UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Análise exploratória de dados de SOC (Security Operation Center) utilizando KNN e árvore de decisão para gerar insights

Silmara Cristina Sol Falcão

Trabalho de Conclusão de Curso
MBA em Inteligência Artificial e Big Data

ICMC USP
SÃO CARLOS

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

S684a

Sol Falcão, Silmara Cristina
Análise exploratória de dados de SOC (Security Operation Center) utilizando KNN e árvore de decisão para gerar insights / Silmara Cristina Sol Falcão; orientador Kelton Augusto Pontara Costa. -- São Carlos, 2024.
52 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. K-Nearest Neighbors (KNN) . 2. Árvore de decisão. 3. Security Operation Center . I. Pontara Costa, Kelton Augusto , orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Análise exploratória de dados de SOC (Security Operation Center) utilizando KNN e árvore de decisão para gerar insights

Silmara Cristina Sol Falcão

USP - São Carlos
2024

Silmara Cristina Sol Falcão

Análise exploratória de dados de SOC (Security Operation Center) utilizando KNN e árvore de decisão para gerar insights

Trabalho de conclusão de curso apresentado ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador:
Prof. Kelton Augusto Pontara da Costa

USP - São Carlos

2024

DEDICATÓRIA

Ao meu esposo e filha pela compreensão, carinho e apoio incansável.

AGRADECIMENTOS

Meus agradecimentos especiais

Ao Prof. Kelton Pontara, pela orientação, paciência e ensinsamentos contribuindo para o meu crescimento científico e para a realização deste trabalho.

A Profa. Solange Rezende, pela tolerância, sabedoria e persistência. Obrigada por não desistir de mim e por me conduzir na escolha no tema num momento que parecia improvável essa conclusão.

A Profa. Roseli Romero, pelos ensinamentos e disponibilidade em complementar o curso sobre machine learning, foi fundamental para a compreensão do tema e para ampliar o horizonte neste tema tão atual.

A minha psicóloga Dra. Fabiana Garcia Gomes, por me dar suporte e cuidar da minha saúde mental, foram períodos de muita frustração, mas ela contribuiu para eu manter o foco.

Ao meu amado esposo por sempre acreditar em mim, me apoiar e incentivar incondicionalmente, mesmo que isso reflita na minha ausência. Obrigada por não desistir e por estar presente na minha vida, sua confiança e apoio me permitiu seguir e concluir.

A minha amada filha, por compreender meus momentos de ausência e por toda contribuição e suporte que me deu para que eu pudesse me dedicar inteiramente aos estudos.

Aos docentes que lecionaram esse MBA e ampliaram meu conhecimento. Muitíssima obrigada.

EPÍGRAFE

Lágrimas, angústia, medo e frustração converteram-se em motivação, inspiração e impulsão para a conclusão deste MBA.

O feito é melhor que o perfeito não feito!

Profª Solange Rezende (2024)

RESUMO

FALCÃO, S. C. S. **Análise exploratória de dados de SOC (Security Operation Center) utilizando KNN e árvore de decisão para gerar insights**. 2024. 52 f. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Este trabalho consiste na análise exploratória dos algoritmos K-Nearest Neighbors (KNN**)** e da árvore de decisão aplicadas ao dataset UNSW-NB15 criado com o objetivo de simular atividades normais e comportamentos de ataque contemporâneos em tráfego de rede, visando representar o ambiente de um Security Operations Center (SOC). O objetivo deste estudo é perceber através da investigação do dataset a consistência dos dados e implementação dos modelos com o propósito de identificar o mais adequado para a classificação de falsos positivos e falsos negativos, informações relevantes para a operação de um SOC. Para a realização desta análise foram utilizados bibliotecas específicas do Python com foco em machine learning, tais como Pandas, Matsploit, Ski-Learn, o código foi desenvolvido na plataforma do collab do Google, que permitiu a inserção textos explicativos combinando com os códigos para facilitar a compreensão do processo. Foram selecionados algumas features e os datasets separados em treinamento e teste para que pudesse analisar a performance do modelo e foram aplicados a dois datasets: um conjunto de treinamento, UNSW_NB15_training-set.csv, com 175.341 registros, e um conjunto de teste, UNSW_NB15_testing-set.csv, com 82.332 registros, divididos em 70/30 para training e testing, respectivamente. Para explorar os modelos foram realizados experimentos com variação de parâmetros em ambos os modelos. A performance dos modelos foram satisfatórias no modo geral.

Palavras-chave: inteligência artificial; security operation center; KNN; árvore de decisão.r

ABSTRACT

FALCÃO, S. C. S. **Análise exploratória de dados de SOC (Security Operation Center) utilizando KNN e árvore de decisão para gerar insights**. 2024. 52 f. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

This work consists of the exploratory analysis of the K-Nearest Neighbors (KNN) algorithms and the decision tree applied to the UNSW-NB15 dataset created with the objective of simulating normal activities and contemporary attack behaviors in network traffic, aiming to represent the environment of a Security Operations Center (SOC). The objective of this study is to understand, through investigation of the dataset, the consistency of the data and implementation of the models with the purpose of identifying the most appropriate one for the classification of false positives and false negatives, information relevant to the operation of a SOC. To carry out this analysis, specific Python libraries focused on machine learning were used, such as Pandas, Matsploit, Ski-Learn, the code was developed on the Google collab platform, which allowed the insertion of explanatory texts combining with the codes to facilitate understanding the process. Some features were selected and the datasets separated into training and testing so that the model's performance could be analyzed and applied to two datasets: a training set, UNSW_NB15_training-set.csv, with 175,341 records, and a test set, UNSW_NB15_testing- set.csv, with 82,332 records, divided 70/30 for training and testing, respectively. To explore the models, experiments were carried out with parameter variation in both models. The performance of the models was generally satisfactory.

Keywords: artificial inteligence; security operation center; KNN; decision tree.

SUMÁRIO

1. INTRODUÇÃO

O objetivo principal da segurança da informação é assegurar aos dados, sistemas e recursos de uma organização ou indivíduo proteção contra acessos não autorizados, uso indevido, alteração não autorizada, destruição acidental ou intencional, e garantir a disponibilidade dos recursos.

A abordagem contextual para segurança da informação refere-se às práticas, políticas e tecnologias projetadas que visam contribuir para a proteção da confidencialidade, da integridade e da disponibilidade das informações, os pilares da segurança da informação, que representam respectivamente a maneira de garantir que apenas pessoas autorizadas possam ter acessos as informações, implantando controles de acessos e medidas de proteção; assegurar que a informação permaneça precisa, completa e íntegra, sem a alteração não autorizada e garantir que as informações estejam disponíveis quando necessárias.

Levando em consideração o ambiente digital contemporâneo, o cenário dinâmico pós-pandemia e o crescimento de inúmeras ameaças e desafios intrínsecos a esses ambientes, a segurança da informação representa uma preocupação extremamente relevante para as organizações exigindo atenção constante as inovações tecnológicas, pois a segurança da informação é essencial para garantir a privacidade e proteção dos dados, prevenir e mitigar ameaças cibernéticas, garantir a continuidade dos negócios, manter relação de confiança entre clientes quando eles acreditam que suas informações estão seguras, proteger a propriedade intelectual permitindo a inovação segura, entre outras.

A segurança da informação não limita-se à prevenção, mas também inclui a capacidade de adaptação e recuperação de incidentes cibernéticos, e a utilização de tecnologias como inteligência artificial (IA) e machine learning (ML) para detecção de ameaças podem ser diferenciais significativos. Essas tecnologias contribuem para a identificação padrões e ameaças emergentes de maneira rápida e eficiente. (DE OLIVEIRA, 2023)

> A inteligência artificial refere-se à capacidade de um sistema ou máquina realizar tarefas que normalmente exigiriam inteligência humana, como aprendizado, raciocínio, resolução de problemas, compreensão de linguagem natural, percepção visual, reconhecimento de padrões e tomada de decisões. (RUSSEL e NORVIG, 2003).

De acordo com Deep (2018), constantemente surgem novos ataques e a descoberta de novas vulnerabilidades, esses fatores exigem que segurança seja combinada com métodos mais proativos para se manter à frente e agir com eficiência e eficácia. Já em Karatas (2018), os Centros de Operações de Segurança (SOC, do inglês, *Security Operation Centers*) atuam nas ações de análise, e o analista tem a responsabilidade de avaliar esses dados e seus comportamentos, visando buscar comportamentos suspeitos. Milhares de registros revisados, precisam de filtros e correlação de dados, destacando descobertas importantes para analistas. Nesse contexto uso da Inteligência Artificial (IA) em problemas deste tipo alcançou resultados promissores.

Segundo Valente (2022) os sistemas de segurança baseados em AI / ML, demonstraram a capacidade de criar automaticamente perfis de comportamento, orquestrar e automatizar respostas a incidentes, além de detectar num elevado quantitativo de dados atividades anômalas, detecção de malware em tempo real, inclusive interagindo com outras ferramentas de segurança permitindo reduzir os incidentes, resultando em menos eventos para o analista atuar e reduzindo significativamente os falsos positivos.

Conforme explana XIN et al, 2017, ainda não há um método mais eficaz de detecção de intrusão estabelecido, e para a análise do seu artigo foram testados os algoritmos de ML (KNN, SVM, Decision Tree e Bayes), e em Deep Learning (DBM, CNN e LSTM), e eles constataram que a abordagem para implementar um sistema de detecção de intrusão tem características próprias, do ponto de vista das vantagens e desvantagens, comprometendo a escolha eficaz.

De supervisionados e não supervisionados a híbridos e DL, novas técnicas estão sendo continuamente desenvolvidas e testadas no setor de cibersegurança (SOUSA, 2022 apud Apruzzese). No sistema proposto por Sousa (2022), foi possível enriquecer os alertas recebidos através de ML, após seguir as etapas da possibilidade de riscos de ameaças, contribuindo com a redução do tempo de análise do profissional do SOC e aumentando sua eficiência de um modo geral.

Este trabalho tem por propósito contribuir através da análise exploratória de dados com foco na geração de insights, investigar e compreender os dados coletados visando auxiliar na identificação de padrões e comportamentos com o intuito de detectar e classificar anomalias em dados de SOC a fim de indicar ameaças à segurança e agilizar as análises de detecção de

eventos no SOC visando minimizar os alertas para falsos positivos e falsos negativos otimizando a operação e a atuação de analistas em incidentes efetivos.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Segurança da Informação e SOC

Segurança da informação é o conjunto de práticas, políticas e procedimentos que propõe proteger a confidencialidade, a integridade e a disponibilidade das informações de uma organização. Atingir objetivo envolve proteção contra ameaças internas e externas, tais como ataques cibernéticos, roubo, perda ou danificação de informações e o acesso não autorizado.

A criptografia neste cenário é considerada uma das técnicas utilizadas para atribuir segurança e visa garantir a confidencialidade e integridade das informações. A crescente ameaça de ataques cibernéticos torna a segurança da informação uma preocupação das empresas de todos os setores (CLARKE; KNAKE, 2015).

Segundo Clarke e Knake (2015), há três pilares para a Segurança da Informação: confidencialidade, integridade e disponibilidade. Esses pilares são aceitos como uma estrutura útil para a construção de um programa de segurança cibernética eficaz e contribuem com as organizações na identificação e avaliação dos riscos de segurança.

Confidencialidade

A confidencialidade é um dos principais pilares da segurança da informação e refere-se à proteção de dados contra acessos não autorizados. Seu objetivo é garantir que informações sensíveis estejam acessíveis apenas a indivíduos, processos ou sistemas autorizados, evitando vazamentos e exposição a terceiros. Essa prática é essencial em ambientes onde o sigilo de dados é crucial, como em instituições financeiras, de saúde, governamentais e em setores empresariais de alta segurança (Stallings, 2017; Pfleeger & Pfleeger, 2006).

É a garantia de que a informação estará protegida contra divulgação ou acesso não autorizado, para isso é importante implementar medidas como criptografia, controle de acesso, firewalls e outras tecnologias de segurança cibernética.

Com o intuito de assegurar a confidencialidade, várias ferramentas e métodos são implementados. A criptografia, por exemplo, codifica os dados de forma que apenas usuários com a chave de decriptação correta possam acessá-los, protegendo informações mesmo que

sejam interceptadas. Já o controle de acesso limita quem pode visualizar ou modificar os dados, enquanto mecanismos de autenticação, como senhas, tokens e biometria, garantem que somente usuários devidamente identificados possam acessar informações protegidas (Anderson, 2008; Tipton & Krause, 2008).

Integridade

A integridade visa garantir que as informações permaneçam completas, corretas e protegidas contra alterações não autorizadas, assegurando a permanência inalterada dados desde a criação até o uso, evitando manipulações acidentais ou maliciosas que poderiam comprometer a precisão e confiabilidade. Desta maneira, a integridade é essencial para a manutenção de dados confiáveis e consequetemente para a tomada de decisões seguras.

Diversas técnicas são utilizadas. Checksums, por exemplo, são valores calculados a partir dos dados que permitem a verificação de sua integridade ao longo do tempo; qualquer modificação no conteúdo altera o checksum, indicando uma possível violação. Hashing, utiliza funções criptográficas gerando uma representação única para cada conjunto de dados; mudanças nos dados originais resultam em um hash diferente, alertando sobre possíveis alterações. Assinaturas digitais, permitem autenticar a origem e integridade dos dados, provendo ainda mais segurança durante a transmissão de informações confidenciais (Anderson, 2008; Whitman & Mattord, 2017).

Essas práticas de integridade são aplicadas em contextos desde armazenamento de dados (dados em repouso) até sua transmissão (dados em uso ou movimento), assegurando a confiabilidade das informações e protegendo-as contra ataques que poderiam comprometer sistemas e processos organizacionais. A manutenção da integridade é, portanto, indispensável para evitar que dados críticos sejam corrompidos, preservando sua acurácia e credibilidade ao longo de seu ciclo de vida.

Disponibilidade

A disponibilidade da informação é um dos pilares fundamentais da segurança da informação, ao lado da confidencialidade e da integridade. Esse princípio assegura que os dados e os sistemas estejam acessíveis quando necessários, sem interrupções causadas por falhas técnicas ou ataques cibernéticos. De acordo com Stallings (2019), a indisponibilidade pode

impactar severamente as operações de uma organização, tornando indispensável a implementação de medidas preventivas e reativas para mitigar riscos.

Para garantir a disponibilidade, é essencial adotar estratégias que mantenham a continuidade dos serviços e minimizem a interrupção de atividades críticas. A redundância dos sistemas permitem que, em caso de falha de um componente ou sistema, outro imediatamente assuma sua função, ou seja, reestabeleça o sistema, garantindo a continuidade operacional.

Realizar cópias de segurança periódicas de dados e sistemas, armazenando-as em locais seguros e separados do ambiente principal, permite a recuperação rápida e eficiente em situações de falha ou comprometimento dos sistemas originais, sendo necessário a realização de testes periódicos a fim de assegurar a eficiência da operação.

Implementar planos de ação que incluam detecção precoce de ameaças, contenção de ataques e recuperação de sistemas comprometidos. Ahmed e Hossain (2020) destacam que ações rápidas e coordenadas são cruciais para reduzir o impacto de incidentes cibernéticos na disponibilidade. Essas práticas, combinadas com políticas de segurança bem definidas e equipes capacitadas, são essenciais para a construção de um ambiente digital resiliente. Além disso, a adesão a normas e padrões reconhecidos internacionalmente, como a ISO 27001, fortalecendo a postura de segurança da organização.

A indisponibilidade de dados ou sistemas pode gerar prejuízos financeiros, perdas de produtividade, danos à reputação e, em alguns casos, riscos à segurança de pessoas ou infraestrutura crítica. Alguns setores como saúde, finanças e governo são especialmente dependentes de alta disponibilidade para garantir a prestação de serviços essenciais.

Ao priorizar a disponibilidade, as organizações demonstram compromisso com a continuidade de negócios, proteção de seus ativos e atendimento às expectativas de seus clientes e parceiros.

SOC

Um Security Operations Center (SOC) é uma estrutura centralizada responsável por monitorar, detectar, analisar e responder a incidentes de segurança cibernética em uma organização. O SOC atua como o núcleo de operações de segurança, integrando processos,

pessoas e tecnologias para proteger os ativos digitais da organização contra ameaças e vulnerabilidades.

De acordo com Bodeau e Graubart (2016), o SOC desempenha funções críticas como o monitoramento contínuo de eventos, análise de logs, investigação de incidentes e orquestração de respostas. Ele utiliza ferramentas avançadas, como SIEM (Security Information and Event Management), SOAR (Security Orchestration, Automation and Response), sistemas de detecção de intrusão (IDS), sistemas de prevenção de intrusão (IPS), firewalls e outras tecnologias de segurança.

Segundo Stallings, W. (2019), o SOC é uma peça fundamental para a segurança cibernética moderna, oferecendo proteção proativa contra ameaças emergentes e permitindo que as organizações mantenham seus sistemas resilientes e seguros. Ele não apenas responde a incidentes, mas também melhora a capacidade organizacional de prevenir ataques futuros, sendo um elemento essencial em um ambiente digital cada vez mais complexo.

Principais Funções do SOC

De acordo com Ahmed & Hossain (2020), para uma organização, um SOC bem estruturado realiza atividades essenciais para a segurança dos dados e sistemas, tais como:

- **Monitoramento contínuo**: com operação 24 horas por dia, 7 dias por semana, supervisiona atividades em redes e sistemas visando detectar comportamentos anômalos e sinais de possíveis ataques cibernéticos.
- **Gestão de incidentes**: incluindo a identificação, contenção, erradicação e recuperação de ameaças, garantindo que os incidentes sejam tratados de forma eficiente e que os impactos sejam minimizados.
- **Análise de logs**: coleta e avalia dados de eventos e registros de atividades em sistemas e dispositivos para identificar vulnerabilidades e padrões de ataque.
- **Inteligência de ameaças**: através de informações atualizadas sobre tendências e padrões de ameaças com o propósito antecipar possíveis ataques e melhorar as defesas.
- **Automação e orquestração**: ferramentas que podem acelerar respostas a incidentes e reduzir o tempo de mitigação.

Benefícios de um SOC

- Quando bem implementado um SOC oferece inúmeros benefícios para as organizações:
- Resposta ágil: redução do tempo de detecção e mitigação de ataques, limitando o impacto sobre os negócios.
- Melhorar a visibilidade: proporcionando uma visão abrangente sobre o ambiente de ameaças, permitindo uma gestão de riscos mais eficiente.
- Conformidade regulatória: auxiliando as organizações a cumprir regulamentações e manter o compliance.
- Redução de riscos: proteção aos ativos críticos, fortalecendo a resiliência contra ataques e reduzindo as consequências financeiras e reputacionais de incidentes cibernéticos.

2.2. Análise exploratória de dados (AED)

A Análise Exploratória de Dados (AED) é uma abordagem inicial no processo de análise de dados, com foco na investigação e no entendimento das características principais de um conjunto de dados. Essa abordagem combina técnicas estatísticas e visualizações gráficas para identificar padrões, tendências, anomalias e relações importantes entre variáveis, sendo uma etapa essencial para a preparação de dados e construção de modelos preditivos ou explicativos.

John Tukey (1977), pioneiro no campo da análise exploratória, destacou que essa etapa é fundamental para "formular questões adequadas, verificar suposições e orientar escolhas de modelos e métodos estatísticos". Em essência, a AED permite que os analistas desenvolvam hipóteses sobre os dados, sem assumir premissas rígidas inicialmente.

Os objetivos principais de uma análise exploratória consistem no entendimento dos dados, compreendendo sua estrutura, dimensão, propriedades gerais; na detecção de regularidade ou comportamento consistentes nas variáveis; assim como na detecção de anomalias.

A AED utiliza uma combinação de métodos estatísticos descritivos e gráficos, incluindo medidas de tendência central e dispersão, visualizações tais como: histogramas, boxplots, entre outros e análise de correlação.

Ferramentas de software, como Python (bibliotecas como Pandas, Matplotlib e Seaborn), R e softwares estatísticos dedicados, são amplamente utilizadas para realizar AED. Essa análise exploratória contribui para garantir a qualidade e a validade dos dados antes de realizar análises mais complexas. Segundo Montgomery, Peck e Vining (2021), "uma AED bem conduzida reduz significativamente o risco de erros em inferências estatísticas ou decisões baseadas em dados". Além disso, ela facilita a comunicação dos achados iniciais para diferentes públicos, tornando os insights acessíveis e úteis para tomada de decisões.

2.3. Algoritmo KNN

O K-Nearest Neighbors (KNN**)** é um dos algoritmos mais simples e intuitivos em aprendizado de máquina, baseada na proximidade de instâncias em um espaço de características, sendo utilizado principalmente para tarefas de classificação e regressão. Fukunaga e Narendra (1975), em seu artigo seminal *"A Branch and Bound Algorithm for Computing K-Nearest Neighbors"*, abordaram um método eficiente para encontrar os vizinhos mais próximos de uma dada amostra, uma tarefa central para a execução do KNN.

O KNN é um algoritmo de aprendizagem supervisionada que classifica novos dados com base nos “K” vizinhos mais próximos no conjunto de treinamento. Esse método é particularmente conhecido por sua simplicidade e eficácia em várias tarefas de classificação, como reconhecimento de padrões e detecção de anomalias. A principal ideia é classificar ou prever o valor de uma amostra desconhecida com base nos k exemplos mais próximos no espaço de características. A proximidade entre os pontos é geralmente medida utilizando métricas como distância euclidiana, Manhattan ou outras métricas específicas. (Montgomery et. al (2021).

Esse algoritmo funciona segundo o princípio de que instâncias semelhantes tendem a estar próximas no espaço de características, o que é conhecido como o princípio da "similaridade" ou "proximidade". Para classificar uma nova instância, o KNN (Figura 1) calcula a distância entre essa instância e todas as outras no conjunto de dados, selecionando as “K” instâncias mais próximas e usando-as para prever a classe ou valor da nova instância. Entre as métricas de distância mais usadas estão a distância Euclidiana, a distância de Manhattan e a distância de Minkowski Parâmetro "K" O valor de “K” é um parâmetro crucial que define o número de vizinhos a serem considerados. Valores pequenos de “K” tornam o modelo sensível ao ruído, enquanto valores muito grandes podem suavizar demais a fronteira de decisão, prejudicando a precisão. Geralmente, valores intermediários são recomendados, e a escolha ideal de “K” depende da estrutura dos dados e pode ser obtida por validação cruzada.

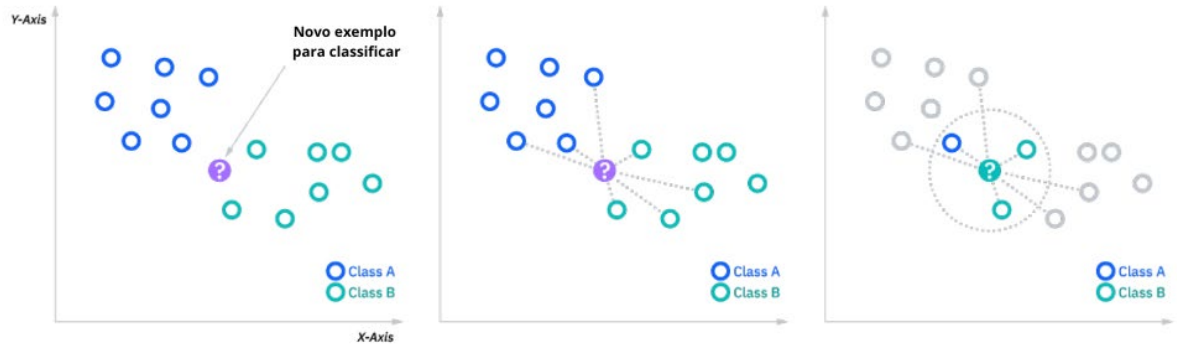

Figura 1. Representação do algoritmo de classificação KNN

Fonte: https://www.ibm.com

Dentre as vantagens do KNN é o fato de ser um algoritmo não-paramétrico, o que significa que não faz suposições sobre a distribuição dos dados, sendo versátil para diferentes tipos de dados, sendo uma boa escolha para tarefas exploratórias iniciais em machine learning. No entanto, o KNN tende a ser computacionalmente intensivo em grandes volumes de dados, pois precisa calcular distâncias para cada ponto de dado e é sensível a dados de alta dimensionalidade, exigindo que técnicas de redução da dimensionalidade sejam executadas para melhorar a eficácia.

No contexto de detecção de anomalias e segurança, o KNN é útil em SOCs, pois permite classificar eventos e identificar comportamentos anômalos com base em padrões de proximidade com eventos anteriores classificados, essa identificação nas detecções de atividades suspeitas em tempo real, permite uma aplicação crescente em segurança de redes.

2.4. Árvore de Decisão

As árvores de decisão são modelos de aprendizado supervisionado amplamente utilizados em tarefas de classificação e regressão. Elas estruturam o processo de tomada de decisão em uma sequência hierárquica de testes em atributos, o que as torna intuitivas e de fácil interpretação. Breiman et al. (1984) definem as árvores de decisão como "estruturas que particionam o espaço de entrada em regiões distintas por meio de divisões sucessivas baseadas em valores dos atributos".

Estrutura e o funcionamento do modelo de uma árvore de decisão (Figuras 2 e 3) consiste em: nós internos que representam condições ou testes em atributos; os ramos cuja função é indicar os resultados de cada teste e as folhas que correspondem às decisões finais (classes ou valores preditos).

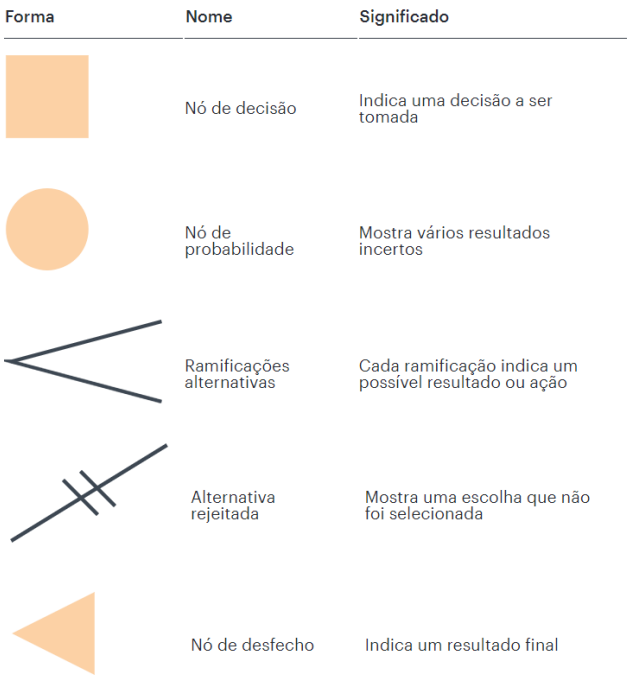

Figura 2. Descrição das formas do fluxograma de uma árvore de decisão

Fonte: https://www.plantareducacao.com.br/arvore-de-decisao/

O processo de construção de uma árvore envolve a escolha do atributo que melhor particiona os dados em cada nó. Essa escolha é guiada por métricas como a entropia e o ganho de informação (Quinlan, 1986), ou o índice Gini (Breiman et al., 1984), dependendo do algoritmo.

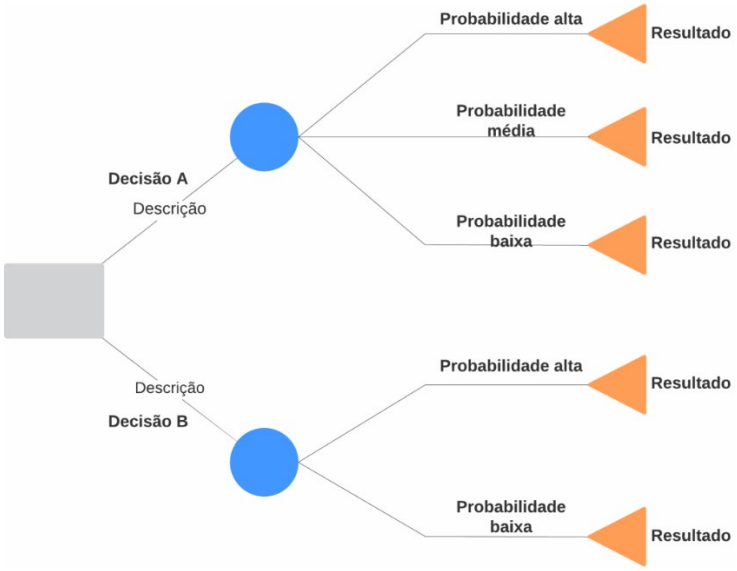

Figura 3. Representação gráfica de uma árvore de decisão

Fonte: https://www.plantareducacao.com.br/arvore-de-decisao/

As árvores de decisão oferecem diversas vantagens:

- **Interpretação intuitiva:** "Árvores de decisão são simples de interpretar, mesmo para usuários não especializados" (Quinlan, 1993).
- **Capacidade de modelar relações não lineare**s: podem capturar interações complexas entre atributos sem a necessidade de transformações prévias.
- **Robustez a dados categóricos e contínuos:** flexíveis em relação aos tipos de dados.

No entanto, as árvores de decisão apresentam alguns desafios:

- **Sobreajuste:** Árvores muito profundas podem se ajustar excessivamente aos dados de treinamento, perdendo capacidade de generalização.
- **Instabilidade:** Pequenas mudanças nos dados podem levar a alterações significativas na estrutura da árvore.

2.5. Métricas de análise

Acurácia

Métrica referente a proporção de previsões corretas em relação ao total de exemplos. Utilizando a fórmula:

$$Acurácia = \frac{Verdadeiros\ Positivos\ (TP) + Verdadeiros\ Negativos\ (VN)}{Total}$$

Precisão

Mede a proporção dos exemplos classificados como positivos, quantos realmente são positivos. Utilizando a fórmula:

$$Precisão = \frac{Verdadeiros\ Positivos\ (TP)}{Verdadeiros\ Positivos\ (TP) + FalsosPositivos\ (FP)}$$

Recall

Qual a porcentagem dos dados classificados como positivos em comparação aos reais positivos existentes na amostra, ou seja, quantos foram classificados corretamente. Utilizando a fórmula:

$$Recall = \frac{Verdadeiros\ Positivos\ (TP)}{Verdadeiros\ Positivos\ (TP) + Falsos\ Negativos\ (FN)}$$

F1-Score

Média harmônica entre precisão e recall. Normalmente usado quando ocorre um trade-off (dois aspectos ou objetivos conflitantes, ou seja, melhorar um implicará piorar o outro). A ideia é ter número único com o propósito de determinar a qualidade geral do modelo. Utilizando a fórmula:

$$F1 = \frac{2 * precisão * recall}{precisão + recall}$$

3. METODOLOGIA

3.1.Coleta de dados

O dataset UNSW-NB15 foi criado com o objetivo de simular atividades normais e comportamentos de ataque contemporâneos em tráfego de rede, gerado pelo IXIA PerfectStorm no laboratório Cyber Range do Australian Centre for Cyber Security (ACCS). Para capturar o tráfego bruto de rede (em arquivos Pcap) de aproximadamente 100 GB, foi utilizada a ferramenta Tcpdump. O dataset é composto por nove tipos de ataques, incluindo Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode e Worms.

As ferramentas Argus e Bro-IDS foram utilizadas junto a doze algoritmos para gerar um conjunto de 49 atributos (ou features) rotulados, que estão descritos no arquivo UNSW-NB15_features.csv. Esses atributos abrangem características de rede essenciais para análises de segurança e detecção de ameaças.

Para facilitar a aplicação em modelos de machine learning, o dataset está dividido em um conjunto de treinamento, UNSW_NB15_training-set.csv, com 175.341 registros, e um conjunto de teste, UNSW_NB15_testing-set.csv, com 82.332 registros, ambos contendo tanto amostras de tráfego normal quanto de diferentes tipos de ataque. Os dados são rotulados, permitindo o uso tanto em tarefas de classificação supervisionada quanto em aprendizado não supervisionado.

Esse dataset é amplamente utilizado para testes de algoritmos de classificação e detecção de anomalias em ambientes de segurança cibernética, sendo um recurso valioso para pesquisa e desenvolvimento de soluções em Security Operations Centers (SOC).

O UNSW-NB15 foi projetado para oferecer um cenário mais representativo das ameaças modernas em redes de computadores. Ele é amplamente usado em desenvolvimento e avaliação de sistemas de detecção de instrusões; estudos de aprendizado de máquina para segurança cibernética, como análise de anomalias e classificação de tráfego de rede e comparação de desempenho entre algoritmo de detecção.

As vantagens deste dataset são a representatividade, pois reflete ameaças modernas com dados gerados em ambientes realistas; diversidade de ataques; uma combinação de características que capturam diferentes dimensões do tráfego de rede e a disponibilidade, os dados estão acessíveis. Entretanto há algumas limitações, algumas categorias possuem menos

registros, que pode comprometer o desempenho do algoritmo e o cenário de segurança evolui exponencialmente exigindo uma atualização constante.

3.2. Processamento e limpeza dos dados

Essa etapa foi dispensada, pois o dataset selecionado já apresenta a separação em treinamento e teste, preparados para a análise acadêmica.

3.3. Implementação do KNN e da Árvore de decisão

3.3.1. Implementando KNN

Para implementar o KNN serão utilizados os datasets UNSW_NB15_training-set.csv e o UNSW_NB15_testing-set.csv, seguindo as seguintes etapas:

Dataset UNSW_NB15_training-set.csv

Etapa 1. Importando o dataset (neste cenário a partir do UNSW_NB15_training-set.csv) (Figura 4)

```
import pandas as pd
import numpy as np
ciber = pd.read_csv('/content/sample_data/UNSW_NB15_training-set.csv')
ciber
```

Figura 4. Importando o dataset para o ambiente de desenvolvimento

Etapa 2. Visualizando o dataset e confirmando a quantidade dos dados (Figura 5)

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	...	1	1	0	0	
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	...	1	2	0	0	
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	...	1	3	0	0	
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	...	1	3	1	1	
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	...	1	40	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
175336	175337	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	13	24	0	0	
175337	175338	0.505762	tcp	-	FIN	10	8	620	354	33.612649	...	1	2	0	0	
175338	175339	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	3	13	0	0	
175339	175340	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	14	30	0	0	
175340	175341	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	16	30	0	0	

175341 rows × 45 columns

Figura 5. Visualização do dataset

Etapa 3. Importando as bibliotecas necessárias do Python para implementar o KNN (Figura 6)

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

Figura 6. Importação das bibliotecas Python

Etapa 4. Selecionando as features (Figura 7) que serão analisadas pelo algoritmo e qual será o target para identificar a relevância do modelo do classificador, neste caso, a coluna label (rótulo), considerando o dataset supervisionado, definindo o que é tráfego normal = 0 e um possível incidente = 1.

```
# Selecionando as features para análise do algoritmo (colunas)
features = ['dur', 'synack', 'ackdat', 'sbytes', 'dbytes', 'rate']
X = ciber[features]
y = ciber['label']
```

Figura 7. Definição das features para aplicar no modelo

Etapa 5. Dividir o dataset entre treinamento e teste (Figura 8)

```
# Dividindo do dataset para treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42) # Examplo: 70/30 split (training/test)
```

Figura 8. Separação do dataset em um conjunto de treinamento e teste

Etapa 6. Inicializando o classificador KNN e estabelecendo a distância dos pontos mais próximos (k) para realizar o agrupamento (Figura 9)

```
# Inicializando o classificador KNN e escolhendo a quantidade dos vizinhos, neste caso k=10, ou seja, baseará nos 10 mais próximos)
knn = KNeighborsClassifier(n_neighbors=5)  # Example: k=5
knn.fit(X_train, y_train)

# Estabelecendo a predição a partir da parte do dataset separado em teste (30%)
y_pred = knn.predict(X_test)

# Avaliando o modelo
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Figura 9. Aplicação do modelo KNN

Etapa 7. Print da avaliação do modelo (k = 5) (Figura 10)

	precision	recall	f1-score	support
0	0.92	0.89	0.90	16772
1	0.95	0.96	0.96	35831
accuracy			0.94	52603
macro avg	0.93	0.92	0.93	52603
weighted avg	0.94	0.94	0.94	52603

Figura 10. Avaliação do modelo

Refazendo a etapa 6 e alterando parâmetros de k, neste exemplo k = 10 (Figura 11)

```
# Inicializando o classificador KNN e escolhendo a quantidade dos vizinhos, neste caso k=10, ou seja, baseará nos 10 mais próximos)
knn = KNeighborsClassifier(n_neighbors=10)  # Example: k=10
knn.fit(X_train, y_train)

# Estabelecendo a predição a partir da parte do dataset separado em teste (30%)
y_pred = knn.predict(X_test)

# Avaliando o modelo
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Figura 11. Código para alteração do parâmetro k

Etapa 7. Print da avaliação do modelo (k = 10) (Figura 12)

	precision	recall	f1-score	support
0	0.91	0.90	0.90	16772
1	0.95	0.96	0.96	35831
accuracy			0.94	52603
macro avg	0.93	0.93	0.93	52603
weighted avg	0.94	0.94	0.94	52603

Figura 12. Visualização da avaliação do modelo após a alteração do parâmetro

Alterando parâmetros de features e target no dataset

Refazer as etapas 1, 2 e 3 (Figuras 4, 5 e 6, respectivamente) e abaixo as alterações consolidadas correspondentes as etapas 4, 5 e 6 (Figura 13); alterando a seleção das features e do target, neste caso optando pela categoria dos ataques.

```python
# Definindo as features e selecionando o target (neste cenário, optando pela categoria do ataque)
features = ciber[['dur', 'spkts', 'dpkts', 'sbytes', 'dbytes', 'sinpkt', 'dinpkt', 'smean', 'rate', 'sttl','synack', 'ackdat']] #Exemplo de colunas
target = ciber['attack_cat'] # Substitua pelo nome da sua coluna alvo

# Dividindo o dataset em treino e teste
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=42)

# Criando o classificador KNN
knn = KNeighborsClassifier(n_neighbors=5)  # Define o número de vizinhos

# Treinando o modelo
knn.fit(X_train, y_train)

# Realiza previsões no conjunto de teste
y_pred = knn.predict(X_test)

# Avaliando o modelo
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Figura 13. Alteração da feature e do target para nova aplicação do modelo KNN

Etapa 7. Print da avaliação do modelo com a alteração dos parâmetros features e target no dataset de training (k = 5) (Figura 14)

```
[[   87     2    94   353     6     0    56     0     0     0]
 [    1    60    97   351     5     0     6     5     2     2]
 [   29    20   834  2513    86    10    48    21     7     0]
 [   54    35   958  8399   261    18   244   157    23     9]
 [    8     3   114   583  3636     6   929    61    38     4]
 [    2     0    37   176    33 11804    16     3     3     1]
 [   61     2    28   270  1476     6 14848    62    18     1]
 [    2     3   134   672    32     8    24  2228    12     0]
 [    1     0    19    50   137    11    59    24    67     0]
 [    0     0     1    11     5     1     4     0     0    16]]
                precision    recall  f1-score   support

      Analysis       0.36      0.15      0.21       598
      Backdoor       0.48      0.11      0.18       529
           DoS       0.36      0.23      0.28      3568
      Exploits       0.63      0.83      0.71     10158
       Fuzzers       0.64      0.68      0.66      5382
       Generic       0.99      0.98      0.99     12075
        Normal       0.91      0.89      0.90     16772
Reconnaissance       0.87      0.72      0.79      3115
     Shellcode       0.39      0.18      0.25       368
         Worms       0.48      0.42      0.45        38

      accuracy                           0.80     52603
     macro avg       0.61      0.52      0.54     52603
  weighted avg       0.79      0.80      0.79     52603
```

Figura 14. Matriz e avaliação do modelo

Dataset UNSW_NB15_testing-set.csv

Refazendo a etapa 1 substituindo o dataset para UNSW_NB15_testing-set.csv.

```
import pandas as pd
import numpy as np
ciber = pd.read_csv('/content/sample_data/UNSW_NB15_testing-set.csv')
ciber
```

Figura 15. Carregando o dataset de testing para aplicar o modelo

Etapa 2. Visualizando o dataset e confirmando a quantidade dos dados (Figura 16)

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login
0	1	0.000011	udp	-	INT	2	0	496	0	90909.090200	...	1	2	0
1	2	0.000008	udp	-	INT	2	0	1762	0	125000.000300	...	1	2	0
2	3	0.000005	udp	-	INT	2	0	1068	0	200000.005100	...	1	3	0
3	4	0.000006	udp	-	INT	2	0	900	0	166666.660800	...	1	3	0
4	5	0.000010	udp	-	INT	2	0	2126	0	100000.002500	...	1	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
82327	82328	0.000005	udp	-	INT	2	0	104	0	200000.005100	...	1	2	0
82328	82329	1.106101	tcp	-	FIN	20	8	18062	354	24.410067	...	1	1	0
82329	82330	0.000000	arp	-	INT	1	0	46	0	0.000000	...	1	1	0
82330	82331	0.000000	arp	-	INT	1	0	46	0	0.000000	...	1	1	0
82331	82332	0.000009	udp	-	INT	2	0	104	0	111111.107200	...	1	1	0

82332 rows × 45 columns

Figura 16. Visualizando o dataset de testing e a quantidade de dados

Refazer as etapas 3 (Figura 6), 4 (Figura 7), 5 (Figura 8) e 6 (Figura 9) semelhante ao apresentado no tópico Dataset UNSW_NB15_training-set.csv.

Etapa 7. Print da avaliação do modelo (k = 5)

	precision	recall	f1-score	support
0	0.90	0.93	0.91	11147
1	0.94	0.92	0.93	13553
accuracy			0.92	24700
macro avg	0.92	0.92	0.92	24700
weighted avg	0.92	0.92	0.92	24700

Figura 17. Visualização da avaliação do modelo no dataset de testing

Etapa 7. Print da avaliação do modelo (k = 10) (Figura 18)

	precision	recall	f1-score	support
0	0.88	0.95	0.92	11147
1	0.96	0.90	0.93	13553
accuracy			0.92	24700
macro avg	0.92	0.93	0.92	24700
weighted avg	0.93	0.92	0.92	24700

Figura 18. Visualização da avaliação do modelo com a alteração do parâmetro k

Alterando parâmetros de features e target no dataset

Refazer as etapas 1, 2 e 3 (Figuras 4, 5 e 6, respectivamente) e abaixo as alterações consolidadas correspondentes as etapas 4, 5 e 6 (Figura 19); alterando a seleção das features e do target, neste caso optando pela categoria dos ataques.

```
# Definindo as features e selecionando o target (neste cenário, optando pela categoria do ataque)
features = ciber[['dur', 'spkts', 'dpkts', 'sbytes', 'dbytes', 'sinpkt', 'dinpkt', 'smean', 'rate', 'sttl','synack', 'ackdat']] #Exemplo de colunas
target = ciber['attack_cat'] # Substitua pelo nome da sua coluna alvo

# Dividindo o dataset em treino e teste
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=42)

# Criando o classificador KNN
knn = KNeighborsClassifier(n_neighbors=5)  # Define o número de vizinhos

# Treinando o modelo
knn.fit(X_train, y_train)

# Realiza previsões no conjunto de teste
y_pred = knn.predict(X_test)

# Avaliando o modelo
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Figura 19. Alteração das features e do target para implementar o modelo KNN

Etapa 7. Print da avaliação do modelo com a alteração dos parâmetros features e target (k = 5) (Figura 20)

```
[[   16    11    33    90    43     0    17     0     0     0]
 [   24     4    20    77    45     0     3     2     1     0]
 [   38    11   282   699    67     4    74     8     2     0]
 [   61    20   279  2546   150    28   233    31     4     0]
 [   41    25    65   226   646     5   771    18     4     0]
 [    0     3    15    93    16  5465    39     3     1     0]
 [    1     0    23   135   528    18 10408    30     4     0]
 [    3     8    28   167     7     5    40   806     0     0]
 [    0     1     4    13    28     4    52     7    10     0]
 [    0     0     1     3     1     0     4     0     0     2]]
                precision    recall  f1-score   support

      Analysis       0.09      0.08      0.08       210
      Backdoor       0.05      0.02      0.03       176
           DoS       0.38      0.24      0.29      1185
      Exploits       0.63      0.76      0.69      3352
       Fuzzers       0.42      0.36      0.39      1801
       Generic       0.99      0.97      0.98      5635
        Normal       0.89      0.93      0.91     11147
Reconnaissance       0.89      0.76      0.82      1064
     Shellcode       0.38      0.08      0.14       119
         Worms       1.00      0.18      0.31        11

      accuracy                           0.82     24700
     macro avg       0.57      0.44      0.46     24700
  weighted avg       0.80      0.82      0.81     24700
```

Figura 20. Visualização da matriz e avaliação do modelo com a alteração de parâmetros

3.3.2. Implementando Árvore de decisão

Os datasets UNSW_NB15_training-set.csv e o UNSW_NB15_testing-set.csv, foram utilizados para implementar o algoritmo da árvore de decisão seguindo as seguintes etapas:

Dataset UNSW_NB15_training-set.csv

Etapa 1. Importando as bibliotecas Python necessárias (Figura 21)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
```

Figura 21. Importando as bibliotecas necessárias para o modelo

Etapa 2. Carregando o dataset (Figura 22)

```
# Carregando o dataset
file_path = '/content/sample_data/UNSW_NB15_training-set.csv'
df = pd.read_csv(file_path)
```

Figura 22. Carregando o dataset para implementar a árvore de decisão

Etapa 3. Visualizando o dataset e conferindo a quantidade dos dados

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	...	1	1	0	0
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	...	1	2	0	0
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	...	1	3	0	0
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	...	1	3	1	1
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	...	1	40	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
175336	175337	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	13	24	0	0
175337	175338	0.505762	tcp	-	FIN	10	8	620	354	33.612649	...	1	2	0	0
175338	175339	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	3	13	0	0
175339	175340	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	14	30	0	0
175340	175341	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	16	30	0	0

175341 rows × 45 columns

Figura 23. Visualização do dataset e o resumo da composição dos dados (rows x columns)

Etapa 4. Selecionando features e target (Figura 24)

```
# Selecionando features e target
features = ['synack', 'ackdat']
target = 'label'
```

Figura 24. Definindo as features para o modelo

Etapa 5. Verificando a existência das colunas selecionadas ao dataset utilizado e definir o conjunto de dados referentes ao treino e teste, foram escolhidos a proporção 70/30 treinamento/teste) (Figura 25)

```
# Verificando se as colunas selecionadas pertencem ao dataset
if all(col in df.columns for col in features + [target]):
    # Dividir os dados em treino e teste
    X = df[features]
    y = df[target]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Figura 25. Conferindo as colunas no dataset e divisão do conjunto de treinamento e teste

Etapa 6. Criação e treinamento da árvore de decisão e definição da profundidade da ramificação através dos parâmetro max_depth, e iniciando as previsões a partir da divisão dos conjuntos de dados de treinameto e teste (Figura 26)

```
# Criando e treinando a árvore de decisão
tree_clf = DecisionTreeClassifier(max_depth=5, random_state=42)
tree_clf.fit(X_train, y_train)

# Fazendo previsões a partir dos conjuntos de treinamento e teste
y_train_pred = tree_clf.predict(X_train)
y_test_pred = tree_clf.predict(X_test)
```

Figura 26. Instanciando o treinamento e teste do modelo e realizando as previsões

Etapa 7. Avaliando o modelo (Figuras 27 e 28)

```
# Avaliando o modelo
print("\nRelatório de Classificação - Conjunto de Treinamento")
print(classification_report(y_train, y_train_pred))

print("\nRelatório de Classificação - Conjunto de Teste")
print(classification_report(y_test, y_test_pred))
```

Figura 27. Código para verificar a avaliação do modelo

```
Relatório de Classificação - Conjunto de Treinamento
              precision    recall  f1-score   support

           0       0.88      0.52      0.65     39228
           1       0.81      0.97      0.88     83510

    accuracy                           0.82    122738
   macro avg       0.84      0.74      0.77    122738
weighted avg       0.83      0.82      0.81    122738


Relatório de Classificação - Conjunto de Teste
              precision    recall  f1-score   support

           0       0.88      0.51      0.65     16772
           1       0.81      0.97      0.88     35831

    accuracy                           0.82     52603
   macro avg       0.84      0.74      0.76     52603
weighted avg       0.83      0.82      0.81     52603
```

Figura 28. Visualização da avaliação dos conjutos de treinamento e teste

Etapa 8. Visualizando a árvore de decisão (Figura 29 e 30)

```
# Visualizar a árvore de decisão
plt.figure(figsize=(20, 10))
plot_tree(tree_clf, feature_names=features, class_names=[str(cls) for cls in tree_clf.classes_],
          filled=True, rounded=True)
plt.title("Árvore de Decisão")
plt.show()
```

Figura 29. Código para visualizar a árvore de decisão

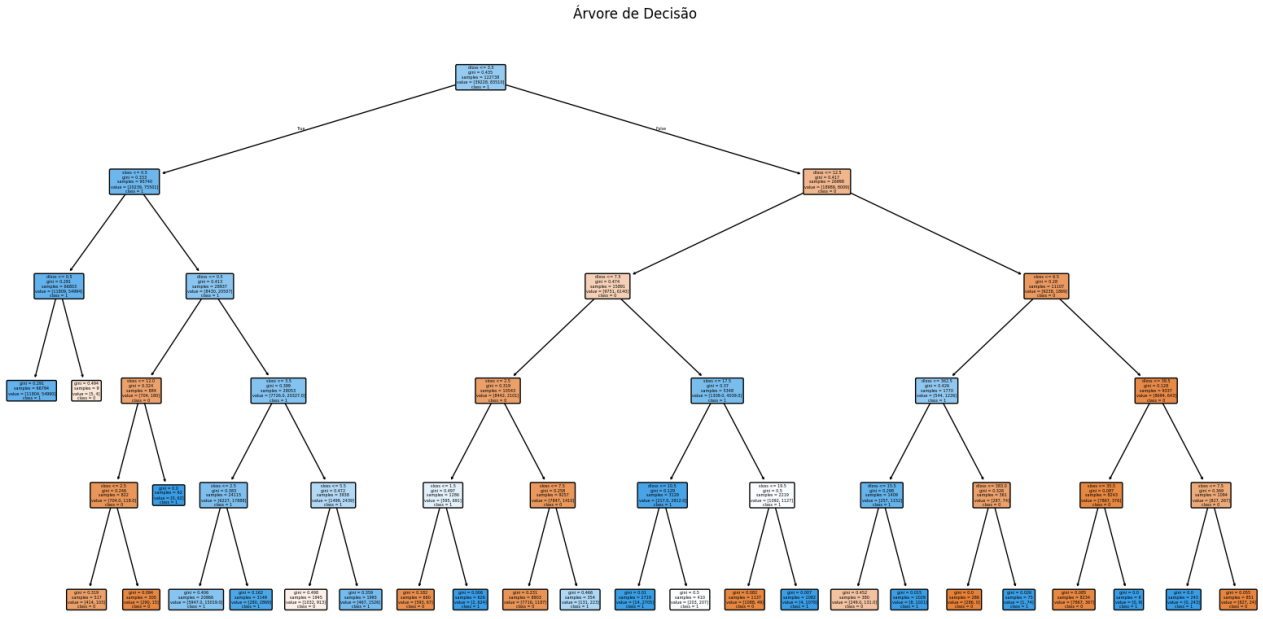

Figura 30. Visualização da árvore de decisão

Dataset UNSW_NB15_testing-set.csv

Replicar a etapa 1 (Figura 21) do dataset UNSW_NB15_training-set.csv

Etapa 2 e 3. Carregando o dataset e visualizando a quantidade de dados

```
import pandas as pd
import numpy as np
ciber = pd.read_csv('/content/sample_data/UNSW_NB15_testing-set.csv')
ciber
```

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd
0	1	0.000011	udp	-	INT	2	0	496	0	90909.090200	...	1	2	0	0
1	2	0.000008	udp	-	INT	2	0	1762	0	125000.000300	...	1	2	0	0
2	3	0.000005	udp	-	INT	2	0	1068	0	200000.005100	...	1	3	0	0
3	4	0.000006	udp	-	INT	2	0	900	0	166666.660800	...	1	3	0	0
4	5	0.000010	udp	-	INT	2	0	2126	0	100000.002500	...	1	3	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
82327	82328	0.000005	udp	-	INT	2	0	104	0	200000.005100	...	1	2	0	0
82328	82329	1.106101	tcp	-	FIN	20	8	18062	354	24.410067	...	1	1	0	0
82329	82330	0.000000	arp	-	INT	1	0	46	0	0.000000	...	1	1	0	0
82330	82331	0.000000	arp	-	INT	1	0	46	0	0.000000	...	1	1	0	0
82331	82332	0.000009	udp	-	INT	2	0	104	0	111111.107200	...	1	1	0	0

82332 rows × 45 columns

Figura 30. Visualizando o dataset e conferindo a quantidade de dados

Etapa 4, 5 e 6 (Figuras 24, 25 e 26, respectivamente) são semelhantes as apresentadas para o dataset UNSW_NB15_training-set.csv.

Etapa 7. Avaliando o modelo (Figura 31)

```
Relatório de Classificação - Conjunto de Treinamento
              precision    recall  f1-score   support

           0       0.80      0.60      0.68     25853
           1       0.73      0.88      0.80     31779

    accuracy                           0.75     57632
   macro avg       0.76      0.74      0.74     57632
weighted avg       0.76      0.75      0.75     57632


Relatório de Classificação - Conjunto de Teste
              precision    recall  f1-score   support

           0       0.80      0.59      0.68     11147
           1       0.72      0.88      0.79     13553

    accuracy                           0.75     24700
   macro avg       0.76      0.73      0.74     24700
weighted avg       0.76      0.75      0.74     24700
```

Figura 31. Avaliação do modelo após a separação do conjunto de treinamento e teste

Etapa 8. Visualizando a árvore de decisão (Figuras 32 e 33)

```
# Visualizar a árvore de decisão
plt.figure(figsize=(20, 10))
plot_tree(tree_clf, feature_names=features, class_names=[str(cls) for cls in tree_clf.classes_],
          filled=True, rounded=True)
plt.title("Árvore de Decisão")
plt.show()
```

Figura 32. Código para visualização da árvore de decisão

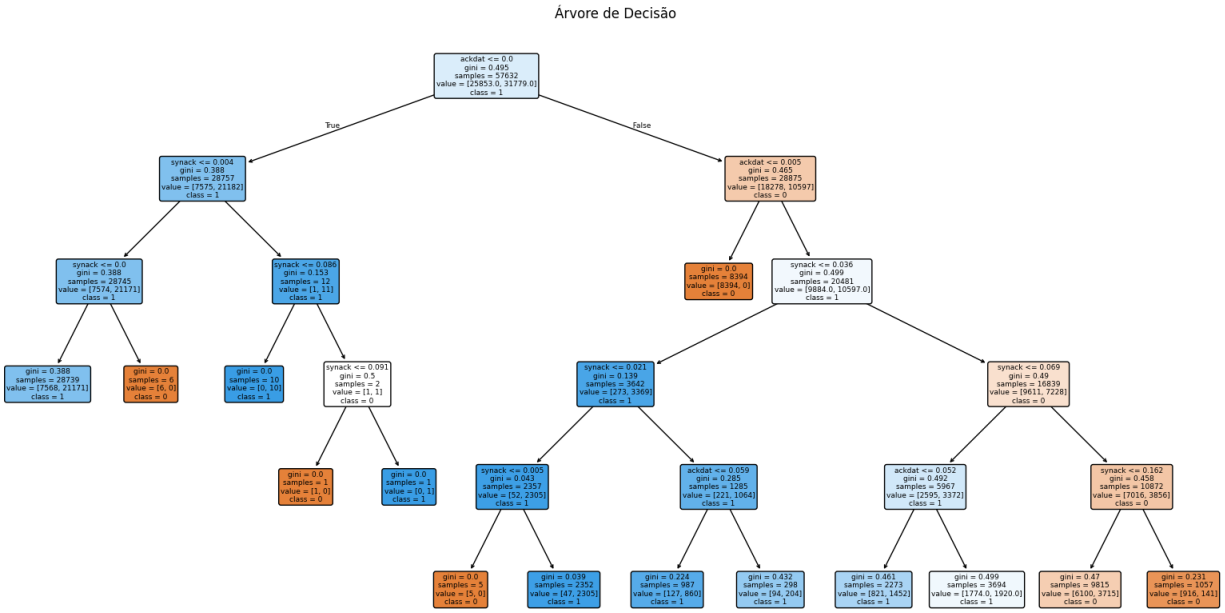

Figura 33. Visualização da árvore de decisão após o treinamento

3.4. Ferramentas utilizadas

Para a realização desta análise exploratória de dados houve a combinação de ferramentas de software e bibliotecas, com o propósito de implementar os algoritmos de classificação K-Nearest Neighbors (KNN) e Árvore de Decisão. São elas:

- **Python:** linguagem de programação para o desenvolvimento do projeto, possui sintaxe clara e concisa e bibliotecas específicas para análise de dados e aprendizado de máquina.
- **Pandas:** biblioteca utilizada para carregar, manipular e explorar os dados. Assim como para a visualização de amostras e a criação de subconjuntos de dados.
- **Numpy:** suporte para operações matemáticas e manipulação de arrays multidimensionais.
- **Matplotlib:** para a criação de gráficos e visualizações, permitindo compreender padrões e tendências nos dados durante a análise exploratória.

- **Scikit-learn:** biblioteca para a implementação dos algoritmos de classificação K-Nearest Neighbors (KNN) e da árvore de decisão; e para avaliação do desempenho calculando as métricas de precisão, recall, F1-score, acurácia e geração de matrizes de confusão.
- **Collab:** Ambiente usado para desenvolvimento do código, permitindo unificar texto explicativo, visualizações e código.

4. RESULTADOS

Após da implementação dos algoritmos de classificação KNN e árvore de decisão nos datasets UNSW_NB15_training-set.csv e UNSW_NB15_testing-set.csv foi possível identificar que algumas classes estavam desbalanceadas então foi proposta o experimento considerando uma seleção específica de features e um target considerando a classificação (label) do tráfego da rede, sendo 0 para normal e 1 para possíveis incidentes.

4.1. Resultados KNN

No dataset UNSW_NB15_training-set.csv foram modificados os parâmetros do valor de k, influenciando na quantidade de dados analisados (podem ser percebidos pela matriz) e apesar da alteração, os dados obtidos nas métricas de precisão, recall, f1-score e acurácia não foram afetados significativamente, como pode ser observado a seguir:

Resultado para k = 5

```
[[14857  1915]
 [ 1288 34543]]
              precision    recall  f1-score   support

           0       0.92      0.89      0.90     16772
           1       0.95      0.96      0.96     35831

    accuracy                           0.94     52603
   macro avg       0.93      0.92      0.93     52603
weighted avg       0.94      0.94      0.94     52603
```

Figura 34. Visualização da matriz do conjunto de treinamento e teste e a avaliação do modelo

Esse resultado apresenta a matriz do treinamento e teste e a avaliação do modelo KNN com valor de k = 5 (Figura 34), aplicado ao dataset UNSW_NB15_training-set.csv. Ele demonstrou um desempenho consistente, com acurácia de 94% e métricas equânimes para ambas as classes, sendo eficaz na previsão da classe 1 quanto da classe 0. Alguns ajustes são necessários para refinar o desempenho, principalmente para a classe 0.

Resultados para k = 10

```
[[15066  1706]
 [ 1510 34321]]
              precision    recall  f1-score   support

           0       0.91      0.90      0.90     16772
           1       0.95      0.96      0.96     35831

    accuracy                           0.94     52603
   macro avg       0.93      0.93      0.93     52603
weighted avg       0.94      0.94      0.94     52603
```

Figura 35. Visualização da matriz e a avaliação do modelo para k = 10

Alterando o valor de k para 10, obteve-se o resultado da matriz do treinamento e teste acima e a uma nova avaliação do modelo (Figura 35), aplicado ao dataset UNSW_NB15_training-set.csv. Manteve a acurácia de 94% similar ao k = 5, porém houve redução de 1% na métricas de precision, e ganho de 1% no recall, mantendo a eficácia na previsão da classe 1. A classe majoritária indica que os exemplos positivos foram identificados corretamente.

Gráfico de dispersão

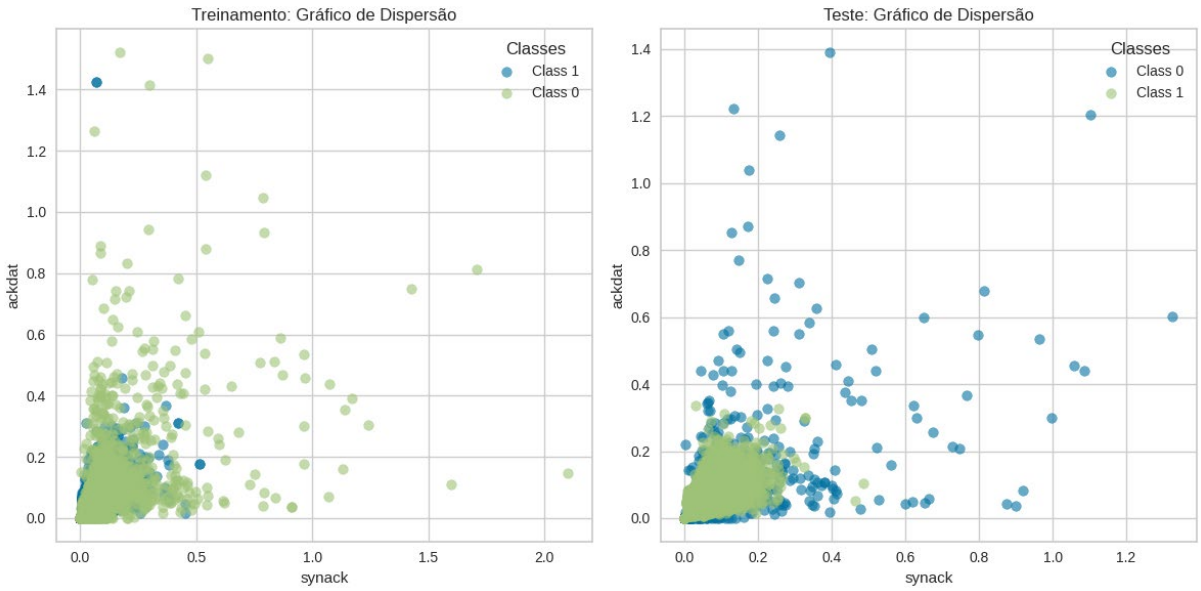

Figura 36. Gráfico de dispersão para os conjuntos de treinamento e teste

Os gráficos de dispersão para os conjuntos de treinamento e teste (Figura 36), retratam a relação entre as variáveis synack (eixo x) e ackdat (eixo y), com a separação das classes (Class 0 e Class 1). O gráfico à esquerda (conjunto de treinamento), concentra os pontos na região inferior esquerda, ou seja, as variáveis apresentam valores baixos, à direita o conjunto de teste, pela similaridade visual, é possível inferir consistência nos dados.

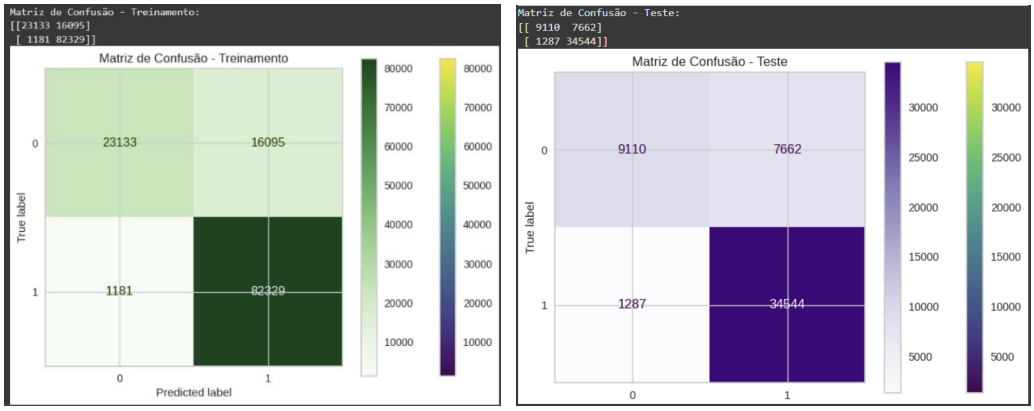

Figura 37. Matriz de confusão para o dataset training

O modelo avaliou corretamente a maioria dos casos (Figura 37), sendo identificados 95% dos casos da classe 1, representando uma alta sensibilidade, entretanto tem uma taxa de falsos positivos elevada, prejudicial para a área da segurança cibernértica, um ambiente extremamente crítico.

No dataset UNSW_NB15_testing-set.csv também foram modificados os parâmetros do valor de k, influenciando na quantidade de dados analisados (podem ser percebidos pela matriz) houve uma pequena melhora na precisão para a classe 1 para o k = 10 a acurácia permaneceu equivalente, como pode ser observado a seguir:

Resultado para k = 5

```
[[10353   794]
 [ 1143 12410]]
              precision    recall  f1-score   support

           0       0.90      0.93      0.91     11147
           1       0.94      0.92      0.93     13553

    accuracy                           0.92     24700
   macro avg       0.92      0.92      0.92     24700
weighted avg       0.92      0.92      0.92     24700
```

Figura 38. Visualização da matriz e da avaliação do modelo do dataset testing

O dataset UNSW_NB15_testing-set.csv equivale a 47% dos dados do training. Com uma base menor o resultado a avaliação do modelo KNN com valor de k = 5 (Figura 38), equivale em acurácia de 92% e métricas equilibradas para ambas as classes.

Resultado para k = 10

```
[[10634   513]
 [ 1401 12152]]
              precision    recall  f1-score   support

           0       0.88      0.95      0.92     11147
           1       0.96      0.90      0.93     13553

    accuracy                           0.92     24700
   macro avg       0.92      0.93      0.92     24700
weighted avg       0.93      0.92      0.92     24700
```

Figura 39. Visualização da matriz e da avaliação do modelo do dataset testing para k = 10

Aumentando a distância entre os pontos dos vizinhos (k = 10), a avaliação do modelo (Figura 39) permite perceber uma redução de 2% em comparação ao k = 5. Um aumento na precisão e redução do recall da classe 1, significando um bom desempenho na previsão dos positivos.

Gráfico de dispersão

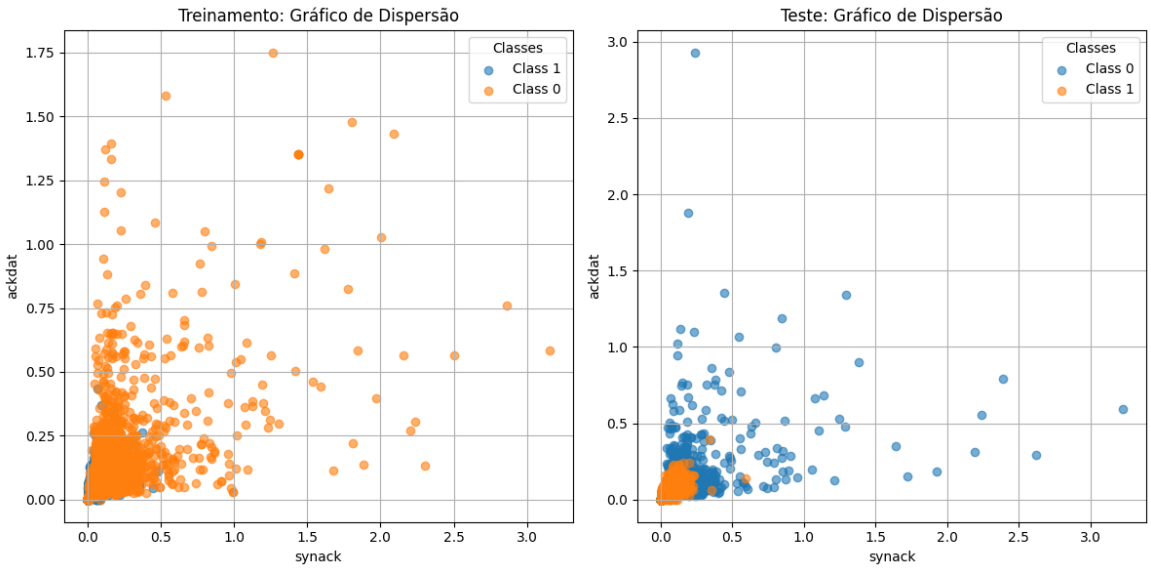

Figura 40. Gráfico de dispersão para os conjuntos de treinamento e teste do dataset testing

Consistência entre os conjuntos de treinamento e teste é visto no gráfico de dispersão (Figura 40). Entretando percebe-se uma sobreposição entre as classes em valores baixos, isso pode comprometer a classificação.

Matriz de Confusão

No geral modelo apresentou desempenho satisfatório, sendo eficiente na identificação dos dados positivos relacionados a classe 1, mas precisa de ajustes para diminuir os falsos positivos. (Figura 41)

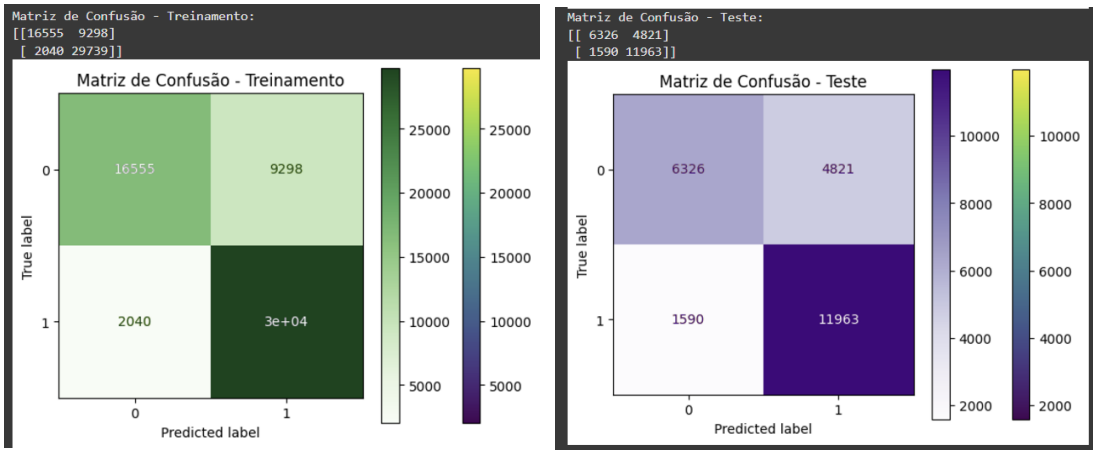

Figura 41. Matriz de confusão para o dataset de testing

Alteração das features e do target

Como o intuito de descobrir possibilidades de melhorar a classificação do dataset training e explorando as features presentes no conjunto de dados, realizou-se uma seleção ampliado as features e alterando o target para a categoria dos ataques, nos estudos anteriores o target selecionado rotulava o dado em 0 e 1. Os resultados obtidos neste cenário evidenciou o desbalanceamento das classes. (Figura 42)

Resultado para k = 5

```
[[   87     2    94   353     6     0    56     0     0     0]
 [    1    60    97   351     5     0     6     5     2     2]
 [   29    20   834  2513    86    10    48    21     7     0]
 [   54    35   958  8399   261    18   244   157    23     9]
 [    8     3   114   583  3636     6   929    61    38     4]
 [    2     0    37   176    33 11804    16     3     3     1]
 [   61     2    28   270  1476     6 14848    62    18     1]
 [    2     3   134   672    32     8    24  2228    12     0]
 [    1     0    19    50   137    11    59    24    67     0]
 [    0     0     1    11     5     1     4     0     0    16]]
                precision    recall  f1-score   support

      Analysis       0.36      0.15      0.21       598
      Backdoor       0.48      0.11      0.18       529
           DoS       0.36      0.23      0.28      3568
      Exploits       0.63      0.83      0.71     10158
       Fuzzers       0.64      0.68      0.66      5382
       Generic       0.99      0.98      0.99     12075
        Normal       0.91      0.89      0.90     16772
Reconnaissance       0.87      0.72      0.79      3115
     Shellcode       0.39      0.18      0.25       368
         Worms       0.48      0.42      0.45        38

      accuracy                           0.80     52603
     macro avg       0.61      0.52      0.54     52603
  weighted avg       0.79      0.80      0.79     52603
```

Figura 42. Visualização da matriz e da avaliação do modelo (k = 5)

A acurácia foi de 80% para o modelo considerando k = 5 ao alterar os parâmetros de feature e target e as métricas de avalição tiveram bom desempenho para classes com maior quantidade de dados, classes menores não performaram muito bem, a ausência de informação comprometeu a predição. (Figura 42)

Resultado para k = 10

```
[[   66     0    94   355     7     0    76     0     0     0]
 [    0    44    99   359     8     0     9     6     2     2]
 [   12     4   730  2614    96    10    71    24     5     2]
 [   30    11   930  8454   278     8   257   161    19    10]
 [    4     1   112   638  3699    12   801    77    32     6]
 [    0     0    37   183    33 11794    22     4     2     0]
 [   52     2    20   274  1574     8 14748    79    14     1]
 [    0     2   119   704    33    10    19  2224     4     0]
 [    0     2    16    51   156    13    44    31    55     0]
 [    0     0     0    21     5     1     3     0     0     8]]
                precision    recall  f1-score   support

      Analysis       0.40      0.11      0.17       598
      Backdoor       0.67      0.08      0.15       529
           DoS       0.34      0.20      0.26      3568
      Exploits       0.62      0.83      0.71     10158
       Fuzzers       0.63      0.69      0.66      5382
       Generic       0.99      0.98      0.99     12075
        Normal       0.92      0.88      0.90     16772
Reconnaissance       0.85      0.71      0.78      3115
     Shellcode       0.41      0.15      0.22       368
         Worms       0.28      0.21      0.24        38

      accuracy                           0.80     52603
     macro avg       0.61      0.48      0.51     52603
  weighted avg       0.79      0.80      0.79     52603
```

Figura 43. Visualização da matriz e da avaliação do modelo (k = 10)

Mesmo com a alteração do valor de k (k = 10), a acurácia do modelo manteve-se em 80%, entretanto houve um aumento de 20% na precisão de backdoor e um alterção significativa na categoria Worms, a precisão alterou de 48% (k = 5) para 28% (k = 10), 50% de redução na métrica de recall, e redução do f1-score também. (Figura 43)

Matriz de confusão

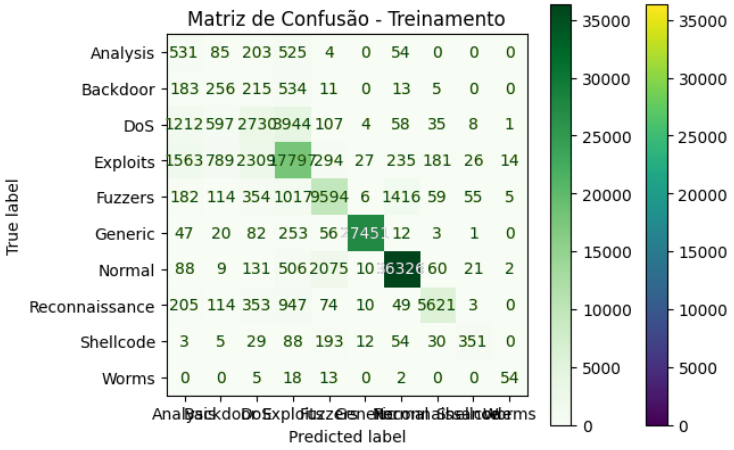

Figura 44. Matriz de confusão para o conjunto de treinamento

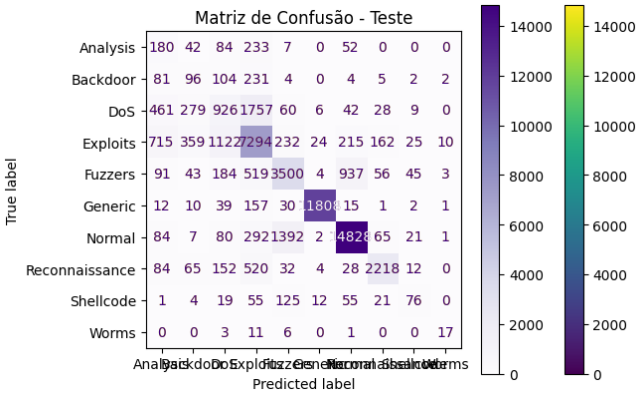

Figura 45. Matriz de confusão para o conjunto de teste

As diagonais representam as amostras classificadas corretamente (verdadeiros positivos), e os valores fora são os erros de classificação. (Figuras 44 e 45)

Algumas classes tiveram um erro significativo de identificação, dentre elas o backdoor, confundido com Analysis e DoS (Deny of Service/Negação de serviço); bem como Exploits e DoS.

Worms não performou bem no modelo pois a classe estava bem desbalanceada. Já a classe Fuzzers obteve um bom desempenho.

4.2. Resultados Árvore de decisão

Resultados obtidos com a implementação do modelo de árvore de decisão para os datasets UNSW_NB15_training-set.csv, visando identificar a acurácia do modelo.

Resultado para o conjunto de treinamento e teste com a escolha das features (synack e ackdat) (Figura 46).

```
features = ['synack', 'ackdat']
target = 'label'
```

Figura 46. Código mostrando a mudança das features para treinar o modelo

```
Relatório de Classificação - Conjunto de Treinamento
              precision    recall  f1-score   support

           0       1.00      0.52      0.68     39228
           1       0.81      1.00      0.90     83510

    accuracy                           0.84    122738
   macro avg       0.91      0.76      0.79    122738
weighted avg       0.87      0.84      0.83    122738


Relatório de Classificação - Conjunto de Teste
              precision    recall  f1-score   support

           0       1.00      0.51      0.68     16772
           1       0.81      1.00      0.90     35831

    accuracy                           0.84     52603
   macro avg       0.90      0.76      0.79     52603
weighted avg       0.87      0.84      0.83     52603
```

Figura 47. Relatório de classificação dos conjuntos de treinamento e teste da árvore de decisão do dataset training

A avaliação apresenta aspectos similares de acordo com as métricas em ambos conjuntos, isso representa uma consistências nos dados de treinamento e teste, observação para a precisão da classe 0, que significa que todos os dados que deveriam ser classificados como verdadeiros foram identificados. (Figura 47)

Matriz de confusão

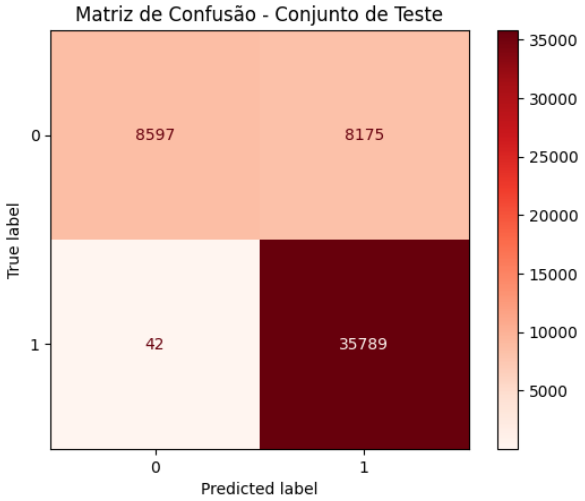

Figura 48. Matriz de confusão para o conjunto de teste

A previsão do modelo para identificar a classe 1 está correto em 81%, uma taxa de bom desempenho, uma acurácia de 84% (Figura 49), porém a taxa de falsos positivos é crítica, 49% dos casos que deveriam ser classificados como 0 e foram atribuídos a classe 1, para aplicação de segurança esse parâmetro é significativo e crucial.

```
    # Avaliando a acurácia
    train_accuracy = accuracy_score(y_train, y_train_pred)
    test_accuracy = accuracy_score(y_test, y_test_pred)

    print(f"Acurácia no Conjunto de Treinamento: {train_accuracy:.2f}")
    print(f"Acurácia no Conjunto de Teste: {test_accuracy:.2f}")

else:
    print("Certifique-se de que as colunas selecionadas existem no dataset!")
```

```
Acurácia no Conjunto de Treinamento: 0.84
Acurácia no Conjunto de Teste: 0.84
```

Figura 49. Código para avaliar a acurácia do modelo

Alterando as features do modelo

```
# Alterando as features
features = ['sloss', 'dloss']
target = 'label'
```

Figura 50. Alterando as features para reaplicar o modelo

Após a alteração das features estes foram os novos valores obtidos para as métricas.

```
Relatório de Classificação - Conjunto de Treinamento
              precision    recall  f1-score   support

           0       0.88      0.52      0.65     39228
           1       0.81      0.97      0.88     83510

    accuracy                           0.82    122738
   macro avg       0.84      0.74      0.77    122738
weighted avg       0.83      0.82      0.81    122738


Relatório de Classificação - Conjunto de Teste
              precision    recall  f1-score   support

           0       0.88      0.51      0.65     16772
           1       0.81      0.97      0.88     35831

    accuracy                           0.82     52603
   macro avg       0.84      0.74      0.76     52603
weighted avg       0.83      0.82      0.81     52603
```

Figura 51. Avaliação do modelo com alteração da feature

Mostra que o modelo tem um desempenho equilibrado em ambas as classes, tratando-as igualmente independente do conjunto de teste e treinamento, com acurácia de 82%. Das previsões realizadas para a classe 0, 88% estavam corretas contra 81% da classe 1, é uma boa performance para o modelo.

Matriz de confusão

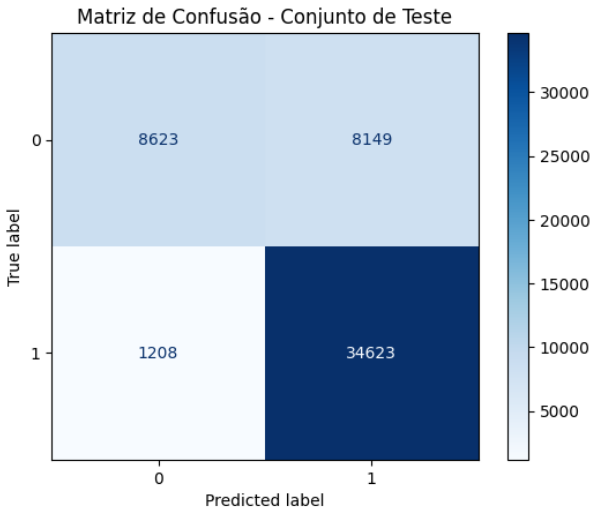

Figura 52. Matriz de confusão para o conjunto de teste após alteração da feature

Exemplos de verdadeiros negativos classificados corretamente equivalem a 16% da amostra total, verdadeiros positivos correspondem a 65% da mostra, 15% representam falsos positivos e 4% são falsos negativos. O modelo apresenta excelente desempenho na detecção da classe positiva (1), com alta sensibilidade e equilíbrio entre precisão e recall. Porém, é insatisfatória a performance para a classe negativa (0), devido a elevada taxa de falsos positivos.

```
    # Avaliando a acurácia
    train_accuracy = accuracy_score(y_train, y_train_pred)
    test_accuracy = accuracy_score(y_test, y_test_pred)

    print(f"Acurácia no Conjunto de Treinamento: {train_accuracy:.2f}")
    print(f"Acurácia no Conjunto de Teste: {test_accuracy:.2f}")

else:
    print("Certifique-se de que as colunas selecionadas existem no dataset!")
```

```
Acurácia no Conjunto de Treinamento: 0.82
Acurácia no Conjunto de Teste: 0.82
```

Figura 53. Código para validar a acurácia do modelo

O modelo apresenta desempenho expressivamente melhor para a classe 1, conforme a matriz de confusão mostra pelo maior número de verdadeiros positivos, pode ser um viés e requer uma atenção e ajustes nos parâmetros para uma validação do modelo. A acurácia para esse conjunto de treinamento e teste é de 82% (Figura 53).

5. CONCLUSÃO

O modelo KNN possui alta sensibilidade para identificar a classe 1, relacionando ao cenário da cibersegurança seria útil na identificação de possíveis incidentes.

A acurácia geral é boa e o F1-score indicam um desempenho consistente para o modelo, quando aplicado em classes balanceada, não performa bem em classes menores. Entretanto com uma taxa de falsos positivos alta, esse modelo representar um problema. Para aplicações em que falsos positivos são críticos como é o caso do SOC.

Em relação ao modelo de árvore de decisão, apresenta excelente desempenho na classe 1, no entanto também apresenta uma taxa elevada de falsos positivos para a classe 0, para aplicação de SOC necessita de revisão. Melhorias no equilíbrio entre precisão e recall, especialmente para a classe 0, podem contribuir para o aumento da eficácia do modelo.

Em suma os modelos tiveram performance satisfatória na classe 1, para projetos futuros é recomendável avaliar o threshold principalmente relacionado a classe 0, realizar o balanceamento das classes, considerar a utilização de uma base mais robusta.

Para aplicações de SOC os modelos ainda necessitam de ajustes, a criticidade e a importância da acuracidade dos incidentes são de extrema relevância para a continuidade dos negócios das organizações e para a atuação de profissionais na investigação, mas os modelos são úteis e contribuem na investigação preliminar de volumes de dados e correlações, com o refinamento é possível ter resultados eficazes.

6. REFERÊNCIAS BIBLIOGRÁFICAS

AGYEMANG, M., Barker, K., & Alhajj, R. (2006). **"A comprehensive survey of distance and similarity measures for clustering applications."** Journal of Knowledge and Information Systems, 9(3), 1-23.

AHMED, M., & HOSSAIN, M. A. (2020). **"Threats and security challenges in SOC operation**s." Journal of Cybersecurity Practice and Research.
ANDERSON, R. (2008). **Security Engineering: A Guide to Building Dependable Distributed Systems**. Wiley.
Árvore de decisão. Disponível em <https://www.plantareducacao.com.br/arvore-de-decisao/>. Acesso em 03/dez/24

BODEAU, D. J., & Graubart, R. (2016). **Cyber Resiliency Engineering Framework**. MITRE Corporation.
BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., & STONE, C. J. (1984). **Classification and Regression Trees.** Wadsworth.

CLARKE, Richard A. and Robert K. Knake. **Guerra Cibernética: A Próxima Ameaça À Segurança E O Que Fazer A Respeito**. 2015 (Editora Brasport)
COVER, T., & HART, P. (1967). **"Nearest neighbor pattern classification."** IEEE Transactions on Information Theory, 13(1), 21-27.

Dataset. Australian Centre for Cyber Security (ACCS). **UNSW-NB15 Dataset**. Disponível em: <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>. Acesso em 28/out/2024.

DE OLIVEIRA, Felipe Barreto; et al. Detecção de ataque DoS utilizando NSL-KDD e abordagens de aprendizado de máquinas. **Revista Ibérica de Sistemas e Tecnologias de Informação**; Lousada Ed. E62, (Oct 2023): 32-45. Disponível em: <https://www.proquest.com/openview/9ec42ec894a0cf76c6ceaa11abb28cb8/1?pq-origsite=gscholar&cbl=1006393>. Acesso em: dez 2023.
DEEP, R., & Goyal, V. (2018). Machine Learning Methods for Cyber Security. **Pramana Research Journal**, Volume 8. Disponível em: <https://www.researchgate.net/publication/325159145_Machine_Learning_and_Deep_Learning_Methods_for_Cybersecurity>. Acesso em: jan 2024.
Documentação sobre scikit para aplicação da curva de validação do KNN. Disponível em: <https://www.scikit-yb.org/en/latest/api/model_selection/validation_curve.html>. Acesso em 02/dez/2024.

FUKUNAGA, K., & NARENDRA, P. M. (1975). **"A branch and bound algorithm for computing k-nearest neighbors."** IEEE Transactions on Computers, 100(7), 750-753.

https://typeset.io/papers/deep-learning-in-intrusion-detection-systems-4f7qmf5u2z>. Acesso em: set 2024.
ISO/IEC 27001:2013. (2013). **Information technology** — Security techniques — Information security management systems — Requirements.
KARATAS, Gozde (2018). Deep Learning in Intrusion Detection Systems. Disponível em <
MCKINNEY, W. (2017). **Python for Data Analysis**. O'Reilly Media.

MONTGOMERY, D. C., PECK, E. A., & VINING, G. G. (2021). **Introduction to Linear Regression Analysis**. Wiley.

MOUSTAFA, N., & SLAY, J. (2015). **"UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)."** 2015 Military Communications and Information Systems Conference (MilCIS), IEEE.

NASSAR, Ahmed e KAMAL, Mostafa. Machine Learning and Big Data Analytics for Cybersecurity Threat Detection: A Holistic Review of Techniques and Case Studies. 2021. Disponível em: <https://journals.sagescience.org/index.php/jamm/article/view/97>. Acesso em: out 2024.
PETERSON, L. E. (2009). **"K-nearest neighbor."** Scholarpedia, 4(2), 1883.

PFLEEGER, C. P., & Pfleeger, S. L. (2006). Prentice Hall.
QUINLAN, J. R. (1986). **"Induction of decision trees."** Machine Learning, 1(1), 81-106.

QUINLAN, J. R. (1993). **C4.5: Programs for Machine Learning**. Morgan Kaufmann.

RUSSEL, S.; NORVIG, P.; **Artificial Inteligence: A Modern Approach**. Prentice-Hall, Second Edition, 2003.
SOUSA, Norberto João Gomes Lopes de. Machine Learning to Improve Security Operations Centers. 2022. Disponível em: <https://recipp.ipp.pt/handle/10400.22/20733>. Acesso em: set 2024.
STALLINGS, W. (2017). **Cryptography and Network Security: Principles and Practice**. Pearson.

STALLINGS, W. (2019). **Network Security Essentials: Applications and Standards**. Pearson.
TUKEY, J. W. (1977). **Exploratory Data Analysis**. Addison-Wesley.

VALENTE, Rodrigo Pinto. Motor de Inferência aplicado à deteção de incidentes de segurança no ciberespaço de uma Organização. 2022. Disponível em: <https://repositorio.iscte-iul.pt/bitstream/10071/27552/1/master_rodrigo_pinto_valente.pdf>. Acesso em: set 2024.
XIN, Yang et al. Machine Learning and Deep Learning Methods for Cybersecurity. 2017. Disponível em: < https://www.researchgate.net/publication/325159145_Machine_Learning_and_Deep_Learning_Methods_for_Cybersecurity>. Acesso em: dez 2023.