UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Marcos Rogério Silveira

Abordagem Semi-Supervisionada para detecção de Domínios Maliciosos em TLDs em sua primeira consulta

São José do Rio Preto
2024

Marcos Rogério Silveira

Abordagem Semi-Supervisionada para detecção de Domínios Maliciosos em TLDs em sua primeira consulta

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiador: NIC.br - Proc. 2764/2018

Orientador:

Prof. Dr. Adriano Mauro Cansian

São José do Rio Preto
2024

S587a

Silveira, Marcos Rogério
Abordagem Semi-Supervisionada para detecção de Domínios Maliciosos em TLDs em sua primeira consulta / Marcos Rogério Silveira. -- São José do Rio Preto, 2024
115 p. : il., tabs.

Tese (doutorado) - Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientador: Adriano Mauro Cansian

1. Ciência da computação. 2. Computadores Medidas de segurança. 3. Crime por computador. 4. Aprendizado de máquinas. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Marcos Rogério Silveira

Abordagem Semi-Supervisionada para detecção de Domínios Maliciosos em TLDs em sua primeira consulta

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiador: NIC.br - Proc. 2764/2018

Comissão Examinadora

Prof. Dr. Adriano Mauro Cansian
Unesp - Câmpus São José do Rio Preto
Orientador

Prof. Dr. André Ricardo Abed Grégio
UFPR - Universidade Federal do Paraná

Prof. Dr. Leandro Alves Neves
Unesp - Câmpus São José do Rio Preto

Prof. Dr. Marcelo Zanchetta do Nascimento
UFU - Universidade Federal de Uberlândia

Prof. Dr. Robson de Oliveira Albuquerque
UnB - Universidade de Brasília

São José do Rio Preto
06 de Agosto de 2024

Agradecimentos

À Deus, pelo dom da vida e saúde.

Aos meus pais, Ana Cristina Donda Silveira e Ademir Silveira, por acreditarem neste sonho comigo, pelo apoio incondicional, e por vibrarem comigo a cada nova conquista.

À minha namorada, Alexa Chaves, por acreditar em mim, por estar ao meu lado em todos os momentos e por ser minha companheira em cada passo dessa jornada.

Aos meus amigos, pela amizade ao longo dos anos, por todas as risadas mesmo nos momentos mais difíceis, e pelo suporte e ajuda em todas as etapas.

Ao meu orientador, Prof. Dr. Adriano Mauro Cansian, pela oportunidade de poder realizar o estudo descrito neste trabalho.

Ao Laboratório ACME! e aos seus membros, que durante estes anos foi possível criar laços de amizade, e colaborar um para com os outros. Gostaria de agradecer em especial ao "Team DNS".

Agradeço, também, à Universidade Estadual Paulista "Júlio de Mesquita Filho", campus de São José do Rio Preto, pelo curso de pós-graduação ali ministrado por professores de excelência.

Ao NIC.br, por todo suporte durante estes anos.

O presente trabalho foi realizado com o apoio do Núcleo de Informação e Coordenação do Ponto BR (NIC.br) - Processo nº 2764/2018 - CCP, o qual agradeço.

"If you want it, you can fly
you just have to trust you a lot."

-Jobs, Steve.

Resumo

Este trabalho propõe um método para a detecção precoce de domínios maliciosos recém-registrados em Top Level Domains (TLDs). A abordagem proposta utiliza apenas a primeira consulta DNS de um domínio recém-registrado coletada de forma passiva e enriquecida pelo sistema ENTRADA, que permite a análise e o armazenamento eficiente e em larga escala desses dados. A detecção desses domínios fica a cargo de uma abordagem semi-supervisionada, composta por um módulo de aprendizado de máquina supervisionado no qual foram utilizados dois algoritmos supervisionados, e um módulo de aprendizado não supervisionado, com a utilização de um algoritmo de clusterização. As saídas de todos os modelos usados são enviadas para um modelo classificador final, com o objetivo de combinar as previsões anteriormente fornecidas e identificar se aquele domínio é malicioso, juntamente com a probabilidade. Para a etapa de treinamento dos modelos supervisionados, os dados são balanceados para que não haja viés. O treinamento do modelo não supervisionado é feito apenas com domínios maliciosos, visando gerar clusters puramente maliciosos. Após a etapa de treinamento, os modelos são testados no ambiente real, avaliando novos domínios recém-registrados e o desempenho da abordagem proposta. Por fim, um módulo de re-treinamento está disponível, sempre que houver uma degradação no desempenho. A abordagem proposta na etapa de treinamento apresentou uma área sob a curva ROC (AUC) de 0,96 (+/- 0,01) e uma Acurácia (ACC) de 0,91. Na etapa de teste, que simula o ambiente de produção, as métricas foram ACC 0,88, taxa de de verdadeiro positivo (TVP) de 0,884, taxa de de verdadeiro negativo (TVN) 0,875, taxa de falso positivo (TFP) 0,124 e taxa de falso negativo (TFN) 0,110.

Palavras-chave: Domain Name System. Domínios Maliciosos. DNS Passivo. Aprendizado de Máquina.

Abstract

This work proposes a method for the early detection of newly registered malicious domains in Top Level Domains (TLDs). The proposed approach utilizes only the first DNS query of a newly registered domain, passively collected and enriched by the ENTRADA system, which enables the efficient and large-scale analysis and storage of these data. The detection of these domains is handled by a semi-supervised approach, consisting of a supervised machine learning module, in which two supervised algorithms were used, and an unsupervised learning module, employing a clustering algorithm. The outputs of all the models used are sent to a final classifier model, with the aim of combining the previously provided predictions and identifying whether the domain is malicious, along with the probability. For the training phase of the supervised models, the data is balanced to avoid bias. The training of the unsupervised model is done exclusively with malicious domains, aiming to generate purely malicious clusters. After the training phase, the models are tested in a real environment, evaluating newly registered domains and the performance of the proposed approach. Finally, a retraining module is available whenever there is a degradation in performance. The proposed approach in the training phase achieved an Area Under the ROC Curve (AUC) of 0.96 (+/- 0.01) and an Accuracy (ACC) of 0.91. In the testing phase, which simulates the production environment, the metrics were ACC 0.88, True Positive Rate (TPR) of 0.884, True Negative Rate (TNR) of 0.875, False Positive Rate (FPR) of 0.124, and False Negative Rate (FNR) of 0.110.

Keywords: *Domain Name System. Malicious Domain. Passive DNS. Machine Learning.*

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AA *Authoritative Answer*

ABC *Artificial Bee Colony*

ACC *Acurácia*

ANCOUNT *Answer count*

ARCOUNT *Additional Information Count*

ASN *Autonomous System Number*

AUC *Área Under ROC Curve*

BIND *Berkeley Internet Name Domain*

CC *Command and Control*

ccTLD *country code Top-Level Domain*

CD *Checking Disabled*

CDN Rede de Distribuição de Conteúdo

CGI.br Comitê Gestor da Internet no Brasil

CNAME *Canonical Name*

DBSCAN *Density-Based Spatial Clustering of Applications with Noise*

DDoS *Distributed Denial of Service*

DGA *Domain Generation Algorithm*

DNS *Domain Name System*

DNSSEC *Domain Name System Security Extensions*

DT *Decision Tree*

E Especificidade

ENTRADA *Enhanced Top-level domain Resilience through Advanced Data Analysis*

FN Falso Negativo

FP Falso Positivo

FQDN *Fully Qualified Domain Name*

GOSS *Gradient-based One-Side Sampling*

gTLD *Generic Top-Level*

ICANN *Internet Corporation for Assigned Names and Numbers*

IETF *Internet Engineering Task Force*

IP *Internet Protocol*

ISC *Internet System Consortium*

ISP *Internet Service Provider*

LightGBM *Light Gradient Boosting Machine*

ML *Machine Learning*

MX *Mail Exchange*

NIC.br Núcleo de Informação e Coordenação do Ponto BR

NS *Name Server*

NXDomain *Non-Existent Domain*

PART *Partial Decision Tree*

PCA *Principal Component Analysis*

PCAP *Packet Capture*

pDNS *Passive DNS*

PR Precisão-Recall

PTR *Pointer*

QNAME *Query Name*

QTYPE *Query Type*

RCODE *Response Code*

RF *Random Forest*

ROC *Receiver Operating Characteristic*

RRs *Resource Records*

RUS *Random Undersampling*

SMOTE *Synthetic Minority Oversampling Technique*

SOA *Start of Authority*

SVM *Support Vector Machine*

TFN Taxa de Falso Negativo

TFP Taxa de Falso Positivo

TLD *Top-Level Domain*

TTL *Time To Live*

TVN Taxa de Verdadeiro Negativo

TVP Taxa de Verdadeiro Positivo

UDP *User Datagram Protocol*

URL *Uniform Resource Locator*

VN Verdadeiro Negativo

VP Verdadeiro Positivo

XGBoost *Extreme Gradient Boosting*

Sumário

Capítulo 1

Introdução

No início da Internet, observou-se a necessidade de traduzir nomes de domínio em IPs, já que os humanos lidam melhor com nomes do que com sequências numéricas. Para atender a essa necessidade, foi desenvolvido um sistema de tradução IPs em nomes de domínios. A resolução dos domínios era realizada por meio de um arquivo chamado *hosts.txt*, que continha nomes de domínios e seus respectivos IPs (HARRENSTIEN K.; FEINLER, 1985). No entanto, essa solução apresentava um problema, pois era centralizada. Portanto, a cada novo nome de domínio registrado, era necessário atualizar manualmente esse arquivo.

Dois anos depois, devido à expansão exponencial da Internet, a resolução de nomes de domínio, que antes era centralizada, tornou-se descentralizada. Graças à sua escalabilidade e resolução distribuída, ela foi capaz de acomodar o crescimento da rede (MOCKAPETRIS, 1987a; MOCKAPETRIS, 1987b).

O *Domain Name System* (DNS) tornou-se um protocolo essencial para o bom funcionamento da Internet, uma vez que sua principal função é realizar a tradução de nomes de domínio em IPs e vice-versa. Devido à sua vital importância para a Internet, usuários mal-intencionados utilizam esse protocolo para realizar atividades maliciosas, como a disseminação de *malware*, *botnets*, *fast-flux domains* e *phishing*.

Uma forma de mitigar esses domínios é o uso de *blocklists*. As *blocklists* são listas previamente analisadas por especialistas nas quais domínios que exibem comportamento malicioso são incluídos. Elas são usadas em redes para bloquear o acesso a tais domínios. Para que um domínio seja incluído nessas listas leva algum tempo uma vez que os usuários precisam denunciar o domínio após a ocorrência de atividades maliciosas e, em seguida, uma análise humana se faz necessária. Segundo trabalho apresentado por Vissers et al. (2017), outro problema inerente a essas listas é que 19,30% dos domínios maliciosos registrados como parte de campanhas nunca estarão

presentes nessas listas, seja devido a uma cobertura incompleta das listas ou porque diversos domínios registrados ainda não foram utilizados para fins maliciosos assim, como à existência de falsos positivos.

Ao analisar o problema de domínios recém-registrados, as *blocklists* não possuem aplicação direta para a mitigação do problema, uma vez que o domínio é recente e a inclusão desse domínio nessas listas demora. Com isso, mais usuários se tornam vítimas.

Portanto, é necessário encontrar uma forma de realizar a detecção precoce desses domínios assim que forem registrados, em sua primeira consulta DNS. Isso permitirá a suspensão desses domínios e impactará o menor número possível de usuários por esses sites maliciosos.

1.1 Motivação e Justificativas

No mundo, existem mais de 722 milhões de domínios registrados, divididos entre os 1.600 *Top Level Domains* (TLDs). Atualmente, existem 341 *country code Top Level Domains* (ccTLDs) que possuem cerca de 40% dos domínios registrados globalmente (STAT, 2024). O TLD brasileiro `.br` é o 14º maior TLD, com aproximadamente 5,2 milhões de domínios registrados, apresentando uma média de mais de 4 mil novos domínios registrados por dia (REGISTRO.BR, 2024).

Diante desta quantidade de domínios registrados diariamente, é necessário garantir que apenas domínios legítimos fiquem disponíveis para que os usuários acessem. Isso ocorre porque, ao acessar domínios maliciosos, usuários desavisados podem se tornar vítimas dos mais diversos golpes mencionados anteriormente, resultando em perdas financeiras e até mesmo no sequestro de dados, independentemente de terem sido fornecidos pelo usuário ou não. Para as empresas, as perdas financeiras são ainda maiores, podendo levar à interrupção de seus negócios, dependendo da gravidade do ataque. Além das perdas financeiras, tais incidentes podem causar um impacto negativo na confiança do consumidor em relação ao produto ou serviço.

De acordo com o Relatório de Ameaças à Internet de 2019, os ataques na web aumentaram cerca de 58% (SYMANTEC, 2019). Ainda segundo a esse mesmo relatório, uma em cada dez URLs analisadas é maliciosa. Outro dado preocupante é fornecido pela Kaspersky, que aponta o Brasil como líder no ranking de usuários atacados por phishers, correspondendo a 21,66% de todos os ataques. Isso representou um aumento de 1,53 pontos percentuais em comparação com o trimestre anterior, quando o país já liderava o ranking (VERGELIS; SHCHERBAKOVA; SIDORINA, 2021). Entre

abril e junho de 2020, cerca de 12,9% dos usuários de Internet no Brasil acessaram pelo menos um site malicioso, enquanto a média global para o mesmo período foi de 8,26% (KULIKOVA; SIDORINA; SHCHERBAKOVA, 2021b). No primeiro trimestre de 2021, um relatório da Kaspersky sobre Spam e *phishing* indicou que o Brasil ficou em nono lugar na distribuição de ataques de *phishing*, respondendo por 7,94% dos ataques (KULIKOVA; SIDORINA; SHCHERBAKOVA, 2021a). No entanto, no relatório do segundo trimestre de 2021, o Brasil voltou à liderança do ranking, com 6,67%, seguido de perto por Israel (6,55%) e França (6,46%) (KULIKOVA; SHCHERBAKOVA, 2021). No relatório mais recente da Kaspersky, referente ao ano de 2023, observa-se que o Brasil ascendeu à sexta posição entre os países responsáveis pelo lançamento de *spams*, representando 2,8% do total. Essa movimentação indica um aumento de sua participação em comparação com anos anteriores. Por fim, este relatório evidencia uma redução significativa no Brasil como alvo de ataques de spam por e-mail em comparação com o ano anterior. A Rússia liderou os registros, representando 16,72% do spam malicioso, mais que o dobro em comparação com os dados de 2022. A Espanha caiu para o segundo lugar (9,53%), apesar de seu aumento na participação, enquanto o Brasil (2,63%) sofreu uma queda de quase metade (KULIKOVA et al., 2024).

Portanto, é evidente que os usuários de Internet no Brasil frequentemente se tornam vítimas de crimes cibernéticos, especialmente devido a domínios maliciosos relacionados ao *phishing*. Os ataques de *phishing* ocorrem quando os atacantes criam sites extremamente semelhantes aos originais, muitas vezes copiando o layout idêntico ao site genuíno para enganar os usuários. Além de replicar o layout, eles registram domínios com erros de grafia ou com partes do nome do domínio, de forma que os usuários, ao acessarem esses sites, acreditam erroneamente estar no site genuíno e acabam fornecendo seus dados pessoais, informações de cartão de crédito, dados bancários e senhas de acesso.

Realizar a detecção precoce desses domínios maliciosos é essencial para prevenir que os usuários caiam em golpes e fraudes. Quanto mais cedo esses domínios maliciosos forem detectados, menor será o número de vítimas. Atualmente, um grande número de domínios é registrado diariamente, com uma média de mais de 4 mil novos registros por dia apenas no TLD .br (REGISTRO.BR, 2024).

É por essa razão que as abordagens baseadas em Aprendizado de Máquina (ML) estão se tornando cada vez mais populares, devido à sua capacidade de detectar automaticamente esses domínios, com base em padrões de reconhecimento.

Na literatura existem, basicamente, três tipos de abordagens para a detecção de

domínios maliciosos. A primeira ocorre na etapa de pré-registro, onde analisa-se aspectos léxicos do domínio e informações pessoais necessárias para a compra do domínio (KIDMOSE et al., 2018; DESMET et al., 2021). Na etapa de pós-registro, que ocorre após a primeira atualização da Zona do DNS, os autores podem optar por utilizar DNS ativo (KOUNTOURAS et al., 2016) ou realizar a coleta passiva de dados do DNS (SILVEIRA et al., 2022). A coleta passiva de DNS pode ser realizada em dois pontos da estrutura do DNS, em servidores recursivos ou autoritativos em TLDs. Ao utilizar o DNS passivo em servidores autoritativos de TLDs, é possível obter uma visão abrangente do domínio (ANTONAKAKIS et al., 2011).

Usando uma abordagem semi-supervisionada em conjunto com o DNS passivo coletado de servidores TLD, a presente tese aborda uma frente até então inexplorada: a detecção de domínios maliciosos recém-registrados na sua primeira consulta. Entre os trabalhos publicados, apenas (ANTONAKAKIS et al., 2011) e (SILVEIRA et al., 2022) utilizam o pDNS (DNS passivo) de servidores autoritativos TLD como fonte de dados. No entanto, os autores do KOPIS Antonakakis et al. (2011) não se concentram na detecção precoce de domínios, enquanto Silveira et al. (2022) realiza a detecção desses domínios na janela de 72 horas após a primeira consulta. Trabalhos como (KIDMOSE et al., 2018) e (DESMET et al., 2021) se preocupam com a detecção precoce, mas utilizam exclusivamente dados da fase de pré-registro, quando o domínio ainda não está disponível para acesso ao público. Apenas (DESMET et al., 2021) emprega uma abordagem semi-supervisionada.

Esta tese apresenta duas hipóteses que são objeto de investigação. A primeira hipótese se concentra na avaliação da eficácia da detecção de domínios maliciosos mediante o uso exclusivo da primeira consulta DNS. A segunda hipótese abordada por este trabalho, é a possível melhora na identificação de domínios maliciosos e legítimos que pode ser analisada pelas métricas TVP, TVN em comparação com o uso exclusivo de uma abordagem supervisionada, tomando como referencial exclusivamente os modelos desenvolvidos no âmbito deste estudo. Adicionalmente, o trabalho demonstra preocupação com a adaptabilidade em relação aos novos possíveis comportamentos dos domínios maliciosos, apresentando a possibilidade de re-treinamento dos modelos sempre que um analista detectar uma queda de desempenho do sistema.

1.2 Objetivos

O presente trabalho tem como objetivo principal realizar a detecção de domínios

maliciosos recém-registrados, utilizando apenas sua primeira consulta DNS coletada passivamente de um servidor TLD e empregando uma abordagem semi-supervisionada para identificar esses domínios maliciosos.

Os objetivos específicos a serem alcançados são os seguintes:

a) Criar um modelo não supervisionado utilizando uma técnica de agrupamento (clusterização);

b) Desenvolver um modelo de aprendizado de máquina supervisionado para classificação binária;

c) Produzir um modelo final baseado em *Ensemble*, no qual esse modelo receberá as saídas dos modelos anteriores e gerará uma pontuação final para o domínio (indicando a probabilidade de pertencer à classe maliciosa);

d) Fazer a coleta passiva de consultas de resolução de DNS de um servidor autoritativo de um TLD e usá-lo como fonte de dados primária.

1.3 Contribuições Obtidas

Com o desenvolvimento deste estudo sobre a detecção precoce de domínios maliciosos em TLDs utilizando DNS passivo (pDNS), foram feitas as contribuições a seguir:

- O desenvolvimento de uma abordagem capaz de detectar domínios maliciosos em sua primeira consulta DNS ao servidor autoritativo TLD;
- A obtenção de uma abordagem semi-supervisionada com Taxa de Verdadeiro Positivo (TVP) e Taxa de Verdadeiro Negativo (TVN) acima de 0,87;
- A obtenção de resultados mais equilibrados na detecção de domínios maliciosos e legítimos ao comparar com o uso de apenas um algoritmo supervisionado;
- A publicação de um artigo intitulado *"Detection of Newly Registered Malicious Domains through Passive DNS"* na *"IEEE International Conference on Big Data (Big Data)"* que aconteceu no ano de 2021 e pode ser acessado através do DOI https://doi.org/10.1109/BigData52589.2021.9671348;
- A publicação de um artigo intitulado *"Early Identification of Abused Domains in TLD through Passive DNS Applying Machine Learning Techniques"* no *"International Journal of Communication Networks and Information Security (IJCNIS)"* no ano de 2022 e pode ser acessado através do DOI https://doi.org/10.17762/ijcnis.v14i1.5256.

1.4 Organização do Trabalho

O presente trabalho está dividido em cinco capítulos. Neste primeiro capítulo, é apresentada a introdução do trabalho, bem como o problema a ser abordado e os objetivos a serem alcançados. O Capítulo 2 apresenta a fundamentação teórica utilizada na construção deste trabalho, incluindo o levantamento bibliográfico realizado. No Capítulo 3, é apresentada a metodologia utilizada no trabalho, juntamente com os cinco módulos que o compõem além da metodologia dos dados utilizados na etapa de treinamento. No Capítulo 4, são apresentados os resultados obtidos pelos modelos desenvolvidos individualmente, bem como o desempenho do modelo final, que representa o desempenho global do trabalho. Por fim, no Capítulo 5, são apresentadas as conclusões, bem como os trabalhos futuros propostos e limitações deste trabalho.

Capítulo 2

Fundamentação Teórica

Este capítulo tem como objetivo expor os conceitos utilizados nesta tese, apresentando suas características e relevância, além de apresentar os trabalhos mais relevantes na literatura sobre a detecção de domínios maliciosos usando abordagens de Aprendizado de Máquina. Na Seção 2.1, é apresentado o processo de registro de um domínio. Na Seção 2.2, é abordado o protocolo DNS, explicando seu funcionamento e destacando os abusos e atividades maliciosas que podem ocorrer por meio desses domínios. Na Seção 2.3, são apresentados conceitos relacionados ao Aprendizado de Máquina, incluindo formas de aprendizado e alguns dos principais algoritmos utilizados. Na Seção 2.4 , são introduzidas as métricas utilizadas para avaliar os modelos construídos. Na Seção 2.5 são discutidas as abordagens para o treinamento e teste dos modelos. Na Seção 2.6, é abordado o balanceamento de classes e as técnicas aplicadas. Na Seção 2.7 são apresentados conceitos de redução de dimensionalidade e os algoritmos mais relevantes nessa aplicação. Na Seção 2.8 , é realizado o levantamento bibliográfico, destacando os principais estudos disponíveis na literatura. Por fim, na Seção 2.9, são apresentadas as considerações parciais deste capítulo.

2.1 Processo de Registro de um Domínio

Para que um domínio fique disponível para o público, o primeiro passo é a aquisição do domínio. Nesta etapa de aquisição de um domínio, há três atores principais, sendo eles o *"Registry"*, o *"Registrar"* e o *"Registrant"*.

O *"Registrant"* é uma pessoa física ou jurídica que procura um *"Registrar"* para adquirir um nome de domínio. Os *"Registrars"* são organizações credenciadas pela ICANN e certificadas pelos registros para vender nomes de domínio. É de responsabilidade deles manter os dados do WHOIS, embora os *"Registries"* também possam

vender domínios diretamente aos *"Registrants"* (HARRENSTIEN; WHITE, 1982; DAIGLE, 2004). Os *"Registries"* são entidades responsáveis por manter os registros de cada TLD. É de responsabilidade deles aceitar as solicitações de registro de domínio, manter um banco de dados contendo todos os dados necessários do registro do nome de domínio e fornecer servidores DNS para a publicação dos dados dos arquivos de zona, tornando o domínio acessível na Internet.

A ICANN, mencionada anteriormente, é a organização sem fins lucrativos responsável pela supervisão e atribuição de endereços IP e nomes de domínio. É de responsabilidade da ICANN o gerenciamento do servidor raiz, a administração do sistema de nomes de TLDs e a celebração de acordos contratuais com *"Registries"* e *"Registrars"* para manter o WHOIS (ICANN, 2021).

O *"Registry"* brasileiro é o Núcleo de Informação e Coordenação do Ponto BR (NIC.br). O NIC.br foi criado para implementar as decisões e os projetos do Comitê Gestor da Internet no Brasil (CGI.br), que é responsável por coordenar e integrar as iniciativas e serviços da Internet no país. Portanto, além de ser o braço executivo do CGI.br, uma de suas atribuições é registrar e manter os nomes de domínio que utilizam o "`.br`" e distribuir endereços IPv4 e IPv6 no país (NIC.BR, 2021).

Na Figura 2.1 é ilustrado o processo de registro de um nome de domínio.

Figura 2.1: Processo de registro de um nome de domínio

Fonte: Adaptado de ICANN (2021).

Ao finalizar o processo de aquisição e registro de um domínio, é necessário configurar o DNS para que esse serviço possa realizar a tradução do nome de domínio no IP correspondente e permitir que os usuários acessem o domínio.

2.2 Sistema de Nomes de Domínio

O Sistema de Nomes de Domínio (DNS) (MOCKAPETRIS, 1987a; MOCKAPETRIS, 1987b), como é conhecido hoje, foi proposto por Paul Mockapetris em 1983. Antes disso, a ARPAnet, como era conhecida, realizava traduções por meio de um único arquivo chamado "hosts.txt" em um único host. Ao reconhecer esse problema, Paul Mockapetris propôs um sistema distribuído e dinâmico. Após a criação formal da Internet Engineering Task Force (IETF) em 1986, o DNS se tornou um dos padrões originais da Internet (INNOVATOR, 2021).

A primeira implementação de software capaz de utilizar a proposta de Paul Mockapetris foi realizada em 1984, utilizando o sistema UNIX, e foi desenvolvida por quatro estudantes da Universidade de Berkeley sob o nome de *"Berkeley Internet Name Daemon"* (BIND). No ano seguinte, Kevin Dunlap da *Digital Equipment Corporation (DEC)*, que posteriormente foi adquirida pela HP, fez uma significativa revisão na implementação original e renomeou-a para *"Berkeley Internet Name Domain"* (BIND). Atualmente, esse software é mantido pelo *Internet Systems Consortium* (ISC).

Normalmente, servidores DNS são máquinas que utilizam o sistema operacional UNIX e empregam o software BIND para realizar consultas e respostas do DNS.

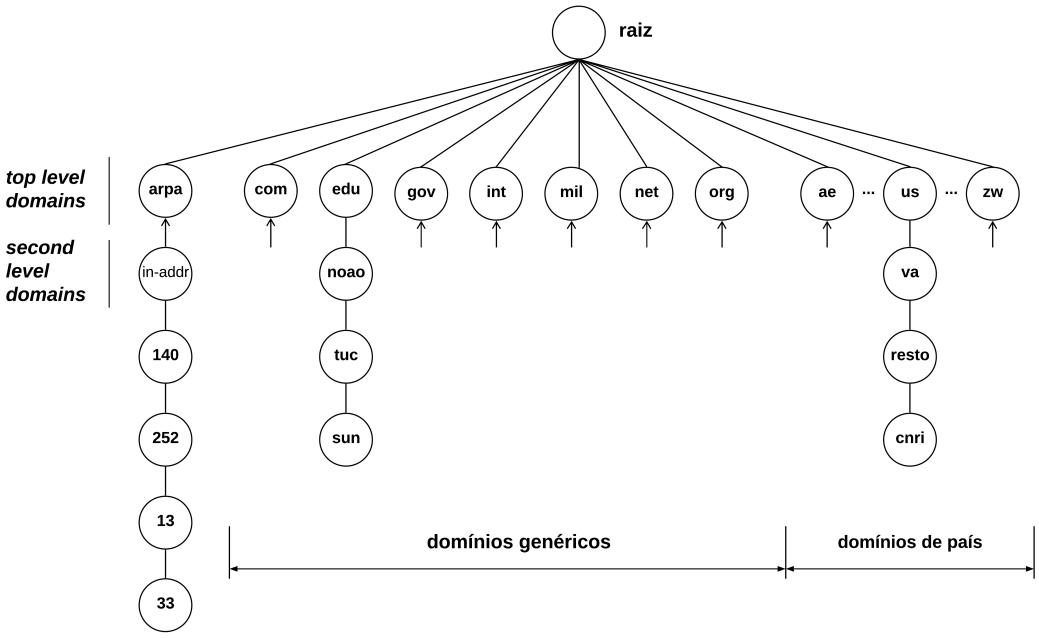
O DNS é composto por um banco de dados distribuído e hierárquico e utiliza um protocolo da camada de aplicação que utiliza UDP na porta 53 para executar essas operações.

A principal função do DNS é traduzir nomes de domínio em endereços IP e vice-versa, uma vez que, para nós, seres humanos, é mais fácil trabalhar com nomes do que com sequências numéricas, especialmente com o advento do IPv6, no qual essas sequências alfanuméricas se tornaram ainda mais longas. Além da tradução de nomes de domínio, o DNS contém outras informações técnicas que podem ser obtidas por meio de seus *"Resource Records"* (RRs) (LYNN, 2001). Esses RRs são obtidos como respostas às consultas feitas pelos *resolvers*. Um servidor DNS pode responder com um RR específico ou vários RRs, dependendo da consulta realizada.

2.2.1 Hierarquia

Devido à sua escalabilidade, o DNS possui uma estrutura hierárquica bem definida e é insensível a letras maiúsculas e minúsculas, assemelhando-se a diretórios e arquivos no sistema de arquivos dos sistemas operacionais (FALL; STEVENS, 2011). Essa hierarquia é apresentada na Figura 2.2.

Figura 2.2: Estrutura hierárquica do DNS



Fonte: Adaptado de Stevens (1994).

A estrutura hierárquica do DNS assemelha-se a uma árvore, com os *Root Servers* no topo da hierarquia, sendo o ponto de partida para consultas DNS. Existem 13 *Root Servers* com vários espelhos distribuídos pelo mundo. No Brasil, há 54 desses espelhos *Root Servers* em todo o país (IANA, 2021).

Abaixo dos *Root Servers*, na estrutura hierárquica, estão os *Top Level Domains* (TLDs). Os TLDs possuem dois tipos, dependendo de seu uso. *Country Code Top Level Domains* (ccTLDs) são TLDs destinados a países, como o .br para o Brasil. Enquanto isso, os *Generic Top Level Domains* (gTLDs) são destinados a uso genérico, e sua distribuição é independente da localização geográfica. Alguns dos gTLDs mais conhecidos incluem .com, .net, .org, .gov, entre outros.

No segundo nível da árvore da estrutura hierárquica, encontram-se os *Second Level Domains*, no terceiro nível os *Third Level Domains* e, assim, sucessivamente, até que se complete a formação do nome de domínio completo, intitulado de *Fully Qualified Domain Name* (FQDN).

A formação dos FQDN ocorre ao percorrer todos os nós da árvore, desde o *root server* até a formação completa do nome de domínio. É possível observar que cada vez que o domínio possui um ponto "." em seu nome, ele tem mais níveis, o que significa que mais servidores foram percorridos para a formação de seu FQDN. É importante ressaltar que os *Root Servers* não possuem rótulos, portanto, os nomes

de domínio não terminam com um ponto ".". Na Figura 2.2, é possível observar a formação de três FQDNs: `cnri.resto.va.us`, `sun.tuc.noao.edu` e, por fim, `33.13.252.140.in-addr.arpa`. Este último é resultado de uma resolução reversa, na qual se consulta o endereço IP para obter o nome de domínio associado a ele.

Devido à sua estrutura hierárquica e distribuída, os servidores possuem ponteiros apenas para os servidores que estão no nível imediatamente abaixo deles, e assim por diante, garantindo, portanto, que as informações sejam distribuídas por toda a hierarquia de servidores DNS.

Além de toda a estrutura apresentada para a resolução do nome de domínio, há um servidor de extrema importância que não faz parte da hierarquia apresentada. Este servidor é responsável por realizar consultas, iniciando nos *Root Servers* e percorrendo todo o caminho até a formação do FQDN. Esse servidor é conhecido como servidor DNS recursivo. Os provedores de serviços de Internet (ISPs) possuem um ou mais servidores recursivos. Quando um host é conectado à rede, ele recebe o endereço IP desses servidores recursivos, que são responsáveis por percorrer toda a árvore na resolução do nome de domínio.

2.2.2 Consultas DNS

As consultas DNS são iniciadas quando um *resolver* envia uma consulta DNS ao seu servidor recursivo com o nome de domínio a ser traduzido. A partir deste passo inicial, todas as consultas à hierarquia DNS são realizadas por meio do servidor DNS recursivo. Após obter o endereço IP do servidor autoritativo do domínio, o servidor DNS recursivo envia essa informação de volta ao *resolver*, permitindo que o usuário acesse o site desejado. Todo esse processo de resolução de um nome de domínio é exemplificado na Figura 2.3 e descrito a seguir:

1. O cliente envia uma mensagem ao servidor DNS recursivo com o nome de domínio a ser traduzido, por exemplo, "`exemplo.acmesecurity.org`".

2. O servidor recursivo recebe essa mensagem e envia uma consulta a um servidor raiz.

3. O servidor raiz identifica o TLD responsável pelo domínio e envia uma mensagem ao servidor recursivo com uma lista de endereços IP dos servidores responsáveis pelo TLD, no caso, "`.org`".

4. O servidor recursivo, por sua vez, retransmite a mensagem de consulta a um desses servidores TLD.

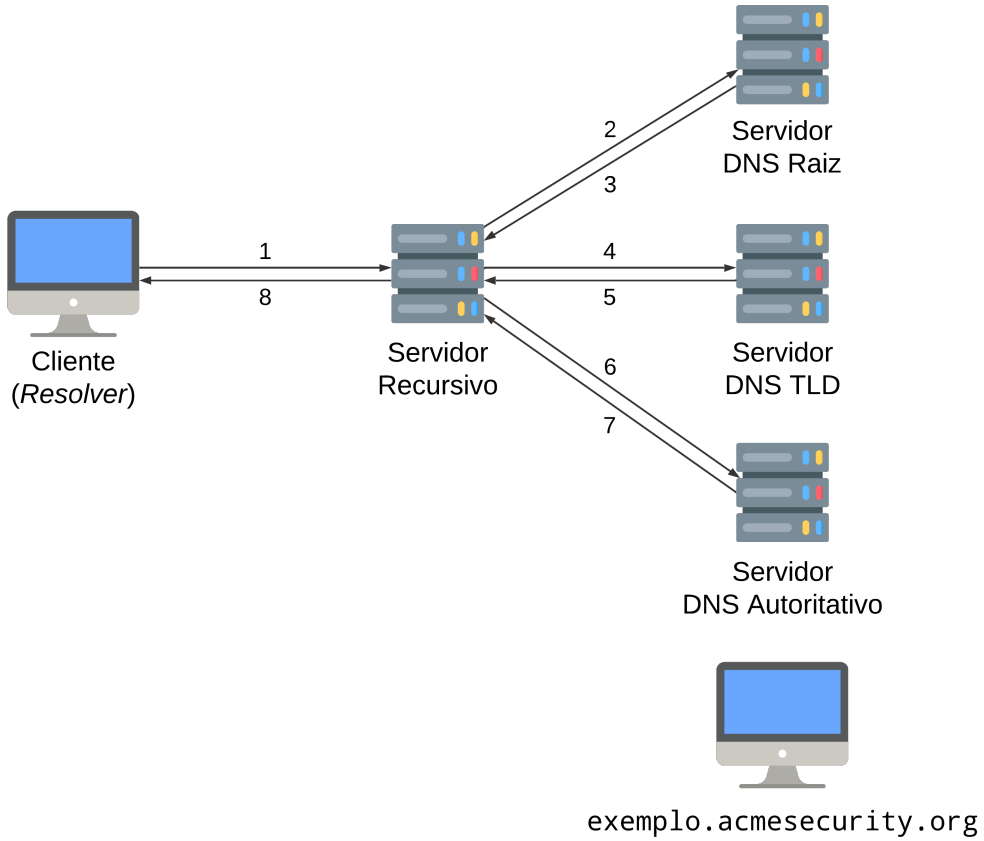
5. O servidor TLD observa o sufixo "`acmesecurity.org`" e envia uma mensagem de resposta ao servidor recursivo contendo o(s) IP(s) responsáveis por esse domínio.

6. O servidor recursivo reenvia a mensagem ao servidor autoritativo informado anteriormente.

7. O servidor autoritativo, que tem autoridade para responder a solicitações para o domínio em questão, responde com o endereço IP de "`exemplo.acmesecurity.org`".

8. O servidor recursivo, agora com o endereço IP em mãos, envia uma mensagem ao cliente, informando o endereço IP do domínio desejado.

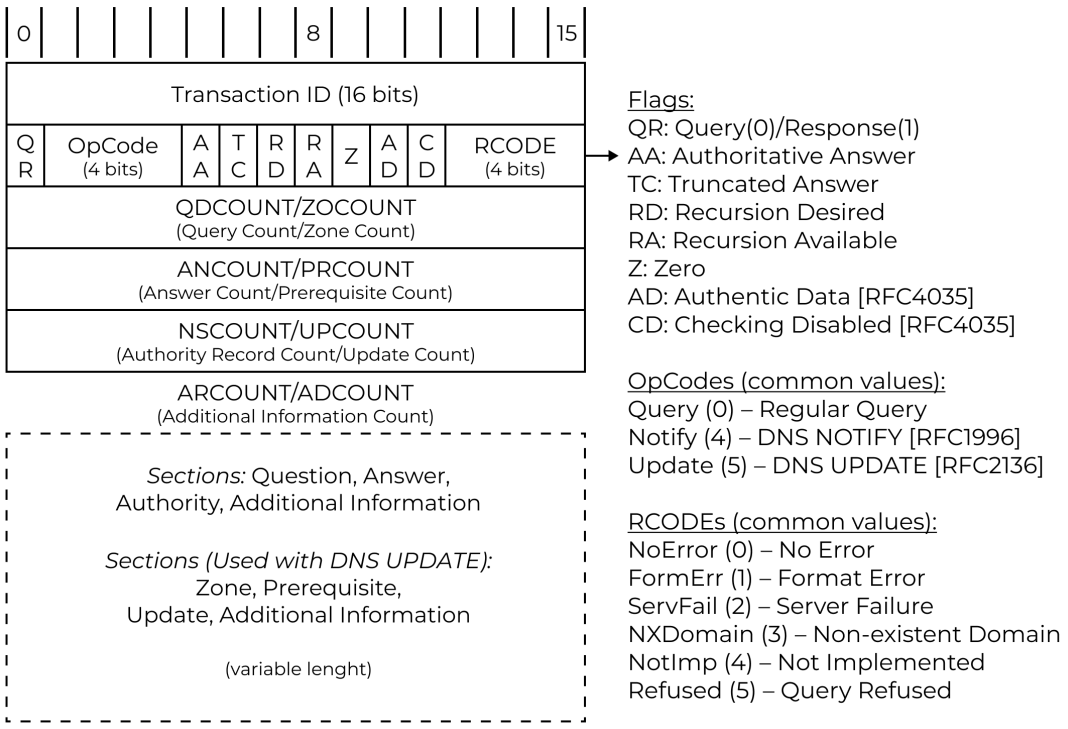
Figura 2.3: Consultas ao sistema de nomes de domínio



Fonte: Adaptado de Kurose e Ross (2013).

Há um formato básico de mensagens DNS que é utilizado para todas as operações, incluindo consultas, respostas, transferências de zona, notificações e atualizações dinâmicas (3RD, 2011). Este formato é ilustrado na Figura 2.4.

Figura 2.4: Formato básico de uma mensagem DNS



Fonte: Adaptado de Fall e Stevens (2011).

A mensagem DNS inicia-se com um cabeçalho fixo seguido por quatro seções de comprimento variável, a saber: perguntas (ou consultas), respostas, registro de autoridade e registros adicionais. Exceto pela primeira seção, todas as outras podem conter um ou mais RRs que podem ser armazenados em cache. As consultas não são armazenadas (FALL; STEVENS, 2011).

O campo *"Response Code"* (RCODE) possui 4 bits que contêm o código de retorno. A Tabela 2.1 apresenta os dez primeiros valores possíveis deste campo, bem como suas descrições. Os valores mais comuns estão no intervalo de 0 a 5.

2.2.3 *Resource Records*

Os *"Resource Records"* (RRs) são registros de recursos que fornecem mapeamentos de hosts para endereços IP e vice-versa. Essas informações são armazenadas nos servidores DNS e enviadas como respostas às solicitações. As consultas DNS podem ser feitas para solicitar um RR específico ou vários RRs. Um RR pode ser definido como uma tupla que contém quatro campos: nome, valor, tipo e *"Time To Live"* (TTL). Os campos nome e valor dependem do tipo de RR a ser fornecido, enquanto o TTL define o tempo (em segundos) durante o qual essa informação pode ser mantida em

Tabela 2.1: Os dez primeiros tipos de erros utilizados no campo RCODE.

Valor	Nome	Descrição e Propósito
0	NoError	Sem erro.
1	FormErr	Erro de formato; query não pode ser interpretada
2	ServFail	Falha de servidor; erro no processamento no servidor
3	NXDomain	Domínio inexistente; domínio desconhecido referenciado
4	NotImp	Não implementado; pedido não suportado no servidor
5	Refused	Servidor se recusou a fornecer resposta
6	YXDomain	O nome existe, mas não deveria (usado com atualizações)
7	YXRRSet	RRSet existe, mas não deveria (usado com atualizações)
8	NXRRSet	RRSet não existe, mas deveria (usado com atualizações)
9	NothAuth	Servidor não autorizado para zona (usado com atualizações)
10	NotZone	Nome não contido na zona (usado com atualizações)

Fonte: Adaptado de Fall e Stevens (2011).

cache pelos servidores DNS (KUROSE; ROSS, 2013). Por fim, serão apresentados alguns tipos de RRs (TCP/IP).

- ***Address* (A):** Define o endereço IPv4 de um FQDN;
- ***Address* (AAAA):** Define o endereço IPv6 de um FQDN;
- ***Canonical Name* (CNAME):** Define um apelido para um FQDN;
- ***Pointer* (PTR):** O PTR é utilizado para consultas reversas. Este tipo de RR indica o FQDN relacionado ao IP consultado;
- ***Name Server* (NS):** Define um servidor DNS autoritativo para um domínio. Obrigatoriamente cada domínio deve possuir ao menos um NS;
- ***Start of Authority* (SOA):** Define o início de uma zona de autoridade;
- ***Mail Exchange* (MX):** Define o nome do host de gerenciamento do email do domínio.

2.2.4 Abusos e uso indevido de DNS

Os domínios podem ser abusados logo após o registro, e toda a estrutura do DNS pode ser usada com más intenções. Essas práticas podem ser implementadas, aprimoradas e/ou servirem como suporte para diversas atividades maliciosas (TORABI et al.,

2018). Abusos podem ocorrer já na etapa de pré-registro de um domínio. Dois abusos comuns nessa fase são o *"Cybersquatting"*, também conhecido como *"Domain Squatting"*, e o *"Typosquatting"*.

- ***Cybersquatting ou Domain Squatting***: Usuários mal-intencionados, com o objetivo de obter lucro, registram domínios que se assemelham ao nome de uma pessoa ou entidade comercial e, em seguida, tentam vender esses domínios à empresa ou indivíduo representado. Um exemplo desse tipo de abuso é o caso de Chris Gillespie, que registrou domínios como "`googlechevron.com`", "`googlecoors.com`", e "`googledonaldtrump.com`" entre 29 de fevereiro e 10 de março de 2010. Isso é um exemplo clássico de *Cybersquatting*, no qual os infratores buscam lucrar com a semelhança entre os domínios registrados e marcas conhecidas. (MCGEE, 2012).

- ***Typosquatting***: Este tipo de ataque visa gerar confusão nos usuários da Internet. Os usuários registram domínios muito semelhantes aos originais amplamente conhecidos, contendo erros de digitação em seus nomes. Domínios como `amercanas.com.br` ou `amaz0n.com` são exemplos desse tipo de ataque. Embora esses domínios nem sempre sejam usados para práticas maliciosas, uma grande parte deles está relacionada a diversas atividades maliciosas, incluindo ataques de *phishing* (ALRWAIS et al., 2014).

Com o objetivo de ocultar atividades maliciosas e dificultar sua detecção, diversos atacantes utilizam a estrutura aberta do DNS para operar e manter domínios maliciosos na internet, fazendo uso de serviços legítimos para implementar suas atividades maliciosas. Dentre os tipos de ataques mais comuns que utilizam essa estratégia estão: *Botnet*, *Fast Flux Domains* e *Malicious Domain Generation Algorithms*.

- ***Botnet***: As *Botnets* são redes de *bots*. *Bots*, como são conhecidos, são computadores que possuem software malicioso instalado por atacantes, permitindo a capacidade de interação com outros *bots* ou com toda a rede *botnet*. Os *bots* são controlados por um *botmaster* que opera toda a *botnet* por meio de servidores de Comando e Controle (C&C) para realizar diversos tipos de ataques, como DDoS, campanhas de spam, coleta de dados, disseminação de malwares, entre outros. As máquinas afetadas não são controladas totalmente pelo atacante; portanto, a operação das *botnets* limita-se à operação das máquinas que foram comprometidas. (TORABI et al., 2018).

- ***Fast Flux Domains e Flux Networks***: O fluxo de domínio *(domain fluxing)* ocorre quando um domínio é resolvido para diversos IPs alternativos. Desenvolvido com o propósito legítimo de balanceamento de carga e Redes de Distribuição de Conteúdo (*Content Delivery Network* - CDN), o fluxo de domínio *(domain name fluxing)* é utilizado por atacantes para ocultar rastros de atividades maliciosas, uma vez que os registros de recursos (RRs) são rapidamente atualizados, alternando, portanto, o endereço IP resolvido para tal domínio. Normalmente, o fluxo rápido *(fast fluxing)* utiliza valores baixos de TTL, permitindo um IP diferente para o domínio em quase todas as solicitações (TORABI et al., 2018).

- ***Malicious Domain Generation Algorithms (DGAs)***: DGAs são uma forma automática de criar nomes de domínio, e, portanto, os atacantes fazem uso desse recurso para gerar domínios maliciosos automaticamente. *Botnets* se beneficiam das DGAs para evitar listas de bloqueio, já que registram diversos domínios e alternam seu uso, além de registrar domínios em vários TLDs, distribuindo-os globalmente pela Internet. O malware *Conficker*, por exemplo, gerou 50.000 nomes de domínio aleatórios sob 113 TLDs diferentes. (TORABI et al., 2018).

Por conta da estrutura aberta do DNS, os atacantes utilizam serviços e aplicações benignas para a execução de suas atividades maliciosas. Por exemplo, o número de resolvers de DNS abertos, servidores DNS configurados para aceitar consultas de qualquer origem na internet, apresentaram discrepâncias nas resoluções DNS. Esses *resolvers* abertos foram implementados maliciosamente com o objetivo de censurar canais de comunicação, injetar anúncios, transferir arquivos maliciosos e até mesmo realizar ataques de *phishing*. Usando ferramentas de varredura, os atacantes podem identificar vulnerabilidades nos servidores e explorá-las para realizar ataques direcionados.

2.2.5 DNS Passivo

Com o objetivo de combater ataques de malware, o DNS passivo, conhecido como *"passive DNS"* (pDNS), foi proposto por Weimer (2005). Utilizando servidores DNS recursivos, ele registrou as respostas recebidas de diferentes servidores DNS. O pDNS pode ser definido como a coleta de comunicações entre servidores DNS, o que permite obter consultas e respostas reais do DNS. Portanto, ele possui uma quantidade significativa de recursos para identificar domínios associados a atividades maliciosas e possibilita o enriquecimento dos dados coletados, como informações sobre Números de Sistemas Autônomos (ASN), WHOIS, geolocalização, entre outros.

Devido à estrutura hierárquica do DNS, é possível coletar pDNS em diferentes pontos. Isso pode ser feito instalando *sniffers* para coletar dados em servidores recursivos de DNS em provedores de serviços de Internet ou em servidores autoritativos de TLDs.

2.3 Aprendizado de Máquina

O aprendizado de máquina, conhecido em inglês como *"Machine Learning"* (ML), pode ser definido como um campo de estudo que confere aos computadores a habilidade de aprender, sem serem explicitamente programados. Em outras palavras, é a ciência da programação de computadores que permite o aprendizado a partir dos dados disponíveis, com o objetivo de identificar padrões (WEISS, 1992). O ML é aplicado em uma ampla variedade de soluções, desde aplicações cotidianas até soluções mais complexas em diversas áreas. Isso se deve ao desenvolvimento do poder computacional e à capacidade de armazenamento de grandes volumes de dados, o que possibilita a utilização de ML e ciência de dados em várias aplicações.

Existem quatro tipos principais de técnicas de ML que podem ser aplicadas para resolver uma ampla gama de problemas. São elas: aprendizado supervisionado, aprendizado não supervisionado, aprendizado semi-supervisionado e aprendizado por reforço.

Os algoritmos de aprendizado supervisionado, conforme explicado por Géron (2019), podem ser aplicados em tarefas de regressão e classificação. Nesse tipo de aprendizado, é fundamental que os dados estejam previamente rotulados, pois o modelo de *machine learning* se baseia nas amostras fornecidas para adquirir conhecimento.

Por outro lado, os algoritmos de aprendizado não supervisionado são capazes de aprenderem sem a necessidade de rótulos nos dados. Esses algoritmos agrupam os dados com base em sua similaridade. Após essa fase de aprendizado, o modelo pode determinar se um novo dado pertence a algum dos grupos formados ou não. Além disso, o aprendizado não supervisionado é frequentemente utilizado para reduzir a dimensionalidade dos conjuntos de dados.

O aprendizado semi-supervisionado constitui uma vertente da aprendizagem de máquina que busca integrar algoritmos de aprendizado supervisionado com os de aprendizado não supervisionado. Geralmente, esses algoritmos buscam aprimorar o desempenho em uma dessas tarefas, fazendo uso de informações tipicamente vinculadas à outras (ENGELEN; HOOS, 2020).

Já na aprendizagem por reforço, a máquina aprende por tentativa e erro. A cada

decisão correta que o algoritmo toma, ele recebe uma recompensa, enquanto decisões erradas resultam em punições.

Portanto, o que distingue essas técnicas de *machine learning* é a maneira como os algoritmos aprendem e identificam padrões nos dados. Nos próximos parágrafos são apresentados, com mais detalhes, os dois tipos de aprendizado que serão utilizados no desenvolvimento deste trabalho para a construção da abordagem semi-supervisionada.

2.3.1 Aprendizado Supervisionado

O aprendizado de máquina supervisionado, pode ser definido como o processo de aprender um conjunto de regras a partir de instâncias, sendo que estas instâncias são os dados de treinamento. Definindo de uma forma mais geral, é o processo de desenvolvimento de um classificador que possa generalizar a partir de instâncias (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007). Na Figura 2.5, é apresentado o processo de aplicação de aprendizado de máquina supervisionado para um problema real.

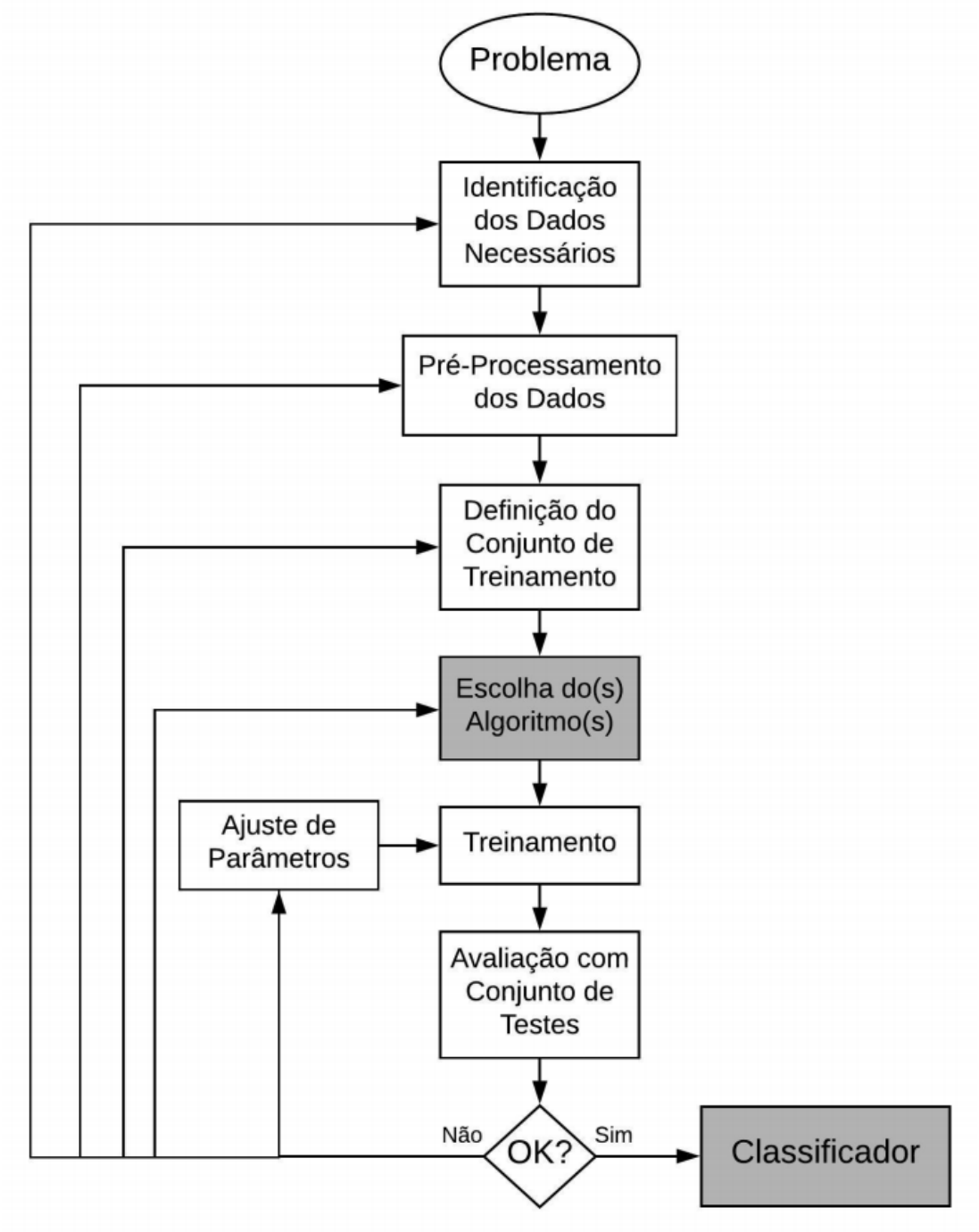
O desenvolvimento de uma abordagem de aprendizado de máquina inicia-se com a identificação de um problema a ser resolvido. Após a identificação dessa necessidade, é necessário determinar quais dados serão necessários para o desenvolvimento da abordagem.

A próxima etapa a ser realizada é o pré-processamento dos dados. É importante destacar que uma etapa essencial no aprendizado de máquina supervisionado é chamada de "engenharia de características". Nessa etapa, as características dos conjuntos de dados são extraídas, e é a partir dessas características que o algoritmo reconhecerá os padrões. É fundamental que as características que compõem o conjunto de dados de treinamento também possam ser extraídas dos dados futuros, para que o modelo possa fazer previsões precisas. Além da engenharia de características, a etapa de pré-processamento de dados inclui ajustes como o balanceamento dos dados, normalização e outros pré-processamentos necessários de acordo com o problema a ser resolvido.

A etapa subsequente é a definição do conjunto de treinamento. Nessa etapa, os dados já precisam estar rotulados, e posteriormente, escolhe-se o algoritmo. Algoritmos aprendem de formas diferentes e seguem abordagens diferentes. Além disso, determinados algoritmos requerem pré-processamentos específicos dos dados. Portanto, a escolha do algoritmo a ser utilizado requer atenção especial, assim como os requisitos necessários para obter bons resultados.

Com o algoritmo escolhido, inicia-se a etapa de treinamento do modelo, e testes são realizados em seu respectivo conjunto com o objetivo de avaliar o desempenho do modelo desenvolvido. Caso o desempenho não esteja satisfatório, é necessária uma

Figura 2.5: Aprendizado de Máquina Supervisionado



Fonte: Adaptado de Kotsiantis, Zaharakis e Pintelas (2007).

etapa de ajuste de parâmetros. Nesse ponto, o modelo é treinado e testado novamente até que apresente um desempenho satisfatório. Ao obter um desempenho satisfatório, é gerado o classificador, e este está pronto para ser utilizado na solução final.

O uso de abordagens de aprendizado de máquina supervisionado é frequentemente

observado no estado da arte para detecção de domínios maliciosos (ZHAUNIAROVICH et al., 2018). Algoritmos de classificação são comumente utilizados, e essa abordagem requer dados rotulados para o seu desenvolvimento.

Dentre os algoritmos de aprendizado de máquina supervisionado, destacam-se aqueles que são baseados em árvores, como a *"Decision Tree"* (DT), o *"Random Forest"* (RF), o *"Extreme Gradient Boost"* (XGBoost) e o *LightGBM*, pois têm demonstrado bom desempenho em trabalhos anteriores.

Árvores de Decisão

O algoritmo de Árvore de Decisão, conhecido como *"Decision Tree"* (DT) em inglês, utiliza a estratégia "dividir para conquistar" ao analisar um problema complexo e dividi-lo em diversos problemas menores e mais simples. O algoritmo faz isso de maneira recursiva até obter a solução para o problema. As soluções desses subproblemas resolvidos podem ser combinadas para, finalmente, produzir a solução para o problema complexo. (CARVALHO et al., 2011).

Basicamente, a Árvore de Decisão (DT) consiste em uma sequência de regras "se-então", como ilustrado na Figura 2.6. Nessa Figura, é apresentado um problema de uma empresa que oferece entregas com frete grátis apenas em seu estado e se a distância for inferior a 100 quilômetros.

Devido ao fato de se ajustar bem ao conjunto de dados fornecido na etapa de treinamento, as Árvores de Decisão (DT) estão propensas a sofrer com o problema de *"overfitting"*. Quando um modelo sofre de *"overfitting"*, ele é capaz de obter bons resultados na etapa de treinamento, no entanto, não consegue generalizar bem para novos dados, levando a resultados extremamente inferiores em comparação com os obtidos na etapa de treinamento.

Safavian e Landgrebe (1991) discutem o funcionamento de um classificador baseado em Árvore de Decisão (DT). Um ponto crítico que os autores destacam é a construção da árvore propriamente dita. Isso ocorre porque toda essa construção é baseada na análise das amostras dos dados fornecidos na etapa de treinamento, com o objetivo de criar relações entre características e rótulos dos dados. Isso destaca ainda mais a importância de uma boa engenharia de características para que o classificador obtenha bons resultados.

Floresta Aleatória

O algoritmo Floresta Aleatória, conhecido como *Random Forest* (RF) em inglês, é um algoritmo que faz uso do método *ensemble*. Ele combina diferentes modelos

Figura 2.6: Exemplo Árvore de Decisão

Fonte: Elaborado pelo autor, 2024.

simples para alcançar um bom desempenho em suas tarefas, em vez de depender de um modelo complexo para resolver o problema (GÉRON, 2019). O termo "floresta" no nome do algoritmo refere-se à combinação de várias árvores de decisão executadas em paralelo. Cada árvore na floresta realiza uma divisão aleatória das linhas e colunas do conjunto de dados, resultando em várias árvores de decisão simples que operam em diferentes partes do conjunto de dados (CUI et al., 2018).

O algoritmo RF é versátil e pode ser utilizado tanto em tarefas de regressão quanto em tarefas de classificação. Para determinar a classe de um dado não rotulado, ele é submetido a todas as árvores que compõem a floresta. Depois de obter os resultados individuais de cada árvore, é realizado um processo de votação para determinar a classe à qual o dado pertence.

Essa abordagem, utilizando o algoritmo RF, oferece uma ampla diversidade no processo de classificação, resultando em previsões mais satisfatórias, tanto em termos de desempenho do modelo quanto de precisão das previsões.

Extreme Gradient Boosting (XGBoost)

O *Extreme Gradient Boost* (XGBoost) foi introduzido por Chen e Guestrin (2016). Este algoritmo é um sistema de árvore ponta-a-ponta escalável que tem conquistado excelentes resultados em desafios de aprendizado de máquina, como os realizados na plataforma de competição de ciência de dados chamada de Kaggle. É importante

observar que, quando se trata de dados estruturados ou tabulares de pequeno a médio porte, os algoritmos baseados em árvores de decisão são considerados os melhores da categoria no momento.

O XGBoost utiliza um processo de *Boosting* para gerar seus modelos. Ele emprega uma técnica que aprimora as árvores de decisão, tentando corrigir as deficiências do modelo anterior a cada nova iteração. Além disso, o algoritmo se destaca pela sua escalabilidade, permitindo sua execução em máquinas comuns ou em configurações distribuídas. O aprendizado paralelo e distribuído acelera o processo de treinamento, possibilitando a exploração mais eficaz do modelo. Outro aspecto notável deste algoritmo é a capacidade de computação fora do núcleo, que possibilita o processamento de centenas de milhões de exemplos em um computador convencional.

Destaca-se também a prevenção de *overfitting* já incorporada ao algoritmo, que é dividida em três técnicas. A primeira consiste na suavização dos pesos finais aprendidos, fazendo com que a técnica tenda a selecionar um modelo que utiliza funções simples e preditivas. A segunda técnica é denominada *Shrinkage*, que reduz a influência de cada árvore individual, permitindo espaço para futuras árvores aprimorarem o modelo. Por fim, a última técnica é a *Column Subsampling*, que previne o *overfitting* e melhora a velocidade de processamento dos cálculos na versão paralelizada do algoritmo.

Para obter o máximo desempenho do algoritmo, são necessários alguns ajustes de hiperparâmetros. Esses ajustes ocorrem antes da etapa de treinamento e controlam o processo de aprendizado em si. Dentre os hiperparâmetros importantes, destacam-se:

- ***learning_rate*:** este hiperparâmetro também é conhecido como *eta*, e refere-se a taxa de aprendizagem definindo o tamanho do passo usado na atualização para evitar *overfitting*. Após cada etapa de *boosting*, obtém diretamente os pesos das novas características;

- ***n_estimators*:** número de árvores com *gradient boosting*. Equivalente ao número de rodadas de reforço;

- ***max_depth*:** profundidade máxima da árvore. Quanto maior este valor, maior a profundidade máxima da árvore e, portanto, mais complexo o modelo se torna e consequentemente maior a probabilidade de *overfitting*;

- ***subsample*:** proporção de subamostra das instâncias de treinamento;

- ***gamma*:** redução de perda mínima necessária para fazer uma partição em um nó folha da árvore. Quanto maior o valor de *gamma*, mais conservador o algoritmo

será;

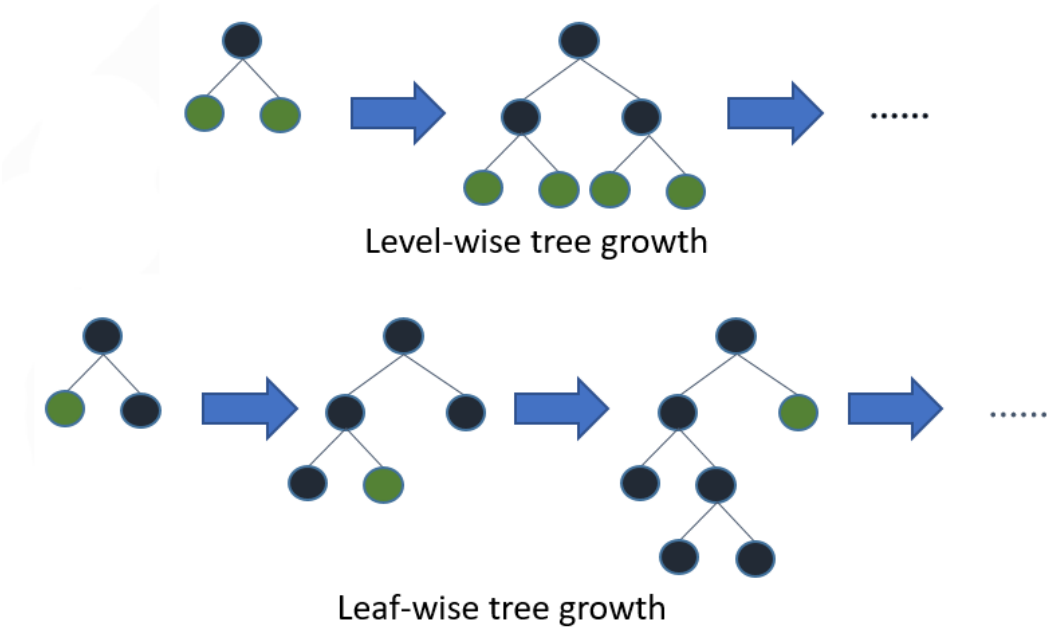
- ***reg_lambda*:** termo de regularização L2 sobre os pesos. Quanto maior o valor que este parâmetro receber, mais conservador o algoritmo se tornará.

LightGBM

O *Light Gradient Boosting Machine* (LightGBM) foi apresentado por Ke et al. (2017) e desenvolvido pela Microsoft. Projetado para possuir alta eficiência e velocidade, usando pouca memória e com a capacidade de lidar com grandes conjuntos de dados, além de oferecer melhor acurácia. Também possui suporte para aprendizado paralelo e em GPUs. O principal objetivo deste algoritmo é acelerar o processo de treinamento, mantendo o mesmo desempenho do XGBoost. Para alcançar esse feito, o algoritmo conta com otimizações durante a construção das árvores (CORPORATION, 2021).

Diferentemente dos algoritmos anteriores, o crescimento da árvore no LightGBM é realizado verticalmente, expandindo em termos de folhas. Esse processo é denominado *"leaf-wise tree growth"*, enquanto nos outros algoritmos, o crescimento é horizontal, expandindo em níveis, conhecido como *"level-wise tree growth"*. O LightGBM seleciona a folha com uma perda maior para crescer, o que pode levar a uma redução mais significativa nas perdas em comparação com algoritmos que se baseiam no crescimento horizontal das árvores. Esse comparativo de crescimento é ilustrado na Figura 2.7.

Figura 2.7: Comparativo de crescimento horizontal e vertical da árvore



Fonte: Extraído de Microsoft Corporation, 2021.

Para aproveitar todas as vantagens apresentadas anteriormente, o LightGBM utiliza duas técnicas: a *Gradient-based One-Side Sampling* (GOSS), responsável pela amostragem dos dados, e a *Exclusive Features Bundling* (EFB), que reduz o número de recursos em conjuntos de dados esparsos durante o treinamento. Da mesma forma que o XGBoost, para obter o melhor desempenho e evitar o *overfitting*, o LightGBM possui hiperparâmetros a serem otimizados. Ao todo, existem mais de 100 hiperparâmetros disponíveis no algoritmo, incluindo

- ***lambda_l1*:** termo de regularização L1;
- ***lambda_l2*:** termo de regularização L2;
- ***num_leaves*:** número máximo de folhas em uma árvore;
- ***feature_fraction*:** seleciona aleatoriamente um subconjuntos de características em cada iteração (árvore). Se definido em 0,7, significa que 70% das características serão escolhidas antes de treinar cada árvore. Este hiperparâmetro pode ser utilizado para acelerar o treinamento do modelo e impedir *overfitting*;
- ***bagging_fraction*:** refere-se a fração de dados que será utilizado em cada iteração, normalmente utilizado para aceleração da velocidade do treino e evitar *overfitting*;

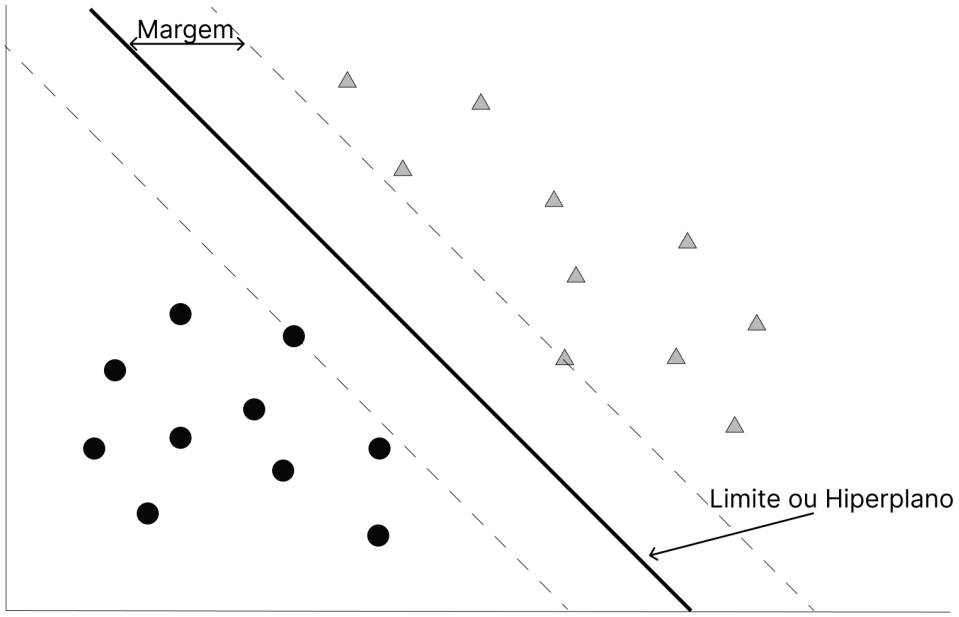
- ***bagging_freq:*** **:** frequência para *bagging*;
- ***min_child_samples*****:** refere-se ao número mínimo de dados em uma folha, podendo ser aplicada para lidar com *overfitting*.

Support Vector Machine

O algoritmo Máquinas de Vetores de Suporte, ou *Support Vector Machine* (SVM) em inglês, é um algoritmo de aprendizado de máquina supervisionado. Ele utiliza um método de classificação baseado em discriminantes lineares de margem máxima, ou seja, tem como objetivo encontrar um hiperplano ótimo que maximize a separação entre as classes analisadas, permitindo assim determinar a qual classe um determinado dado pertence. Além disso, o SVM é capaz de encontrar o limite de decisão não linear ótimo entre as classes, o que corresponde a um hiperplano em algum espaço de alta dimensão não linear por meio do truque de kernel (ZAKI; JR, 2020).

O princípio de funcionamento do SVM é ajustar um limite a uma região de pontos, todos os quais pertencem a uma classe. Uma das vantagens deste algoritmo é que, uma vez definido este limite, a maioria dos dados de treinamento se torna redundante, uma vez que o SVM precisa de um conjunto central de pontos que podem ajudar a identificar e fixar esse limite. Tais pontos são chamados de vetores de suporte, porque suportam o limite. O termo "vetores" vem da ideia de que cada ponto de dados é um vetor, ou seja, uma linha de dados que contém valores para diversos atributos diferentes. Este limite é tradicionalmente chamado de hiperplano, que pode ser uma linha reta ou curva para dados bidimensionais, um plano ou uma superfície complexa irregular para dados tridimensionais, e em dimensões mais altas, que são mais difíceis de visualizar, é chamado de hiperplano. A etapa de teste do modelo ocorre da seguinte forma: para quaisquer novos pontos que precisam ser classificados, é verificado se eles estão dentro do limite previamente definido ou não.(KOTU; DESHPANDE, 2018). A Figura 2.8 ilustra o funcionamento do algoritmo SVM.

Figura 2.8: Separação de classes por um modelo SVM



Fonte: Elaborado pelo autor, 2024.

Uma das desvantagens do SVM é que, em geral, é necessário calcular o produto escalar para cada classificação durante a etapa de treinamento, o que pode resultar em tempos de computação lentos, especialmente em dimensões extremamente altas ou com um grande número de atributos. No entanto, essa desvantagem é compensada pelo fato de que, após a construção do modelo, pequenas alterações nos dados de treinamento não resultarão em alterações significativas nos coeficientes do modelo, desde que os vetores de suporte não mudem. Essa resistência ao *overfitting* é uma das razões pelas quais o SVM se destaca como um dos algoritmos de aprendizado de máquina mais versáteis (KOTU; DESHPANDE, 2018).

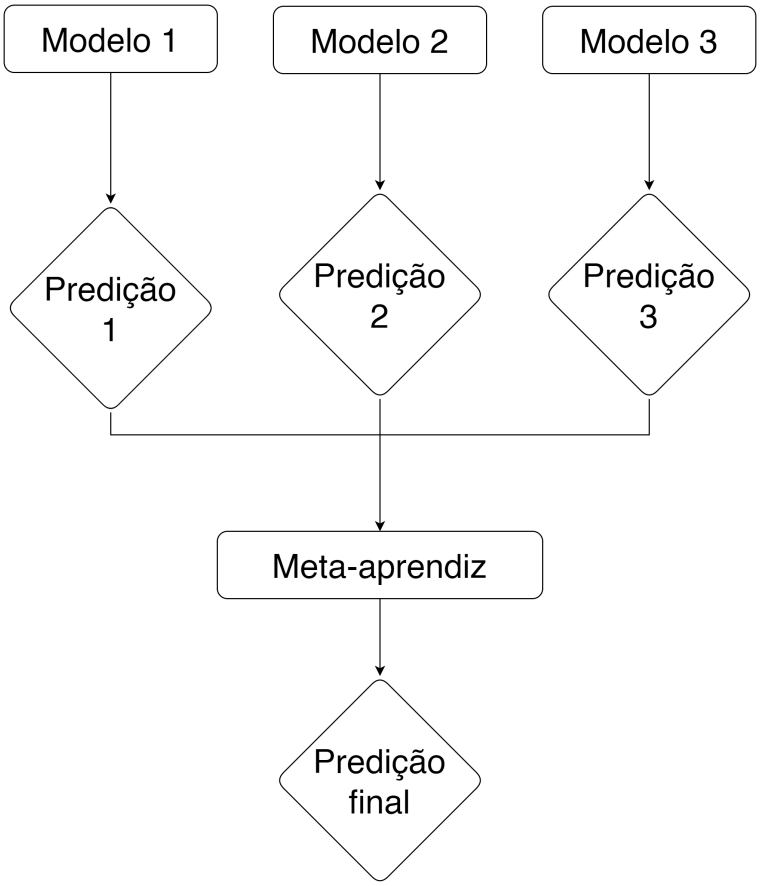
Ensemble Classifier

Um *ensemble classifier*, ou classificador de ensemble, é uma técnica no campo de aprendizado de máquina que envolve a combinação de múltiplos classificadores individuais para aprimorar o desempenho geral na tarefa de classificação. Em vez de depender de um único modelo de aprendizado, os classificadores de ensemble combinam as previsões de vários modelos, aproveitando as forças de cada um e mitigando suas fraquezas. Essa abordagem é fundamentada na premissa de que a combinação de múltiplas perspectivas pode resultar em previsões mais precisas e robustas (GÉRON,

2019).

Existem duas estratégias principais para a construção de ensembles: *"voting"* e *"stacking"*. No *"voting"*, os classificadores individuais emitem seus votos para uma classe específica em uma instância dada, e a classe mais votada é selecionada como a previsão final. Essa técnica é conhecida como *"hard voting"*. No entanto, os classificadores também podem fornecer probabilidades de classe, permitindo a técnica de *"soft voting"*, na qual a classe com a maior probabilidade média é escolhida como a previsão final. O *"stacking"* envolve o uso dos resultados dos classificadores individuais como entradas para outro modelo, que aprende como combinar essas previsões para tomar uma decisão mais informada, conforme apresentado na Figura 2.9. Isso permite que os classificadores de *ensemble* se adaptem a qualquer conjunto de treinamento, resultando em previsões mais precisas.

Figura 2.9: Visualização de um ensemble com método *"stacking"* a partir de três modelos



Fonte: Adaptado de Huyen (2022).

Os classificadores de *ensemble* podem aprimorar o desempenho da classificação ao combinar as previsões de vários modelos, superando as limitações dos modelos individuais e reduzindo o risco de *overfitting* (KOTU; DESHPANDE, 2018). Eles são frequentemente usados em tarefas de aprendizado supervisionado, como classificação e regressão, sendo especialmente úteis em problemas complexos nos quais diferentes modelos podem capturar diferentes aspectos dos dados. No entanto, os *ensembles* também apresentam desafios, como o aumento da complexidade do modelo, o custo computacional e a redução da interpretabilidade.

2.3.2 Aprendizado de Máquina Não Supervisionado

O aprendizado de máquina não supervisionado opera com dados de treinamento não rotulados, buscando identificar padrões nos dados fornecidos e agrupando-os com base em semelhanças (GÉRON, 2019). Dois exemplos clássicos de tarefas de aprendizado de máquina não supervisionado são o agrupamento e a redução de dimensionalidade (GHAHRAMANI, 2003). Além disso, algoritmos não supervisionados podem ser empregados na detecção de anomalias, onde aprendem o padrão dos dados considerados normais e o utilizam para identificar instâncias anormais (GÉRON, 2019).

Os algoritmos de clusterização analisam os dados fornecidos e os agrupam sem necessariamente conhecer os rótulos das classes. Isso os torna ferramentas úteis para tarefas de aprendizado semi-supervisionado e redução de dimensionalidade. Alguns dos algoritmos de clusterização mais notáveis incluem o K-Means, o DBSCAN, a Análise Hierárquica de Cluster (HCA) e o Agrupamento Aglomerativo.

K-Means

O K-Means é um algoritmo simples sendo capaz de agrupar conjuntos de dados de maneira rápida e eficiente, normalmente sendo necessário poucas iterações. Proposto no ano de 1957 por Stuart Lloyd no Bell Labs, mas só foi publicado fora da empresa no ano de 1982 (LLOYD, 1982). Este algoritmo é destinado a agrupar dados que possuem características semelhantes em clusters. (GÉRON, 2019)

Seu funcionamento é baseado em centroides, o que significa que ele define clusters representados por pontos no espaço de características que são centrais em relação aos pontos dos dados. Seu funcionamento é baseado quatro etapas principais: inicialização dos centroides; atribuição de pontos aos clusters; atualização dos centroides e iteração (AHMED; SERAJ; ISLAM, 2020).

A inicialização dos centroides é o primeiro passo do K-Means. Normalmente os

centroides são escolhidos de forma aleatória a partir do conjunto de dados, mas há outras opções como a utilização do `k-means ++` para inicialização. Apesar da facilidade da inicialização aleatória, ela apresenta algumas desvantagens, como o risco de convergir para mínimos locais em vez de mínimos globais, o que pode afetar a eficiência dos clusters. A inicialização por K-Means ++ é uma abordagem mais sofisticada que consiste em dois passos, sendo o primeiro a escolha aleatória do primeiro centroide, o segundo passo é a seleção de centroides subsequentes, no quais são escolhidos baseados em probabilidade ponderada, que é inversamente proporcional à distância do centroide mais próximo já escolhido. Garantindo assim que os centroides iniciais estejam espalhados de maneira uniforme e representativa. Dentre as vantagens deste método de inicialização destaca-se uma maior robustez, devido a esta forma mais inteligente de escolher os centroides iniciais; uma menor probabilidade de mínimos locais, o que consequentemente eleva a chance de encontrar uma solução próxima ao mínimo global resultando assim em melhores resultados de clusterização, tendendo os clusters a apresentarem uma maior qualidade (ARTHUR; VASSILVITSKII, 2007).

A segunda etapa consiste na atribuição de pontos aos clusters, onde cada ponto de dados é atribuído ao cluster cujo centroide é o mais próximo a partir da distância euclidiana.

Na terceira etapa realiza-se a atualização dos centroides. Uma vez que os pontos iniciais já estão atribuídos aos clusters, recalcula-se os centroides de cada cluster como ponto médio utilizando média aritmética dos pontos atribuídos a esse cluster, resultando assim em um deslocamento do centroide para o centro real dos pontos do cluster.

Por fim, o quarto passo é a iteração. Basicamente consiste em repetir a execução da segunda e terceira etapa apresentados anteriormente até que a atribuição dos pontos aos cluster não mude significativamente entre iterações ou que um critério de parada seja atendido, resultado assim numa estabilização dos clusters.

Após apresentado o funcionamento do algoritmo, observa-se que ele funciona com base em um número de cluster que deve ser informado para que ele possa realizar as etapas de aprendizado, este valor é o valor do parâmetro K. Tal valor caso não informado corretamente pode afetar significativamente os resultados da clusterização. Dentre as formas de selecionar o valor de K, será apresentado o "Método do Cotovelo" e o "Método da Silhueta" por serem considerados os métodos estados da arte (MASUD et al., 2018).

A curva do cotovelo (*Elbow Method*) é uma abordagem visual para determinar o valor de K. Envolve a execução do algoritmo K-Means para uma variedade de valores

de K e o cálculo da soma dos quadrados das distâncias entre os pontos de dados e os centroides dos clusters (conhecida como inércia). A Curva do Cotovelo ajuda a identificar o ponto em que a inércia começa a diminuir mais lentamente, formando uma aparência de "cotovelo" na curva. A rápida redução da inércia sugere que os clusters são bem definidos e compactos. Quando o valor da inércia começa a reduzir mais lentamente, a qualidade da clusterização pode estar diminuindo, indicando uma quantidade excessiva de clusters. O ponto na curva onde o valor da inércia começa a nivelar ou formar um cotovelo pode ser escolhido como o valor ideal de K. Entretanto, a interpretação da curva do cotovelo pode ser subjetiva e depende do conjunto de dados (KETCHEN; SHOOK, 1996).

O método da Silhueta é uma métrica que avalia o quão bem os objetos estão agrupados em seus clusters fornecendo uma medida numérica da qualidade dos clusters, variando de -1 a 1. Este cálculo é realizado para um dos pontos em relação ao seu próprio cluster e aos clusters vizinhos mais próximos. Esta métrica considera dois aspectos (ROUSSEEUW, 1987; RODRIGUEZ et al., 2019).

O primeiro aspecto considerado mede a similaridade média do ponto com os outros pontos do mesmo cluster. Quanto maior o valor, mais semelhante o ponto é entre os demais pontos do mesmo cluster. Este aspecto será chamado de *"a"*. O segundo aspecto mede a dissimilaridade média do ponto em relação aos pontos do cluster vizinho mais próximo. Quanto maior este valor, menos semelhante o ponto é aos pontos do cluster vizinho. Este aspecto será chamado de *"b"*.

O coeficiente de Silhueta é calculado da seguinte forma:

$$s = \frac{b - a}{max(a,b)} \tag{2.1}$$

- Se o valor de s estiver próximo de 1, indica que o ponto está bem classificado em seu cluster e distante dos outros clusters.
- Se o valor de s estiver próximo de -1, indica que o ponto pode estar mais apropriado em outro cluster.
- Se o valor de s estiver próximo de 0, indica que o ponto está próximo do limite entre dois clusters e pode estar ambíguo.

O valor médio do coeficiente de Silhueta para todos os pontos de dados em um conjunto é usado para avaliar a qualidade geral da clusterização. O valor máximo da média deste coeficiente quando calculado para diferentes valores de K, normalmente indica o número ideal de clusters.

Apesar de ser um algoritmo rápido e escalável, o K-Means não se comporta muito bem quando os clusters possuem tamanhos variados, densidades diferentes ou formas não esféricas.

2.4 Métricas de Avaliação

A avaliação do desempenho de um modelo é uma etapa importante, pois é nela que podemos inferir a capacidade do modelo de generalizar o aprendizado adquirido para novas instâncias. Há diversas métricas para avaliação do modelo, dentre as quais se destacam a matriz de confusão, acurácia, precisão, *recall*, *F1-score*, além da *"Receiver Operating Characteristic Curve"* (Curva ROC) e da *"Area Under ROC Curve"* (AUC). (GÉRON, 2019).

A matriz de confusão, ilustrada na Figura 2.10, é uma tabela que apresenta dados quantitativos sobre as classes reais e as classes preditas corretamente e erroneamente, bem como a classe que deveria ter sido predita. Essa matriz é composta pelas seguintes informações: Verdadeiro Positivo (VP), que são os dados positivos que foram preditos corretamente; Verdadeiro Negativo (VN), que são dados negativos preditos corretamente; Falso Positivo (FP), que são dados negativos que foram preditos erroneamente como verdadeiros; Falso Negativo (FN), que são dados verdadeiros que foram preditos equivocadamente como negativos. A partir da definição dos termos acima, podemos estabelecer que, para este trabalho, VP e VN serão utilizados para domínios maliciosos e domínios legítimos preditos corretamente, respectivamente, enquanto FP representa domínios legítimos classificados como maliciosos incorretamente, e FN representa domínios maliciosos classificados como legítimos erroneamente.

Figura 2.10: Matriz de Confusão

		Valores Preditos	
		Positivo	**Negativo**
Valores Reais	**Positivo**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Negativo	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Fonte: Elaborado pelo autor, 2024.

A partir da extração das métricas anteriormente apresentadas, torna-se possível calcular métricas mais complexas, como acurácia, precisão, *recall* e F1-score.

A acurácia (ACC), apresentada na equação 2.2, é compreendida como a fração das previsões que o modelo acertou.

$$Acurácia = \frac{VP+VN}{VP+VN+FP+FN} \tag{2.2}$$

A precisão indica, dentre todas as predições de domínios da classe maliciosa que o modelo fez, quantas estão corretas. Tal métrica pode ser calculada pela equação 2.3.

$$Precisão = \frac{VP}{VP+FP} \tag{2.3}$$

O *Recall*, definido pela equação 2.4, também é conhecido como sensibilidade ou revocação e apresenta a proporção de domínios maliciosos preditos, quantos estavam corretos.

$$Recall = \frac{VP}{VP+FN} \tag{2.4}$$

Por fim, o *F1-score* apresenta a combinação entre precisão e *recall*, calculando a média harmônica entre as duas métricas. O *F1-score* é definido pela equação 2.5.

$$F1-Score = 2.\frac{Recall.Precisão}{Recall+Precisão} \tag{2.5}$$

Uma das métricas mais utilizadas para avaliação de desempenho de modelos classificadores binários é a AUC (Área sob a Curva ROC). O valor da AUC é obtido através da Curva ROC *(Receiver Operating Characteristic)*, uma métrica que avalia a capacidade do modelo em diferenciar entre as classes. A Curva ROC é composta por dois eixos: no eixo *x*, temos a Taxa de Falso Positivo (TFP), e no eixo *y*, a Taxa de Verdadeiro Positivo (TVP). (FAWCETT, 2004).

A *TFP*, definida pela equação 2.6, representa a taxa de amostras negativas que foram classificadas erroneamente como positivas. Também pode ser calculada como 1 menos a Taxa de Verdadeiro Negativo (TVN).

$$TFP = \frac{FP}{VN+FP} \tag{2.6}$$

A *TVP*, também conhecida como *recall*, é definida pela Equação 2.4, como apresentado anteriormente, e indica a taxa de amostras positivas que foram classificadas corretamente.

A Taxa de Verdadeiro Negativo (*TVN*), conhecida também como especificidade

(E), é definida na equação 2.7, representando as amostras negativas que foram corretamente classificadas.

$$TVN = \frac{VN}{VN + FP} \tag{2.7}$$

A Taxa de Falso Negativo *(TFN)*, definida pela equação 2.8, ou por *1-TVN*, indica a taxa de amostras negativas que foram erroneamente classificadas.

$$TFN = \frac{FN}{VP + FN} \tag{2.8}$$

A cada *threshold*, calcula-se *TVP* e *TFP* e gera-se a Curva ROC. Portanto, quanto menor a *TFP* e maior a *TVP* por *threshold*, mais a curva ROC estará próxima do canto superior esquerdo, o que indica um melhor desempenho do modelo. Na diagonal do gráfico da Curva ROC, há uma linha pontilhada, que representa um modelo classificador aleatório, acertando, portanto, 50% de suas predições. Um exemplo de Curva ROC é apresentada na Figura 2.11 (FAWCETT, 2004).

Com o objetivo de simplificar a análise da Curva ROC, utiliza-se a *AUC*. Tal métrica é calculada a partir da curva ROC, resultando em um número que corresponde ao valor da Área Sob a Curva ROC (*AUC*), o que torna a comparação entre os modelos mais fácil.

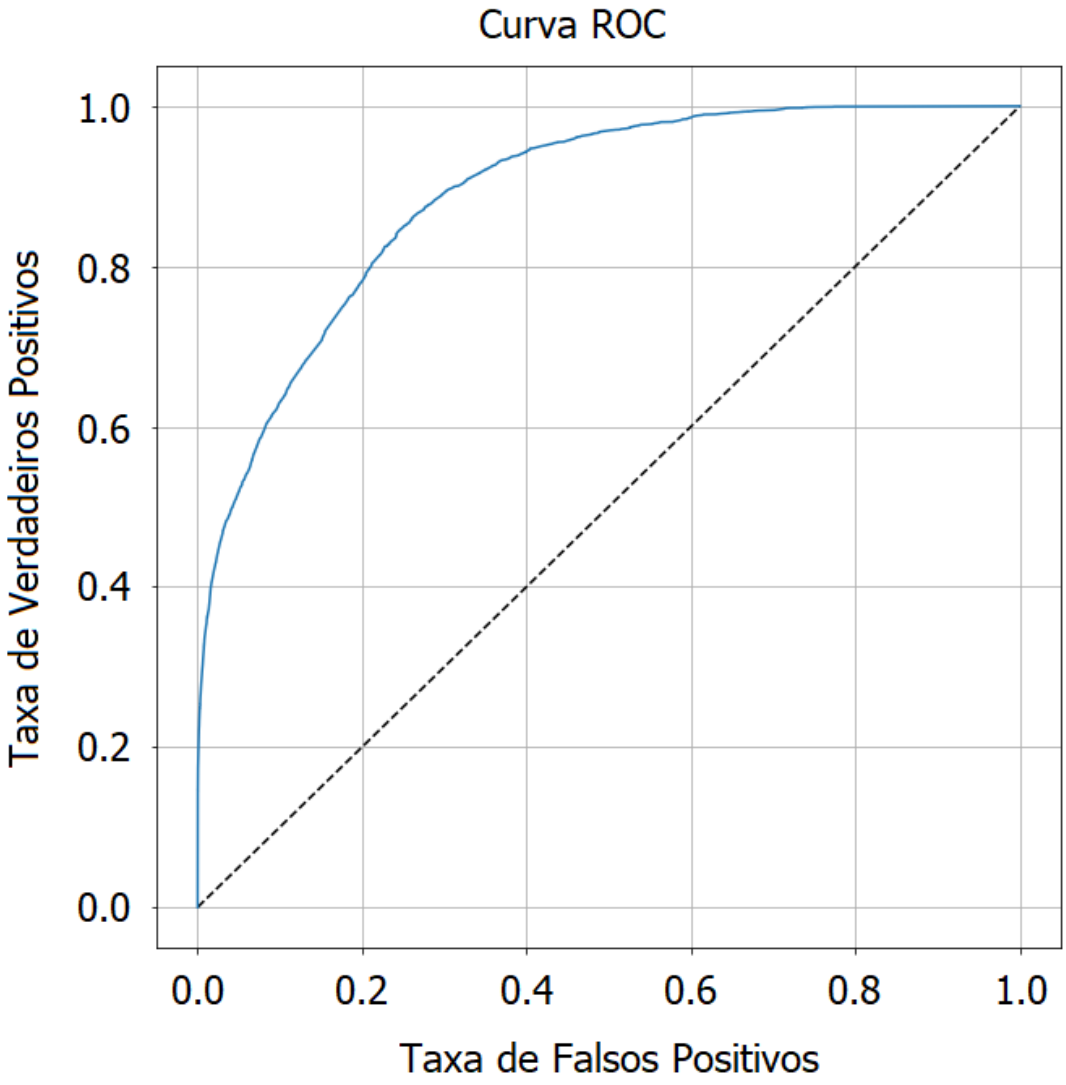
A *AUC* é uma boa métrica para a avaliação do desempenho de modelos classificadores, como apresentado anteriormente. No entanto, ela não deve ser utilizada caso os dados estejam muito desbalanceados, assim como a acurácia. Isso ocorre porque, caso o modelo consiga identificar apenas a classe majoritária, isso já seria suficiente para resultar em uma elevada acurácia. Além disso, a TFP para conjuntos desequilibrados é reduzida a um grande número de verdadeiros negativos. Para trabalhar com classes desbalanceadas, utiliza-se a Curva de Precisão-Recall *(PR)* e a *PR AUC* (DAVIS; GOADRICH, 2006).

2.5 Treino, validação e teste do Modelo

Durante a construção do modelo, é necessário realizar o treinamento para adquirir conhecimento e, em seguida, a etapa de teste para medir o desempenho do modelo. Essa etapa é denominada de treino e teste, e destacam-se duas técnicas: a *hold-out* e a validação cruzada (ou *Cross-validation* em inglês).

A técnica *hold-out* é mais simples, pois envolve a divisão do conjunto de dados em treino e teste. Normalmente, utiliza-se uma proporção de 80% dos dados para treina-

Figura 2.11: Exemplo de Curva ROC

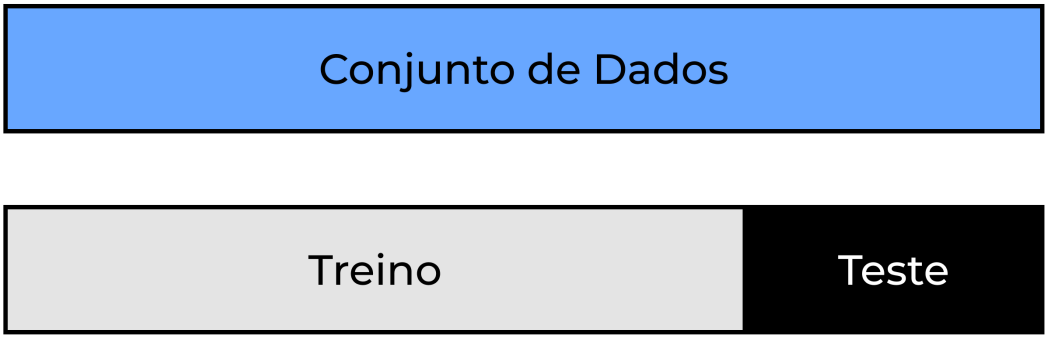


Fonte: Elaborado pelo autor, 2024.

mento e 20% para testes, ou 70% para treinamento e 30% para testes, por exemplo. O modelo é ajustado aos dados de treinamento, cujas classes são conhecidas, e, a partir dele, tenta-se fazer previsões nos dados de teste. A Figura 2.12 ilustra a técnica *hold-out* (RASCHKA, 2018).

Uma técnica mais completa e amplamente utilizada para a validação de modelos é conhecida como *K-Fold cross-validation*. A principal vantagem dessa técnica de validação cruzada é que cada amostra do conjunto de dados tem a oportunidade de ser testada (RASCHKA, 2018). Nessa técnica, o conjunto de dados é dividido em K partes, e o processo é iterado K vezes. Em cada iteração, treina-se o modelo com K-1 partes e valida-se o desempenho com a parte não utilizada para o treinamento, como

Figura 2.12: Hold-out



Fonte: Adaptado de Raschka (2018)

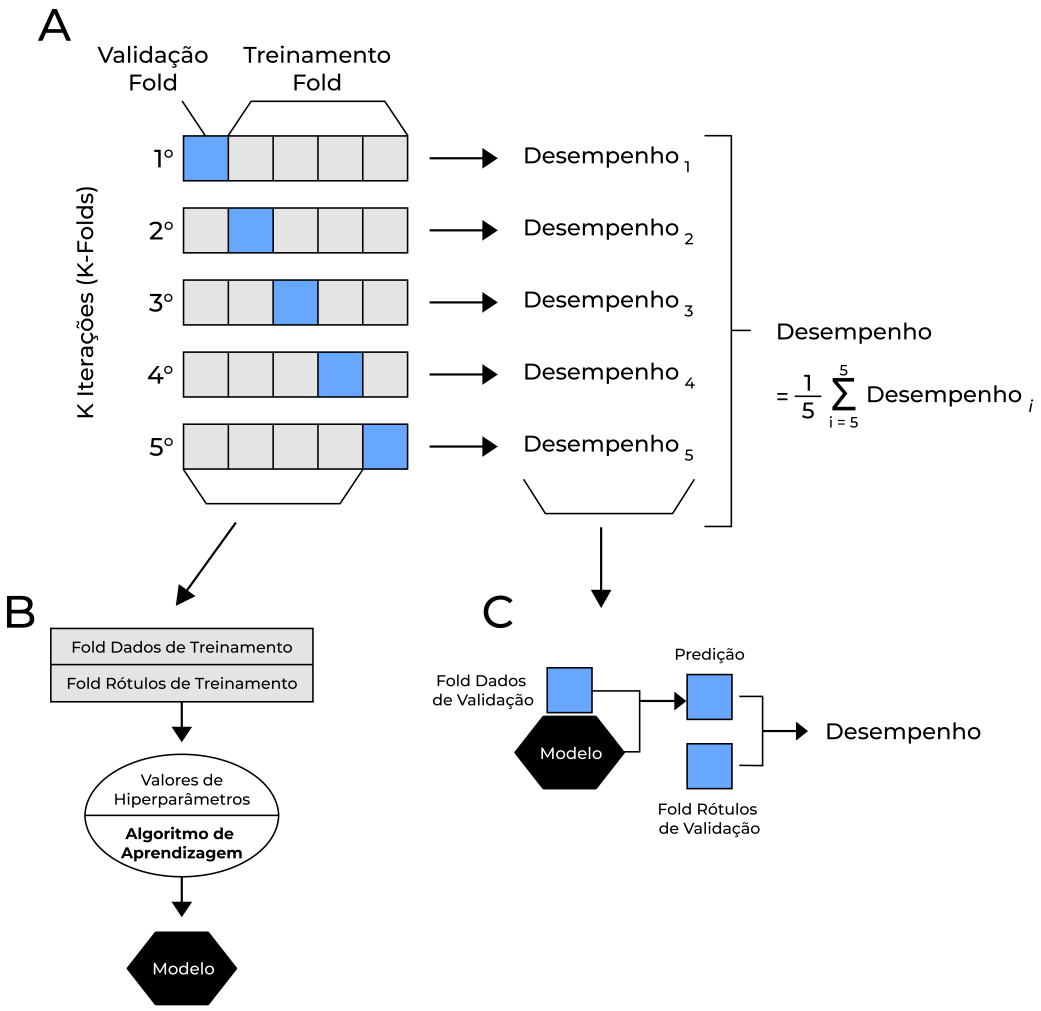
exemplificado na Figura 2.13.

Um problema inerente à técnica *K-Fold Cross Validation* é que, ao dividir esses *folds*, não é garantido que haverá a mesma proporção de dados para cada classe em comparação com o conjunto de dados original, uma vez que a divisão é feita de forma aleatória. Para resolver esse problema, existe uma extensão desse método chamada de *stratified cross-validation*. Nesse método, ao dividir o conjunto de dados em *K folds*, cada classe presente é distribuída de forma uniforme por esses *folds*, resultando em distribuições semelhantes às do conjunto de dados original e preservando, assim, a proporção de classes (BREIMAN JEROME FRIEDMAN, 1984).

2.6 Balanceamento de Classes

O desbalanceamento de classes ocorre em conjuntos de dados nos quais há uma quantidade muito maior de dados em uma classe em comparação com as outras classes. Isso é comum em problemas de detecção de fraudes, e também se aplica à detecção de domínios maliciosos, onde há significativamente mais domínios legítimos registrados em comparação com o número de domínios maliciosos. Esse desequilíbrio afeta o desempenho dos algoritmos de aprendizado supervisionado convencionais, pois esses modelos tendem a ficar enviesados em direção à classe majoritária e, como resultado, têm dificuldade em classificar dados das classes minoritárias. Para resolver esse problema, é necessária a aplicação de técnicas de balanceamento de classes. Essas técnicas envolvem a reamostragem dos dados antes de treinar o algoritmo, para que o modelo seja treinado com dados equilibrados. As técnicas de reamostragem são geralmente classificadas em três grupos: sobreamostragem, subamostragem e métodos

Figura 2.13: *K-fold Cross Validation*



Fonte: Adaptado de Raschka (2018)

híbridos (FERNÁNDEZ et al., 2018).

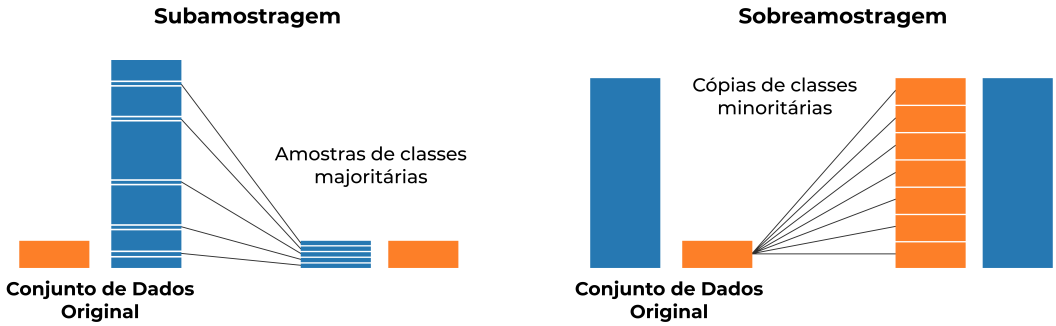
O balanceamento de classes usando técnicas de sobreamostragem envolve a criação de réplicas dos dados originais da classe minoritária ou a geração de dados sintéticos para essa classe, resultando em um conjunto de dados expandido em relação ao conjunto de dados original. Um exemplo notável de algoritmo nessa categoria é o *K-Means SMOTE* (DOUZAS; BACAO; LAST, 2018).

Os métodos de balanceamento de classes por subamostragem resultam em um subconjunto do conjunto de dados original, reduzindo o número de amostras da classe majoritária até que seja alcançado um equilíbrio entre as classes. Um exemplo notável de algoritmo nessa categoria é o *"Random Undersampling"* (RUS) (BATISTA; PRATI; MONARD, 2004).

Além disso, existem técnicas de balanceamento de classes que utilizam métodos híbridos, ou seja, combinam abordagens de sobreamostragem e subamostragem para equalizar o conjunto de dados. Esses métodos híbridos podem ser eficazes na resolução de problemas de desbalanceamento de classes, permitindo que o modelo seja treinado com um conjunto de dados mais equilibrado.

A Figura 2.14 apresenta métodos de sobreamostragem e subamostragem.

Figura 2.14: Sobreamostragem e subamostragem



Fonte: Adaptado de Vaz (2019)

Um problema decorrente do balanceamento de classes é que, ao aplicar técnicas de subamostragem, corre-se o risco de perder informações importantes para o aprendizado do modelo, uma vez que dados são removidos. Se forem aplicadas técnicas de sobreamostragem, isso pode resultar em *overfitting* do modelo, pois os dados gerados por essa técnica são sintéticos ou réplicas dos dados originais.

2.7 Redução de Dimensionalidade

A redução de dimensionalidade visa diminuir a distância em um espaço latente entre as distribuições de diferentes conjuntos de dados. Observa-se também que os resultados obtidos com a ajuda da redução de dimensionalidade são preferíveis àqueles em que a técnica não foi aplicada (Salih Hasan; ABDULAZEEZ, 2021; GEORGE; VIDYAPEETHAM, 2012).

Entre as vantagens da aplicação da redução de dimensionalidade em conjuntos de dados, destacam-se:

a) Redução do número de dimensões e espaço de armazenamento de dados;

b) Requer menos tempo de processamento;

c) Dados irrelevantes, ruidosos e redundantes podem ser excluídos;

d) A qualidade dos dados pode ser otimizada;

e) Ajuda os algoritmos a funcionarem de forma mais eficiente e melhora sua acurácia;

f) Simplifica a classificação e melhora o desempenho.

Existem diversas técnicas para realizar a redução de dimensionalidade dos dados, das quais este trabalho destaca a mais usada, denominada PCA (Análise de Componentes Principais).

2.7.1 Principal Component Analysis

Originalmente proposta por Karl Pearson em 1901, a Análise dos Componentes Principais (PCA), do inglês *Principal Component Analysis*, é uma técnica que visa descobrir como as variáveis se relacionam entre si. A ideia principal por trás do PCA é combinar várias variáveis preditoras numéricas em um conjunto menor de variáveis, que são combinações lineares ponderadas do conjunto original. Esse conjunto menor de variáveis é conhecido como "componentes principais" e ajuda a explicar a maior parte da variabilidade presente no conjunto completo de variáveis, ao mesmo tempo que reduz o espaço dimensional dos dados (BRUCE; BRUCE, 2019).

Os componentes principais são calculados por meio de um método estatístico clássico baseado na matriz de correlação dos dados ou na matriz de covariância. Esse cálculo é realizado de forma eficiente, sem depender de iterações (BRUCE; BRUCE, 2019).

2.8 Levantamento Bibliográfico

Diversos trabalhos têm sido desenvolvidos com o objetivo de realizar a detecção de domínios maliciosos utilizando pDNS. Esses trabalhos se diferenciam em termos de abordagem, como o algoritmo escolhido, as características extraídas do conjunto de dados pDNS, bem como a fonte dos dados, que pode ser a partir de servidores recursivos ou autoritativos de DNS. Essas diferentes abordagens oferecem visões distintas do mesmo problema e diferentes maneiras de abordá-lo. Nesta seção, apresentam-se trabalhos correlatos que possuem uma conexão mais direta com o problema explorado neste estudo.

Sabe-se que a coleta de dados de DNS passivo é uma maneira de armazenar informações de forma anônima sobre consultas e respostas de domínios em registros DNS específicos (WEIMER, 2005).

Antonakakis et al. (2010) apresentaram o primeiro sistema de reputação dinâmica chamado *Notos*. Eles assumiram que domínios legítimos receberiam uma pontuação

maior do que domínios maliciosos. Para construir o sistema, eles usaram recursos baseados em rede, zonas e evidências para gerar a reputação do domínio. Usando uma base de dados de pDNS de um ISP com tráfego DNS de 1,4 milhão de usuários e o algoritmo DT *(Decision Tree)*, o sistema conseguiu detectar domínios maliciosos com uma Taxa de Verdadeiro Positivo (TVP) de 96,8% e uma Taxa de Falso Positivo (TFP) de 0,38%.

O *Exposure*, apresentado por Bilge et al. (2011), foi projetado para detectar vários tipos de domínios maliciosos relacionados a *spam*, *malware*, F*ast Flux Domains* e DGAs. Eles utilizaram dados de pDNS coletados em servidores DNS recursivos para extrair 15 recursos, incluindo 9 inéditos. Esta abordagem exigiu menos tempo de treinamento em comparação com o Notos. Os autores usaram o algoritmo *Decision Tree* (DT) para classificação desses domínios.

Em contraste com os trabalhos anteriores, Antonakakis et al. (2011) apresentaram o *Kopis*, que utiliza dados de pDNS coletados de servidores TLDs *(Top-Level Domains)* e servidores de nome de domínio autoritativo, proporcionando uma visibilidade global. O objetivo do *Kopis* é detectar domínios relacionados a *malware*. Os dados de pDNS coletados são separados em épocas, onde cada época corresponde a um dia, e os recursos são calculados diariamente. Os autores optaram pelo uso do algoritmo *Random Forest* (RF) para a classificação.

Em um estudo de Lison e Mavroeidis (2017), foi apresentada uma abordagem neural para a classificação de domínios. Os autores usaram uma base de dados de pDNS, juntamente com duas redes neurais: uma Rede Neural Recorrente (RNN) e uma Rede Neural Feedforward. O sistema é capaz de identificar domínios em três tipos: maliciosos, legítimos ou *"sinkhole"*. Os autores afirmam que foram os primeiros a usar abordagens neurais para identificação de domínios. Eles obtiveram resultados promissores, com o melhor modelo nos experimentos alcançando um F1-score de 0,96 na detecção de domínios maliciosos.

No trabalho de Weber, Wang e Zhou (2018), uma abordagem não supervisionada para a detecção de campanhas de domínios maliciosos é apresentada. Essas campanhas geralmente envolvem conjuntos massivos de domínios registrados e implantados simultaneamente. Os autores destacam os algoritmos DBSCAN e Agglomerative Clustering com conectividade restrita como aqueles que obtiveram os melhores resultados nessa tarefa.

Bao, Wang e Lan (2019) propõem uma abordagem para detectar domínios maliciosos relacionados a DGAs, bem como domínios pornográficos, com base no vetor de palavras em combinação com o ambiente de rede chinês. Eles utilizam o algoritmo

XGBoost e aplicam o Downsampling para reduzir o desequilíbrio nos dados coletados ao longo de 3 dias. Isso reduz a proporção de 1:20 para 1:5.

O sistema KSDOM, proposto por Wang et al. (2020), utiliza o algoritmo de classificação CatBoost e o método de resampling K-Means SMOTE para tratar o desbalanceamento dos dados. Nessa abordagem, os autores combinam dados de pDNS e DNS ativo para detecção, com um conjunto de treinamento contendo 10.000 domínios legítimos e 1.000 maliciosos.

Silveira et al. (2020) utilizam apenas o tráfego DNS como fonte de dados, extraindo 15 recursos exclusivos desse tráfego para a detecção de domínios maliciosos com o algoritmo XGBoost. Os autores usam o método de undersampling para equilibrar as classes e também aplicam a otimização Bayesiana para selecionar os melhores valores de hiperparâmetros no algoritmo. Seus experimentos demonstram que o modelo não sofre de overfitting, com uma média de AUC de 0,9763.

Em um estudo bioinspirado, Darwish, Anber e Mesbah (2021) utilizaram características baseadas em URL e pDNS para detectar domínios maliciosos. Eles empregaram a técnica Artificial Bee Colony (ABC) para a seleção eficiente de recursos e para a classificação. O algoritmo ABC foi usado para agrupar os dados, resultando em uma acurácia de 96,6%. As características usadas neste trabalho foram apresentadas em Watkins et al. (2017).

O *PREMADOMA*, apresentado por Desmet et al. (2021), é um sistema projetado para atuar na etapa de pré-registro de domínios no TLD europeu `(.eu)` com o objetivo de prevenir o registro de domínios maliciosos. Os autores relatam um conjunto de dados desequilibrado, no qual apenas 2,53% dos domínios são maliciosos, e aplicaram técnicas de subamostragem para equilibrar as classes. Eles desenvolveram uma abordagem semi-supervisionada que envolveu o uso do algoritmo de clusterização *Agglomerative Clustering*, devido à sua capacidade de trabalhar com distâncias personalizadas. O algoritmo supervisionado empregado foi o *Partial Decision Tree* (PART), que constrói árvores de decisão parciais C4.5. Por fim, os autores aplicaram a votação por maioria para criar conjuntos de preditores.

Silveira et al. (2022) apresentam uma abordagem capaz de detectar domínios maliciosos recém-registrados 72 horas após sua primeira consulta. Eles utilizaram pDNS coletado de um servidor TLD e extraíram 20 características, das quais 5 foram exclusivas deste trabalho. Os autores observaram um desequilíbrio significativo em seu conjunto de dados, e para o treinamento do modelo, utilizaram uma abordagem de balanceamento híbrido, combinando as técnicas de *Cluster Centroids* e *K-Means SMOTE* para subamostragem e superamostragem, respectivamente. Também aplicaram otimi-

zação bayesiana para ajustar os hiperparâmetros do modelo. O desempenho do modelo na etapa de treinamento resultou em uma AUC de 0,97630 e um F1-score de 0,905. Na etapa de validação, alcançaram uma TVP de 0,8656 e uma TFP de 0,3471.

Por fim, proposto por Gao et al. (2024), o SemiDom é um modelo semi supervisionado de detecção de domínios maliciosos baseado em meta pseudo-rotulagem, utilizando dados de tráfego DNS. Os autores exploram métodos para detectar domínios maliciosos na Internet, focando em abordagens associativas. Embora eficazes, esses métodos enfrentam desafios ao lidar com domínios maliciosos isolados e dependem fortemente de dados rotulados para precisão. Este modelo emprega redes neurais para gerar pseudo rótulos em dados não rotulados, aumentando assim a quantidade de dados disponíveis para treinamento. Além disso, o SemiDom otimiza continuamente uma rede professora com base no *feedback* de desempenho de uma rede estudante, melhorando a geração de pseudo rótulos precisos. Experimentos demonstram que com apenas 50% dos dados rotulados, o SemiDom alcança resultados superiores, com uma precisão de 0,9610, recall de 0,9789 e F1-Score de 0,9676, comparado a métodos que requerem o uso de 100% dos dados rotulados.

2.9 Considerações Parciais

O capítulo apresenta a fundamentação teórica que sustenta este trabalho, além de destacar diferentes abordagens desenvolvidas para detectar domínios maliciosos. Observa-se que diversos trabalhos, como (ANTONAKAKIS et al., 2010), (BILGE et al., 2011), (ANTONAKAKIS et al., 2011), (SILVEIRA et al., 2020), (SILVEIRA et al., 2022) e (BAO; WANG; LAN, 2019), desenvolveram suas abordagens com base em aprendizado de máquina supervisionado, e o uso de pDNS é uma prática comum, evidenciando a importância desse recurso para a detecção de domínios maliciosos.

Uma semelhança entre os trabalhos é o uso de algoritmos baseados em árvores de decisão, como *Decision Trees* (DT), *Random Forest* (RF), *XGBoost*, *LightGBM* e *CatBoost*, que demonstraram fornecer resultados sólidos para esses trabalhos. As abordagens variam em termos de recursos, como a forma como são extraídos e enriquecidos.

É interessante notar que apenas o *Kopis* (ANTONAKAKIS et al., 2011) e o estudo de Silveira et al. (2022) utilizam pDNS coletado de servidores de domínio de nível superior (TLD), proporcionando uma visão global e única em comparação com os demais modelos. Além disso, o estudo de Silveira et al. (2022) demonstra uma preocupação com a detecção precoce de domínios maliciosos na fase de pós-registro,

detectando esses domínios 72 horas após a primeira consulta.

No que diz respeito às abordagens não supervisionadas e semi-supervisionadas, destacam-se o uso de algoritmos como *Agglomerative Clustering* e *DBSCAN*, que demonstraram bons resultados em seus respectivos trabalhos. A abordagem semi-supervisionada apresentada por Desmet et al. (2021) não faz uso de recursos do DNS; em vez disso, os autores desenvolveram a abordagem para funcionar na fase de pré-registro de domínios, antes que eles se tornem disponíveis para usuários da Internet. Eles se destacam pela combinação eficaz de algoritmos de clusterização com algoritmos baseados em árvores de decisão, o que resultou em excelentes resultados. Por fim, destaca-se o estudo de Gao et al. (2024), que emprega duas redes neurais, uma professora e uma aluna, utilizando uma abordagem associativa por meio de grafos, alcançando excelentes resultados mesmo com apenas 50% dos dados rotulados. No entanto, essa abordagem considera o tráfego DNS, que fornece informações abrangentes sobre as interações entre clientes, resolvedores e servidores DNS de alto nível. Os dados são extraídos e combinados, diferentemente deste trabalho, que se concentra exclusivamente na primeira consulta DNS ao servidor autoritativo do TLD.

A abordagem adotada neste trabalho difere das outras apresentadas no estado da arte, pois se concentra na detecção da maliciosidade de um domínio já em sua primeira consulta DNS ao servidor de domínio de nível superior (TLD). Isso contrasta com as abordagens em que as consultas DNS eram agregadas por domínio, como visto em (SILVEIRA et al., 2022) e (ANTONAKAKIS et al., 2011). A fonte de dados utilizada neste trabalho se assemelha à visão adotada por *Kopis* (ANTONAKAKIS et al., 2011) e o estudo de (SILVEIRA et al., 2022), uma vez que ambos fazem uso de pDNS coletado de servidores TLD.

Outro destaque desta pesquisa é a combinação de técnicas de aprendizado de máquina supervisionado e não supervisionado, em que o resultado final é apresentado por um terceiro modelo, como um algoritmo de *Ensemble*, semelhante ao que foi feito por (DESMET et al., 2021). No entanto, enquanto esses autores aplicaram essa abordagem na fase de pré-registro, nesta tese, ela será aplicada na detecção de domínios maliciosos em sua primeira consulta DNS ao servidor autoritativo. E essa combinação de modelos resulta em um desempenho sólido, com uma baixa taxa de TFP e uma alta TVP, uma vez que os modelos estarão operando em conjunto e fornecendo uma decisão consensual sobre a classe do domínio.

Uma preocupação central neste trabalho é o equilíbrio das classes, e, para atacar esse problema, optou-se por uma abordagem híbrida, combinando técnicas de sobreamostragem e subamostragem. Além disso, destaca-se o uso de algoritmos e infraes-

trutura escaláveis, o que torna este trabalho potencialmente reprodutível em diferentes TLDs, suportando, assim, o tráfego DNS desses domínios.

Capítulo 3

Metodologia

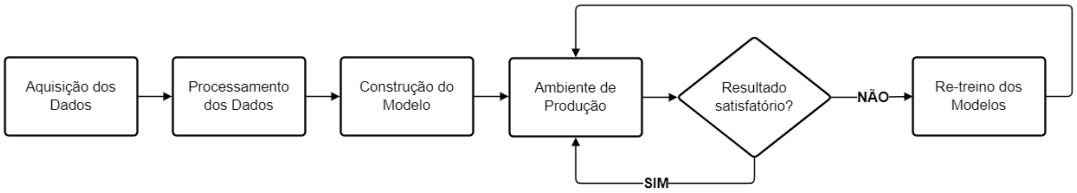
Este capítulo tem como objetivo apresentar a metodologia utilizada neste trabalho. Todos os experimentos foram conduzidos em uma máquina servidora com processador Intel Xeon E52650 v3, 32 GB de memória RAM, armazenamento de 300 GB (SSD) e sistema operacional Ubuntu Server 18.04 LTS.

A linguagem de programação escolhida para o desenvolvimento foi o Python na versão 3.6, e o ambiente de desenvolvimento integrado (IDE) utilizado foi o *Jupyter Notebook*. Dentre as bibliotecas utilizadas, destaca-se o *Apache Spark*, empregado para a extração e filtragem dos dados no ENTRADA (WULLINK et al., 2016). Para o tratamento, transformação e manipulação dos dados, bem como a extração das características, foi utilizado o *Pandas*. O *Imbalanced-learn* foi empregado para o balanceamento das classes, enquanto o *Scikit-learn* foi utilizado para a etapa de treinamento, teste e métricas de avaliação dos modelos.

Além dos algoritmos contidos na biblioteca *Scikit-learn*, também utilizou-se as bibliotecas XGBoost e LightGBM para o treinamento de seus respectivos algoritmos.

O desenvolvimento envolve cinco módulos essenciais, que vão desde a aquisição de dados, processamento, construção de modelos, testes em um ambiente de produção com a inserção de novos dados para avaliar o desempenho dos modelos, até a extração de métricas para determinar se é necessário re-treinar os modelos em produção, conforme ilustrado na Figura 3.1.

Figura 3.1: Diagrama geral da aplicação do sistema



Fonte: Elaborado pelo autor, 2024.

Na seção 3.1, será apresentada uma visão detalhada do sistema desenvolvido. A seção 3.2 abordará o processo de aquisição dos dados utilizados neste trabalho. Na seção 3.3, serão discutidos os procedimentos de processamento dos dados adquiridos, incluindo a rotulação dos dados e informações detalhadas sobre o conjunto de dados. A seção 3.4 apresentará a etapa de construção dos modelos que compõem a abordagem semi-supervisionada proposta. A seção 3.5 descreverá o ambiente de produção e o comportamento dos modelos mencionados anteriormente. Na seção 3.6, será apresentado o último módulo da abordagem proposta, que envolve o re-treinamento dos modelos caso o desempenho deles venha a deteriorar com o tempo. Na seção 3.7, será discutida a metodologia dos dados, como a análise dos dados de treinamento, balanceamento dos dados e redução de dimensionalidade. Por fim, na seção 3.8, serão apresentadas as considerações parciais sobre a metodologia.

3.1 Visão Geral do Sistema

O sistema desenvolvido tem como objetivo realizar a detecção precoce de domínios maliciosos em um TLD, e os dados necessários para este fim serão adquiridos por meio da coleta de pDNS. Assim, como ilustrado na Figura 3.1, o processo tem início com a coleta desses dados.

Após a coleta de dados no Módulo I, o Módulo II é acionado para realizar o processamento dos dados que serão utilizados no Módulo III, o qual se dedica à construção dos modelos, uma vez que a abordagem proposta combina modelos supervisionados e não supervisionados.

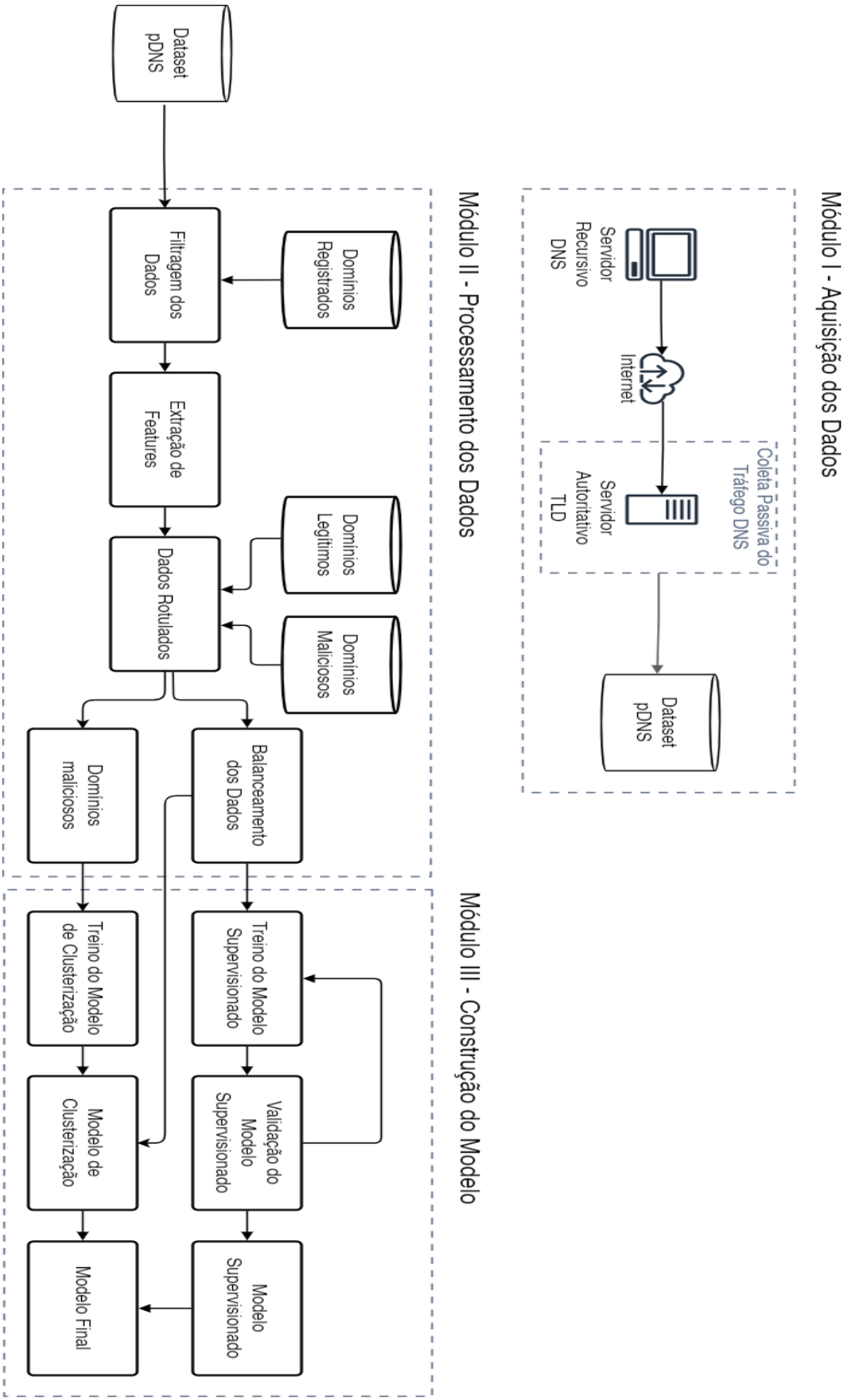
Com os modelos gerados no Módulo III, a abordagem é testada no ambiente de produção (Módulo IV) com novos dados provenientes do DNS passivo, que incluem domínios recém-registrados no TLD a cada dia.

Por fim, a última etapa envolve uma avaliação do desempenho da abordagem. Se for observado que os modelos estão se tornando obsoletos, com queda de desempenho na detecção de domínios maliciosos, devido a mudanças no comportamento de

usuários mal-intencionados, ocorrerá uma etapa de re-treinamento. Isso permitirá que os modelos se adaptem ao novo comportamento, mantendo assim o desempenho na detecção precoce de domínios recém-registrados.

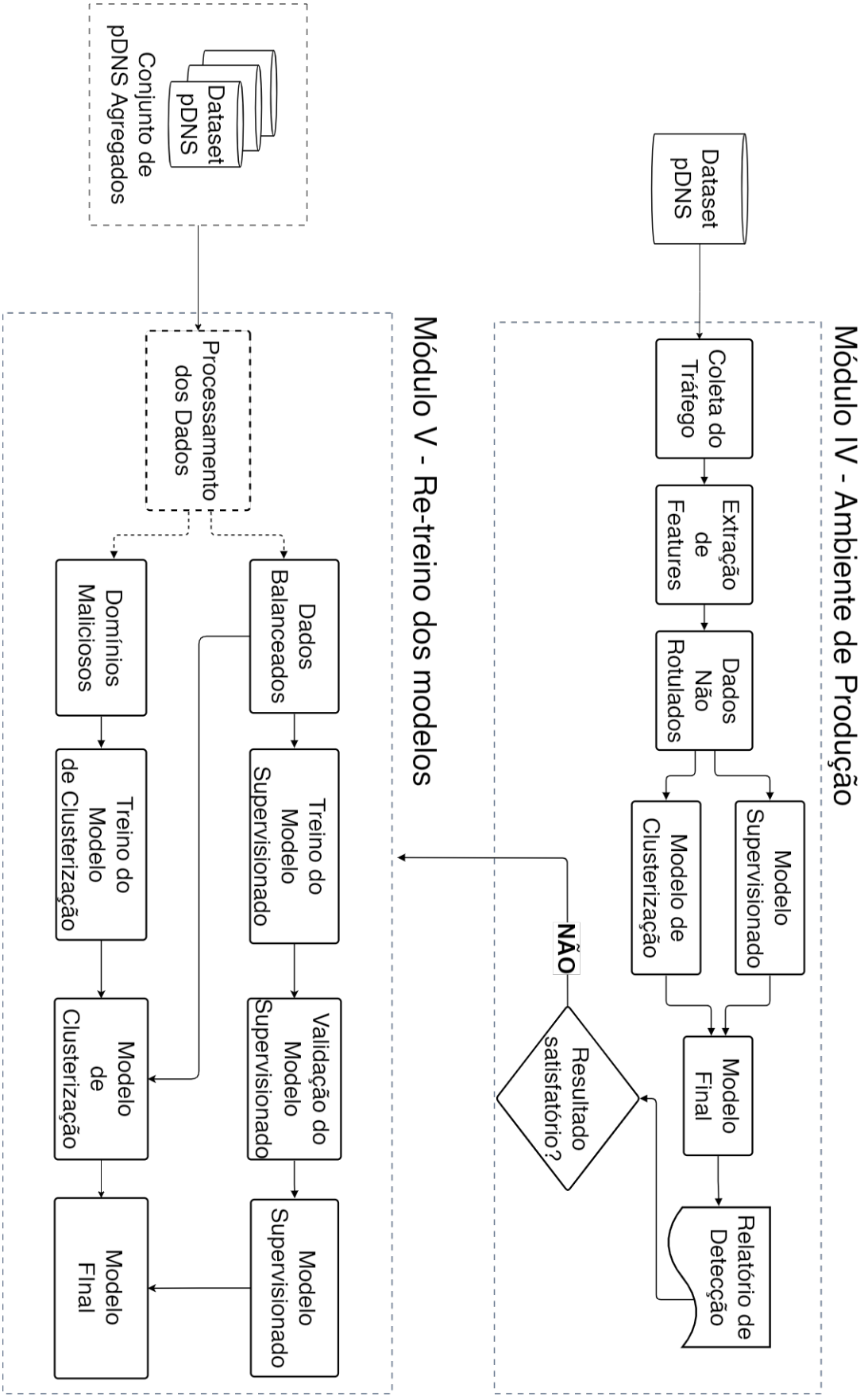
Nas Figuras 3.2 e 3.3, apresenta-se um diagrama detalhado dos módulos que compõem esta abordagem. Nas próximas subseções, cada um desses módulos será analisado em detalhes para fornecer uma compreensão mais aprofundada do funcionamento do sistema.

Figura 3.2: Diagrama de funcionamento do sistema detalhado parte 1



Fonte: Elaborado pelo autor, 2024.

Figura 3.3: Diagrama de funcionamento do sistema detalhado parte 2

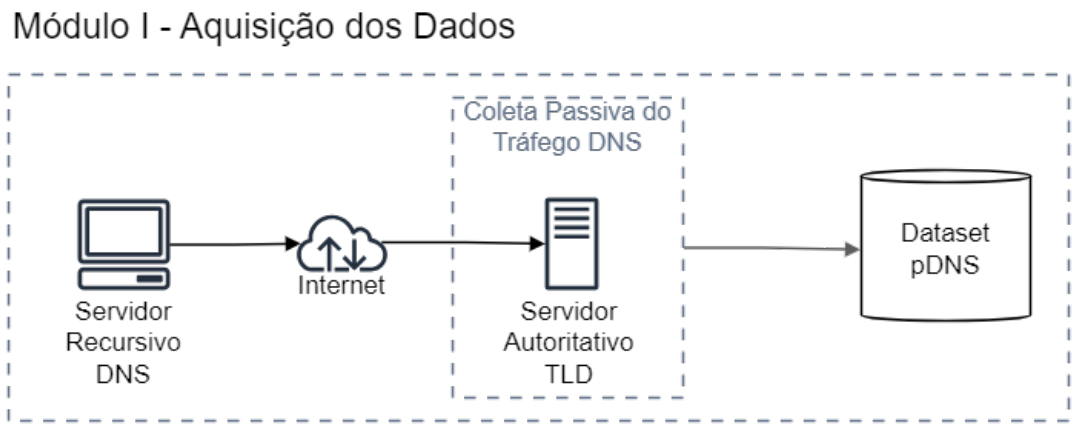


Fonte: Elaborado pelo autor, 2024.

3.2 Aquisição dos Dados

A Figura 3.4 ilustra o primeiro módulo deste trabalho denominado de “Aquisição dos dados”. A seguir uma descrição deste módulo é apresentada.

Figura 3.4: Diagrama de funcionamento da etapa de aquisição dos dados



Fonte: Elaborado pelo autor, 2024.

Como explicado anteriormente no Capítulo 2, a consulta DNS envolve um usuário acessando um site registrado sob um TLD específico. Quando um usuário digita o nome de domínio desejado, uma mensagem é enviada a um servidor recursivo DNS, que inicia a busca pelo IP correspondente ao domínio, começando pelos servidores raíz.

O segundo passo dessa consulta é enviar uma mensagem ao servidor TLD do domínio. É nesse ponto que o pDNS é coletado, registrando a consulta realizada e a resposta enviada pelo servidor autoritativo do TLD. Para realizar essa coleta, estamos utilizando o ENTRADA (WULLINK et al., 2016).

A coleta é realizada por meio de um *sniffer* instalado na rede, que captura os pacotes DNS em formato PCAP em janelas de 5 minutos e os envia ao ENTRADA. O formato PCAP, abreviação de *"Packet Capture"*, é um formato de arquivo padrão usado para armazenar dados de captura de pacotes de rede. Ele permite a gravação de todos os detalhes dos pacotes capturados, incluindo informações sobre a origem, destino, protocolo, horário e conteúdo dos pacotes (SIKOS, 2020). Esses arquivos PCAP são então enviados ao ENTRADA, que os converte para um formato de dados otimizado em colunas chamado *Apache Parquet*. Este formato de arquivo colunar é eficiente para armazenar grandes conjuntos de dados estruturados. *O Apache Parquet* é um formato de arquivo colunar projetado para armazenar grandes conjuntos de dados de forma

eficiente. Em vez de armazenar os dados em linhas, como em um banco de dados tradicional, o *Parquet* armazena os dados em colunas. Isso proporciona uma compressão eficiente e melhora o desempenho de consultas, especialmente em cenários onde apenas algumas colunas são necessárias para uma determinada consulta. Além disso, o *Parquet* suporta compressão de dados, o que reduz ainda mais o espaço de armazenamento necessário e ajuda a acelerar o processamento de consultas (WHITE, 2015).

O ENTRADA não apenas converte os dados, mas também os enriquece e os armazena em um banco de dados distribuído, criando assim o conjunto de dados pDNS.

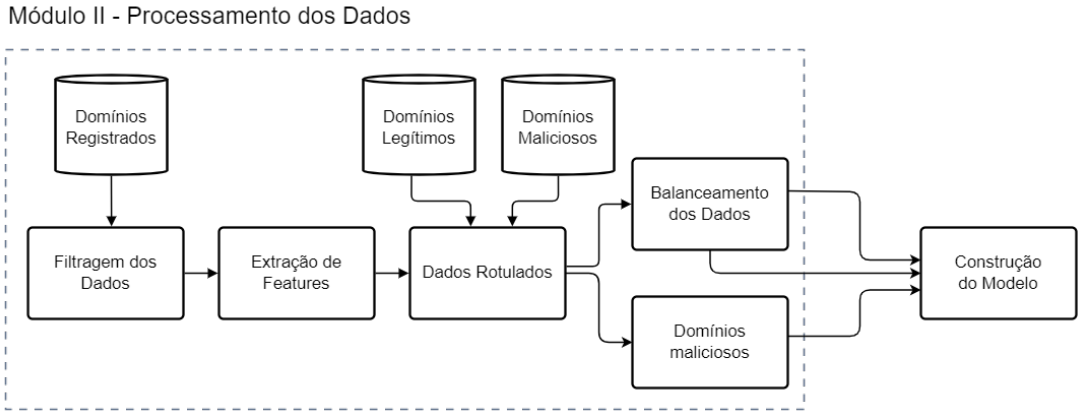
Cada consulta e resposta coletada no servidor TLD é enriquecida com informações como geolocalização, ASN, detecção de *resolvers* públicos e tempo de ida e volta TCP (RTT).

Entre as vantagens de usar o ENTRADA, destacam-se a utilização do banco de dados distribuído *Hadoop*, que é altamente compatível com o ecossistema *Hadoop*. Outra vantagem é a utilização de uma ferramenta desenvolvida especificamente para lidar com grandes volumes de dados de TLDs, juntamente com sua compatibilidade com bibliotecas Python para análises posteriores e desenvolvimento da abordagem desenvolvida.

3.3 Processamento dos Dados

O segundo módulo deste trabalho é denominado de “Processamento dos Dados”, ilustrado na Figura 3.5 e descrito nesta seção.

Figura 3.5: Diagrama de funcionamento da etapa de processamento dos dados



Fonte: Elaborado pelo autor, 2024.

Após a coleta de todo o tráfego de pDNS e seu armazenamento, é necessária uma filtragem dos dados. Como mencionado anteriormente, a coleta abrange todo o tráfego, o que significa que o conjunto de dados gerado até o momento inclui todas as consultas realizadas ao servidor TLD, inclusive aquelas feitas a servidores secundários que podem ter acordos de cooperação com outros países. Essa situação pode resultar em um grande número de consultas respondidas por esses servidores.

Para efetuar a filtragem dos dados, uma lista de domínios recém-registrados é fornecida, juntamente com a data de registro de cada domínio. Essa lista é essencial para filtrar e obter um conjunto de dados que contenha apenas os domínios recém registrados.

Com o pDNS agora contendo apenas os domínios recém registrados, procede-se à extração das *features* que serão utilizadas pelos modelos. A Tabela 3.1 apresenta as *features* obtidas, juntamente com suas descrições correspondentes.

Tabela 3.1: Features extraídas e suas descrições

Feature	**Descrição**
`do_epp_provider`	Provedor EPP
`nb_days_until_collect`	Número de dias do registro até a primeira consulta
`ttl`	TTL
`ipv`	Versão do IP, 4 ou 6
`prot`	Protocolo, 6 (TCP) ou 17 (UDP)
`srcp`	Porta de origem
`aa`	*Authoritative Answer (AA)*
`cd`	Checking Disabled (CD) – Domain Name System Security Extensions (DNSSEC)
`ancount`	*Answer Count (ANCOUNT)*
`arcount`	*Additional Information Count* (ARCOUNT)
`nscount`	*Authority Count* (NSCOUNT)
`rcode`	*Response Code* (RCODE)
`qtype`	Type (QTYPE)
`country`	País
`asn`	*Autonomous System Number*
`labels`	QNAME *labels*
`res_len`	Tamanho da mensagem de resposta DNS

Fonte: Elaborado pelo autor, 2024.

Após a extração das *features*, os dados passam por uma etapa de rotulação. Nessa etapa, são fornecidas duas listas: uma lista de domínios maliciosos, previamente detectados por especialistas que combatem domínios maliciosos no TLD, onde os dados foram obtidos, e outra lista contendo domínios legítimos. Neste contexto, os domínios legítimos são considerados todos aqueles que não foram identificados como maliciosos.

No entanto, como já era esperado e conforme demonstrado anteriormente em trabalhos correlatos, os dados rotulados frequentemente apresentam um desequilíbrio sig-

nificativo. Para lidar com esse desequilíbrio, os dados seguem três caminhos distintos:

1. O primeiro caminho é para os dados que serão usados no treinamento do modelo supervisionado. Nesse caminho, os dados passam por uma etapa de balanceamento, com o objetivo de equilibrar as classes no conjunto de dados. Esse balanceamento é realizado por meio de uma técnica de *undersampling* para reduzir a classe majoritária e uma técnica de *oversampling* para aumentar a quantidade de dados na classe minoritária;

2. No segundo caminho, apenas os domínios maliciosos são selecionados e enviados para o treinamento do modelo não supervisionado. Nesse caso, opta-se por uma técnica de clusterização para lidar com os dados maliciosos;

3. O terceiro caminho envolve o uso dos dados balanceados, que são enviados para o modelo não supervisionado após a etapa de treinamento, permitindo que o modelo não supervisionado faça suas devidas predições sobre todo o conjunto de dados utilizados na etapa de treinamento do modelo supervisionado.

3.3.1 Balanceamento dos Dados

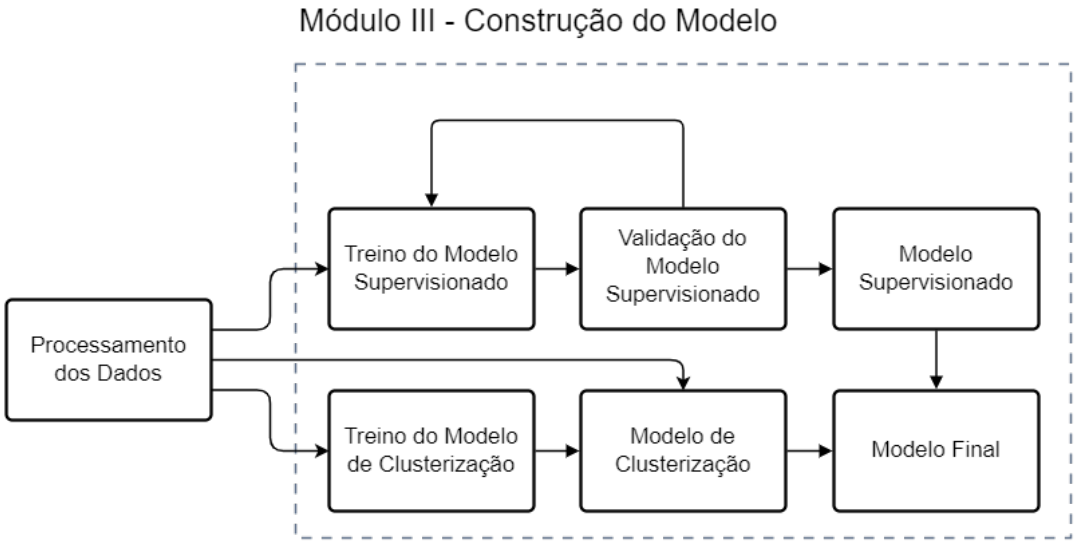
Para realizar essa etapa, a biblioteca Python *"imbalanced-learn"* é utilizada, e é necessário definir um valor para o campo *"sampling_strategy"*. Essa estratégia de amostragem, ou *sampling strategy* em inglês, corresponde ao valor *alpha(us)*, que pode ser definido como o número de amostras da classe minoritária após a reamostragem, dividido pelo número de amostras da classe majoritária. Após o *undersampling* realizado pelo algoritmo *"Random Undersampling"* (ZUECH; HANCOCK; KHOSHGOFTAAR, 2021), é executado o *oversampling* pelo algoritmo *"K-Means SMOTE"* (DOUZAS; BACAO; LAST, 2018) para alcançar o balanceamento entre as classes.

O valor da estratégia de amostragem foi testado de 0,1 a 0,9, variando de 0,1 em 0,1, e o conjunto de dados gerado foi utilizado como treinamento para três diferentes algoritmos de árvore de decisão: Random Forest, XGBoost e LightGBM. Todos os algoritmos foram executados com suas configurações padrão da biblioteca *scikit-learn* (PEDREGOSA et al., 2011). Para a etapa de treinamento e teste dos modelos, foi utilizado o *"Stratified K-fold Cross Validation"*, e a métrica de desempenho escolhida foi a AUC. O objetivo deste experimento é observar como o balanceamento dos dados afeta o desempenho dos modelos, uma vez que o *undersampling* pode resultar em perda de informações, e um grande número de dados gerados sinteticamente pode levar ao *overfitting* do modelo.

3.4 Construção do Modelo

O módulo III, denominado "Construção do Modelo", é responsável pelo desenvolvimento dos modelos que serão utilizados para a predição de novos domínios maliciosos recém-registrados. Essa etapa é ilustrada na Figura 3.6 e detalhada a seguir.

Figura 3.6: Diagrama de funcionamento da etapa de construção dos modelos



Fonte: Elaborado pelo autor, 2024.

Após o processamento dos dados realizado no módulo anterior, este módulo utiliza esses dados para treinar os modelos. Os dados balanceados são enviados para o treinamento e validação dos modelos supervisionados. Nesse contexto, algoritmos que suportam técnicas de *boosting* e processamento paralelo, como XGBoost e LightGBM, são preferenciais devido à sua escalabilidade e desempenho.

Os dados dos domínios maliciosos são enviados para o treinamento do modelo não supervisionado. Na abordagem proposta, foi utilizado o algoritmo de clusterização K-Means, cujo objetivo é agrupar exclusivamente os domínios maliciosos.

Após o treinamento do modelo não supervisionado, os dados balanceados são enviados a ele para realizar a predição. O objetivo é determinar a qual cluster malicioso cada domínio está mais próximo e qual é a sua distância euclidiana em relação a esse cluster.

Uma vez que os dois modelos tenham sido treinados e demonstrado eficácia na identificação de domínios maliciosos, os resultados gerados por ambos são utilizados como entrada para o modelo final. Esse modelo final, com base em uma estratégia de ensemble, produz uma saída definitiva, indicando se o domínio em questão é malicioso

ou não, juntamente com a probabilidade prevista. Com isso, encerra-se o treinamento dos três modelos presentes na abordagem.

3.4.1 Modelos de Aprendizado Supervisionado

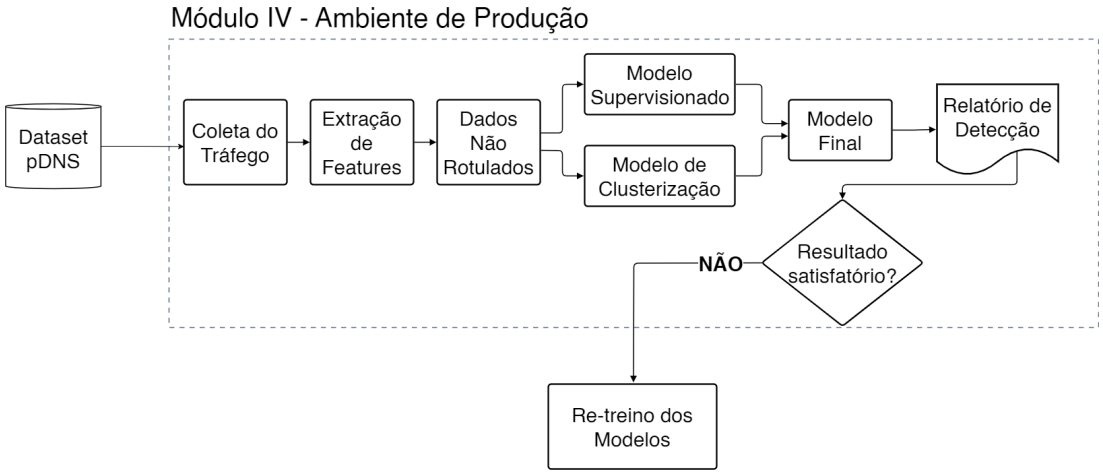
Para o desenvolvimento dos modelos supervisionados, utilizaram-se três algoritmos: XGBoost, LightGBM e SVM. Os dois primeiros algoritmos são baseados em árvores e foram executados em seus modos padrão nas respectivas bibliotecas, enquanto o SVM busca encontrar um hiperplano que separe de maneira ideal as duas classes analisadas, também executado em seu modo padrão, utilizando *kernel* "rfb" com a ativação da previsão de probabilidade. Para o treinamento desses modelos, utilizou-se o conjunto de dados apresentado e analisado na Seção 3.7.1, balanceado de forma híbrida por meio das técnicas RUS e *K-Means SMOTE*, conforme explicado na Seção 3.7.2. Foi adotado um valor de *"sampling strategy"* de 0,5, resultando em um conjunto de dados contendo aproximadamente 3.580 domínios, igualmente divididos entre as classes maliciosas e legítimas, com uma diferença máxima de 2 registros maliciosos em comparação com os domínios legítimos. Toda a etapa de treinamento e validação foi executada com o uso de *Stratified K-fold cross-validation*, com um valor de K igual a 5, definido empiricamente tendo como base a quantidade de dados de treinamento.

Com objetivo avaliar o desempenho do modelo em diferentes configurações e escolher a que melhor se adapta aos para a classificação final, foram criados três tipos de pipelines para esses três algoritmos. Cada etapa do *pipeline* tem um propósito específico, e a ordem em que essas etapas são executadas pode afetar o resultado final do modelo. Além disso, é possível que a ordem em que essas etapas são executadas afete o resultado final do modelo. O primeiro *pipeline* é composto por normalização dos dados, redução de dimensionalidade e balanceamento dos dados. O segundo *pipeline* segue a ordem de normalização dos dados, balanceamento e, por fim, redução de dimensionalidade. O terceiro e último *pipeline* consiste apenas na normalização e balanceamento dos dados. Em todos os pipelines que incluem redução de dimensionalidade, buscou-se uma variância explicada de pelo menos 80%. No *pipeline* do tipo 1, isso resultou em 10 *features*, e no pipeline do tipo 2, em 9 *features*. Essa diferença ocorreu devido à variação da variância explicada pelas *features* em diferentes situações. É importante ressaltar que o estado da arte não apresenta uma ordem clara de execução para a redução de dimensionalidade e o balanceamento dos dados (NASERIPARSA; KASHANI, 2014; GARCÍA; SÁNCHEZ; MOLLINEDA, 2011).

3.5 Ambiente de Produção

Na Figura 3.7 é ilustrado o módulo IV denominado “Ambiente de Produção” no qual refere-se aos modelos realizando a detecção de novos domínios maliciosos que foram recém registrados, bem como o desempenho dos modelos neste ambiente.

Figura 3.7: Diagrama de funcionamento dos modelos no ambiente de produção



Fonte: Elaborado pelo autor, 2024.

Após a etapa anterior de treinamento e validação dos modelos, eles são utilizados no ambiente de produção. Portanto, esse módulo começa realizando a coleta de tráfego, conforme descrito na Seção 3.2. A etapa de extração de *features* também ocorre como previamente descrito na Seção 3.3. No entanto, no ambiente de produção, os dados são inéditos e, portanto, não são rotulados.

Após a extração das *features*, os dados são enviados diretamente aos modelos supervisionado e não supervisionado. A saída de ambos os modelos é inserida no modelo de votação, no qual é decidido se o domínio em questão é malicioso ou legítimo. Após essa decisão, é gerado um relatório de detecção para que os profissionais do TLD possam acompanhar o comportamento desse domínio desde a primeira consulta DNS e tomar as medidas necessárias, caso o domínio seja identificado como malicioso.

No primeiro momento, especialistas do TLD são convidados a confirmar ou negar se a predição do modelo estava correta. Inicialmente, espera-se que o modelo apresente bons resultados. Entretanto, com o passar do tempo, é esperado que os atacantes alterem o comportamento de seus domínios para torná-los cada vez mais semelhantes a domínios legítimos. Isso pode resultar em uma queda no desempenho do modelo anteriormente treinado.

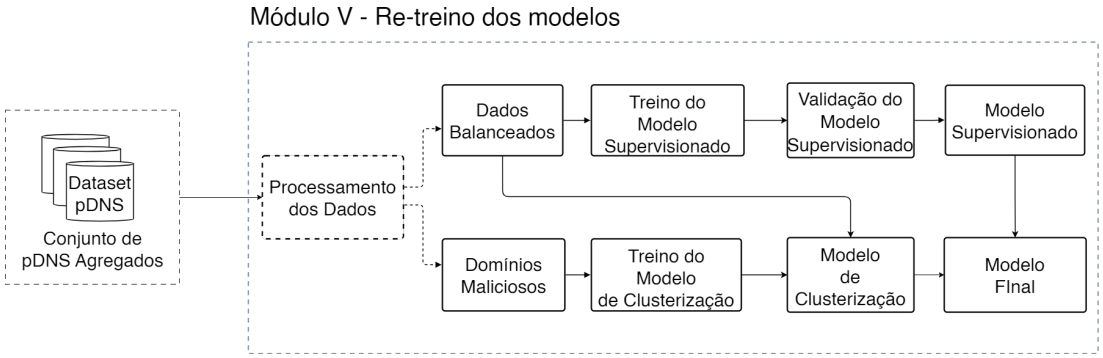
Ao perceber que o resultado do modelo não está sendo satisfatório, ou seja, sua

precisão está diminuindo, é necessário que os especialistas do TLD iniciem o processo de re-treinamento do modelo. Portanto, esse módulo é invocado para atualizar e reajustar os modelos com base nos novos padrões de comportamento dos domínios maliciosos, mantendo assim a eficácia da detecção de domínios recém-registrados.

3.6 Re-treino dos Modelos

O último módulo deste trabalho é denominado de "Re-treino dos Modelos" é ilustrado na Figura 3.8 e descrito nesta seção.

Figura 3.8: Diagrama de funcionamento da etapa de re-treino dos modelos desenvolvidos



Fonte: Elaborado pelo autor, 2024.

Quando é identificada uma queda significativa no desempenho do modelo, é necessário realizar um re-treinamento. O objetivo deste módulo é atualizar o conhecimento adquirido pelos modelos até o momento, e detectar mudanças no comportamento dos domínios.

Este módulo utiliza toda a base de dados que foi previamente aplicada para o treinamento inicial do modelo, acrescida de todo o conhecimento adquirido até então, juntamente com todos os domínios posteriormente registrados e corretamente identificados e rotulados. Portanto, os dados são processados conforme descrito na Seção 3.3. Os modelos são re-treinados, e são validados para avaliar o desempenho dos modelos agora atualizados. Em seguida, esses novos modelos são inseridos no ambiente de produção, novamente com o conhecimento agregado ao longo do tempo em que a versão anterior dos modelos estava em produção. Isso permite que os modelos se adaptem a novos padrões de comportamento e continuem a efetivamente detectar domínios maliciosos recém-registrados.

3.7 Metodologia dos dados

Esta seção apresenta a análise detalhada dos registros de domínios durante um período de 1 ano e 7 meses, incluindo 2.641.449 domínios, dos quais 8.184 foram identificados como maliciosos. Serão discutidas a metodologia de análise da primeira consulta DNS, o balanceamento dos dados devido ao desequilíbrio entre classes e os experimentos de redução de dimensionalidade utilizando PCA.

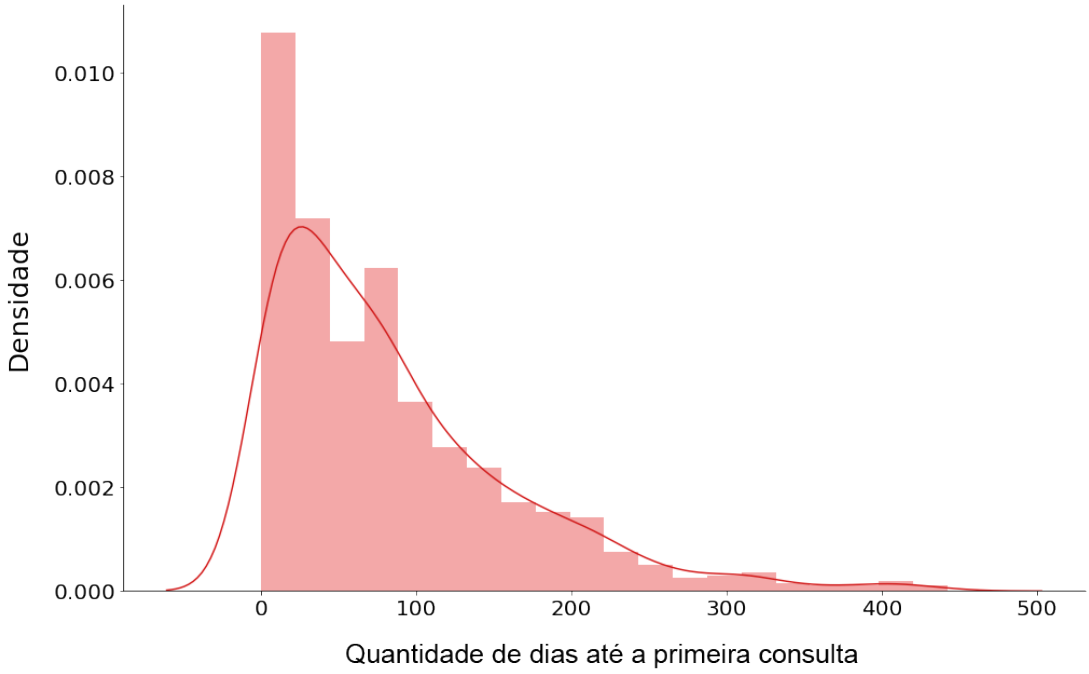
3.7.1 Análise dos Dados de Treinamento

A análise dos dados apresentada aqui leva em consideração um período de 1 ano e 7 meses de registros de domínios no TLD analisado, abrangendo o período de 01/03/2020 a 31/10/2021. Durante esse período foram registrados 2.641.449 domínios, dos quais 8.184 foram detectados como maliciosos, representando 0,31% do total de registros.

A abordagem deste trabalho considera a primeira consulta DNS de um domínio, e não apenas a data de registro. A escolha se justifica pelo fato de que nem todos os domínios têm sua primeira consulta DNS imediatamente após o registro; visto que alguns domínios apresentam um intervalo de tempo até a primeira consulta. Portanto, a coleta de dados de *Passive DNS* (pDNS) foi realizada no intervalo de 01/03/2020 a 09/02/2022 para possibilitar a análise de um maior número de domínios registrados no período anterior.

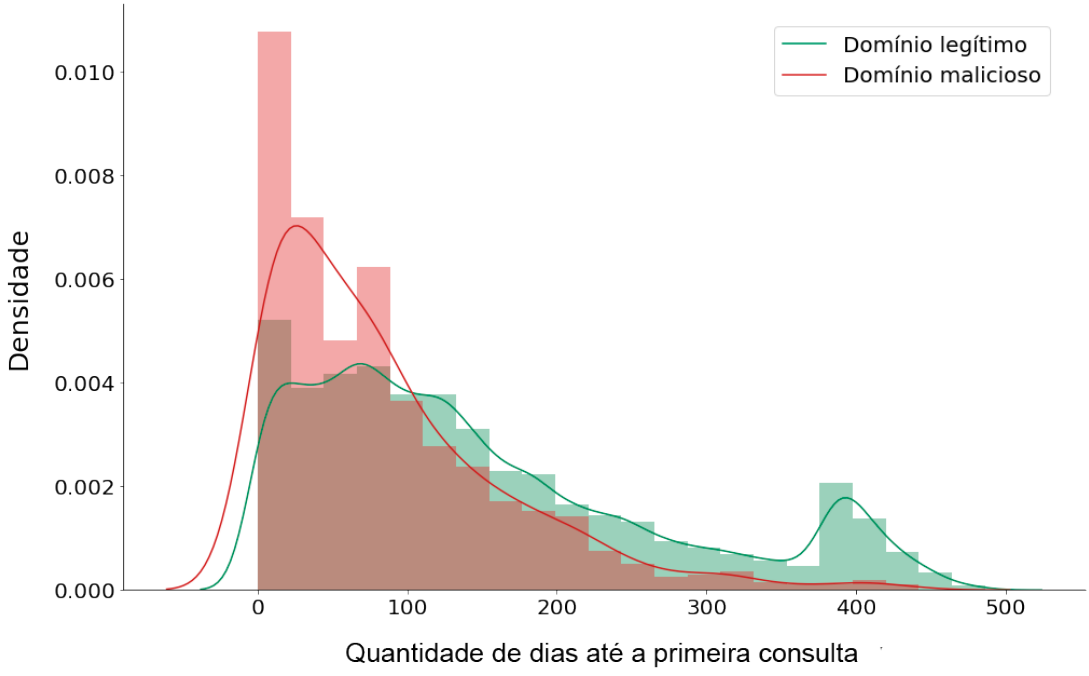
Uma observação importante é que os domínios maliciosos tendem a serem usados logo após seu registro, como evidenciado na Figura 3.9, enquanto os domínios legítimos tendem a terem sua primeira consulta DNS distribuída ao longo do tempo, conforme demonstrado na Figura 3.10. Ambas as figuras apresentam a densidade dos dados em relação ao número de dias até a primeira consulta DNS do domínio.

Figura 3.9: Densidade total de consultas aos domínios maliciosos



Fonte: Elaborado pelo autor, 2024.

Figura 3.10: Densidade total de consulta a todos domínios recém registrados



Fonte: Elaborado pelo autor, 2024.

Na Figura 3.10, que representa tanto os domínios maliciosos quanto os legítimos,

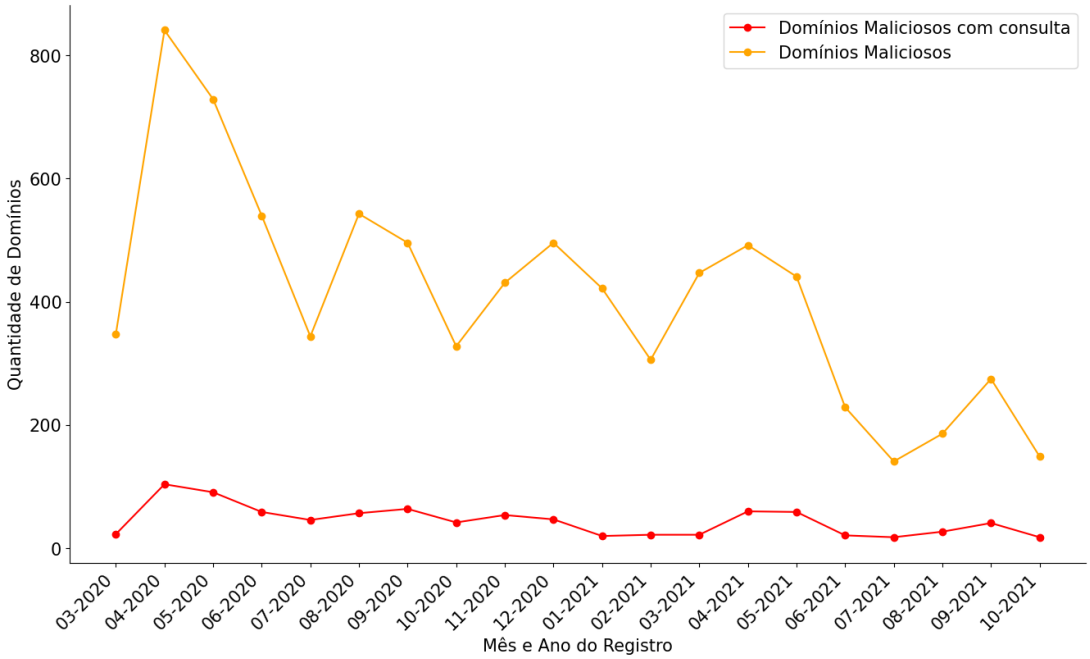
é evidente um pico na densidade próximo aos 400 dias até a primeira consulta DNS dos domínios. Isso sugere que esses domínios não foram usados imediatamente após o registro (normalmente por um período de 365 dias) e que, após a renovação, passaram a ser utilizados.

Durante o período de coleta do *Passive DNS* (pDNS), foram observados três tipos de códigos de resposta RCODE: -1, 0 e 3. Os tipos 0 e 3 foram explicados anteriormente na seção 2.2. O RCODE do tipo -1 é usado pelo ENTRADA para indicar que nenhuma resposta do servidor foi encontrada para aquela consulta DNS. Portanto, as consultas com esse RCODE foram removidas da base de dados, uma vez que não continham as informações necessárias para a análise.

Os domínios que obtiveram uma resposta RCODE 3 foram mantidos, pois indicavam que a consulta do domínio havia ocorrido, e que o servidor autoritativo do TLD respondeu. Mesmo que a resposta fosse baseada em dados em cache, as informações eram consideradas válidas.

Após todas as etapas de filtragem e tratamento dos dados, obteve-se a coleta de pDNS da primeira consulta DNS para 895 domínios maliciosos e 80.671 domínios legítimos. A diferença na quantidade de domínios maliciosos registrados e aqueles que tiveram a primeira consulta DNS coletada se deve ao esforço do TLD em evitar ao máximo a disponibilidade de domínios com atividades maliciosas ao público. Isso é ilustrado na Figura 3.11, que mostra uma comparação mensal entre a quantidade de domínios maliciosos registrados e aqueles para os quais aconteceu a primeira consulta. O conjunto de dados de pDNS até o momento é composto por 81.566 domínios, dos quais 1,09% do tráfego coletado é malicioso.

Figura 3.11: Análise dos domínios maliciosos registrados



Fonte: Elaborado pelo autor, 2024.

Ao analisarmos o segundo, terceiro e sétimo dia de vida do domínio após a primeira consulta DNS, os resultados são semelhantes. Se analisada a quantidade de consultas para esses domínios recém registrados em até 7 dias após a primeira consulta, apenas 7,69% desses domínios tiveram mais de uma consulta.

3.7.2 Balanceamento dos Dados

A necessidade de balanceamento dos dados é evidente devido ao desequilíbrio entre as classes no conjunto de dados, onde apenas 1,09% do conjunto de dados consiste em dados de domínios maliciosos. Como explicado na seção 3.3, sobre o processamento dos dados, a etapa de balanceamento de dados foi realizada por meio de uma técnica híbrida, envolvendo tanto o *oversampling* (aumento da classe minoritária) quanto o *undersampling* (redução da classe majoritária).

O conjunto de dados original utilizado neste experimento é composto por 80.671 dados de domínios legítimos e 895 domínios maliciosos. Os resultados deste experimento estão apresentados na Tabela 3.2, e o comparativo entre as médias das AUCs geradas pelos modelos é apresentado na Figura 3.12.

Os resultados apresentados na Tabela 3.2 e na Figura 3.12 indicam que os modelos baseados em árvores executados em seus modos padrão obtiveram resultados semelhantes, com pequenas variações. Fica evidente que os modelos tiveram um desempe-

Tabela 3.2: Impacto da estratégia de amostragem no treinamento dos modelos

Sampling Strategy	**Domínios legítimos após undersampling**	**Domínios maliciosos após oversampling**	**RF**	**XGBoost**	**LightGBM**
0,1	8950	8956	0,9959	0,9957	0,9958
0,2	4475	4480	0,9912	0,9912	0,9918
0,3	2983	2990	0,9848	0,9861	0,9868
0,4	2237	2243	0,9800	0,9801	0,9812
0,5	1790	1796	0,9742	0,9741	0,9753
0,6	1491	1498	0,9705	0,9706	0,9721
0,7	1278	1285	0,9648	0,9631	0,9652
0,8	1118	1124	0,9597	0,9595	0,9619
0,9	994	1001	0,9544	0,9566	0,9590

Fonte: Elaborado pelo autor, 2024.

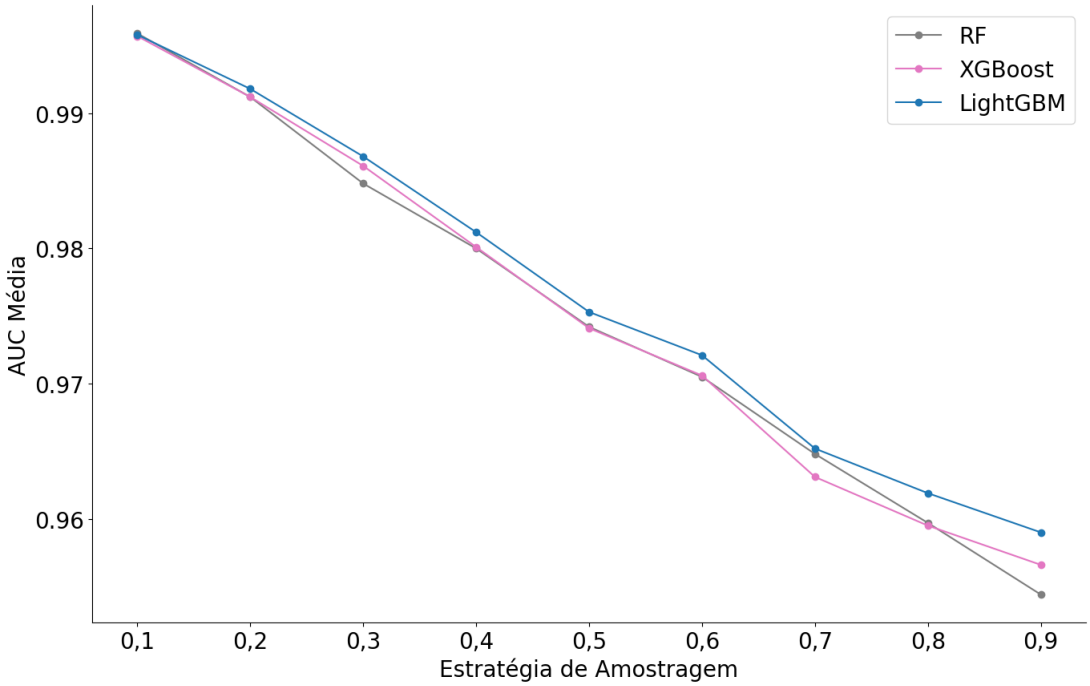
nho melhor com um valor de estratégia de amostragem mais baixo, e esse desempenho foi se deteriorando à medida que o valor da estratégia de amostragem aumentou.

Com um valor de estratégia de amostragem de 0,1, a quantidade de domínios maliciosos no conjunto de dados de treinamento aumentou de 895 para 8956, o que representa um aumento de cerca de 10 vezes. Isso pode levar a um *overfitting* dos modelos, o que explica a AUC média de 0,9958 obtida pelos três modelos. Por outro lado, ao aumentar o valor da estratégia de amostragem, a quantidade de dados sintéticos de domínios maliciosos diminui, resultando em uma redução no desempenho do modelo.

Ao usar um valor de 0,5 para o campo *"sampling strategy"*, a quantidade de domínios maliciosos dobrou em comparação com o conjunto de dados original. O treinamento foi realizado com 1790 domínios legítimos e 1796 domínios maliciosos, resultando em uma AUC média de 0,9745 entre os três modelos.

Por fim, o último valor testado para a *"sampling strategy"* foi 0,9, no qual os modelos obtiveram uma AUC média de 0,9566. Nesse cenário, o conjunto de dados de treinamento era composto por 994 domínios legítimos e 1001 domínios maliciosos. Esses valores extremos de estratégia de amostragem indicam indícios de *overfitting* (*sampling strategy* = 0,1) e *underfitting* (*sampling strategy* = 0,90).

Figura 3.12: Comparativo de AUC entre modelos diversificando estratégia de amostragem



Fonte: Elaborado pelo autor, 2024.

A escolha de um valor de *"sampling strategy"* de 0,5 para duplicar a quantidade de domínios maliciosos no conjunto de treinamento parece ser apropriada, considerando o cenário e os objetivos deste trabalho.

3.7.3 Redução de Dimensionalidade

O conjunto de dados obtido após a etapa de extração das características possui 17 dimensões. Com o objetivo de reduzir a dimensionalidade deste conjunto de dados, ou seja, encontrar um conjunto menor de características que represente o conjunto de dados original com a mesma eficácia, executou-se experimentos utilizando a técnica de redução de dimensionalidade.

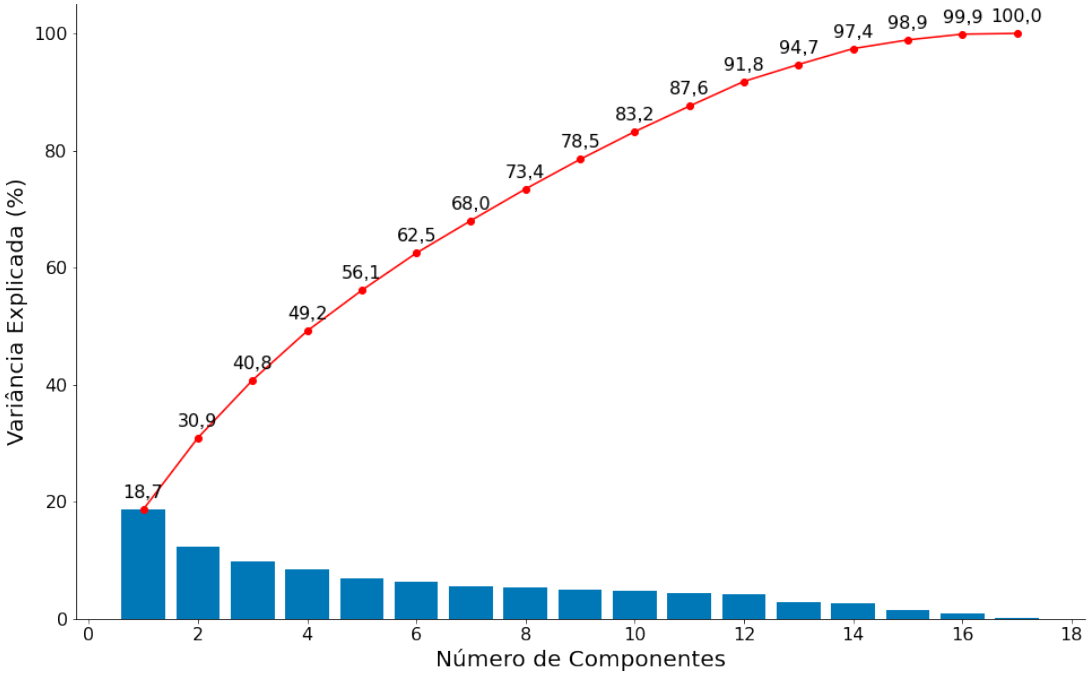
Para reduzir as dimensões de um conjunto de dados, é necessário escolher a quantidade de dimensões para a qual o conjunto de dados original será reduzido. Uma forma de realizar essa seleção é por meio da variância explicada em função do número de dimensões. Ao gerar um gráfico com os eixos da variância explicada em relação às dimensões, é possível observar uma "inflexão" no qual a variância explicada deixa de crescer rapidamente, conforme discutido por Géron (2019). Como alternativa, pode-se escolher os primeiros componentes de modo que a variância cumulativa exceda um

limite de 80%, como sugerido por Bruce e Bruce (2019).

Para aplicar a técnica de Análise de Componentes Principais (PCA), é essencial que os dados estejam normalizados, uma vez que essa técnica é muito sensível às variações relativas das características originais. Os dados foram normalizados usando a técnica *Standard Scaler*, presente no *Scikit-Learn*.

A Figura 3.13 ilustra os componentes gerados pelo PCA após a normalização dos dados. É possível observar que com 10 componentes, obtém-se aproximadamente 83,2% da variância explicada.

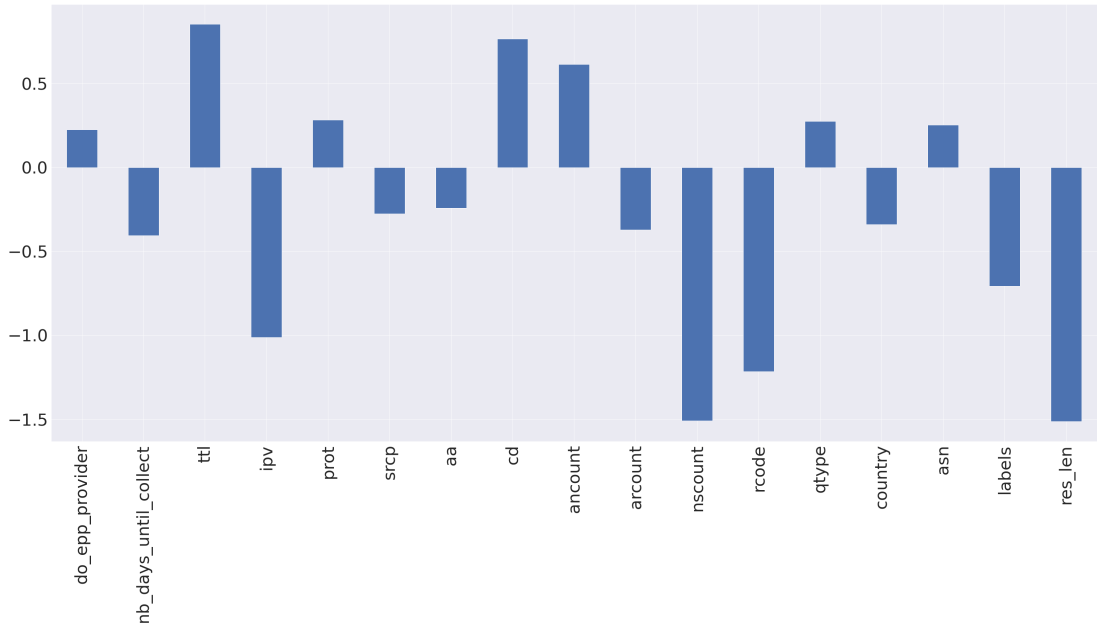
Figura 3.13: Análise da variância explicada componentes gerados



Fonte: Elaborado pelo autor, 2024.

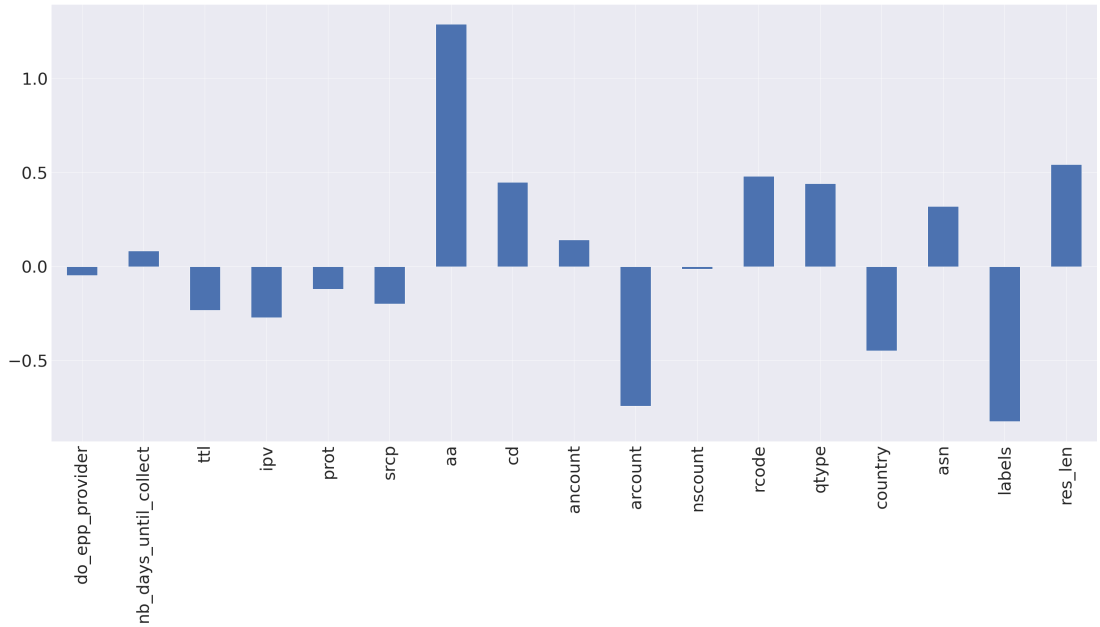
Da Figura 3.14 a Figura 3.23 é apresentada a covariância entre cada um dos 10 primeiros componentes gerados pelo PCA e as características originais.

Figura 3.14: Análise de carga dos componentes do PCA - Componente 1



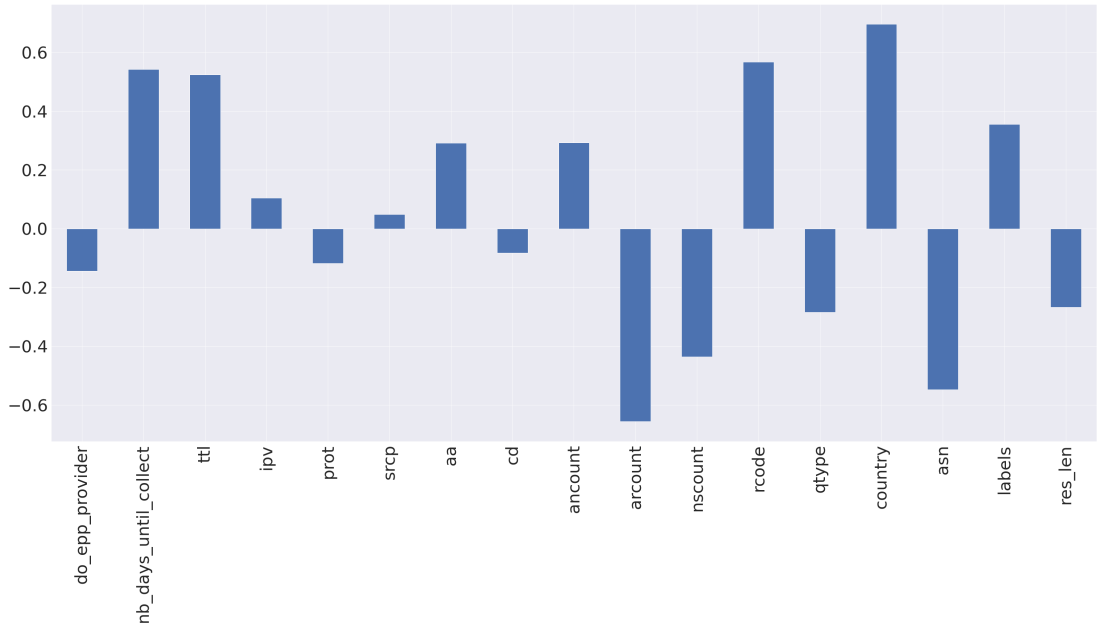
Fonte: Elaborado pelo autor, 2024.

Figura 3.15: Análise de carga dos componentes do PCA - Componente 2



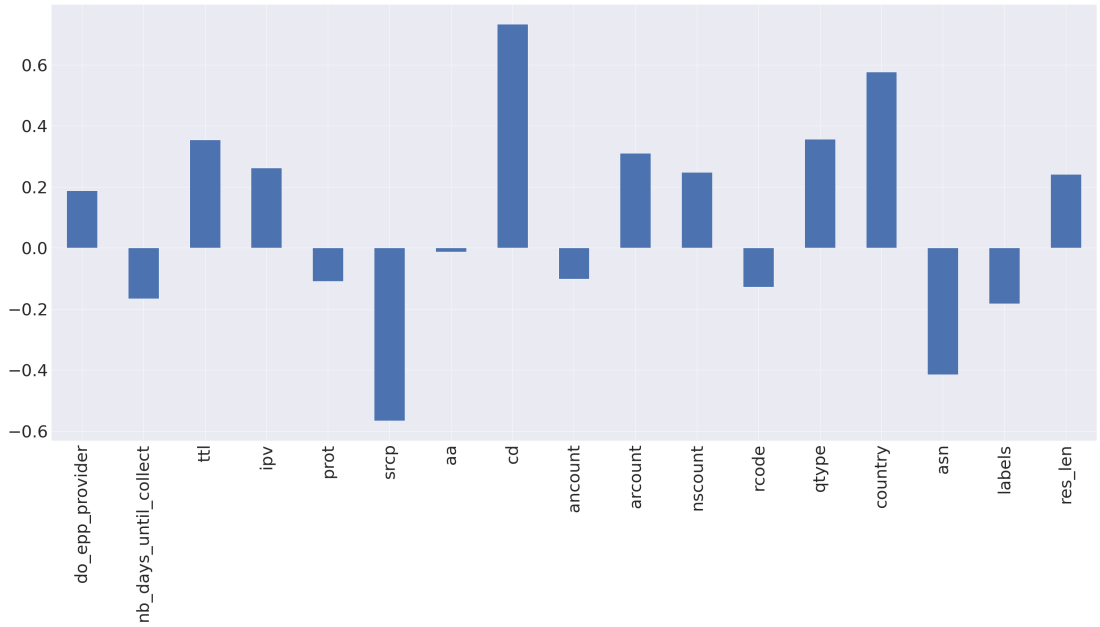
Fonte: Elaborado pelo autor, 2024.

Figura 3.16: Análise de carga dos componentes do PCA - Componente 3



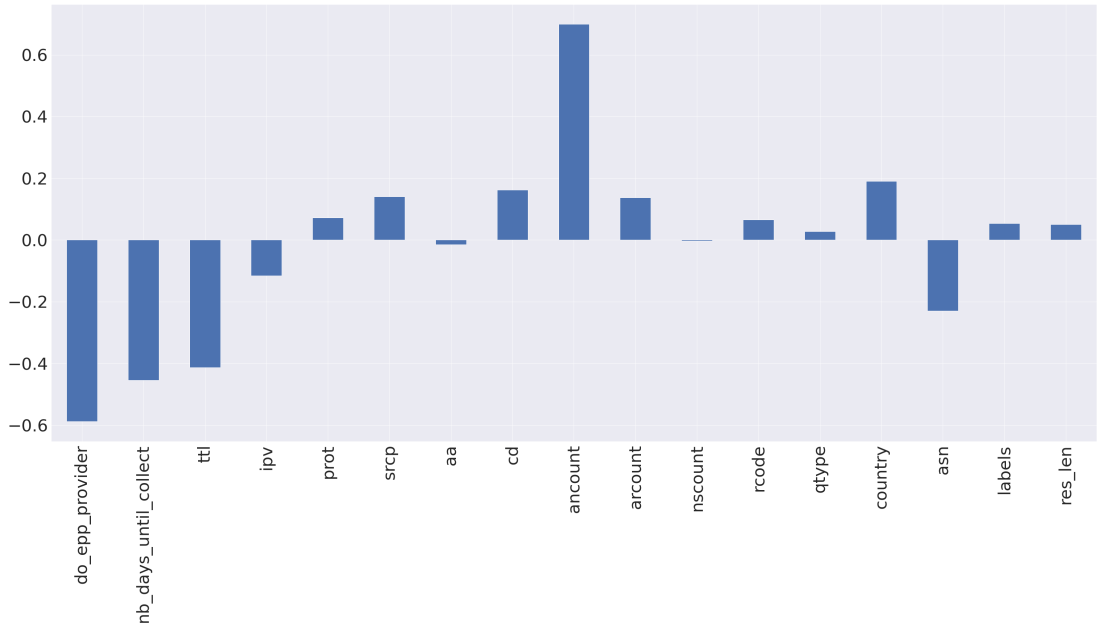
Fonte: Elaborado pelo autor, 2024.

Figura 3.17: Análise de carga dos componentes do PCA - Componente 4



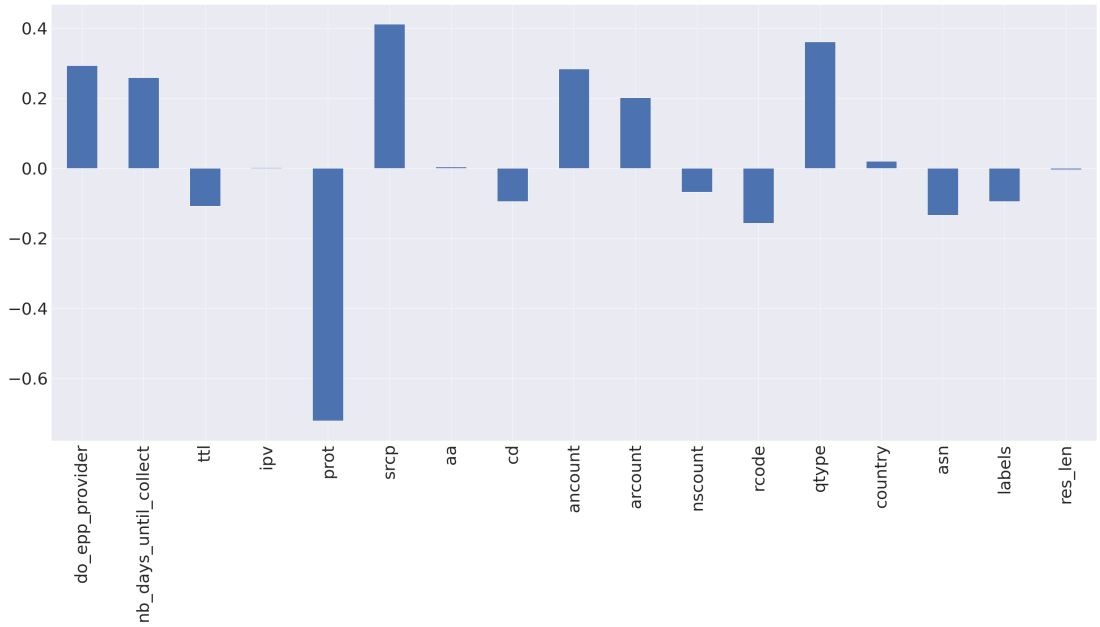
Fonte: Elaborado pelo autor, 2024.

Figura 3.18: Análise de carga dos componentes do PCA - Componente 5



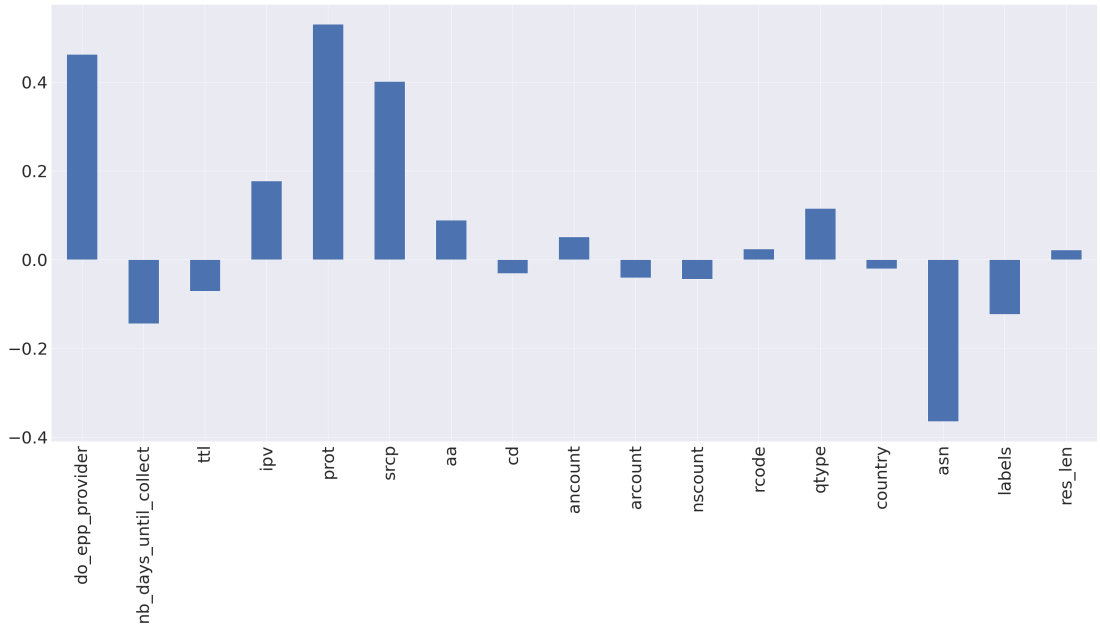
Fonte: Elaborado pelo autor, 2024.

Figura 3.19: Análise de carga dos componentes do PCA - Componente 6



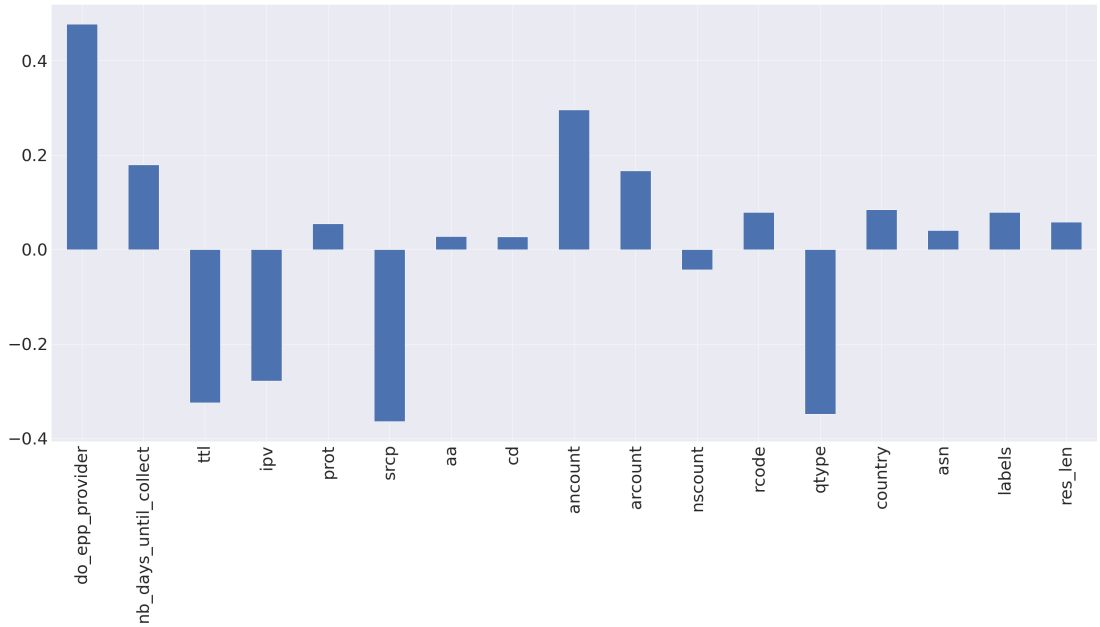
Fonte: Elaborado pelo autor, 2024.

Figura 3.20: Análise de carga dos componentes do PCA - Componente 7



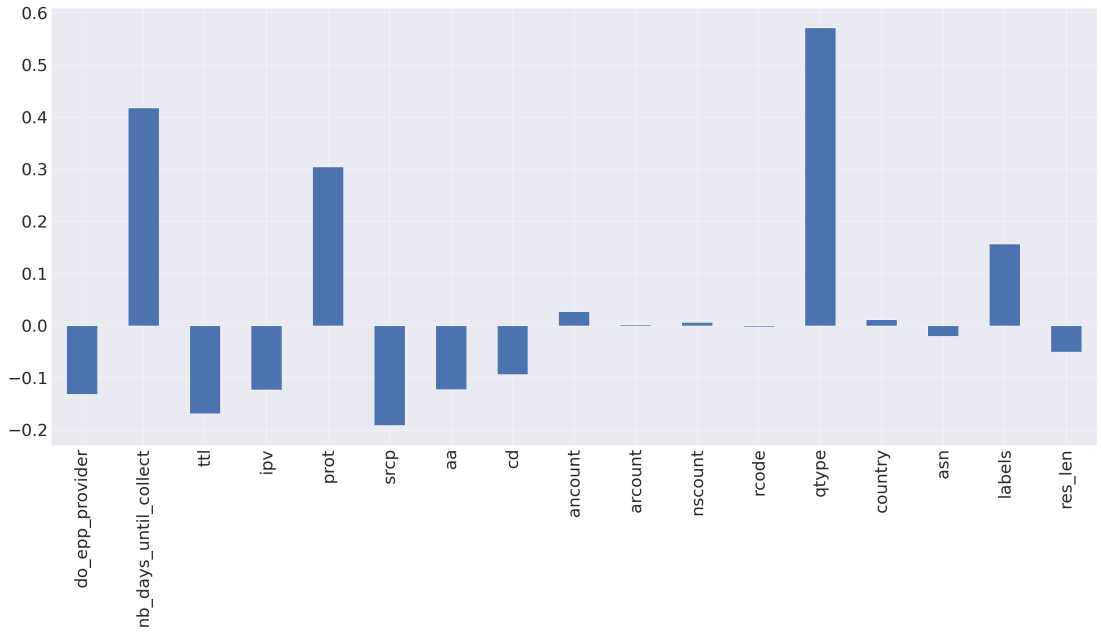
Fonte: Elaborado pelo autor, 2024.

Figura 3.21: Análise de carga dos componentes do PCA - Componente 8



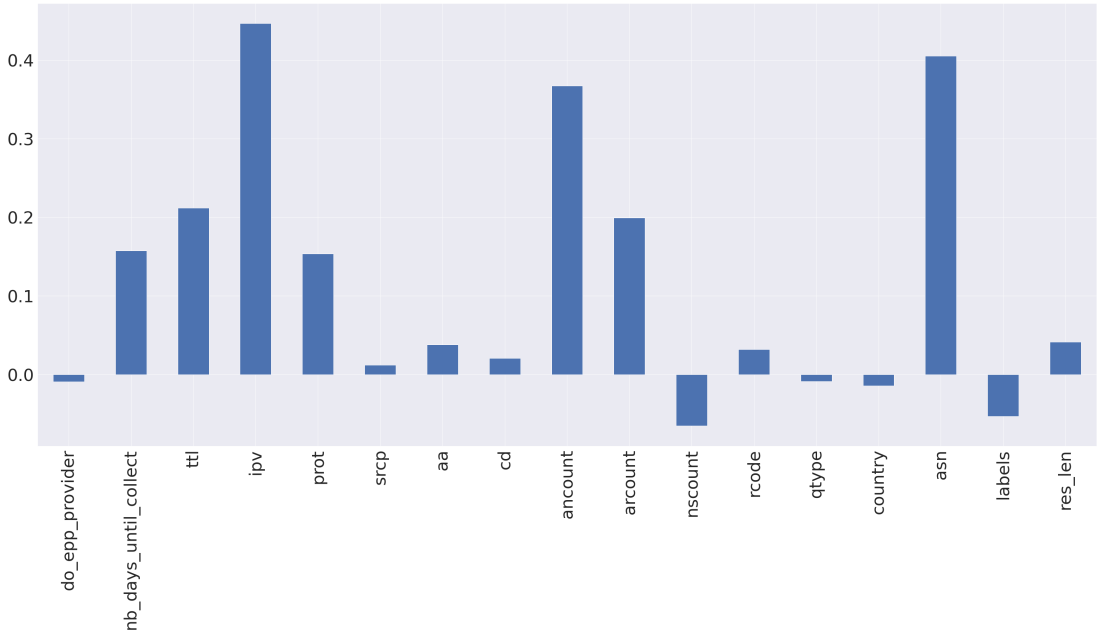
Fonte: Elaborado pelo autor, 2024.

Figura 3.22: Análise de carga dos componentes do PCA - Componente 9



Fonte: Elaborado pelo autor, 2024.

Figura 3.23: Análise de carga dos componentes do PCA - Componente 10



Fonte: Elaborado pelo autor, 2024.

3.8 Considerações Parciais

A abordagem apresentada neste trabalho utiliza DNS passivo coletado de um servidor autoritativo TLD como fonte de dados para a detecção precoce de domínios maliciosos no TLD. Essa abordagem envolve o desenvolvimento de cinco módulos, desde a coleta de dados por meio de *"sniffers"* na rede até o re-treinamento dos modelos para que possam se adaptar a novas estratégias de evasão. Para aprimorar o desempenho do modelo classificador, ou seja, melhorar a qualidade da detecção de domínios maliciosos e aumentar a precisão, elevando as taxas de verdadeiros positivos (TVP) e verdadeiros negativos (TFP) dos modelos nas etapas de validação e produção, optou-se por uma abordagem semi-supervisionada. A expectativa é alcançar um melhor desempenho na identificação desses domínios maliciosos, logo em sua primeira consulta.

A abordagem consiste em modelos de aprendizagem supervisionada trabalhando em conjunto com um modelo não supervisionado, resultando em uma abordagem semi-supervisionada. Ao analisar os dados de treinamento, observa-se que os domínios maliciosos tendem a ser utilizados imediatamente após o registro, enquanto os domínios legítimos são usados ao longo do tempo. Além disso, o TLD no qual os dados estão sendo coletados atua para remover tais domínios maliciosos o mais rápido possível, uma vez que apenas uma parte dos domínios registrados possui consulta associada.

Inicialmente, foram extraídas 17 características dos dados coletados, o que possibilitou a redução de dimensionalidade por meio da técnica de Análise de Componentes Principais (PCA). O objetivo era alcançar uma variância explicada de cerca de 80%, resultando em 10 componentes quando aplicada no *pipeline* 1 (normalização, PCA e balanceamento dos dados) e 9 componentes quando aplicada no *pipeline* 2 (normalização dos dados, balanceamento e PCA).

Devido à quantidade de dados e ao desbalanceamento das classes, técnicas de *oversampling* (aumento da classe minoritária) e *undersampling* (redução da classe majoritária) são necessárias para equilibrar as classes no conjunto de dados. Foi necessária a aplicação de técnicas de balanceamento, escolhendo o *Random Under Sampling* (RUS) para reduzir a classe majoritária (domínios legítimos) e o *K-Means SMOTE* para aumentar a classe minoritária (domínios maliciosos). Na Seção 2.6, ficou definido que a estratégia de amostragem 0,5, que resulta em dobrar a quantidade de domínios maliciosos no conjunto de dados, enquanto mantém aproximadamente a mesma quantidade de domínios legítimos para compor o conjunto de treinamento, seria a melhor abordagem.

Antecipando o surgimento de novas formas de evasão por parte dos atacantes que poderiam tornar a abordagem obsoleta, um módulo de re-treinamento dos modelos foi desenvolvido. Isso garantirá a atualização contínua do conhecimento e a adaptabilidade dos modelos aos novos desafios ao longo do tempo. Finalmente, ao se preocupar com o tempo de detecção de novos domínios maliciosos em sua primeira consulta, a abordagem aqui apresentada se torna ainda mais relevante no combate a esses domínios maliciosos, ajudando a evitar que os usuários da Internet caiam em golpes planejados por esses atacantes.

Capítulo 4

Resultados

Este capítulo tem como objetivo apresentar os resultados obtidos, e está dividido em 5 seções. A seção 4.1 apresenta os resultados e comparações dos modelos de aprendizado de máquina supervisionado. A seção 4.2 apresenta os resultados obtidos com técnicas de aprendizado de máquina não supervisionado. Na seção 4.3, são discutidos os experimentos e resultados do modelo final da abordagem proposta neste trabalho. A seção 4.4 oferece uma discussão sobre os resultados obtidos e pontos importantes do desenvolvimento. Por fim, a seção 4.5 apresenta considerações parciais deste capítulo.

4.1 Modelos de Aprendizado Supervisionado

Como explicado na seção 3.4, foi testado três algoritmos de aprendizado supervisionado em diferentes pipelines, o que resultou em diferentes modelos. A Tabela 4.1 apresenta os resultados da etapa de treinamento e testes dos modelos. O código do modelo é composto por três letras maiúsculas que indicam o algoritmo utilizado (XGB para XGBoost, LGB para LightGBM e SVM para SVM), seguido da letra *"p"* acompanhada de um número, o qual indica qual *pipeline* foi executado (1, 2 ou 3). Em seguida, há uma letra *"f"* acompanhada do número 0 ou 1, em que 0 indica que todas as características foram utilizadas no *pipeline* de dados e 1 indica a remoção das características apresentadas no segundo experimento.

Com base nos resultados apresentados na Tabela 4.1, é possível observar que os modelos tiveram desempenhos semelhantes em termos de acurácia (ACC). Além disso, os modelos baseados em árvores que utilizaram o *pipeline* 3 apresentaram as melhores médias de AUC, com valores de 0,97 (+/- 0,02) e 0,97 (+/- 0,02) para os algoritmos XGBoost e LightGBM, respectivamente.

Para realizar os testes desses modelos, simulando o ambiente de produção (Módulo

Tabela 4.1: Resultado da etapa de treino e validação dos modelos

Código do Modelo	Algoritmo	Pipeline	AUC	ACC
XGBp1f0	XGBoost	1	0,96 (+/- 0,02)	0,92
XGBp2f0	XGBoost	2	0,95 (+/- 0,03)	0,91
XGBp3f0	XGBoost	3	0,97 (+/- 0,02)	0,92
LGBp1f0	LightGBM	1	0,96 (+/- 0,02)	0,91
LGBp2f0	LightGBM	2	0,96 (+/- 0,02)	0,91
LGBp3f0	LightGBM	3	0,97 (+/- 0,02)	0,91
SVMp1f0	SVM	1	0,95 (+/- 0,02)	0,91
SVMp2f0	SVM	2	0,95 (+/- 0,02)	0,91
SVMp3f0	SVM	3	0,95 (+/- 0,02)	0,91

Fonte: Elaborado pelo autor, 2024.

IV), utilizou-se um conjunto de dados composto por domínios registrados nos meses de novembro e dezembro de 2021 e que possuíam consultas DNS até 10 de maio de 2022. No total, foram registrados 212.292 domínios nesses meses, dos quais 211.802 eram legítimos e 490 eram maliciosos. Entre os domínios maliciosos, 104 tiveram consultas DNS, enquanto 7.802 domínios legítimos também receberam consultas. Portanto, o conjunto de dados de teste contém 104 domínios maliciosos e 7.802 domínios legítimos. O desempenho dos modelos desenvolvidos na etapa de teste é apresentado na Tabela 4.2, e tomando como referência o conjunto de métricas de TVP, TVN, TFP e TFN é possível destacar os modelos XGBp3f0 e SVMp3f0.

Tabela 4.2: Resultado da etapa de teste dos modelos

Código do Modelo	ACC	TVP	TVN	TFP	TFN
XGBp1f0	0,88	0,817	0,875	0,124	0,182
XGBp2f0	0,88	0,817	0,875	0,124	0,182
XGBp3f0	**0,89**	**0,846**	**0,887**	**0,110**	**0,153**
LGBp1f0	0,88	0,817	0,877	0,122	0,180
LGBp2f0	0,87	0,846	0,869	0,130	0,153
LGBp3f0	0,89	0,836	0,887	0,112	0,163
SVMp1f0	0,85	0,971	0,846	0,152	0,028
SVMp2f0	0,85	0,971	0,847	0,152	0,028
SVMp3f0	**0,85**	**0,971**	**0,847**	**0,152**	**0,028**

Fonte: Elaborado pelo autor, 2024.

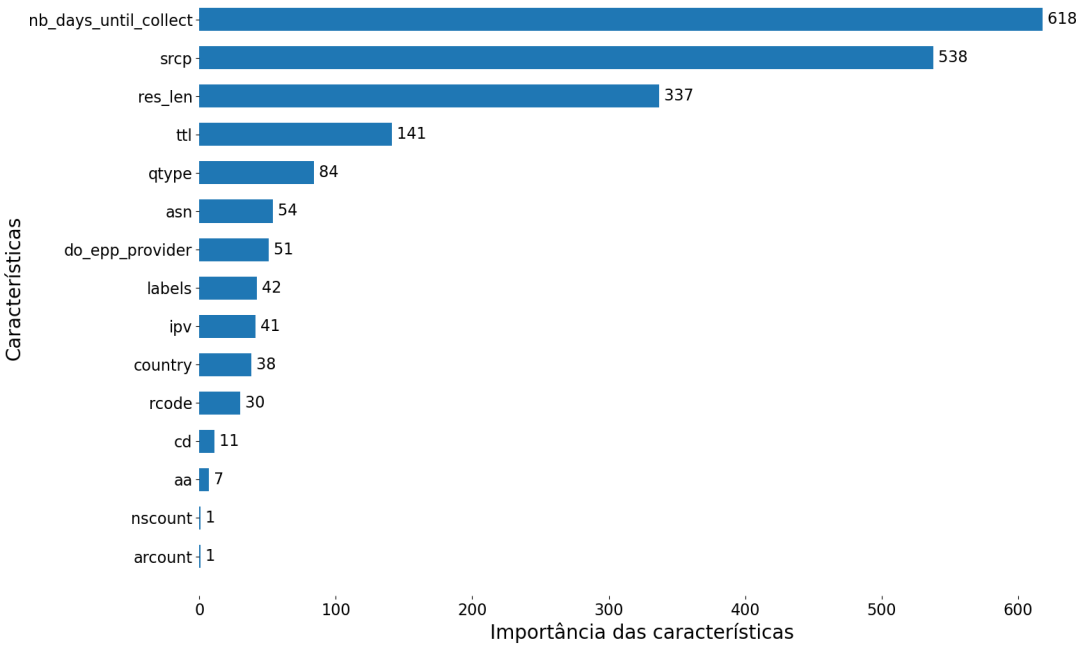
Os resultados apresentados na Tabela 4.2 para os modelos XGBp1f0, XGBp2f0, SVMp1f0 e SVMp2f0 são idênticos, indicando que a ordem de execução do PCA e

do balanceamento dos dados não afetou o desempenho desses algoritmos na etapa de teste. No entanto, os resultados dos modelos `LGBp1f0` e `LGBp2f0` foram diferentes.

É importante destacar que os modelos SVM demonstraram uma TVP superior à obtida pelos modelos `XGBp3f0` e `LGBp2f0`, mostrando sua eficiência na detecção de domínios maliciosos. Além disso, os modelos `XGBp3f0` e `LGBp3f0` apresentaram TVN superiores aos modelos SVM na detecção de domínios legítimos.

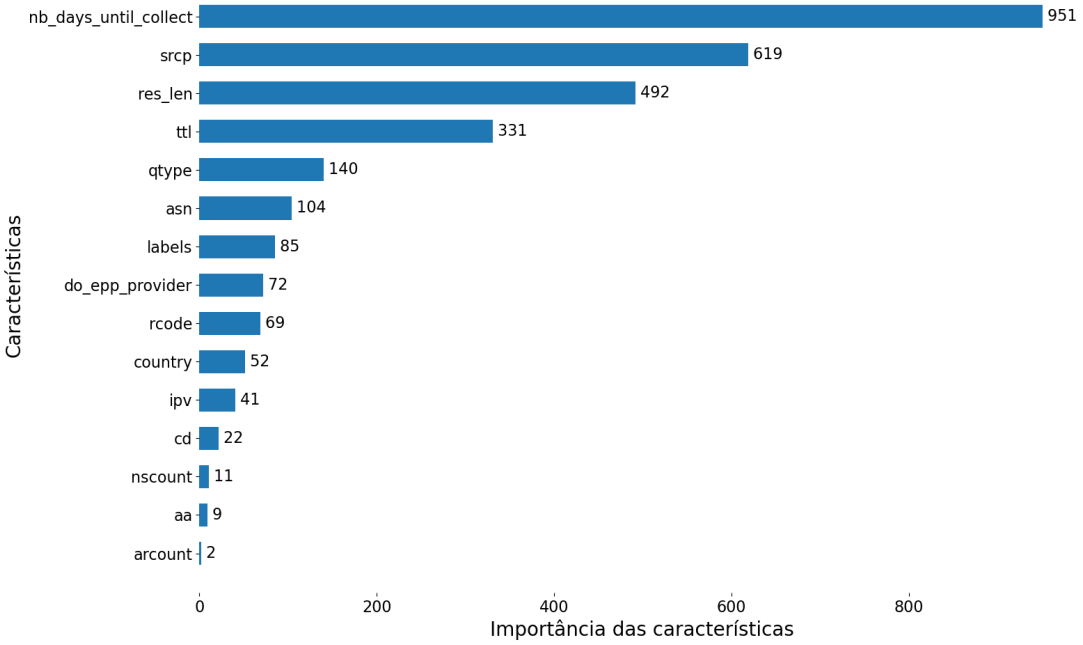
Quanto à importância das características utilizadas, observa-se um comportamento semelhante nos modelos que utilizaram o *pipeline* de dados número 3 em relação às características de menor impacto. Além disso, duas características não apresentaram impacto significativo e não estão representadas nos gráficos, a saber: "`prot`" e "`ancount`". A importância das características nos modelos `XGBp3f0` e `LGBp3f0` é apresentada nas Figuras 4.1 e 4.2, respectivamente. Isso refere-se à medida de quão influentes são as diferentes características ou variáveis em um modelo de *machine learning* para fazer previsões e é usada para entender quais recursos têm o maior impacto nas decisões do modelo.

Figura 4.1: Importâncias das características utilizadas no modelo `XGBp3f0`



Fonte: Elaborado pelo autor, 2024.

Figura 4.2: Importâncias das características utilizadas no modelo `LGBp3f0`



Fonte: Elaborado pelo autor, 2024.

Diante da baixa importância das características "`prot`", "`ancount`", "`arcount`", "`nscount`" e "`aa`" nos modelos, realizou-se uma nova etapa de treinamento e validação, utilizando o *pipeline* 1 e 3 e removendo essas características. Os resultados desse experimento estão apresentados na Tabela 4.3.

Tabela 4.3: Resultado da etapa de treino e validação dos modelos com redução das características originais

Código do Modelo	**Algoritmo**	**Pipeline**	**AUC**	**ACC**
`XGBp3f1`	`XGBoost`	3	0,97 (+/- 0,02)	0,92
`LGBp3f1`	`LightGBM`	3	0,98 (+/- 0,02)	0,92
`SVMp3f1`	`SVM`	3	0,95 (+/- 0,02)	0,91
`XGBp1f1`	`XGBoost`	1	0,97 (+/- 0,01)	0,91
`LGBp1f1`	`LightGBM`	1	0,97 (+/- 0,02)	0,91
`SVMp1f1`	`SVM`	1	0,95 (+/- 0,02)	0,91

Fonte: Elaborado pelo autor, 2024.

Os modelos apresentados na Tabela 4.3 mostram resultados ligeiramente melhores em termos de AUC média na etapa de treinamento e resultados iguais em termos de ACC após a remoção das características mencionadas. Os modelos foram então avaliados na etapa de teste, e os resultados são apresentados na Tabela 4.4.

Tabela 4.4: Resultado da etapa de teste dos modelos com redução das características originais

Código do Modelo	ACC	TVP	TVN	TFP	TFN
XGBp3f1	0,89	0,826	0,892	0,107	0,173
LGBp3f1	0,89	0,836	0,891	0,108	0,163
SVMp3f1	0,85	0,961	0,849	0,150	0,030
XGBp1f1	0,88	0,740	0,878	0,121	0,259
LGBp1f1	0,88	0,740	0,875	0,124	0,259
SVMp1f1	0,85	0,951	0,850	0,149	0,048

Fonte: Elaborado pelo autor, 2024.

A etapa de teste revela resultados semelhantes para os modelos XGBp3f1 e LGBp3f1, com TVP ligeiramente inferiores em comparação com o melhor modelo, XGBp3f0, que obteve uma TVP de 0,846, enquanto XGBp3f1 e LGBp3f1 alcançaram 0,826 e 0,836, respectivamente. O modelo SVM se destaca dos demais, com uma TVP mais baixa no SVMp3f1 (0,961) em comparação com o SVMp3f0 (0,971). No entanto, quando se observa a TVN, XGBp3f1 e LGBp3f1 superam XGBp3f0, SVMp3f0 e SVMp3f1, com valores de 0,892 e 0,891, em comparação com 0,887, 0,847 e 0,849, respectivamente.

Os outros modelos, XGBp1f1 e LGBp1f1, apresentam resultados semelhantes entre si. Ambos têm ACC compatíveis com os melhores modelos desenvolvidos, mas uma TVP consideravelmente inferior, com um valor de 0,740. Essa queda de desempenho nesses modelos pode ser atribuída à redução de dimensionalidade que busca uma variância explicada de 80%, o que pode resultar na perda de informações prejudicial para o desempenho na etapa de validação. O modelo SVMp1f1 também sofreu um impacto na TVP, que foi de 0,951. No entanto, seu desempenho na métrica TVN é inferior ao de XGBp1f1 e LGBp1f1, com um valor de 0,850 em comparação com 0,878 e 0,875, respectivamente.

Os modelos escolhidos para compor o modelo final são XGBp3f0 e SVMp3f0, devido à abordagem diferente na detecção de domínios maliciosos. Acredita-se que eles podem se complementar na etapa de construção do modelo final, uma vez que XGBp3f0 se destaca por sua TVN e ACC, enquanto SVMp3f0 possui uma TVP elevada.

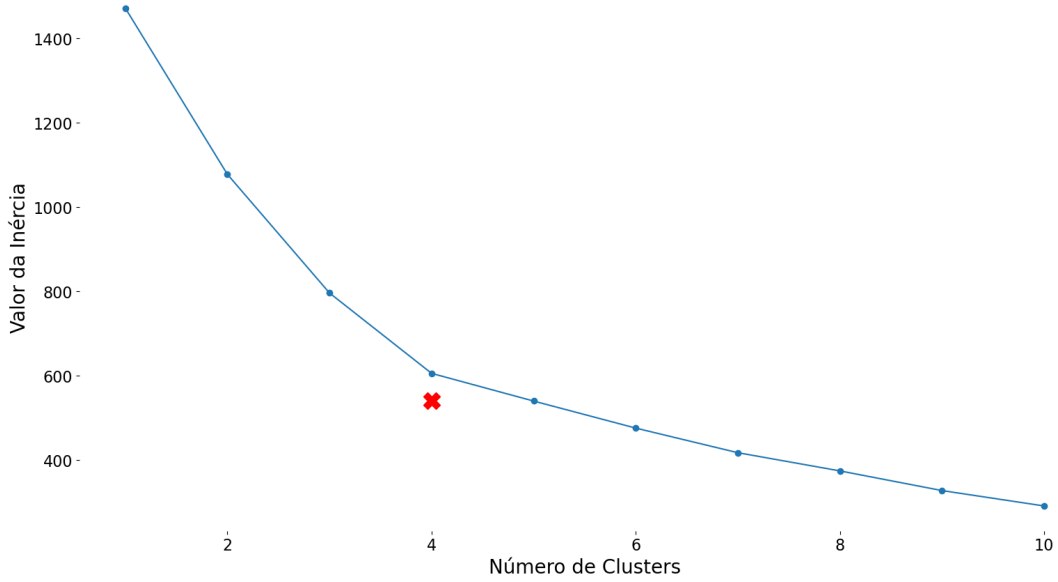
4.2 Modelos de Aprendizado Não Supervisionado

A etapa não supervisionada do Módulo III envolveu a utilização do algoritmo K-Means, que se baseia em centroides para agrupar os dados. O treinamento deste algoritmo foi realizado apenas com os 895 domínios maliciosos do conjunto de treinamento e utilizou as cinco características mais importantes (’`nb_days_until_collect`’, ’`srcp`’, ’`res_len`’, ’`ttl`’, ’`asn`’) com base na análise feita nos modelos supervisionados. Essas características foram normalizadas com o *StandardScaler*. A ideia foi criar *clusters* exclusivamente com domínios maliciosos e, posteriormente, medir a distância de novos pontos de dados em relação aos *clusters* formados. Quanto menor a distância de um novo dado em relação aos agrupamentos, maior a probabilidade de ser um domínio malicioso.

O algoritmo K-Means exige a definição de um valor *K*, que representa a quantidade de *clusters* a serem formados. A escolha do valor de *clusters* foi baseada em dois métodos: (i) a curva do cotovelo e (ii) o score de silhueta, ambos apresentados na Seção 2.3.2 deste trabalho.

No primeiro método, a curva do cotovelo, a melhor quantidade de clusters parece ser 4, com inicialização do K-Means usando o método `k-means++` e um valor de inércia de 604,88, em comparação com 757,23 ao usar o modo aleatório. O gráfico da curva do cotovelo está na Figura 4.3.

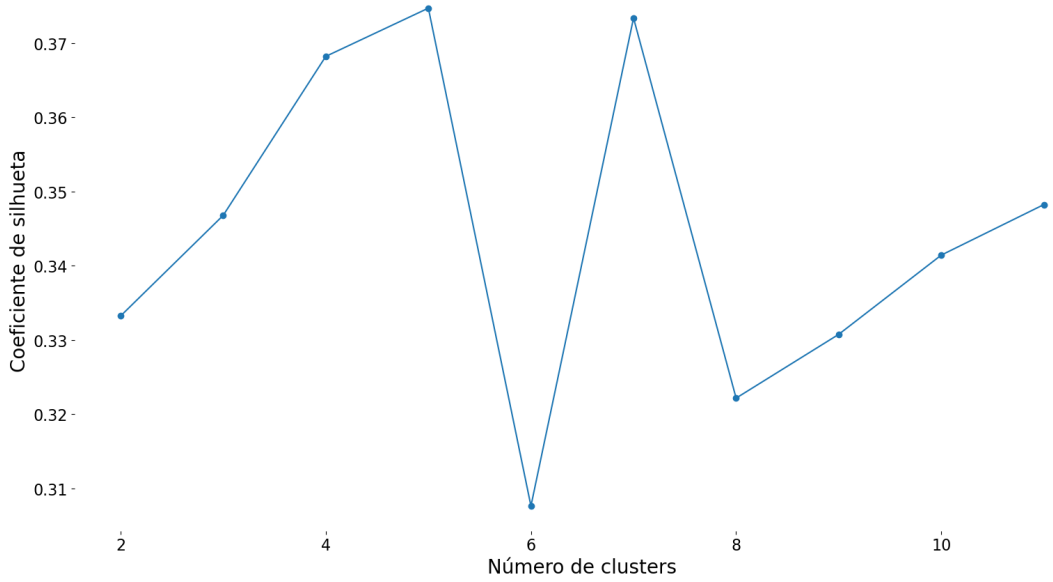
Figura 4.3: Valores de inércia por números de clusters



Fonte: Elaborado pelo autor, 2024.

O segundo método envolve a observação do valor do coeficiente de silhueta *(silhouette score)*, onde valores mais altos indicam uma melhor qualidade na formação dos clusters. Os melhores resultados foram obtidos com 5, 7 e 4 clusters, apresentando coeficientes de 0,3747, 0,3733 e 0,3682, respectivamente. Esse resultado pode ser visualizado na Figura 4.4.

Figura 4.4: Valores do coeficiente de silhueta por número de clusters



Fonte: Elaborado pelo autor, 2024.

Devido à divergência dos métodos sobre qual seria o melhor número de *clusters* para o problema em questão, foram testadas as duas melhores soluções. O método do cotovelo indicou que a melhor solução foi obtida com 4 *clusters*, enquanto o coeficiente de silhueta apontou que a melhor solução foi obtida com 5 *clusters*.

A distribuição de domínios maliciosos na etapa de treinamento para o algoritmo K-Means, utilizando 4 e 5 *clusters*, está apresentada nas Tabelas 4.5 e 4.6, respectivamente.

Tabela 4.5: Distribuição da quantidade de domínios maliciosos pelos 4 *clusters* do `K-Means`

Número do Cluster	Quantidade de domínios maliciosos
0	384
1	148
2	7
3	356

Fonte: Elaborado pelo autor, 2024.

Tabela 4.6: Distribuição da quantidade de domínios maliciosos pelos 5 *clusters* do `K-Means`

Número do Cluster	Quantidade de domínios maliciosos
0	342
1	330
2	7
3	82
4	134

Fonte: Elaborado pelo autor, 2024.

Por se tratar de um modelo não supervisionado, não há etapa de treino e validação, apenas a etapa de treino dos modelos. Portanto, ambos os modelos foram testados na composição da abordagem final, para avaliação conjunta com todo o sistema, e a partir desses dados, definiremos qual dos modelos deverá ser utilizado no ambiente de produção.

Para ambos os modelos usando o `K-Means`, a saída que o modelo deverá gerar será o número do *cluster* ao qual o dado pertence e a distância euclidiana desse dado até aquele centróide.

4.3 Modelo Classificador final

Para concluir a abordagem, é necessário que os resultados vindos dos dois modelos supervisionados (`XGBp3f0` e `SVMp3f0`) e do modelo não supervisionado sejam combinados de forma que o resultado final indique a probabilidade de um domínio ser malicioso ou legítimo. Como anteriormente não era possível realizar um teste dos *clusters* gerados, serão testadas aqui duas configurações do conjunto de dados: uma

com a saída do modelo não supervisionado resultando em 4 *clusters* e a outra com 5 *clusters*. Os conjuntos de dados de treino deste modelo final são compostos por 6 colunas, conforme apresentadas na Tabela 4.7. Essas colunas incluem os resultados dos 2 melhores modelos supervisionados treinados anteriormente e o resultado da execução do modelo não supervisionado sob este conjunto de dados que passou pela etapa de *sampling* das classes. A proporção de domínios maliciosos e legítimos se mantém a mesma apresentada no Capítulo 3.7.2, pois aqui utilizou-se a saída da etapa de treinamento.

Tabela 4.7: Características utilizadas no treinamento do modelo final

Feature	**Descrição**
`y_pred_xgb`	Classe predita pelo modelo classificador XGB
`y_pred_proba_xgb`	Probabilidade predita pelo modelo XGB do domínio ser malicioso
`y_pred_svm`	Classe predita pelo modelo classificador SVM
`y_pred_proba_svm`	Probabilidade predita pelo modelo SVM do domínio ser malicioso
`cluster_pred_kmeans`	Cluster predito pelo modelo K-Means
`distance_centroids_kmeans`	Distância euclidiana do domínio até o centróide do cluster predito
`y`	Classe real do domínio

Fonte: Elaborado pelo autor, 2024.

Foram treinados e testados três algoritmos, um utilizando a estratégia de *bagging (Random Forest)* e dois utilizando *boosting (XGBoost e LightGBM)* para os conjuntos de dados contendo as saídas de `K-Means` com 4 e 5 *clusters*. Por conta disso utilizou-se o seguinte código de modelo: as primeiras letras maiúsculas indicam o algoritmo utilizado, seguido das letras minúsculas "`fi`" indicando que este é um modelo da etapa final do sistema proposto, e por fim a letra "`K`" seguido do valor 4 para a combinação com um `K-Means` utilizando 4 como quantidade de *clusters*, ou 5 para o `K-Means` com 5 clusters.

Nas Tabelas 4.8 e 4.9 são apresentados os resultados de treino/validação e teste dos modelos finais respectivamente.

Na Tabela 4.8 que apresenta os resultados de treino e validação dos modelos é possível observar que todos apresentam uma AUC e ACC similares não havendo grandes diferenças em suas performances.

Tabela 4.8: Resultado da etapa de treino e validação dos modelos finais

Código do Modelo	Algoritmo	AUC	ACC
RFfiK4	Random Forest	0,96 (+/- 0,01)	0,91
XGBfiK4	XGBoost	0,96 (+/- 0,01)	0,90
LGBfiK4	LightGBM	0,96 (+/- 0,01)	0,90
RFfiK5	Random Forest	0,96 (+/- 0,01)	0,90
XGBfiK5	XGBoost	0,96 (+/- 0,01)	0,90
LGBfiK5	LightGBM	0,97 (+/- 0,01)	0,90

Fonte: Elaborado pelo autor, 2024.

A etapa de teste assim como a etapa de treinamento, apresenta resultados bem similares entre os modelos, repetindo o equilíbrio de desempenho e comportamento entre os modelos, tal comportamento pode ser observado na Tabela 4.9.

Tabela 4.9: Resultado da etapa de teste dos modelos finais

Código do Modelo	ACC	TVP	TVN	TFP	TFN
RFfiK4	0,88	0,884	0,875	0,124	0,110
XGBfiK4	0,88	0,836	0,878	0,121	0,163
LGBfiK4	0,88	0,875	0,875	0,124	0,125
RFfiK5	0,87	0,865	0,866	0,133	0,134
XGBfiK5	0,87	0,855	0,872	0,127	0,144
LGBfiK5	0,87	0,884	0,868	0,131	0,115

Fonte: Elaborado pelo autor, 2024.

Na etapa de treino e validação, destaca-se o algoritmo LightGBM que apresenta uma maior AUC, utilizando tanto K-means com 4 e 5 clusters, seguido do modelo RFiK4, no qual apresenta 0,01 de ACC sobre os demais modelos.

Na etapa de teste, os melhores resultados obtidos foram por meio dos modelos RFfiK4 e LGBfiK5, apresentando um valor igual na TVP entretanto com o RFfiK4 sendo 0,007 superior na TVN. A quantidade de domínio exatas e sua respectiva classificação é apresentada na matriz de confusão 4.10 para o modelo RFfiK4 e 4.11 para o modelo LGBfiK5.

Tabela 4.10: Matriz de confusão do modelo RFfiK4 utilizando dados de teste

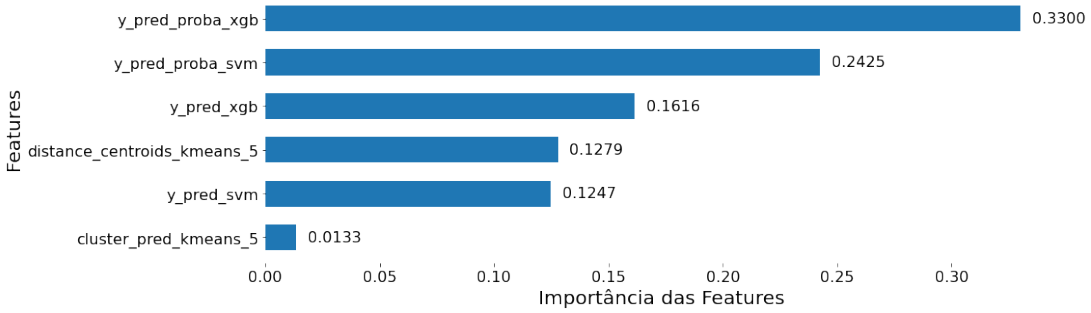
Valor Real	**Legítimo**	6830	972
	Malicioso	12	92
		Legítimo	**Malicioso**
		Valor Predito	

Tabela 4.11: Matriz de confusão do modelo `LGBfiK5` utilizando dados de teste

Valor Real	**Legitimo**	6778	1024
	Malicioso	12	92
		Legítimo	**Malicioso**
		Valor Predito	

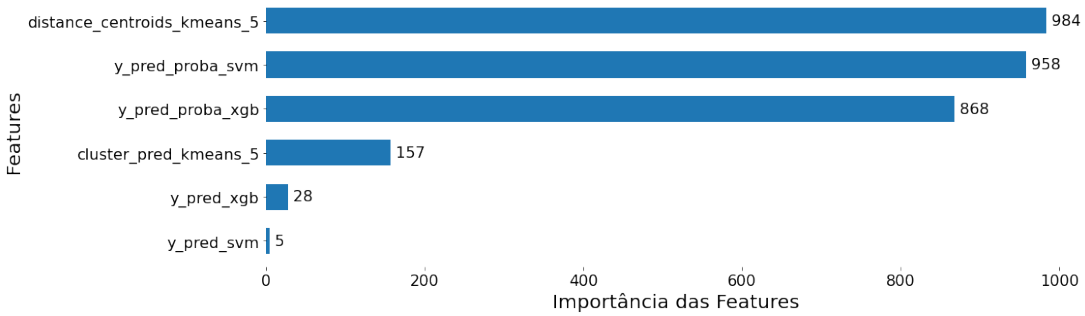
A importância das características utilizadas nos dois melhores modelos citados acima é apresentada respectivamente nas Figuras 4.5, 4.6.

Figura 4.5: Importâncias das características utilizadas no modelo RFfiK4



Fonte: Elaborado pelo autor, 2024.

Figura 4.6: Importâncias das características utilizadas no modelo LGBfiK5



Fonte: Elaborado pelo autor, 2024.

Dentre os modelos finais aqui apresentados observa-se uma consistência de desempenho, apresentando métricas muito similares. A partir dos gráficos de importância das características é observado que modelos baseados em *boosting* deram uma alta importância às saídas probabilísticas dos modelos `SVM` e `XGBoost` além da distância dos centroides fornecidos pelo modelo não supervisionado, enquanto que o modelo de `Random Forest` pesa mais a saída probabilística do modelo `XGBoost` e `SVM`. O modelo escolhido como modelo final é o `RFfiK4`, pois apesar de apresentar um resultado similar, este modelo gera uma quantidade menor de TFP e TFN e consequentemente

apresenta métricas com TVP e TVN superiores ao modelo LGBfiK5. É importante observar como algoritmos com estratégias diferentes *(bagging e boosting)* conseguiram chegar em resultados similares e com pesos diferentes na importância de suas características. Outro ponto de observação é que os dois melhores modelos aqui apresentados, utilizam a saída do modelo não supervisionado com diferentes valores de clusters, onde o LGMfik5, utiliza o *K5*, o que apresenta uma granularidade um pouco maior dos dados clusterizados, se comparado ao *K4* utilizado pelo RFfiK4.

4.4 Discussão

Na etapa de treinamento de modelos supervisionados, utilizou-se os seguintes algoritmos: XGBoost, LighGBM e SVM, todos sem ajustes de hiperparâmetros. Para treinamento dos modelos, além dos dois pipelines apresentados, utilizou-se também um terceiro no qual consiste apenas em uma normalização dos dados. Na etapa de treinamento a AUC média variou de 0,95 a 0,97 com SVMp3f0 e LGBp3f0 respectivamente. Ainda na etapa de treinamento todos modelos apresentaram uma ACC de 0,91 com exceção de dois modelos XGBp1f0 e XGBp3f0. Na etapa de validação desta-se o resultado do modelo XGBp3f0, que foi o melhor modelo dentre os modelos que utilizaram algoritmos baseados em árvore e aos resultados consistentes dos modelos SVM, os quais obtiveram a melhor TVP do experimento com um valor de 0,971 contra 0,846 obtido pelos modelos XGBp3f0 e LGBp2f0. Consequentemente os modelos SVM também obtiveram uma menor taxa de TFN sendo de 0,028. Sua TVN ficou 0,04 abaixo dos modelos XGBp3f0 e LGBp2f0.

Observou-se então que 5 características possuíam pouca ou nenhuma importância para os modelos baseados em XGBoost e LightGBM e com a remoção destas, um novo treinamento aconteceu com todos os modelos, o que possibilitou observar uma leve melhora na AUC média na etapa de treinamento. Entretanto, na etapa de validação houve uma leve queda de desempenho dos modelos baseados em árvores nos quais utilizaram o *pipeline* 3 se comparado ao experimento anterior. O destaque porém fica por conta dos modelos XGBp1f1 e LGBp1f1 nos quais houve uma queda brusca em TVP, ambos apresentando 0,740. Nos modelos SVM houve uma pequena queda de desempenho em sua TVP apresentando resultados de 0,961 e 0,951 para os *pipelines* 3 e 1 respectivamente, ainda assim foram os melhores modelos se comparados a TVP.

No treinamento do modelo não supervisionado utilizou-se o algoritmo K-Means, no qual dois métodos de avaliar o melhor valor de K divergiram entre os valores de 4 e 5. Portanto o treinamento ocorreu com ambas configurações de modelos para

uma avaliação posterior de desempenho do modelo final. A etapa de treinamento destes modelos foi realizada utilizando apenas dados de domínios maliciosos, isso porque, pretende-se formar *clusters* onde quanto mais próximo destes *clusters* maior a probabilidade deste domínio ser malicioso. Após o treinamento, o conjunto de dados balanceado passou pelo modelo para gerar as saídas necessárias para juntar com as saídas dos modelos anteriores e realizar o treinamento e validação do modelo final.

Entre os algoritmos testados para ser o modelo final, o LightGBM apresentou o melhor desempenho quando utilizado com o K-Means com 5 *clusters*, enquanto o *Random Forest* mostrou-se como a melhor escolha com 4 *clusters*. Como esperado, ambos se mostraram mais equilibrados que os modelos supervisionados anteriormente mencionados. O modelo escolhido como o modelo final e parâmetro de comparação com os demais modelos foi o *Random Forest*, pois este apresenta métricas de desempenho superiores ao *LightGBM*, resultando assim em uma maior assertividade na predição e, consequentemente, gerando menos falso positivos. Na Tabela 4.12 é apresentado um comparativo de desempenho entre o resultado do modelo final selecionado e os modelos supervisionados que compõem tal abordagem.

Tabela 4.12: Comparativo de desempenho entre modelos individuais e modelo final

	Treinamento/Validação		**Teste**				
Código do Modelo	**AUC**	**ACC**	**ACC**	**TVP**	**TVN**	**TFP**	**TFN**
`XGBp3f0`	0,97 (+/- 0,02)	0,92	0,89	0,846	0,887	0,110	0,153
`SVMp3f0`	0,95 (+/- 0,02)	0,91	0,85	0,971	0,847	0,152	0,028
`RFfiK4`	**0,96(+/- 0,01)**	**0,91**	**0,88**	**0,884**	**0,875**	**0,124**	**0,110**

Fonte: Elaborado pelo autor, 2024.

De uma forma geral, o modelo final, treinado com os dados das saídas dos algoritmos supervisionados e não supervisionados, apresenta um desempenho mais equilibrado e seu desempenho fica entre os modelos treinados separadamente.

Em comparação com o modelo `XGBp3f0`, ele superou em 3,8% na TVP e 4,3% na TFN. Quanto menor o TFN, menor a quantidade de falsos negativos gerados. Essas métricas indicam que o modelo é melhor na detecção desses domínios maliciosos e gerou uma quantidade menor de falsos negativos. No entanto, o modelo final foi inferior em 1,0% na ACC, 1,2% na TVN e 1,4% na TFP. Isso indica que o modelo final gerou 1,4% a mais de falsos positivos, detectando 1,2% a menos de domínios legítimos corretamente.

Quando comparado com o modelo `SVMp3f0`, ele mostrou ser superior em 3,0% na ACC, 2,8% na TVN e 2,8% na TFP. Esse desempenho mostra que o modelo final

consegue detectar mais domínios legítimos corretamente e gera menos falsos positivos. No entanto, ele é inferior em 8,7% na TVP e 8,2% na TFN, reforçando os pontos positivos do modelo SVM, que se destacava pela sua assertividade na detecção de domínios maliciosos.

Em resumo, o modelo final se apresenta mais equilibrado, com resultados mais semelhantes tanto na detecção de domínios maliciosos quanto de legítimos.

A partir dos gráficos de importância das características, observa-se que modelos que utilizam a estratégia de *boosting* atribuíram uma importância maior às características vindas do modelo não supervisionado em comparação com os modelos que utilizaram a estratégia de *bagging*.

Comparando o desempenho da abordagem semi-supervisionada desenvolvida neste trabalho com trabalhos anteriores, Antonakakis et al. (2011) e Silveira et al. (2022) são os que mais se assemelham, pois atacam o problema de domínios maliciosos em TLDs utilizando DNS passivo. Entretanto, a abordagem aqui desenvolvida não agrega consultas, ao contrário do que é feito nos trabalhos citados e outros correlatos. Ela realiza a detecção usando uma abordagem semi-supervisionada e como fonte de dados apenas a primeira consulta de nome de domínio.

Ao destacar as semelhanças e diferenças, quando comparamos as métricas das abordagens desenvolvidas, os autores do Kopis (ANTONAKAKIS et al., 2011), desenvolvido para o TLD `.ca`, apresentam uma taxa de verdadeiro positivo de 98,4% contra todos os domínios relacionados a *malware* e 73,6% contra novos domínios relacionados a *malware* não classificados anteriormente. O trabalho desenvolvido aqui obteve uma TVP de 88,4% para novos domínios registrados e uma TFP de 12,4%, em comparação com 0,5% do *Kopis*. Silveira et al. (2022), que também faz uso de pDNS de servidores autoritativos, apresenta resultados de TVP de 0,8656 e 0,8682 na etapa de teste dos modelos que detectam domínios maliciosos no período de 3 e 7 dias, respectivamente.

A partir desses dados, é possível concluir que a abordagem desenvolvida aqui apresenta um caráter inédito, uma vez que não há registros de abordagens que foquem na detecção de domínios maliciosos em TLDs logo em sua primeira consulta, nem que explorem tal abordagem para o problema apresentado. Quanto ao desempenho, a abordagem apresenta resultados de TVP superiores aos apresentados em Silveira et al. (2022) e também superiores aos de Antonakakis et al. (2011), quando se compara a detecção de novos domínios relacionados a *malware* não classificados anteriormente, e inferiores a este mesmo trabalho quando comparado ao seu poder de detecção para todos os domínios relacionados a *malwares*.

4.5 Considerações Parciais

A abordagem apresentada neste trabalho consiste em modelos de aprendizagem supervisionada trabalhando em conjunto com um modelo não supervisionado, resultando em uma abordagem semi-supervisionada. Esta abordagem visa a detecção de domínios maliciosos recém-registrados logo em sua primeira consulta.

Através dos experimentos realizados, observou-se que a abordagem desenvolvida obteve resultados promissores em comparação com trabalhos correlatos. Ao analisar o desempenho dos modelos individualmente, destacam-se o modelo `SVMp3f0`, que obteve uma alta Taxa de Verdadeiro Positivo (TVP), e o modelo `XGBp3f0`, que apresentou uma Taxa de Verdadeiro Negativo (TVN) superior aos demais durante a etapa inicial de desenvolvimento.

O treinamento do modelo não supervisionado utilizou apenas dados reais de domínios maliciosos e se concentrou nas 5 características mais importantes, conforme identificado pelos modelos supervisionados. Isso resultou na formação de clusters de domínios puramente maliciosos. O modelo final foi treinado usando as saídas dos modelos previamente treinados (`SVMp3f0` e `XGBp3f0`), combinadas com as saídas dos modelos `K-Means` com 4 ou 5 clusters.

Os melhores modelos resultantes dessas combinações foram o `LGBfiK5` e o `RFfiK4`. No entanto, o `RFfiK4` foi escolhido como modelo final, pois apresentou resultados ligeiramente superiores ao `LGBfiK5`, com 0,7% a mais de TVN, 0,7% a menos de TFP e 0,5% a menos de TFN. Isso significa que o modelo tem a mesma capacidade de detectar domínios maliciosos (TVP igual) e uma capacidade um pouco superior de identificar domínios legítimos, resultando em menos falsos positivos, como comprovado pela matriz de confusão dos modelos.

A abordagem semi-supervisionada desenvolvida neste trabalho demonstra a capacidade de detectar domínios maliciosos logo em sua primeira consulta DNS, apresentando resultados sólidos e mais equilibrados em comparação ao desempenho dos algoritmos individuais. Assim, esta abordagem se diferencia dos outros trabalhos no estado da arte, pois se concentra na detecção de domínios maliciosos recém-registrados na primeira consulta DNS. Além disso, a estratégia semi-supervisionada desenvolvida para essa finalidade é única, permitindo identificar esses domínios o mais cedo possível, reduzindo o número de vítimas e tornando a Internet mais segura.

Capítulo 5

Conclusões

Este capítulo tem como objetivo apresentar as conclusões gerais obtidas por meio do trabalho desenvolvido, os trabalhos futuros, as limitações desta tese, a conclusão final e, por fim, as produções geradas.

Na seção 5.1, serão expostas as conclusões gerais do trabalho. Na seção 5.2, são discutidas sugestões para trabalhos futuros. Na seção 5.3, são abordadas as limitações deste estudo. Na seção 5.4, serão delineadas as conclusões finais. Por fim, na seção 5.5, serão descritas as produções geradas a partir do estudo aqui apresentado.

5.1 Conclusões Gerais

Nesta tese, foi desenvolvida uma abordagem semi-supervisionada para a detecção de domínios maliciosos em TLDs, utilizando apenas sua primeira consulta. A análise dos resultados obtidos com esta abordagem não apenas valida sua eficácia, mas também destaca sua contribuição para fortalecer a segurança da internet.

Este trabalho se destaca por apresentar uma abordagem inédita e significativa na detecção precoce de domínios maliciosos em TLDs, concentrando-se na análise da primeira consulta desses domínios e empregando uma abordagem semi-supervisionada, que utiliza pDNS de um servidor autoritativo como fonte de dados. A utilização de uma abordagem semi-supervisionada neste contexto representa uma inovação. Além disso, o emprego pDNS como fonte de dados primária deve ser destacado, uma vez que a obtenção de tal informação, é pouco explorado na literatura, conferindo uma distinção significativa a esta tese.

Através da coleta passiva do DNS de um servidor autoritativo do TLD, foi possível construir a base de dados utilizada no desenvolvimento deste trabalho, que poderá ser empregada em futuras pesquisas e no avanço de ferramentas analíticas avançadas. Ini-

cialmente, os dados foram coletados em formato PCAP e posteriormente armazenados pelo ENTRADA em seu *cluster Hadoop* em formato *Parquet*, sendo enriquecidos com informações de GeoIP. Essa estruturação permitiu uma consulta eficiente dos dados utilizando linguagens como *Python*, aproveitando suas bibliotecas compatíveis para análise e interpretação dos dados de forma adequada.

Durante a etapa de processamento dos dados, que incluiu a seleção da primeira consulta dos domínios recém-registrados e a rotulação desses como maliciosos ou legítimos, foi possível avançar para o desenvolvimento dos modelos supervisionados e não supervisionados.

A análise após a rotulação dos dados revelou um desequilíbrio extremo entre as classes, levando à opção de balanceamento híbrido durante a etapa de treinamento dos modelos supervisionados. Essa abordagem consistiu em aumentar a quantidade de casos da classe minoritária enquanto reduzia da classe majoritária. Com base nesses dados e nos resultados obtidos durante a etapa de treinamento e validação, dois modelos com estratégias distintas foram selecionados para integrar a solução final: um modelo *XGBoost* e um modelo *SVM*. Essa diversidade de estratégias entre os modelos complementares contribui para uma abordagem mais robusta na solução final.

O treinamento do modelo não supervisionado utilizou apenas os dados reais rotulados na etapa anterior, sem qualquer manipulação, exceto pela exclusão dos domínios legítimos. É conhecido que quanto maior a dimensionalidade, mais complexo o modelo se torna. Para este treinamento, foram selecionadas as cinco características mais importantes, com base na importância das características apresentadas nos modelos supervisionados. Além de reduzir a dimensionalidade do conjunto de dados, essa seleção concentra o modelo nas características mais relevantes e informativas, potencialmente melhorando a qualidade da segmentação dos dados e contribuindo para um processo de clusterização mais eficaz e interpretável.

Essa abordagem resultou na formação de clusters contendo apenas domínios maliciosos. Intuitivamente, quanto mais próximo um domínio desconhecido estiver de algum *cluster* gerado, maior a probabilidade de ser um domínio malicioso, apresentando comportamento semelhante aos exemplos utilizados no treinamento. Destaca-se mais uma vez que, na abordagem deste trabalho não há clusters de domínios legítimos, uma vez que o objetivo desta etapa é agrupar apenas os domínios maliciosos afim de de encontrar novos domínios registrados que apresentem o mesmo comportamento dos já mapeados. Após esta etapa de treinamento, onde foram gerados os clusters, os dados balanceados utilizados anteriormente foram inseridos para que o modelo realizasse a predição. Nesse caso, foram gerados dois modelos, um com quatro clusters e outro

com cinco. A diferença na quantidade de clusters gerados ocorreu devido a divergências nos métodos utilizados para determinar a melhor quantidade de clusters, portanto, ambos os resultados foram analisados e validados na composição da solução final da abordagem.

As informações de saída deste modelo não supervisionado - número do *cluster* ao qual o domínio pertence e sua distância até o centroide - juntamente com as saídas dos modelos supervisionados, constituíram o conjunto de dados de treinamento do modelo final, permitindo uma estratégia de criação de um modelo *ensemble*.

Reitera-se que a abordagem proposta é semi-supervisionada. Desse modo, como descrito no capítulo 3, o sistema proposto é composto por modelos supervisionados, como *XGBoost* e *SVM*, e por clusters, cujo processo de aprendizagem é não supervisionado. Essa configuração de duas formas de aprendizado foi necessária pois trouxe um melhor resultado a abordagem final, fazendo com o que o modelo final fosse capaz de detectar domínios maliciosos e legítimos com capacidade similar, melhorando assim as fraquezas dos modelos individuais. Assim, o modelo final é alimentado pelos resultados de modelos representantes de duas formas de aprendizado distintas, aproveitando as vantagens de cada abordagem. Os resultados gerados pelos modelos anteriormente treinados e selecionados compuseram o conjunto de dados de treinamento final, conforme apresentado na Tabela 4.7 A partir desse conjunto de dados, o modelo final foi treinado e validado. O algoritmo escolhido para o modelo final foi o *Random Forest* pois, quando comparado aos outros modelos finais considerados, ele apresentou os melhores resultados e por isso foi empregado como modelo final do sistema proposto.

Os resultados gerados pelos modelos anteriormente treinados e selecionados compuseram o conjunto de dados de treinamento final. A partir desse conjunto de dados, o modelo final foi treinado e validado. O resultado deste modelo final é o que é informado ao analista no relatório de detecção, fornecendo a probabilidade do domínio analisado ser malicioso. As métricas de desempenho obtidas neste classificador final são as métricas desta tese, e através do treinamento individual dos modelos realizados anteriormente, foi possível mensurar o comportamento mais equilibrado da abordagem em relação aos resultados dos modelos individuais.

Com o propósito de garantir a acurácia dos testes dos modelos, tanto individualmente quanto como etapa final do processo, foi simulado um ambiente de produção. Neste contexto, foram incluídos dados de domínios registrados nos meses de novembro e dezembro de 2021. Essa abordagem visa assegurar não apenas a integridade dos testes dos modelos, mas também uma simulação mais próxima do ambiente real de produção. Vale ressaltar que os dados de treinamento correspondem a domínios regis-

trados até a data limite de 31 de outubro, enquanto todos os domínios incluídos nos testes tiveram suas primeiras consultas realizadas até a data de 10 de maio de 2022. Essa estratégia visa maximizar a fidelidade da simulação ao ambiente de produção real.

Uma preocupação inerente a esta tese é o potencial impacto da detecção de domínios maliciosos na alteração do comportamento dos agentes maliciosos responsáveis pelo registro desses domínios. Diante da possibilidade de uma resposta adaptativa por parte desses agentes, esta abordagem adota uma medida proativa para lidar com essa situação. Um módulo de re-treino do modelo é disponibilizado, permitindo a atualização contínua dos modelos sempre que houver uma degradação no desempenho da abordagem. Essa capacidade de adaptação contínua é essencial para manter a eficácia do sistema de detecção ao longo do tempo, garantindo que ele permaneça resiliente diante das mudanças no comportamento dos domínios maliciosos.

A abordagem desenvolvida neste trabalho revelou-se eficaz na detecção precoce de domínios maliciosos em sua primeira consulta. Os resultados obtidos destacam a capacidade distintiva do modelo `SVMp3f0` e do `XGBp3f0` na detecção de domínios maliciosos e legítimos, respectivamente. Além disso, o modelo final `RFfiK4` demonstrou um desempenho equilibrado, integrando as saídas dos modelos mencionados anteriormente com a saída do modelo *K-Means* de 4 clusters. Essa abordagem resultou em um desempenho mais robusto e uma melhoria nas limitações dos modelos supervisionados quando considerados individualmente.

Durante as etapas de treinamento e validação, a abordagem alcançou uma AUC de 0,96 (+/- 0,01) e uma ACC de 0,91. Nos dados de teste, projetados para simular o ambiente de produção, o modelo alcançou um bom desempenho com as seguintes métricas: TVP: 0,884, TVN: 0,875, TFP: 0,124 e TFN: 0,11. Esses resultados evidenciam não apenas a eficácia da abordagem na identificação de domínios maliciosos em sua primeira consulta, mas também sua consistência e equilíbrio superiores em comparação com a utilização de modelos supervisionados individuais, uma prática comum em trabalhos correlatos.

Apesar dos resultados promissores, destaca-se uma redução na TVP do modelo final em comparação com o bom desempenho do modelo `SVMp3f0` nessa métrica específica. Essa observação sugere um potencial menos explorado do algoritmo *SVM* em comparação com os algoritmos baseados em árvores em trabalhos relacionados. Portanto, há espaço para trabalhos futuros, e essas sugestões são apresentadas na Seção 5.2

5.2 Trabalhos futuros

O trabalho atual testou o algoritmo não supervisionado *K-Means* com o objetivo de criar clusters puramente maliciosos e medir a distância dos novos dados em relação a esses clusters. Uma sugestão para trabalhos futuros é explorar algoritmos não supervisionados com diferentes características que permitam medir distâncias e avaliar o desempenho da abordagem de detecção de domínios maliciosos. Isso pode incluir a composição da abordagem, como feito neste trabalho, ou a criação de um sistema de classificação mais preciso para tipos específicos de maliciosidade presentes em domínios analisados, permitindo identificar quais domínios apresentam indícios de *phishing*, DGA, entre outros.

Além disso, domínios maliciosos podem apresentar mudanças de comportamento ao longo do tempo, seja de forma sazonal, abrupta ou permanente. Se ocorrerem alterações no comportamento dos domínios, é esperado que os modelos de detecção se degradem. Uma sugestão para trabalhos futuros é realizar estudos de longo prazo com um espaço amostral maior para observar essas mudanças. Conceitos como *"concept drift"* podem ser aplicados para identificar automaticamente alterações de comportamento nos modelos e desenvolver *pipelines* de re-treinamento automático ou treinamento incremental, caso os algoritmos utilizados suportem essa abordagem. Isso ajudaria a combater a degradação natural dos modelos ao longo do tempo e a se adaptarem as possíveis mudanças bruscas no comportamento dos domínios maliciosos, à medida que os atacantes tentam se adaptar às estratégias de detecção.

5.3 Limitações

Apesar dos avanços apresentados neste trabalho e da inovação na abordagem desenvolvida, há algumas limitações intrínsecas que merecem destaque:

- A utilização de pDNS coletado apenas de um TLD pode limitar a generalização do sistema, uma vez que o comportamento dos domínios pode variar entre diferentes TLDs. Portanto, caso o sistema treinado seja aplicado em outro TLD, pode ser necessário realizar um retreinamento dos modelos para adaptá-los às características específicas do novo conjunto de dados. Uma possibilidade de melhoria nesta limitação seria o treinamento do modelo com dados de diferentes domínios recém-registrados, abrangendo uma diversidade de TLDs. Isso poderia resultar em uma melhor generalização do modelo, tornando o sistema de detecção mais eficiente e aplicável a um maior número de TLDs. Com uma vari-

edade de comportamentos dos domínios incorporados durante o treinamento, os modelos presentes na abordagem estariam preparados para identificar padrões convergentes ou divergentes, ampliando assim a aplicabilidade da abordagem a diversos TLDs sem a necessidade de retreinamento imediato do sistema.

- A abordagem se restringe à análise dos dados da primeira consulta, o que possibilita a detecção precoce de domínios maliciosos. No entanto, essa limitação impede uma visão contínua do ciclo de vida desses domínios, dificultando a identificação de padrões de comportamento ao longo do tempo. Nesse sentido, considera-se o desenvolvimento de uma nova abordagem, levantando a hipótese de que seja possível analisar os padrões de comportamento de domínios que já possuem um longo tempo de registro. Dessa forma, seria viável identificar mudanças comportamentais em suas consultas, tanto em termos quantitativos quanto de geolocalização. Tais mudanças poderiam indicar um uso indevido do domínio ou alguma anomalia, levando em conta seu histórico de uso saudável.

- Devido à estratégia adotada de utilizar o pDNS como fonte de dados, cada domínio passa pelo sistema apenas uma vez, após sua primeira consulta. Isso significa que a abordagem pode perder informações importantes sobre o comportamento dos domínios após a primeira interação.

- A etapa de re-treino do modelo depende da intervenção de um analista, o que pode gerar atrasos na atualização do sistema em resposta a mudanças no comportamento dos domínios maliciosos. Uma abordagem automatizada para acionar o re-treino do modelo poderia melhorar a agilidade do sistema em se adaptar a novas ameaças. Os conceitos de *concept drift* e *data drift* são potencialmente válidos na exploração da solução dessa limitação. Por meio deles, seria possível identificar uma mudança de comportamento nos dados de entrada do modelo, o que poderia acionar um retreinamento automático sempre que tal mudança fosse detectada. Isso poderia afetar as métricas dos modelos mas, possivelmente, ao aplicar esses conceitos e desenvolver um fluxo automático de retreinamento dos modelos, é esperado que suas métricas de desempenho oscilem dentro de um limite aceitável, tendendo a se estabilizar.

5.4 Conclusão Final

Conclui-se que o presente estudo desempenhou um papel significativo no avanço da detecção de domínios maliciosos. Por meio da integração de técnicas de apren-

dizado de máquina supervisionado e não supervisionado, foi elaborada uma abordagem inovadora para identificar prontamente domínios suspeitos logo em sua primeira consulta. A validação da metodologia proposta com dados reais, simulando um ambiente de produção, reforça a sua eficácia e pertinência no contexto da cibersegurança. Destaca-se, assim, que este trabalho se destina a servir como ponto de referência para pesquisadores e profissionais, fornecendo não apenas *insights* valiosos, mas também métodos comprovados para investigações futuras nesta área. Essa contribuição representa um avanço significativo no campo da cibersegurança, oferecendo novas perspectivas e estratégias para lidar com ameaças digitais em todo o mundo.

Enfim, vale ressaltar que este trabalho não resolve todos os problemas de identificação de domínios maliciosos em TLDs, mas pode ser utilizado como mais uma camada de segurança trabalhando cooperativamente com abordagens baseadas na etapa de pré-registro que utilizam características léxicas e cadastrais, e pós-registro que utiliza dados provenientes de coleta ativa de DNS a fim de identificar alguma anomalia antes de que o domínio possua uma consulta. Além disso, é importante considerar o acompanhamento de vida do domínio por meio de pDNS, o que representa uma camada adicional de proteção. Essa prática permite a análise de comportamentos ao longo do tempo, possibilitando a detecção de alterações sutis que podem indicar atividades maliciosas.

5.5 Produções

Quanto às produções geradas neste trabalho, foram publicados dois artigos. O primeiro, intitulado *"Detection of Newly Registered Malicious Domains through Passive DNS"*, foi publicado na *"IEEE International Conference on Big Data (Big Data)"* no ano de 2021 e encontra-se disponível através do DOI https://doi.org/10.1109/BigData52589.2021.9671348. O segundo, com o título *"Early Identification of Abused Domains in TLD through Passive DNS Applying Machine Learning Techniques"*, foi publicado no *"International Journal of Communication Networks and Information Security (IJCNIS)"* no ano de 2022 e pode ser consultado no DOI https://doi.org/10.17762/ijcnis.v14i1.5256.

Referências

3RD, D. E. E. *Domain Name System (DNS) IANA Considerations*. [S.l.]: RFC Editor, 2011. RFC 6195. (Request for Comments, 6195).

AHMED, M.; SERAJ, R.; ISLAM, S. M. S. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, MDPI, v. 9, n. 8, p. 1295, 2020.

ALRWAIS, S.; YUAN, K.; ALOWAISHEQ, E.; LI, Z.; WANG, X. Understanding the dark side of domain parking. In: *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. [S.l.: s.n.], 2014. p. 207–222.

ANTONAKAKIS, M.; PERDISCI, R.; DAGON, D.; LEE, W.; FEAMSTER, N. Building a dynamic reputation system for DNS. *Proceedings of the 19th USENIX Security Symposium*, p. 273–289, 2010.

ANTONAKAKIS, M.; PERDISCI, R.; LEE, W.; VASILOGLOU, N.; DAGON, D. Detecting malware domains at the upper DNS hierarchy. *Proceedings of the 20th USENIX Security Symposium*, p. 411–426, 2011.

ARTHUR, D.; VASSILVITSKII, S. K-means++ the advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. [S.l.: s.n.], 2007. p. 1027–1035.

BAO, Z.; WANG, W.; LAN, Y. Using Passive DNS to Detect Malicious Domain Name. *ACM International Conference Proceeding Series*, 2019. Disponível em: https://dl.acm.org/doi/abs/10.1145/3387168.3387236.

BATISTA, G. E.; PRATI, R. C.; MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, ACM New York, NY, USA, v. 6, n. 1, p. 20–29, 2004.

BILGE, L.; KIRDA, E.; KRUEGEL, C.; BALDUZZI, M.; ANTIPOLIS, S. EXPOSURE : Finding Malicious Domains Using Passive DNS Analysis. *Ndss*, p. 1–17, 2011. ISSN 10949224. Disponível em: https://sites.cs.ucsb.edu/ chris/research/doc/ndss11_exposure.pdf.

BREIMAN JEROME FRIEDMAN, R. A. O. C. J. S. L. *Classification and regression trees*. [S.l.]: CRC, 1984. (The Wadsworth statistics / probability series). ISBN 0-412-04841-8.

BRUCE, A.; BRUCE, P. *Estatística Prática para Cientistas de Dados*. [S.l.]: Alta Books, 2019.

CARVALHO, A.; FACELI, K.; LORENA, A.; GAMA, J. Inteligência artificial: uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*, 2011.

CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, v. 13-17-Augu, p. 785–794, 2016.

CORPORATION, M. *LightGBM Documentation*. 2021. https://lightgbm.readthedocs.io/en/latest/. Accessed on October 13, 2021.

CUI, H.; HUANG, D.; FANG, Y.; LIU, L.; HUANG, C. Webshell detection based on random forest–gradient boosting decision tree algorithm. In: IEEE. *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*. [S.l.], 2018. p. 153–160.

DAIGLE, L. *WHOIS Protocol Specification*. RFC Editor, 2004. RFC 3912. (Request for Comments, 3912). Disponível em: https://rfc-editor.org/rfc/rfc3912.txt.

DARWISH, S. M.; ANBER, A. E.; MESBAH, S. Bio-inspired machine learning mechanism for detecting malicious url through passive dns in big data platform. In: *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*. [S.l.]: Springer, 2021. p. 147–161.

DAVIS, J.; GOADRICH, M. The relationship between precision-recall and roc curves. In: ACM. *Proceedings of the 23rd international conference on Machine learning*. [S.l.], 2006. p. 233–240.

DESMET, L.; SPOOREN, J.; VISSERS, T.; JANSSEN, P.; JOOSEN, W. Premadoma: An Operational Solution to Prevent Malicious Domain Name Registrations in the .Eu TLD. *Digital Threats: Research and Practice*, v. 2, n. 1, p. 1–24, 2021. ISSN 2692-1626. Disponível em: https://remote-lib.ui.ac.id:2075/10.1145/3419476.

DOUZAS, G.; BACAO, F.; LAST, F. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, Elsevier, v. 465, p. 1–20, 2018.

ENGELEN, J. E. V.; HOOS, H. H. A survey on semi-supervised learning. *Machine learning*, Springer, v. 109, n. 2, p. 373–440, 2020.

FALL, K. R.; STEVENS, W. R. *TCP/IP illustrated, volume 1: The protocols*. [S.l.]: addison-Wesley, 2011.

FAWCETT, T. ROC Graphs: Notes and Practical Considerations for Researchers BT - Tech Report HPL-2003-4, HP Laboratories. p. 1–38, 2004. Disponível em: http://binf.gmu.edu/mmasso/ROC101.pdf.

FERNÁNDEZ, A.; GARCÍA, S.; GALAR, M.; PRATI, R. C.; KRAWCZYK, B.; HERRERA, F. *Learning from imbalanced data sets*. [S.l.]: Springer, 2018. v. 10.

GAO, Y.; YUAN, F.; YANG, J.; WANG, D.; CAO, C.; LIU, Y. Semi-supervised malicious domain detection based on meta pseudo labeling. In: SPRINGER. *International Conference on Computational Science*. [S.l.], 2024. p. 312–324.

GARCÍA, V.; SÁNCHEZ, J. S.; MOLLINEDA, R. A. Classification of high dimensional and imbalanced hyperspectral imagery data. In: SPRINGER. *Iberian conference on pattern recognition and image analysis*. [S.l.], 2011. p. 644–651.

GEORGE, A.; VIDYAPEETHAM, A. Anomaly detection based on machine learning dimensionality reduction using pca and classification using svm. *International Journal of Computer Applications*, Citeseer, v. 47, n. 21, p. 5–8, 2012.

GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. [S.l.]: O'Reilly Media, 2019.

GHAHRAMANI, Z. Unsupervised learning. In: SPRINGER. *Summer School on Machine Learning*. [S.l.], 2003. p. 72–112.

HARRENSTIEN, K.; WHITE, V. *NICNAME/WHOIS*. RFC Editor, 1982. RFC 812. (Request for Comments, 812). Disponível em: https://rfc-editor.org/rfc/rfc812.txt.

HARRENSTIEN K., S. M.; FEINLER, E. *DoD Internet host table specification*. RFC Editor, 1985. RFC 952. (Request for Comments, 952). Disponível em: https://rfc-editor.org/rfc/rfc952.txt.

HUYEN, C. *Designing machine learning systems*. [S.l.]: "O'Reilly Media, Inc.", 2022.

IANA. *Root Servers*. 2021. https://www.iana.org/domains/root/servers. Accessed on October 13, 2021.

ICANN. *Domain Name Registration Process*. 2021. https://www.icann.org/. Accessed on October 13, 2021.

INNOVATOR, I. H. of F. *Paul Mockapetris*. 2021. https://internethalloffame.org/inductees/paul-mockapetris. Accessed on October 13, 2021.

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. *Advances in neural information processing systems*, v. 30, p. 3146–3154, 2017.

KETCHEN, D. J.; SHOOK, C. L. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, Wiley Online Library, v. 17, n. 6, p. 441–458, 1996.

KIDMOSE, E.; LANSING, E.; BRANDBYGE, S.; PEDERSEN, J. M. Detection of malicious and abusive domain names. *Proceedings - 2018 1st International Conference on Data Intelligence and Security, ICDIS 2018*, p. 49–56, 2018.

KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, v. 160, p. 3–24, 2007.

KOTU, V.; DESHPANDE, B. *Data science: concepts and practice*. [S.l.]: Morgan Kaufmann, 2018.

KOUNTOURAS, A.; KINTIS, P.; LEVER, C.; CHEN, Y.; NADJI, Y.; DAGON, D.; ANTONAKAKIS, M.; JOFFE, R. Enabling network security through active dns datasets. In: SPRINGER. *International Symposium on Research in Attacks, Intrusions, and Defenses*. [S.l.], 2016. p. 188–208.

KULIKOVA, T.; SHCHERBAKOVA, T. *Spam and phishing in Q1 2021*. 2021. https://securelist.com/spam-and-phishing-in-q2-2021/103548/. Accessed on October 15, 2021.

KULIKOVA, T.; SIDORINA, T.; SHCHERBAKOVA, T. *Spam and phishing in Q1 2021*. 2021. https://securelist.com/spam-and-phishing-in-q1-2021/102018/. Accessed on October 15, 2021.

KULIKOVA, T.; SIDORINA, T.; SHCHERBAKOVA, T. *Spam and phishing in Q2 2020*. 2021. https://securelist.com/spam-and-phishing-in-q2-2020/97987/. Accessed on October 15, 2021.

KULIKOVA, T.; SVISTUNOVA, O.; KOVTUN, A.; SHIMKO, I.; DEDENOK, R. *Spam and phishing in Q1 2021*. 2024. https://securelist.com/spam-phishing-report-2023/112015/. Accessed on March 15, 2024.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet*. [S.l.: s.n.], 2013. 1–658 p. ISBN 9788543014432.

LISON, P.; MAVROEIDIS, V. Neural reputation models learned from passive DNS data. *Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017*, v. 2018-Janua, n. June, p. 3662–3671, 2017.

LLOYD, S. Least squares quantization in pcm. *IEEE transactions on information theory*, IEEE, v. 28, n. 2, p. 129–137, 1982.

LYNN, M. S. A Unique, Authoritative Root for the DNS. *The Internet Protocol Journal*, v. 4, n. 3, 2001. Disponível em: https://www.icann.org/resources/pages/unique-authoritative-root-2012-02-25-en.

MASUD, M. A.; HUANG, J. Z.; WEI, C.; WANG, J.; KHAN, I.; ZHONG, M. I-nice: A new approach for identifying the number of clusters and initial cluster centres. *Information Sciences*, Elsevier, v. 466, p. 129–151, 2018.

MCGEE, M. *Google Wins 750+ Domains From Cybersquatter Who Wants 'Google' Trademark Canceled*. 2012.

MOCKAPETRIS, P. *Domain names - concepts and facilities*. [S.l.]: RFC Editor, 1987. RFC 1034. (Request for Comments, 1034).

MOCKAPETRIS, P. *Domain names - implementation and specification*. [S.l.]: RFC Editor, 1987. RFC 1035. (Request for Comments, 1035).

NASERIPARSA, M.; KASHANI, M. M. R. Combination of pca with smote resampling to boost the prediction rate in lung cancer dataset. *arXiv preprint arXiv:1403.1949*, 2014.

NIC.BR. *Sobre o NIC.br*. 2021. https://www.nic.br/sobre/. Accessed on October 13, 2021.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

RASCHKA, S. Model evaluation, model selection, and algorithm selection in machine learning. *CoRR*, abs/1811.12808, 2018. Disponível em: http://arxiv.org/abs/1811.12808.

REGISTRO.BR. *Domínios .br registrados até o momento*. 2024. https://registro.br/estatisticas.html. Accessed on February 14, 2024.

RODRIGUEZ, M. Z.; COMIN, C. H.; CASANOVA, D.; BRUNO, O. M.; AMANCIO, D. R.; COSTA, L. d. F.; RODRIGUES, F. A. Clustering algorithms: A comparative approach. *PloS one*, Public Library of Science San Francisco, CA USA, v. 14, n. 1, p. e0210236, 2019.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, Elsevier, v. 20, p. 53–65, 1987.

SAFAVIAN, S. R.; LANDGREBE, D. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, IEEE, v. 21, n. 3, p. 660–674, 1991.

Salih Hasan, B. M.; ABDULAZEEZ, A. M. A Review of Principal Component Analysis Algorithm for Dimensionality Reduction. *Journal of Soft Computing and Data Mining*, v. 02, n. 01, p. 20–30, 2021.

SIKOS, L. F. Packet analysis for network forensics: A comprehensive survey. *Forensic Science International: Digital Investigation*, v. 32, p. 200892, 2020. ISSN 2666-2817. Disponível em: https://www.sciencedirect.com/science/article/pii/S1742287619302002.

SILVEIRA, M. R.; da Silva, L. M.; CANSIAN, A. M.; KOBAYASHI, H. K. Detection of Newly Registered Malicious Domains through Passive DNS. p. 3360–3369, 2022. Disponível em: https://ieeexplore.ieee.org/document/9671348.

SILVEIRA, M. R.; SILVA, L. M. da; CANSIAN, A. M.; KOBAYASHI, H. K. Xgboost applied to identify malicious domains using passive dns. In: IEEE. *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*. [S.l.], 2020. p. 1–4.

STAT, D. N. *The overall number of domains registered, by TLD type*. 2024. https://domainnamestat.com/statistics/overview. Accessed on February 14, 2024.

STEVENS, W. R. *TCP/IP illustrated vol. I: the protocols*. [S.l.]: Pearson Education India, 1994.

SYMANTEC. Internet Security Threat Report VOLUME 21, February 2019. *Network Security*, v. 21, n. February, p. 61, 2019. ISSN 13534858. Disponível em: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/istr-24-cyber-security-threat-landscape.

TORABI, S.; BOUKHTOUTA, A.; ASSI, C.; DEBBABI, M. Detecting internet abuse by analyzing passive DNS traffic: A survey of implemented systems. *IEEE Communications Surveys and Tutorials*, IEEE, v. 20, n. 4, p. 3389–3415, 2018. ISSN 1553877X.

VAZ, A. L. *Como lidar com dados desbalanceados em problemas de classificação*. Medium, 2019. Disponível em: https://medium.com/data-hackers/como-lidar-com-dados-desbalanceados-em-problemas-de-classifica%C3%A7%C3%A3o-17c4d4357ef9.

VERGELIS, M.; SHCHERBAKOVA, T.; SIDORINA, T. *Spam and phishing in Q1 2019*. 2021. https://securelist.com/spam-and-phishing-in-q1-2019/90795/. Accessed on October 15, 2021.

VISSERS, T.; SPOOREN, J.; AGTEN, P.; JUMPERTZ, D.; JANSSEN, P.; Van Wesemael, M.; PIESSENS, F.; JOOSEN, W.; DESMET, L. Exploring the Ecosystem of Malicious Domain Registrations in the.eu TLD. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 10453 LNCS, p. 472–493, 2017. ISSN 16113349.

WANG, Q.; LI, L.; JIANG, B.; LU, Z.; LIU, J.; JIAN, S. Malicious Domain Detection Based on K-means and SMOTE. In: . Springer, Cham, 2020. p. 468–481. ISBN 978-3-030-50416-8. Disponível em: http://link.springer.com/10.1007/978-3-030-50417-5
.

WATKINS, L.; BECK, S.; ZOOK, J.; BUCZAK, A.; CHAVIS, J.; ROBINSON, W. H.; MORALES, J. A.; MISHRA, S. Using semi-supervised machine learning to address the big data problem in dns networks. In: IEEE. *2017 IEEE 7th Annual*

Computing and Communication Workshop and Conference (CCWC). [S.l.], 2017. p. 1–6.

WEBER, M.; WANG, J.; ZHOU, Y. Unsupervised clustering for identification of malicious domain campaigns. In: *Proceedings of the First Workshop on Radical and Experiential Security*. [S.l.: s.n.], 2018. p. 33–39.

WEIMER, F. *Passive DNS Replication*. 2005.

WEISS, E. A. Biographies: Eloge: Arthur lee samuel (1901-90). *IEEE Annals of the History of Computing*, IEEE, v. 14, n. 3, p. 55–69, 1992.

WHITE, T. *Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale*. O'Reilly Media, 2015. ISBN 9781491901700. Disponível em: https://books.google.com.br/books?id=6BmkBwAAQBAJ.

WULLINK, M.; MOURA, G. C.; MÜLLER, M.; HESSELMAN, C. Entrada: A high-performance network traffic data streaming warehouse. In: IEEE. *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. [S.l.], 2016. p. 913–918.

ZAKI, M. J.; JR, W. M. *Data mining and machine learning: Fundamental concepts and algorithms*. [S.l.]: Cambridge University Press, 2020.

ZHAUNIAROVICH, Y.; KHALIL, I.; YU, T.; DACIER, M. A survey on malicious domains detection through dns data analysis. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 51, n. 4, jul 2018. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/3191329.

ZUECH, R.; HANCOCK, J.; KHOSHGOFTAAR, T. M. Detecting web attacks using random undersampling and ensemble learners. *Journal of Big Data*, SpringerOpen, v. 8, n. 1, p. 1–20, 2021.