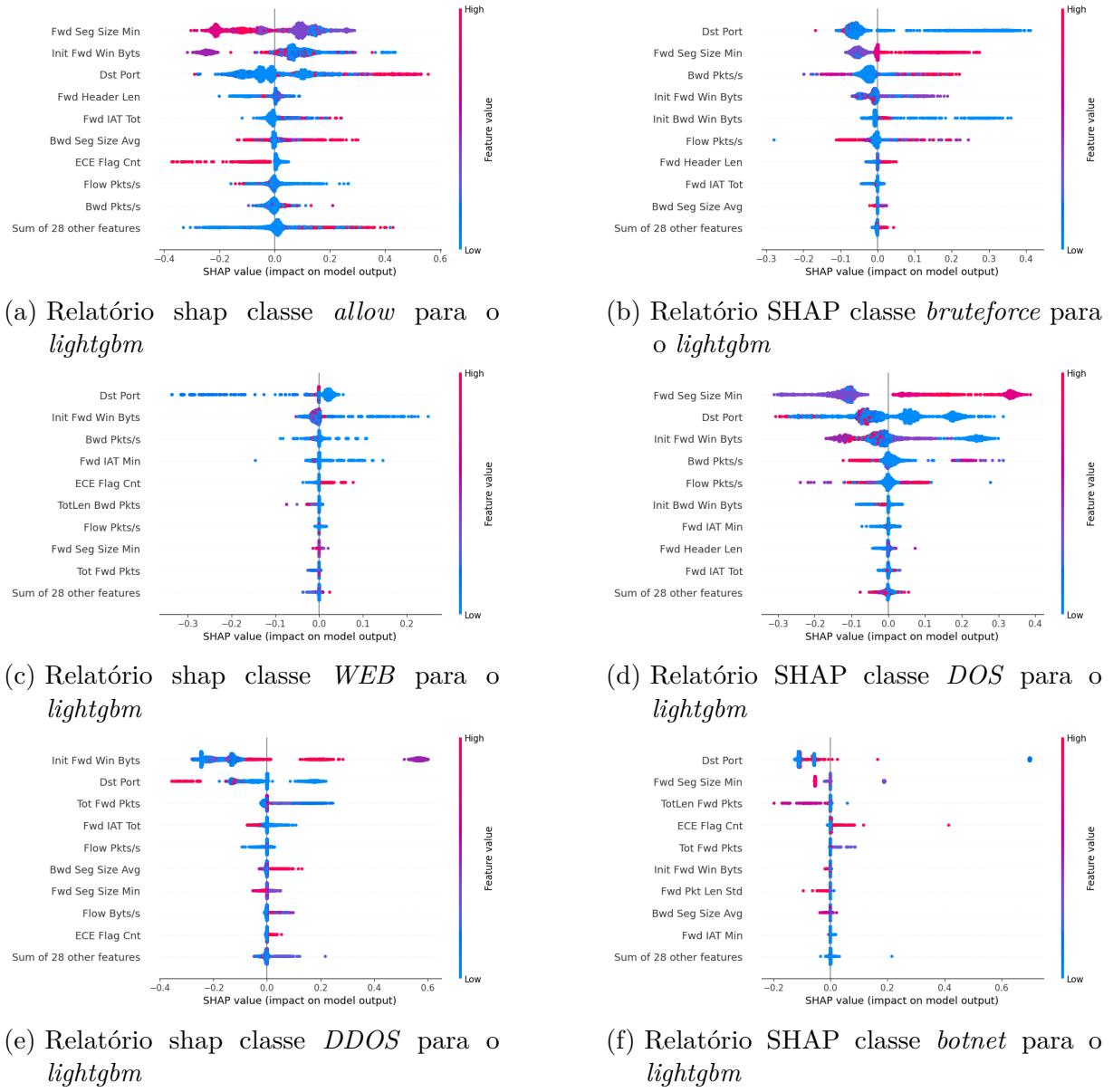


Figura 16 – Comparação dos relatórios SHAP para diferentes classes no *lightgbm* para a base 2

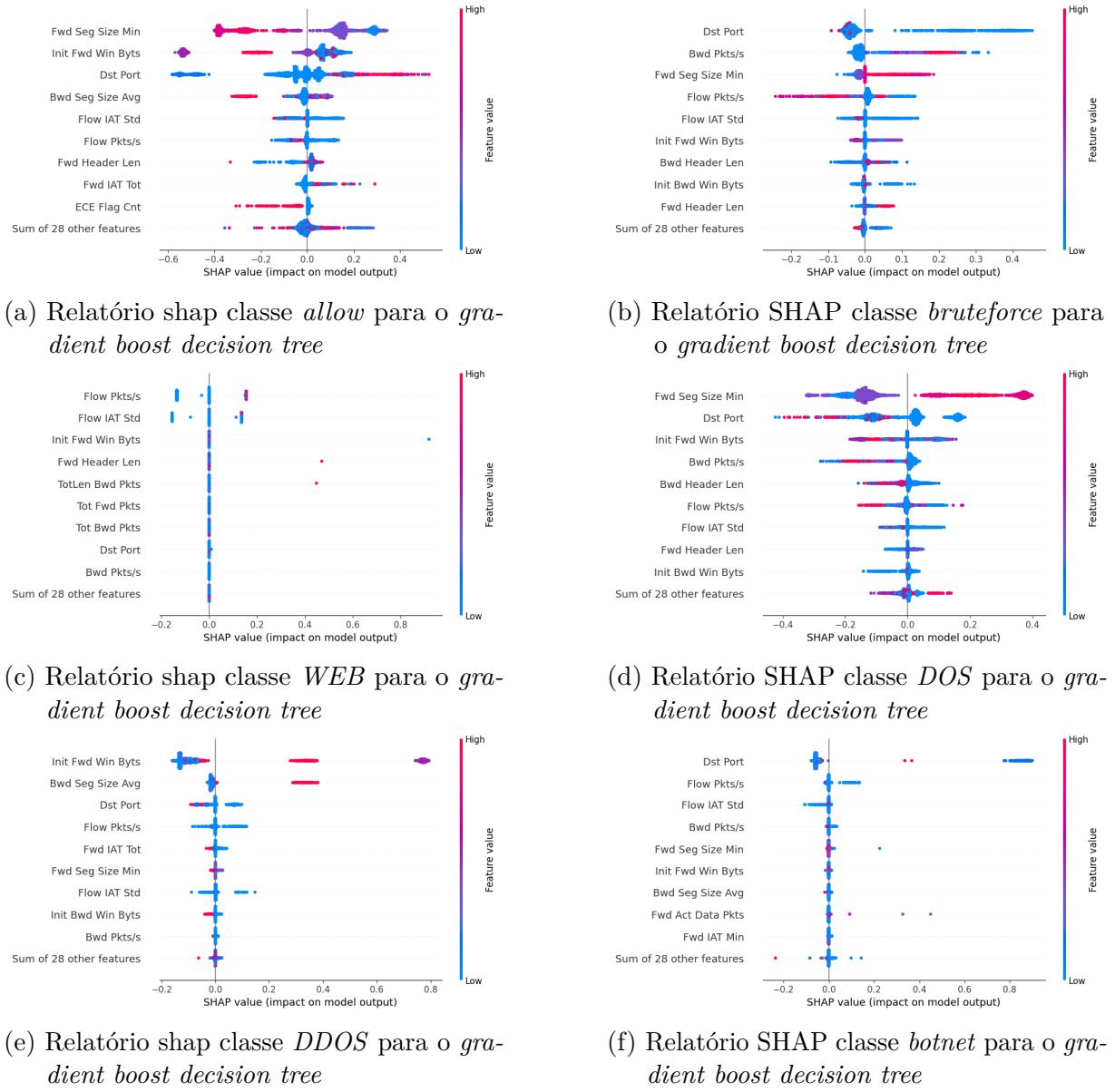


Figura 17 – Comparaçao dos relatórios SHAP para diferentes classes no *gradient boost decision tree* para a base 2

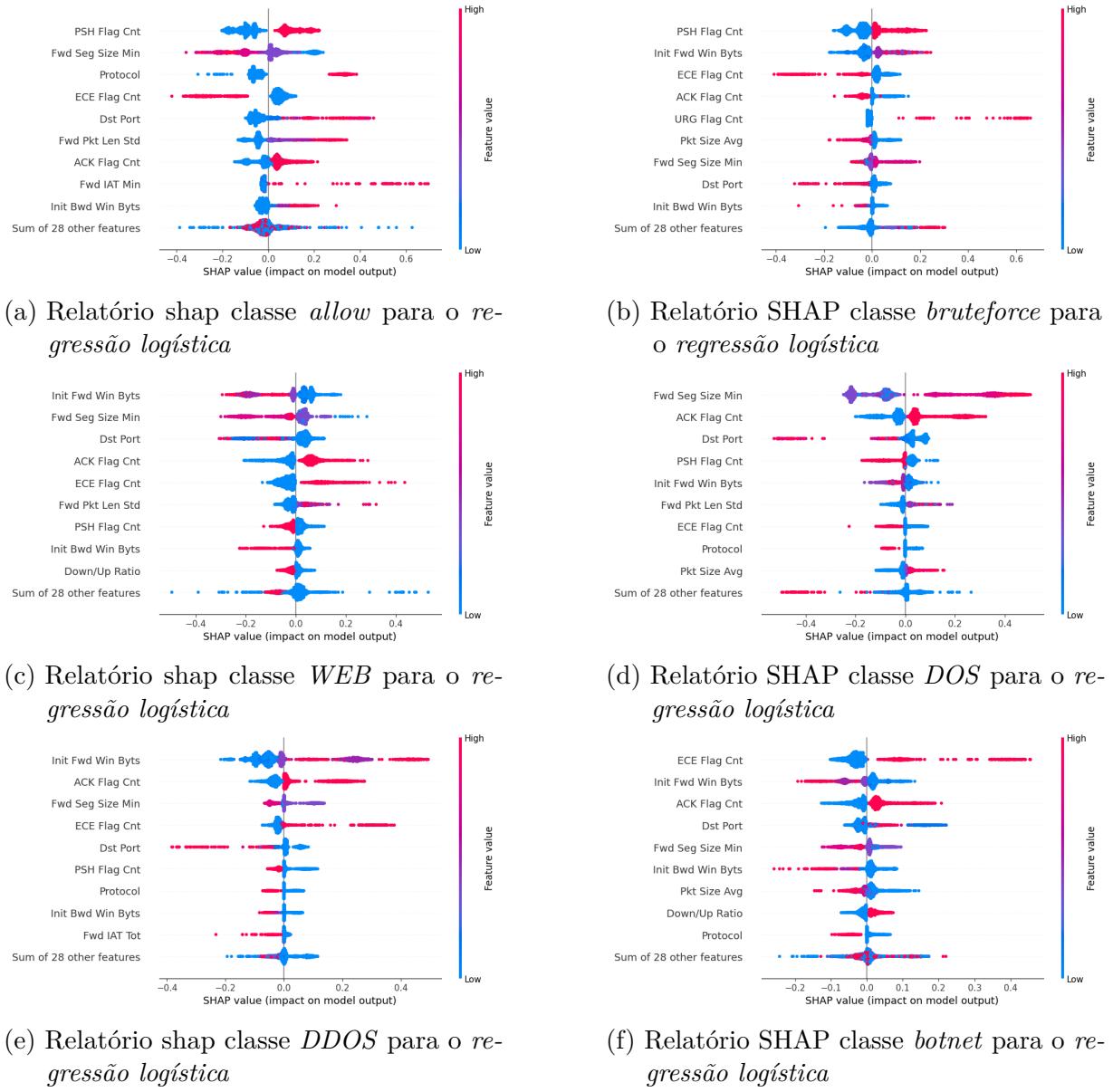


Figura 18 – Comparação dos relatórios SHAP para diferentes classes no regressão logística para a base 2

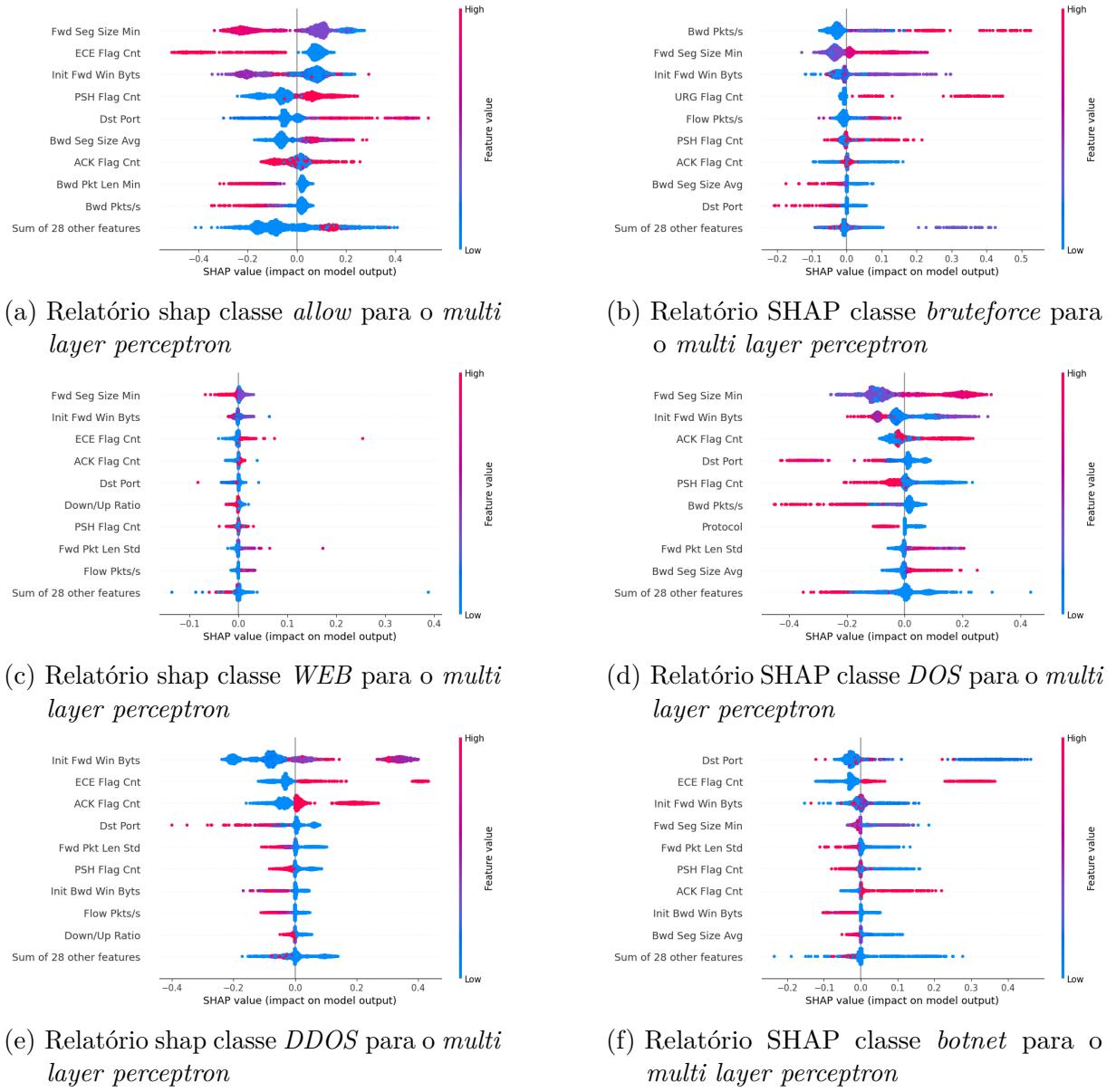


Figura 19 – Comparação dos relatórios SHAP para diferentes classes no *multi layer perceptron* para a base 2

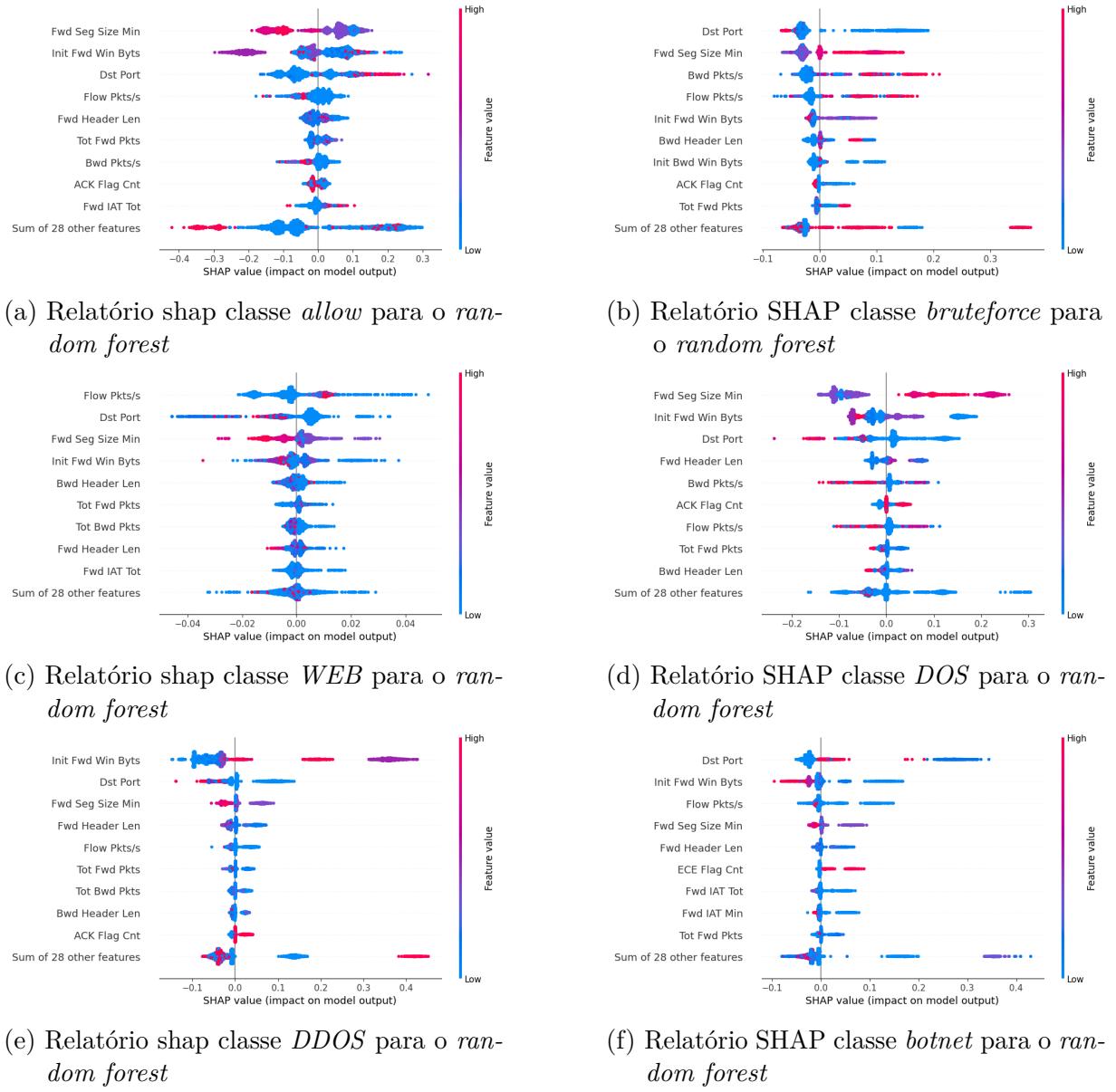


Figura 20 – Comparação dos relatórios SHAP para diferentes classes no *random forest* para a base 2

APÊNDICE D – TELAS DO CLIENTE DE NAVEGADOR

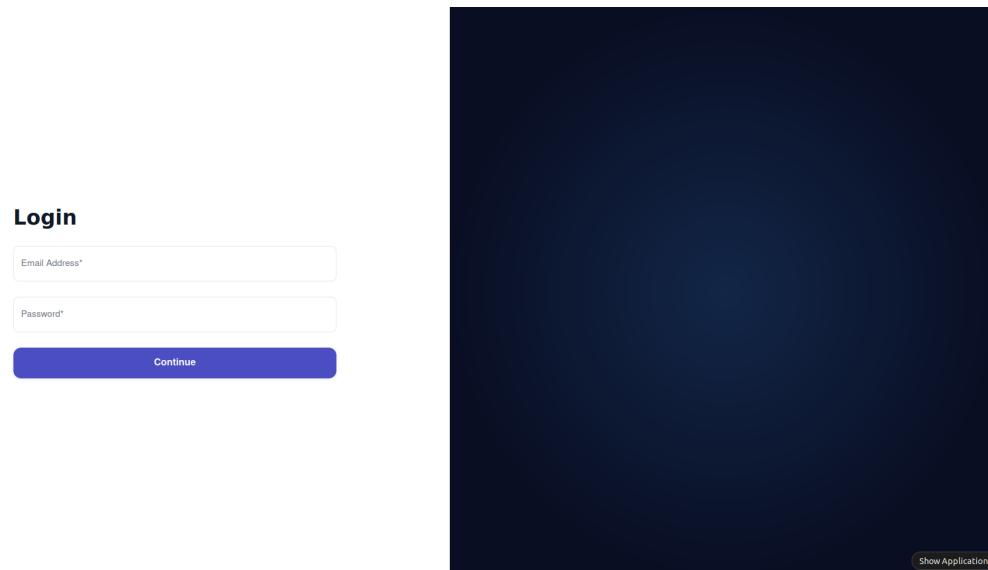


Figura 21 – Tela de Login

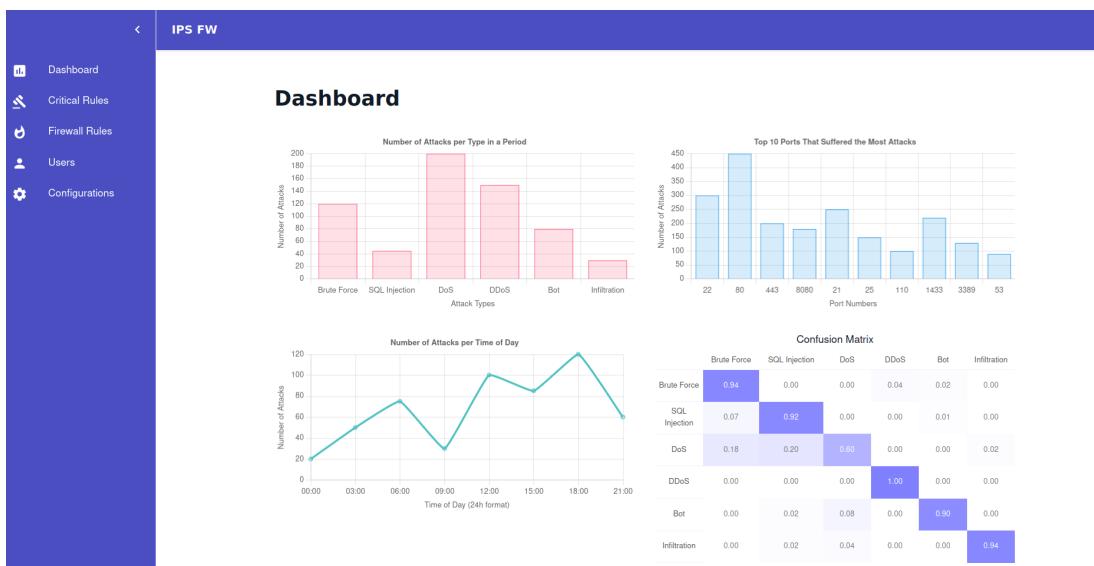


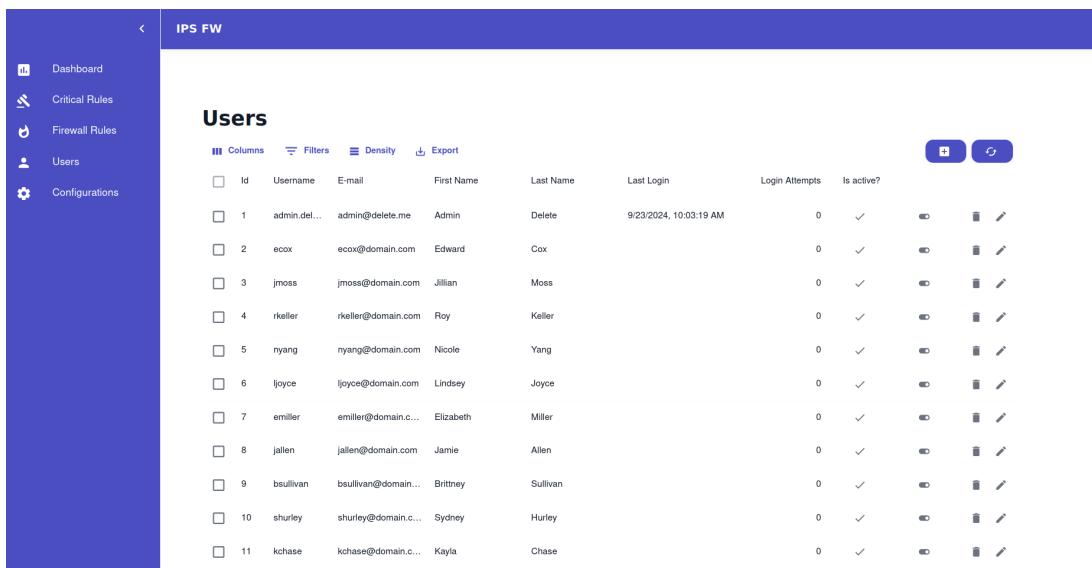
Figura 22 – Dashboard do cliente

Id	Title	Action	Protocol	Dest Port	Last Update
1	Block from 21.188.28.53 to 218.103.121.206	block	udp		9/23/2024, 1:02:37 PM
2	Allow from 198.20.172.124 to 222.104.124...	allow	tcp		9/23/2024, 1:02:37 PM
3	Block from 71.145.111.47	block			9/23/2024, 1:02:37 PM
4	Allow from 15.26.92.211:36688 to 83.103....	allow	udp		9/23/2024, 1:02:37 PM
5	Block from 38.170.104.191 to 41803	block		41803	9/23/2024, 1:02:37 PM
6	Block from 219.173.196.68 to 66.127.7.244	block	tcp		9/23/2024, 1:02:37 PM
7	Block from 210.91.69.192	block			9/23/2024, 1:02:37 PM
8	Allow	allow			9/23/2024, 1:02:37 PM
9	Block	block	udp		9/23/2024, 1:02:37 PM
10	Block	block			9/23/2024, 1:02:37 PM
11	Block from 69.235.30.137 to 200.77.117.1...	block	tcp	61938	9/23/2024, 1:02:37 PM

Figura 23 – Gerenciamento de Regras Críticas

Id	Action	Protocol	Src Address	Src Port	Dest Port	Last Update
101	allow	udp				9/23/2024, 1:10:16 PM
102	allow	tcp				9/23/2024, 1:10:16 PM
103	block	udp		14539		9/23/2024, 1:10:16 PM
104	allow			55190		9/23/2024, 1:10:17 PM
105	allow	udp		19205		9/23/2024, 1:10:17 PM
106	allow	udp		13648		9/23/2024, 1:10:17 PM
107	block	udp		6204		9/23/2024, 1:10:17 PM
108	allow	udp				9/23/2024, 1:10:17 PM
109	allow					9/23/2024, 1:10:17 PM
110	block	udp		52369		9/23/2024, 1:10:17 PM
111	allow	tcp				9/23/2024, 1:10:17 PM

Figura 24 – Tela de regras de firewall



The screenshot shows the 'Users' section of the IPS FW web interface. The left sidebar includes links for Dashboard, Critical Rules, Firewall Rules, Users (selected), and Configurations. The main area has a blue header bar with the text 'IPS FW'. Below it, the title 'Users' is centered above a table. The table has columns: Id, Username, E-mail, First Name, Last Name, Last Login, Login Attempts, and Is active?. The table contains 11 rows of user data. At the bottom right of the table are two buttons: a blue one with a magnifying glass icon and a white one with a refresh symbol.

	Id	Username	E-mail	First Name	Last Name	Last Login	Login Attempts	Is active?		
	1	admin.del...	admin@delete.me	Admin	Delete	9/23/2024, 10:03:19 AM	0	✓		
	2	ecox	ecox@domain.com	Edward	Cox		0	✓		
	3	jmoss	jmoss@domain.com	Jillian	Moss		0	✓		
	4	rkeller	rkeller@domain.com	Roy	Keller		0	✓		
	5	nyang	nyang@domain.com	Nicole	Yang		0	✓		
	6	ljoyce	ljoyce@domain.com	Lindsey	Joyce		0	✓		
	7	emiller	emiller@domain.c...	Elizabeth	Miler		0	✓		
	8	jallen	jallen@domain.com	Jamie	Allen		0	✓		
	9	bsullivan	bsullivan@domain.c...	Brittney	Sullivan		0	✓		
	10	shurley	shurley@domain.c...	Sydney	Hurley		0	✓		
	11	kchase	kchase@domain.c...	Kayla	Chase		0	✓		

Figura 25 – Tela de usuários

APÊNDICE E – API: DOCUMENTAÇÃO SWAGGER

Neste apêndice, mostramos os *endpoints* disponíveis na API do serviço do lado servidor. A documentação interativa completa, aqui exibida, pode ser encontrada no endpoint `/docs#/` do endereço/porta onde está sendo executado o serviço.

Firewall Rule Prediction API 1.0.0 OAS 3.1

[/openapi.json](#)

[Authorize](#)

Users

- POST** /users/login Login
- GET** /users/ Get All
- POST** /users/ Create
- GET** /users/me Get User Me
- GET** /users/{id} Get One
- PUT** /users/{id} Update
- DELETE** /users/{id} Delete One
- PUT** /users/{id}/toggle Toggle Active

Critical Rules

- GET** /critical-rules/ Get All
- POST** /critical-rules/ Create
- DELETE** /critical-rules/ Delete Multiple
- GET** /critical-rules/{id} Get One
- PUT** /critical-rules/{id} Update
- DELETE** /critical-rules/{id} Delete One

Figura 26 – Swagger: usuários e regras críticas

Firewall Rules

- GET /firewall-rules/ Get All
- GET /firewall-rules/{id} Get By Id
- POST /firewall-rules/train Train Ensemble
- POST /firewall-rules/create_static_rules Train Ensemble

Development

- POST /dev/mock Mock Data

Ensemble

- POST /firewall-rules/train Train Ensemble
- POST /firewall-rules/create_static_rules Train Ensemble

default

- GET / Root

Figura 27 – Swagger: outros endpoints

Schemas

- Action > Expand all string
- Body_login_users_login_post > Expand all object
- CriticalRuleCreateModel > Expand all object
- CriticalRuleOutModel > Expand all object
- CriticalRuleUpdateModel > Expand all object
- FirewallRuleOutModel > Expand all object
- GetAllCriticalRules > Expand all object
- GetAllFirewallRules > Expand all object
- GetAllUsers > Expand all object
- HTTPValidationError > Expand all object
- MockConfig > Expand all object
- User > Expand all object
- UserCreateModel > Expand all object
- UserOutModel > Expand all object
- UserUpdateModel > Expand all object
- ValidationError > Expand all object

Figura 28 – Swagger: modelos