UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Martins Abreu

Análise dos Efeitos do PCA e MDI nos *Datasets* UNSW-NB15 e NSL-KDD, Visando a Federação de Máquinas de Aprendizado

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Martins Abreu

Análise dos Efeitos do PCA e MDI nos *Datasets* UNSW-NB15 e NSL-KDD, Visando a Federação de Máquinas de Aprendizado

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Luís Fernando Faina

Coorientador: Márcia Aparecida Fernandes

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2025

Gabriel Martins Abreu

Análise dos Efeitos do PCA e MDI nos *Datasets* UNSW-NB15 e NSL-KDD, Visando a Federação de Máquinas de Aprendizado

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 19 de setembro de 2025

Luís Fernando Faina
Orientador

Márcia Aparecida Fernandes
Coorientadora

Uberlândia, Brasil
2025

Resumo

Este trabalho apresenta uma análise detalhada dos *datasets* UNSW-NB15 e NSL-KDD, com o objetivo de avaliar o impacto de diferentes técnicas de pré-processamento no desempenho de modelos de Inteligência Artificial *Random Forest* (RF) e *Multilayer Perceptron* (MLP), aplicados à detecção de intrusões (Intrusion Detection Systems – IDS). Foram aplicadas técnicas de normalização (*Min-Max* e *Z-Score*) e redução de dimensionalidade (PCA e MDI), além da realização de experimentos de avaliação cruzada e treinamento conjunto entre os dois *datasets*, buscando analisar a viabilidade de integração federada entre modelos treinados em domínios distintos.

Os resultados mostram que o NSL-KDD, apesar de mais antigo e simples, produz resultados consistentes, enquanto o UNSW-NB15, mais realista e heterogêneo, apresenta maior desafio de generalização. A aplicação das técnicas de pré-processamento mostrou-se essencial para estabilizar o aprendizado e reduzir a redundância dos dados. Nos experimentos de avaliação cruzada, observou-se que o treinamento conjunto entre ambos os conjuntos aumentou a capacidade de generalização dos modelos, comportamento análogo à fusão de conhecimento esperada em cenários de aprendizado federado.

Assim, este estudo contribui para o entendimento da relação entre pré-processamento, generalização e federação em IDS baseados em ML, oferecendo subsídios para o desenvolvimento de sistemas distribuídos mais robustos e alinhados à complexidade dos tráfegos de rede modernos.

Palavras-chave: Segurança de Redes, Aprendizado de Máquina, Federação de Máquinas de Aprendizado, *Random Forest*, UNSW-NB15.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IA	Inteligência Artificial
SO	Sistema Operacional
ML	*Machine Learning*
DoS	*Denial of Service*
IDS	*Intrusion Detection System*
RF	*Random Forest*
MLP	*Multi-Layer Perceptron*
PCA	*Principal Component Analysis*
MDI	*Mean Decrease in Impurity*
FE	*Feature Extraction*
FS	*Feature Selection*
KCV	*K-fold Cross Validation*
PCAP	*Packet Capture*
CSV	*Comma-Separated Values*
CVE	*Common Vulnerabilities and Exposures*
AUC-ROC	*Area Under the Receiver Operating Characteristic Curve*
TPR	*True Positive Rate*
FPR	*False Positive Rate*
NIDS	*Network-based Intrusion Detection System*
HIDS	*Host-based Intrusion Detection System*
HFL	*Horizontal Federated Learning*
VFL	*Vertical Federated Learning*
FTL	*Federated Transfer Learning*

Sumário

1 Introdução

O acesso a redes de computadores tem crescido de forma exponencial nas últimas décadas, ocasionando um grande aumento dos dados sensíveis individuais e corporativos, confidenciais e importantes sendo compartilhados por meio da Internet, trazendo à tona riscos cada vez maiores de ataques cibernéticos, causando acesso indevido e a possível alteração desses dados. Com o objetivo de prever e evitar tais ataques, há um ramo de estudo da cibersegurança conhecido como Sistemas de Detecção de Intrusões (Intrusion Detection Systems - IDSs) (ABDULGANIYU; TCHAKOUCHT; SAHEED, 2023).

Os IDSs podem ser divididos em dois tipos: baseados em assinatura e baseados em anomalias. Os sistemas baseados em assinatura reconhecem padrões, mas não detectam invasões desconhecidas, têm alta taxa de falsos negativos e são dependentes do Sistema Operacional (SO), dificultando a implementação em diferentes SOs. Já os sistemas baseados em anomalias requerem muitos dados de tráfego de rede para modelar comportamentos normais e identificar anomalias, usando algoritmos estatísticos, de mineração de dados e de Aprendizado de Máquina (ou *Machine Learning* - ML). A abordagem de ML permite que sistemas aprendam autonomamente a partir de dados e melhorem seu desempenho ao longo do tempo (SARKER, 2021). A vantagem desse tipo de sistema é a forte capacidade de descobrir novos ataques, apesar de apresentar uma maior taxa de falsos positivos (OTOUM; NAYAK, 2021).

Para o desenvolvimento de um IDS eficiente, é essencial ter uma grande quantidade e qualidade de dados para o treinamento e teste (ASHFAQ et al., 2017). Por esse motivo, pesquisadores têm explorado métodos de detecção baseados em assinaturas e, mais recentemente, em anomalias, utilizando técnicas de inteligência artificial, como *Deep Learning* e Aprendizado de Máquina, para detectar intrusões em redes (OJHA; ABRAHAM; SNášEL, 2017). Foram realizadas diversas experiências em laboratórios de tecnologia, simulando e documentando uma variedade de ataques cibernéticos em meio a conexões normais em ambientes de rede controlados (MOUSTAFA, 2015). Vale destacar a relevância do *dataset* utilizado no treinamento e na testagem dos IDSs para os resultados obtidos nessas experiências. Um Sistema de Detecção treinado e com detecção eficiente, ao ser testado em um *dataset* específico, não necessariamente irá obter resultados satisfatórios sobre os registros de outros *datasets* (MEDINA-ARCO, J. G.; MAGAN-CARRIÓN, R.; RODRÍGUEZ-GÓMEZ, R. A.; GARCÍA-TEODORO, P. M, 2024). Nesse contexto, pesquisas recentes têm explorado também a federação de IDSs, técnica que possibilita a utilização de múltiplos conjuntos de dados (MARKOVIC et al., 2022).

O desenvolvimento de Sistemas de Detecção de Intrusões (IDS) baseados em in-

teligência artificial é de extrema relevância, tanto para a sociedade quanto para a academia. Com o aumento exponencial de dispositivos conectados e a complexidade das Redes de Computadores, a segurança cibernética tornou-se uma prioridade global. Estatísticas indicam que 80% das violações de segurança envolvem identidades comprometidas (ARMELLINI, 2024), e a aplicação de IA pode reduzir o impacto dessas brechas em até 48% (APARECIDO, 2024). Para a academia, o uso de técnicas avançadas de aprendizado de máquina e redes neurais em IDS representa um campo fértil para pesquisa, com potencial para desenvolver soluções inovadoras que podem ser aplicadas em diversos setores, desde a indústria até a infraestrutura crítica (TURCATO, 2020).

Os trabalhos discutidos a seguir abordam importantes questões acerca do desenvolvimento de IDSs, especialmente em relação aos métodos e conjuntos de dados utilizados. O trabalho de (THAKKAR; LOHIYA, 2020) foca na necessidade de atualizar os conjuntos de dados utilizados na avaliação de IDSs, destacando a eficácia das técnicas de Aprendizado de Máquina e Mineração de Dados na classificação do tráfego de rede e na identificação de padrões de ataque, concentrando sua discussão nos *datasets* "CIC-IDS-2017" e "CSE-CIC-IDS-2018", ambos criados no Canadian Institute for Cybersecurity (CIC). Já o trabalho de (NETO, 2021) examina a aplicação de técnicas de aprendizado de máquina na detecção de *botnets*, enfatizando a importância da seleção de atributos e do processamento de dados para aumentar a eficiência dos modelos de ML. Os dados utilizados para o estudo foram coletados em testes realizados no Cyber Range Lab of UNSW Canberra, um ambiente de simulação e testes de cibersegurança, que deu origem ao *dataset* "UNSW-NB15".

Este trabalho possui como objetivo investigar diferentes máquinas de aprendizado treinadas usando *datasets* para detecção de intrusões de rede, além de realizar uma análise detalhada do *dataset* UNSW-NB15, avaliando a forma como ele foi gerado, seus atributos registrados, como esses atributos foram extraídos do ambiente em que foram coletados, bem como a forma como eles foram calculados. Esta pesquisa está inserida em um contexto mais amplo de estudo sobre o uso de técnicas de Aprendizado de Máquina em Sistemas IDSs, preocupando-se com as técnicas de redução de dimensionalidade de modo a preservar a origem do atributo no *dataset*. Ao preservar parte dos atributos do *dataset* original ao mesmo tempo em que se reduz a dimensionalidade, espera-se alcançar máquinas de aprendizado treinadas com *datasets* com um razoável número de atributos comuns, abrindo espaço para a federação. Embora a implementação de cenários de Aprendizado Federado não seja abordada de forma aprofundada neste trabalho, busca-se oferecer uma análise comparativa entre os *datasets* e técnicas de pré-processamento, além de experimentos iniciais combinando os subconjuntos de treino e teste do UNSW-NB15 e do NSL-KDD, utilizando os atributos comuns a ambos os *datasets*. Esses resultados e análises iniciais podem subsidiar pesquisas futuras no contexto de Federação de Sistemas IDS, em que o presente trabalho está inserido.

No desenvolvimento deste trabalho, será utilizada a linguagem de programação Python, devido à sua versatilidade e ampla adoção na comunidade científica. As bibliotecas *scikit-learn* (PEDREGOSA et al., 2011) e *pandas* (TEAM, 2024) são essenciais, pois a primeira oferece diversas ferramentas para aprendizado de máquina, incluindo algoritmos de classificação e seleção de atributos, enquanto a biblioteca *pandas* facilita a manipulação e análise de grandes volumes de dados, como é o caso dos *datasets* analisados na área de desenvolvimento de IDSs. Além disso, serão aplicadas técnicas como a Análise de Componentes Principais (PCA) e *Mean Decrease Impurity* (MDI) para a seleção de atributos, com o objetivo de reduzir a dimensionalidade dos dados e identificar as *features* mais relevantes para o treinamento do modelo. Espera-se alcançar métricas iguais ou melhores do que as observadas em trabalhos anteriores que utilizam o *dataset* UNSW-NB15.

Além desses recursos, dispõe-se dos resultados obtidos pela pesquisa do discente Miguel Borges de Rezende Costa, com a mesma metodologia, que possui o *dataset* NSL-KDD como principal objeto de estudo. Esses resultados são essenciais para realizar uma comparação entre as principais características dos dois *datasets*, além de avaliar o desempenho de diferentes técnicas de ML em cada um deles, tendo por base a redução de dimensionalidade, buscando alcançar conjuntos semelhantes de *features* entre os dois *datasets*. A compreensão desses resultados permite identificar as oportunidades e os desafios no uso dos *datasets* UNSW-NB15 e NSL-KDD combinados com técnicas de normalização e pré-processamento, a fim de treinar diferentes técnicas de ML para a detecção de ataques de rede.

A partir deste contexto, a presente pesquisa busca responder às seguintes perguntas de investigação: Como as técnicas de normalização e redução de dimensionalidade, aplicadas aos *datasets* UNSW-NB15 e NSL-KDD, impactam o desempenho de diferentes algoritmos de Aprendizado de Máquina em cenários de detecção de intrusões? Qual é o conjunto mínimo de atributos aceitáveis em registros de cada um dos *datasets* utilizados para treinar as MLs de modo a abrir espaço para que um registro não utilizado no treino de uma ML possa ser utilizado na Federação de MLs?

Estas perguntas de pesquisa orientam toda a investigação apresentada neste trabalho, direcionando a análise comparativa entre os *datasets*, a aplicação de técnicas de pré-processamento e a avaliação do desempenho de diferentes algoritmos de aprendizado de máquina, visando o contexto de Federação de Máquinas de Aprendizado.

2 Fundamentação Teórica

Para o desenvolvimento deste trabalho, é fundamental a compreensão acerca de alguns conceitos importantes para a área de Segurança da Informação e Inteligência Artificial. Na seção 2.1, detalha-se o conceito de Sistema IDS, na seção 2.2, descreve-se a importância dos atributos nos *datasets* para Sistemas IDS, na seção 2.3 descrevem-se os algoritmos de Redução de Dimensionalidade utilizados na pesquisa, na seção 2.4 são apresentadas as Máquinas de Aprendizado exploradas e na seção 2.5 detalha-se o conceito de Federação de Máquinas de Aprendizado.

2.1 Sistemas IDSs

Sistemas IDS (*Intrusion Detection Systems*) são componentes essenciais da arquitetura de segurança cibernética, responsáveis por monitorar o tráfego de rede e identificar atividades suspeitas ou maliciosas, permitindo que administradores de rede detectem e respondam a ameaças em tempo real (ABDULGANIYU; TCHAKOUCHT; SAHEED, 2023). Dentre suas principais funções, destacam-se: i) análise de configurações e vulnerabilidades do sistema; ii) avaliação da integridade dos sistemas e arquivos; iii) reconhecimento de padrões típicos de ataques e iv) identificação de padrões anômalos de atividade e rastreamento de violações de políticas dos usuários (ASHOOR; GORE, 2010).

Sistemas IDS podem ser classificados quanto a sua localização ou quanto ao método de detecção usado. Quanto à localização, a literatura destaca três tipos principais: i) IDSs baseados em Rede (*Network-based* IDS - NIDS); ii) IDSs baseados em *Host* (*Host-based* IDS - HIDS) e iii) IDSs Híbridos (*Hybrid* IDS). Os NIDs são posicionados ao longo de uma rede, a fim de monitorar todo o tráfego, enquanto os HIDs são colocados em um único *host* para escanear e monitorar todos os processos envolvendo esse *host* na rede. Já os IDSs Híbridos apresentam a gestão e o alerta de dispositivos de detecção de intrusões tanto baseados em rede quanto baseados em *host* (ASHOOR; GORE, 2010).

Em relação ao método de detecção utilizado, os IDSs são baseados em Anomalias (*Anomaly-based* IDS) ou em Assinatura (*Signature-based* IDS). Sistemas IDSs baseados em Assinatura (também chamados de IDSs baseados em uso indevido, *Misuse-based*) são muito eficazes contra ataques conhecidos, pois dependem do recebimento de atualizações regulares de padrões, mas não são capazes de detectar ameaças não conhecidas. Os sistemas IDSs baseados em Anomalias, por outro lado, dependem da classificação do comportamento na rede em normal e anômalo, sendo essa classificação baseada em regras ou heurísticas, no lugar de padrões ou assinaturas. Para a implementação desse tipo de IDS, é necessário conhecer o comportamento normal da rede. Diferentemente do sistema baseado

em uso indevido, o sistema baseado em anomalias pode detectar ameaças desconhecidas anteriormente, mas a probabilidade de falsos positivos é maior (ASHOOR; GORE, 2010).

O conceito de IDS foi introduzido em 1972, quando (ANDERSON, 1972) publicou um relatório da Força Aérea dos Estados Unidos discutindo a necessidade de detectar brechas de segurança em sistemas computacionais. Até a década de 1980, os operadores de segurança dependiam da análise manual de *logs* e dados de auditoria para as verificações nos sistemas. No entanto, com o avanço das tecnologias, esses métodos não eram mais suficientes para lidar com a complexidade dos sistemas, o que gatilhou pesquisas visando o desenvolvimento de IDSs automatizadas. (ANDERSON, J. P., 1980) também propôs a automatização de IDSs ao isolar comportamentos anômalos em dados de auditoria. (DENNING; NEUMANN, 1985) desenvolveram o primeiro sistema de detecção em tempo real baseado em regras.

O desenvolvimento comercial de tecnologias de detecção de intrusão começou no início dos anos 1990. A *Haystack Labs* foi a primeira fornecedora comercial de ferramentas IDS com a linha *Stalker*, focada em produtos baseados em *host*. Ao mesmo tempo, a *Science Applications International Corporation* (SAIC) desenvolveu o Computer Misuse Detection System (CMDS), também voltado para detecção de intrusão baseada em host (*Host-based Intrusion Detection* - HID). Paralelamente, o Centro de Suporte Criptológico da Força Aérea dos EUA criou o *Automated Security Incident Measurement System* (ASIM) para monitorar o tráfego de rede, superando problemas de escalabilidade e portabilidade em produtos de detecção de intrusão baseados em rede (*Network-based Intrusion Detection* - NID) (ASHOOR; GORE, 2010).

2.2 Atributos de *Datasets* para Sistemas IDS

Sistemas IDS baseados em anomalias utilizam técnicas de Aprendizado de Máquina para distinguir comportamentos normais de anômalos, por meio de partições do *dataset* para treino, contendo dados previamente coletados e rotulados, a fim de permitir a classificação correta dos dados (MEDINA-ARCO, J. G.; MAGAN-CARRIÓN, R.; RODRÍGUEZ-GÓMEZ, R. A.; GARCÍA-TEODORO, P. M, 2024). Os atributos em um *dataset* são características do tráfego de rede que descrevem padrões normais e anômalos, cruciais para o treinamento de algoritmos de ML e permitem que o sistema aprenda a diferenciar entre tráfego benigno e ataques. A qualidade e a relevância dos atributos influenciam diretamente a precisão e a capacidade do IDS de detectar ameaças (THAKKAR; LOHIYA, 2020).

Os *datasets* usados para o treinamento e teste dos sistemas IDS podem ser: i) sintéticos, ii) reais ou iii) compostos, de acordo com a origem dos dados que os compõem. Os *datasets* sintéticos são formados por dados gerados em ambientes controlados e garantem

a correta rotulação dos dados, mas podem não refletir com precisão padrões de tráfego reais. Os *datasets* reais capturam o tráfego de rede em ambientes reais, fornecendo padrões autênticos de uso, mas que podem ser desbalanceados, exigindo rotulação posterior. Por fim, os *datasets* compostos apresentam uma combinação de dados reais e sintéticos para introduzir padrões de ataque (MEDINA-ARCO, J. G.; MAGAN-CARRIÓN, R.; RODRÍGUEZ-GÓMEZ, R. A.; GARCÍA-TEODORO, P. M, 2024).

É extremamente importante conhecer o formato e os atributos dos dados representados nos *datasets*, a fim de permitir a sua classificação e manipulação. Informações de tráfego de rede são normalmente capturadas no formato baseado em pacotes ou em fluxo. Os dados baseados em pacotes são comumente capturados no formato PCAP (*Packet Capture File Format*) e contêm um campo de *payload*. Além disso, os metadados disponíveis neles dependem da rede e dos protocolos de transporte usados. Já os dados baseados em fluxo aparecem em um formato mais compacto, que condensa todos os pacotes que compartilham as mesmas propriedades dentro da mesma janela de tempo em um fluxo único, geralmente sem um campo de *payload* (RING et al., 2019).

O primeiro *dataset* para Detecção de Intrusões foi criado em 1998 pelo Laboratório Lincoln do Instituto de Tecnologia de Massachussets (MIT *Lincoln Laboratory)*, chamado de DARPA, sob o projeto fundado pela Agência de Projetos de Pesquisa Avançada de Defesa (*Defense Advanced Research Projects Agency* - DARPA) (CUNNINGHAM et al., 1999). Posteriormente, no ano de 1999, os arquivos `tcpdump` do DARPA foram refinados e processados por pesquisadores da Universidade da Califórnia para formar o *dataset* KDD CUP99 (THAKKAR; LOHIYA, 2020). Apesar de este último ser vastamente utilizado para a avaliação de Sistemas IDSs, ele ainda apresentava muitos registros redundantes, que foram removidos para formar o *dataset* NSL-KDD (TAVALLAEE et al., 2009). Em 2015, foi criado o *dataset* UNSW-NB15, com o objetivo de fornecer uma base mais atualizada e representativa para a avaliação de Sistemas IDS. Ele foi gerado no *Cyber Range Lab* da Universidade de New South Wales (UNSW) de Canberra, através de simulações em um ambiente controlado (MOUSTAFA, 2015). Analisar os atributos em comum entre diferentes *datasets* feitos para IDSs é essencial para garantir uma avaliação confiável e justa dos modelos de ML em diversos cenários de ataque e redes, além de possibilitar a criação de uma base de dados universal mais abrangente (SARHAN; LAYEGHY; PORTMANN, 2022).

2.3 Algoritmos de Redução de Dimensionalidade PCA e MDI

A Redução de Dimensionalidade é uma etapa de pré-processamento amplamente utilizada em análises, visualizações e modelagens de dados de alta dimensão. Esse pré-processamento pode ser feito pela Seleção de Atributos (*Feature Selection* - FS), que esco-

lhe as dimensões mais importantes, ou pela Extração de Atributos (*Feature Extraction* - FE), que transforma o espaço original em um subespaço de menor dimensão, mantendo as informações essenciais. Os métodos FE são classificados em supervisionados e não supervisionados, dependendo se há o uso de rótulos de classe ou não (CHUMERIN; HULLE, 2006).

Um dos métodos de Redução de Dimensionalidade mais usados para FE é a Análise de Componentes Principais (*Principal Component Analysis* - PCA), que se baseia em usar a menor quantidade de componentes possível para reduzir as dimensões dos dados analisados (HASAN; ABDULAZEEZ, 2021). Essa redução é feita ao decompor uma matriz de dados em duas matrizes menores, que capturam os padrões essenciais dos dados. Isso permite representar dados de alta dimensão (como frequências de absorção) em um espaço de menor dimensão (concentrações e espectros), preservando as informações mais relevantes (WOLD; ESBENSEN; GELADI, 1987).

Um método de FS bastante usado baseia-se na avaliação da Redução Média de Impureza (*Mean Decrease in Impurity* - MDI), uma métrica usada para avaliar a importância das variáveis em modelos baseados em árvores de decisão, como *Random Forests* (RFs). No contexto de árvores de decisão, o MDI mede a redução de impureza (diversidade dentro do nó da árvore) que uma variável específica causa ao ser utilizada para dividir os dados em cada nó. Quanto maior a redução de impureza causada por uma variável, mais ela é considerada importante (BREIMAN, 2001).

2.4 Máquinas de Aprendizado

Os Perceptrons de Múltiplas Camadas (*Multilayer Perceptrons* – MLPs) são um tipo de rede neural artificial amplamente utilizada em tarefas de classificação, regressão e reconhecimento de padrões. Inspirados no funcionamento do cérebro humano, os MLPs utilizam neurônios artificiais organizados em camadas: uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída (TAUD; MAS, 2018). Cada neurônio aplica uma função de ativação não-linear (como $ReLU$, sigmoide ou tangente hiperbólica) sobre a soma ponderada das entradas recebidas, permitindo que o modelo capture relações complexas entre variáveis.

O processo de aprendizado é supervisionado e ocorre por meio do algoritmo de retropropagação (*backpropagation*), no qual os pesos das conexões são ajustados de acordo com o erro entre a saída prevista e a saída real (OJHA; ABRAHAM; SNášEL, 2017). Essa capacidade de adaptação faz com que os MLPs sejam modelos flexíveis e poderosos para a detecção de intrusões, já que podem aprender representações não-lineares de tráfego de rede e identificar padrões anômalos em diferentes cenários. Entretanto, os MLPs podem demandar maior custo computacional no treinamento e são sensíveis a problemas como

desbalanceamento dos dados e escolha inadequada da arquitetura de rede (ABDULGANIYU; TCHAKOUCHT; SAHEED, 2023).

As Florestas Aleatórias (*Random Forests* – RF), por sua vez, são um método de aprendizado baseado em *ensemble*, no qual múltiplas árvores de decisão são construídas de forma independente a partir de subconjuntos aleatórios dos dados e dos atributos disponíveis (BREIMAN, 2001). A aleatoriedade introduzida na escolha de amostras e variáveis aumenta a diversidade entre as árvores, reduzindo o risco de *overfitting* e resultando em um modelo robusto e estável. A classificação final é realizada por votação majoritária entre as árvores, enquanto em tarefas de regressão é utilizada a média das previsões.

No contexto de IDS, o RF é especialmente atrativo por apresentar boa capacidade de generalização, lidar de forma eficiente com conjuntos de dados de alta dimensionalidade e fornecer estimativas da importância relativa dos atributos – recurso que fundamenta a técnica de seleção de atributos *Mean Decrease in Impurity* (MDI) empregada neste trabalho. Além disso, estudos anteriores mostram que o RF pode alcançar alto desempenho em tarefas de detecção de intrusões mesmo sem ajustes extensivos de hiperparâmetros, apresentando um equilíbrio interessante entre acurácia, interpretabilidade e custo computacional (KOCHER; KUMAR, 2021).

Assim, tanto os MLPs quanto os RFs representam abordagens complementares: o primeiro, capaz de aprender representações complexas de dados não lineares, e o segundo, eficiente em cenários de alta dimensionalidade e interpretabilidade dos atributos. A escolha dessas duas técnicas, portanto, justifica-se pelo seu amplo uso em trabalhos relacionados e pela relevância de compará-las em diferentes condições de pré-processamento, como a normalização e a redução de dimensionalidade, no contexto de detecção de intrusões em redes.

2.4.1 Métricas Utilizadas para os Testes

A avaliação de modelos de aprendizado de máquina supervisionado geralmente é feita com base em métricas que refletem diferentes aspectos da qualidade das predições, como Precisão, Acurácia, *Recall* e F1-Score (SOKOLOVA; LAPALME, 2009). Ressalta-se que essas métricas devem ser analisadas em conjunto, pois, isoladamente, podem oferecer uma visão distorcida do desempenho real do modelo. Por exemplo, um modelo que classifica todas as instâncias como pertencentes à classe majoritária pode alcançar alta Acurácia ou Precisão, mas falhar completamente na identificação de casos relevantes da classe minoritária. A análise do desempenho dos modelos sobre os *datasets* antes e depois da aplicação das técnicas de normalização e redução de dimensionalidade busca identificar o impacto delas na qualidade dos *datasets* para o treinamento de modelos IDS.

No contexto de técnicas de aprendizado de máquina, a Matriz de Confusão é uma

ferramenta essencial para o cálculo dessas métricas, avaliadas no presente trabalho. A fim de compreender por completo como essas métricas são calculadas a partir da Matriz de Confusão, é essencial compreender como ela se apresenta e o que ela significa para diferentes problemas de classificação.

A Tabela 1 ilustra como funciona uma Matriz de Confusão para o problema de classificação de registros como representantes de comportamento normal ou ataques de rede (Classificação Binária: cada registro pode pertencer a uma entre essas duas classes). Supondo um modelo de aprendizado de máquina qualquer, treinado sobre um *dataset* de treino e testado sobre um *dataset* de teste, as linhas da Tabela indicam a classe prevista pelo modelo e as colunas representam a classe real dos registros no *dataset*.

Tabela 1 – Matriz de Confusão Binária.

Previsto/Real	**Não Ataque**	**Ataque**
Não Ataque	TN	FN
Ataque	FP	TP

A partir da Matriz de Confusão, obtêm-se os seguintes valores: i. TN (*True Negative*, ou Verdadeiros Negativos): equivale à quantidade de registros previstos como comportamento normal, ou seja, não são ataques de rede e que realmente representam comportamento normal; ii. FN (*False Negative*, ou Falsos Negativos): equivale à quantidade de registros previstos como comportamento normal, mas que na realidade representam ataques de rede; iii. FP (*False Positive*, ou Falsos Positivos): representa a quantidade de registros previstos como ataque de rede, mas que na realidade são representantes de comportamento normal de rede; e iv. TP (*True Positive*, ou Verdadeiros Positivos): representa a quantidade de registros previstos como ataque de rede e que realmente representam ataques de rede.

Vale destacar que os termos ”positivo” e ”negativo” podem ter diferentes significados, a depender do objetivo principal do modelo de classificação estudado. No contexto de um modelo classificador como do exemplo anterior, que tem como foco identificar os ataques de rede em um conjunto formado por ataques e registros normais, os registros previstos como ataques de rede são chamados de ”positivos” e os registros de comportamento normal são chamados de ”negativos”.

A princípio, considerou-se a avaliação das máquinas de aprendizado neste estudo com base em uma tarefa de classificação binária, ou seja, a capacidade dos modelos de distinguir entre conexões normais e ataques, conforme indicado pelo atributo alvo *label* do UNSW-NB15. No entanto, após reavaliação da metodologia a ser aplicada, entende-se que a identificação da classe específica de ataque de rede é um aspecto tão fundamental para o desenvolvimento de sistemas IDS quanto a simples distinção entre comportamento normal e malicioso.

Nesse contexto, a Matriz de Confusão assume uma forma expandida, com múltiplas classes. Cada classe do *dataset* é tratada, iterativamente, como classe positiva, enquanto as demais são consideradas negativas. A partir disso, é possível calcular métricas como Precisão, *Recall*, F1-Score e AUC-ROC individualmente para cada classe. Os valores de TP, FP, FN e TN para cada classe são obtidos com base na Matriz de Confusão multiclasse, e os detalhes sobre esse cálculo podem ser consultados no Apêndice A.

Nesse trabalho, as métricas Precisão, Acurácia, *Recall* e *F1-Score* são calculadas individualmente para cada classe, considerando-a como a classe positiva em relação às demais. Posteriormente, os valores são combinados por meio de uma média ponderada, onde o peso de cada classe é proporcional à quantidade de amostras daquela classe no *dataset*. Essa abordagem é conhecida como média ponderada e é particularmente útil quando há desbalanceamento entre as classes, pois evita que métricas de classes com maior número de amostras dominem a avaliação do modelo. Além dessas métricas, também foi incluída a avaliação da métrica *AUC-ROC Score*, a fim de qualificar a capacidade dos modelos em distinguir entre as diferentes classes dos *datasets*.

Precisão

A Precisão mede a proporção de registros realmente pertencentes à Classe i, entre todos os registros classificados como dessa Classe pelo modelo, sendo fundamental quando o custo de um falso positivo é alto (POWERS, 2008).

$$Precisão_i = \frac{TP_i}{TP_i + FP_i} \tag{2.1}$$

Acurácia

A Acurácia representa a proporção de predições corretas sobre o total de amostras da Classe i, sendo uma métrica especialmente útil em *datasets* com classes equilibradas (POWERS, 2008).

$$Acurácia_i = \frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i} \tag{2.2}$$

Recall

Quantifica a proporção de registros corretamente identificados como da Classe i sobre o total de registros pertencentes àquela Classe (inclusive os falsos negativos). Mede a eficiência do modelo em identificar registros da Classe i (POWERS, 2008).

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{2.3}$$

F1-Score

Mede o quanto as previsões positivas feitas pelo classificador se alinham com os rótulos positivos reais do conjunto de dados. Ou seja, essa métrica avalia o equilíbrio entre a precisão e o *recall* (POWERS, 2008).

$$F1\_Score_i = 2 * \frac{Precisão_i * Recall_i}{Precisão_i + Recall_i} \tag{2.4}$$

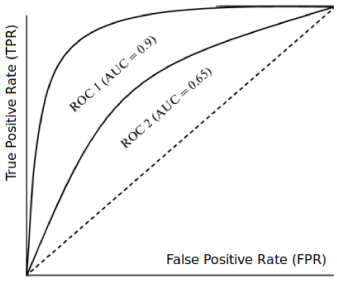
AUC-ROC

A métrica AUC-ROC (*Area Under the Receiver Operating Characteristic Curve*) representa a área sob a curva ROC, que é um gráfico gerado ao se plotar a taxa de verdadeiros positivos (*True Positive Rate* – TPR, ou *Recall*, descrita na eq. 2.5) contra a taxa de falsos positivos (*False Positive Rate* – FPR, descrita na eq. 2.6) em diferentes limiares de decisão do classificador.

$$\text{TPR}_i = \frac{TP_i}{TP_i + FN_i} \tag{2.5}$$

$$\text{FPR}_i = \frac{FP_i}{FP_i + TN_i} \tag{2.6}$$

Ao variar o limiar de decisão do classificador, altera-se o valor mínimo de probabilidade necessário para que uma instância seja classificada como positiva. Limiares mais baixos tendem a aumentar a taxa de verdadeiros positivos, mas também elevam a taxa de falsos positivos; limiares mais altos fazem o oposto, tornando o modelo mais conservador. O gráfico na imagem 1 exemplifica diferentes curvas ROC, com diferentes parâmetros de qualidade, junto de uma Tabela indicando como os valores de AUC-ROC são classificados. Um modelo de classificação que possui um *AUC-ROC Score* de 0.5 equivale a uma decisão aleatória das classes para cada registro, enquanto um modelo de classificação com um AUC-ROC igual a 1 (modelo ideal) é capaz de distinguir eficientemente registros de ataque e normais, independentemente da distribuição dos registros classificados.



Valor AUC-ROC	Qualidade do Resultado
0.9 - 1.0	Excelente
0.8 - 0.9	Muito Bom
0.7 - 0.8	Bom
0.6 - 0.7	Razoável
0.5 - 0.6	Insatisfatório

Figura 1 – Exemplo de gráfico AUC-ROC, adaptado de (TRIFONOVA; LOKHOV; ARCHAKOV, 2014)

A área sob a curva (AUC) pode ser calculada por aproximação numérica via método do trapézio, conforme a eq. 2.7:

$$\text{AUC} = \sum_{i=1}^{n-1} (FPR_{i+1} - FPR_i) \cdot \frac{TPR_{i+1} + TPR_i}{2} \tag{2.7}$$

Como os conjuntos de dados utilizados (UNSW-NB15 e NSL-KDD) envolvem tarefas de classificação em várias classes, foi necessário empregar uma técnica de adaptação da curva ROC para esse cenário. Neste trabalho, foi adotada a abordagem *One vs Rest* (OvR), na qual cada classe é comparada individualmente contra todas as demais. Com isso, o problema de múltiplas classes é transformado em diversos problemas binários, permitindo a aplicação da métrica AUC-ROC de maneira tradicional (TREVISAN, 2022).

Para cada classe, é calculada uma curva ROC considerando-a como classe positiva, enquanto todas as demais são agrupadas como classe negativa. Ao final, os valores de AUC obtidos para cada uma dessas curvas são agregados por meio de uma média ponderada, resultando em um valor final que representa a performance geral do modelo sob a perspectiva OvR.

Cabe destacar que, além da abordagem OvR, existe também a técnica *One vs One* (OvO), em que são geradas curvas ROC para cada par possível de classes, considerando apenas os registros pertencentes às duas classes em questão (TREVISAN, 2022). No entanto, devido à maior complexidade computacional dessa técnica, a abordagem OvR foi considerada mais adequada para os experimentos realizados neste trabalho.

2.5 Federação de Máquinas de Aprendizado

O Aprendizado Federado (*Federated Learning* - FL) é um cenário de aprendizado de máquina em que diversos clientes (dispositivos ou organizações, por exemplo) treinam o mesmo modelo gerenciado em um servidor central, mantendo os dados de treinamento descentralizados (KAIROUZ, P. et al., 2021). O FL incorpora o princípio da Coleta de Dados Focada, proposto no relatório da Casa Branca de 2012, que defende que a coleta de dados deve ser limitada ao mínimo necessário para o propósito específico (KONEčNý et al., 2016).

Os dados de amostra usados são divididos em *Feature Space* (atributos que descrevem cada amostra em um *dataset*) e *Sample ID Space* (identificação única de cada amostra ou exemplo no *dataset*). Baseado em como os dados são distribuídos para várias partes no *Feature Space* e no *Sample ID Space*, as técnicas de FL podem ser categorizadas como *Horizontal* FL, *Vertical* FL e *Federated Transfer Learning* (FTL) (YANG et al., 2019).

O Horizontal (ou *sample-based*) FL é usado nos cenários em que os *datasets* compartilham os mesmos atributos, mas possuem amostras diferentes. Sistemas de FL *sample-based* geralmente assumem participantes honestos e um servidor "honesto mas curioso", ou seja, apenas o servidor é capaz de comprometer a privacidade dos dados participantes (PHONG et al., 2017). O Vertical (ou *feature-based*) FL é aplicável nos casos em que dois *datasets* dividem o mesmo espaço de amostras (*Sample ID Space*), mas possuem atributos diferentes. Esse processo tipicamente assume participantes "honestos mas curiosos", isto é, as partes seguem corretamente o protocolo do sistema, mas podem tentar explorar qualquer informação disponível durante o processo para obter dados confidenciais (YANG et al., 2019). Por último, o FTL é aplicado nos casos em que os *datasets* diferem tanto em amostras quanto em atributos. Sistemas FTL tipicamente assumem participantes "honestos mas curiosos", assim como no *Vertical* FL (GAO et al., 2019).

Recentemente, os avanços no FL se concentraram em melhorar a segurança e a eficiência da comunicação entre os clientes e o servidor central, assim como em abordar os desafios de alta dispersão de dados. Por exemplo, soluções de criptografia homomórfica foram propostas para agregar os parâmetros do modelo de forma segura (PHONG et al., 2017). Além disso, técnicas como o *Deep Gradient Compression* foram introduzidas para reduzir significativamente a largura de banda na comunicação em treinamentos distribuídos em larga escala (PHONG et al., 2017).

3 Trabalhos Relacionados

O trabalho de (MOUSTAFA, 2015) descreve como se deram os processos de coleta, análise e organização dos dados que compõem o *dataset* UNSW-NB15, realizados em 2015 na Universidade de New South Wales de Canberra, na Austrália. Além disso, o trabalho destaca incongruências presentes nos *datasets* KDD98, KDD-CUP e NSL-KDD, como a obsolescência deles em relação aos tráfegos de rede modernos (principalmente quanto às intrusões de baixo impacto, ou *low footprint intrusions*), distribuição desigual de dados e a presença de registros duplicados, por exemplo. O UNSW-NB15 foi desenvolvido com a justificativa de suprir essa lacuna, pois contém uma mistura de registros de tráfego de rede moderno e simulações de ataques contemporâneos.

O *dataset* foi criado utilizando ferramentas como *Argus* e *Bro-IDS*, que extraem características de fluxos e atividades de rede, gerando atributos detalhados como contagem de pacotes, *bytes* de origem e destino, estado da conexão e tempos entre pacotes. Além desses métodos tradicionais, foram incorporadas novas características, incluindo variáveis de conteúdo de transações HTTP e FTP, métricas de tempo e parâmetros específicos de segurança, que auxiliam na identificação de atividades anômalas. Os ataques foram simulados na ferramenta *IXIA PerfectStorm* e categorizados conforme o tipo de ataque, formando um conjunto de dados robusto para análise e detecção de intrusões em sistemas de rede modernos. Esse trabalho será fundamental para o projeto, pois ele detalha os processos envolvidos na coleta e transformação dos dados que compõem o UNSW-NB15, o que permite que seja feita uma análise aprofundada dos atributos que influenciam a classificação dos dados. Esse tipo de informação pode ser muito útil para definir quais são os atributos mais relevantes e como seria possível usar os seus dados em conjunto com outro *dataset* em uma Federação de Máquinas de Aprendizado.

O trabalho de (SIRISHA et al., 2021) apresenta uma análise comparativa entre algoritmos de aprendizado supervisionado e não supervisionado aplicados à detecção de intrusões em redes, utilizando os *datasets* NSL-KDD e CICIDS. Entre os modelos avaliados, destacam-se o *Random Forest* (RF), *Decision Tree* (DT), *Support Vector Machine* (SVM), *Naïve Bayes* (NB) e *K-Means*. Para seleção de atributos preditivos, foi utilizada a técnica *Mean Decrease in Impurity* (MDI) com um limiar de importância de 10%, reduzindo significativamente a dimensionalidade dos conjuntos de dados. A avaliação de desempenho foi conduzida por meio das métricas Acurácia, Precisão, *Recall*, *F1-Score* e AUC-ROC. Os resultados experimentais indicaram que o RF obteve o melhor desempenho entre os métodos supervisionados, alcançando acurácia de 99,4%, enquanto o *K-Means* apresentou desempenho inferior, evidenciando a maior eficácia de modelos supervisionados para os *datasets* avaliados. Este trabalho relaciona-se diretamente com a presente

pesquisa ao empregar a técnica MDI para seleção de atributos e ao evidenciar a robustez do RF na detecção de intrusões, fornecendo embasamento para a escolha e configuração de algoritmos de aprendizado de máquina a serem aplicados no contexto de Federação de Máquinas de Aprendizado proposto.

Kocher e Kumar (KOCHER; KUMAR, 2021) avaliam diferentes algoritmos de ML para detectar intrusões em redes, utilizando o conjunto de dados UNSW-NB15, que representa ataques modernos. Os autores testaram os algoritmos *K-Nearest Neighbors* (KNN), *Stochastic Gradient Descent* (SGD), *Random Forest* (RF), *Logistic Regression* (LR) e Naïve Bayes (NB), aplicando uma técnica de seleção de *features* baseada em Qui-quadrado para melhorar o desempenho, removendo atributos preditivos irrelevantes. O *Random Forest* obteve o melhor desempenho, com precisão de 99,64% e menor taxa de Erro Médio Quadrático (*Mean Squared Error* - MSE), destacando-se como o mais eficaz para o *dataset* utilizado. Esse trabalho relaciona-se com o projeto ao analisar quais algoritmos de ML são mais eficazes para detecção de intrusões sobre os dados do UNSW-NB15. Essa análise pode ser útil na hora de definirmos quais técnicas de ML serão utilizadas e combinadas ao se trabalhar com esse *dataset* em uma Federação de IDSs.

Markovic et. al. (MARKOVIC et al., 2022) propõe o desenvolvimento de um sistema de detecção de intrusões baseado em *Federated Learning* (FL) utilizando o algoritmo *Random Forest* (RF). O objetivo principal é criar um modelo que preserve a privacidade dos dados ao treinar modelos localmente em múltiplos clientes, agregando-os posteriormente em um servidor central para formar um modelo global. O método empregou uma abordagem horizontal de aprendizado federado (*Horizontal Federated Learning* - HFL), conduzindo os testes em cima de quatro *datasets* para detecção de intrusões: KDD99, NSL-KDD, UNSW-NB15 e CIC-IDS-2017, todos divididos em conjuntos de teste, validação e treino, com uma distribuição 70%-10%-20%. Para cada um desses *datasets*, os dados referentes a pacotes de rede foram pré-processados para definir os atributos, e cada registro foi rotulado como comportamento normal ou algum tipo de ataque de rede. Após esse pré-processamento, os *datasets* foram divididos a fim de gerar subconjuntos que simulem um ambiente de HFL, e para cada *dataset* foi escolhido um atributo como critério de divisão (protocolo, tipo de serviço ou porta de destino) e removido do *dataset* de treinamento usado pelos algoritmos de ML. Após isso, foi feito um estudo comparativo para determinar a melhor combinação de hiperparâmetros para cada *dataset*, usando o seu conjunto de validação. Utilizando a combinação de hiperparâmetros escolhida na etapa anterior, uma máquina de aprendizado RF (que o artigo define como ”RFs locais”) foi individualmente treinada e testada sobre os dados do seu próprio subconjunto, sendo cada experimento desses realizado 5 vezes (o trabalho usa os valores médios para comparação). Por fim, as diferentes RFs locais foram combinadas usando diferentes métodos de combinação. O modelo global foi testado em todo o conjunto de teste criado para cada *dataset*. Para os quatro *datasets*, a acurácia encontrada pelo modelo global desenvolvido foi superior

à acurácia média alcançada pelos modelos locais. Este trabalho relaciona-se diretamente com a presente pesquisa, que também utiliza os *datasets* UNSW-NB15 e NSL-KDD, explorando a federação como uma estratégia para combinar informações provenientes de diferentes fontes de dados. Além disso, ele demonstra a eficácia do uso de RF em cenários federados e o impacto do HFL nos resultados, servindo como um referencial importante para a construção e validação de modelos de Aprendizado de Máquina no contexto de Federação de Sistemas IDS.

Em contraste com os trabalhos mencionados, que em sua maioria focam na avaliação de algoritmos específicos ou na implementação direta de abordagens federadas para detecção de intrusões, o presente trabalho adota uma perspectiva exploratória e comparativa sobre os efeitos de técnicas de normalização e redução de dimensionalidade (PCA e MDI) aplicadas aos *datasets* UNSW-NB15 e NSL-KDD. Além disso, este trabalho implementa apenas experimentos superficiais no contexto da Federação de Sistemas IDS, mas busca oferecer uma base teórica e experimental que subsidie pesquisas futuras nessa área de estudo. Essa base para pesquisas futuras é estabelecida ao investigar como diferentes estratégias de pré-processamento impactam o desempenho de algoritmos de aprendizado de máquina em cenários com múltiplos conjuntos de dados, além da possibilidade de utilizar registros do UNSW-NB15 para treinar um modelo capaz de classificar registros do NSL-KDD, e vice-versa.

4 Desenvolvimento

Neste trabalho, a metodologia utilizada para a pesquisa é apresentada em quatro diferentes seções. Na seção 4.1, é feita uma análise detalhada dos *datasets* UNSW-NB15 e NSL-KDD, além de uma comparação entre as principais características de cada um deles. Após isso, na seção 4.2, descrevem-se as etapas de pré-processamento aplicadas sobre cada *dataset*, a fim de permitir o treinamento e teste das máquinas de aprendizado RF e MLP sobre o UNSW-NB15 e o NSL-KDD. A seção 4.3 detalha como as técnicas PCA e MDI foram aplicadas para implementar a redução de dimensionalidade sobre os *datasets* e, por fim, a seção 4.4 detalha os resultados obtidos pelas máquinas RF e MLP após a aplicação das técnicas de redução de dimensionalidade citadas, além dos experimentos de avaliação cruzada entre os *datasets*.

4.1 Análise Geral dos *Datasets*

Dois *datasets* amplamente utilizados para o desenvolvimento de IDSs são o NSL-KDD e o UNSW-NB15. Este trabalho foca sua análise nesses *datasets*, com o objetivo de explorar a viabilidade de usá-los em uma federação de máquinas de aprendizado.

O *dataset* NSL-KDD foi criado para resolver limitações do KDD-CUP99, um dos primeiros *datasets* para IDSs da história, corrigindo redundâncias e desequilíbrios nos dados originais. O *dataset* UNSW-NB15, por sua vez, foi desenvolvido em 2015 na Universidade de New South Wales, em Canberra, Austrália, para representar de forma realista cenários de tráfego de rede atual, com ataques modernos e de baixo impacto. Esse *dataset* busca superar limitações de *datasets* anteriores, incorporando novos atributos que refletem comportamentos recentes de ataques e tráfegos legítimos.

Nas próximas seções, descreve-se a estrutura e os processos envolvidos na geração e coleta dos dados dos *datasets* UNSW e NSL-KDD.

4.1.1 UNSW-NB15

A pesquisa de (MOUSTAFA, 2015) utiliza o método PCAP(*Packet Capture*) para capturar os dados de rede nos experimentos realizados, além das ferramentas *IXIA PerfectStorm*, *tcpdump*, *Argus* e *Bro-IDS*, principalmente. O *dataset* original é composto por 2.540.044 registros, sendo 2.218.761 registros de tráfego normal e 321.283 registros de comportamento malicioso.

O método PCAP (SPASOJEVIC, 2023) é usado para captura, armazenamento e análise de tráfego de rede e de registros de pacotes de rede interceptados ou capturados,

contendo informações básicas de pacotes, como endereços de origem e destino, carimbo de data/hora, protocolo e carga útil. Esses registros de pacotes são chamados de arquivos PCAP, e são comumente capturados e armazenados por ferramentas farejadoras de pacotes de rede, por exemplo, *Wireshark* e *tcpdump*.

A ferramenta *IXIA PerfectStorm* é usada para criar um ambiente moderno híbrido de tráfego de rede normal e anormal. Para o tráfego anormal na rede, 9 classes de diferentes ataques cibernéticos foram contempladas (Tabela 2). A definição dessas classes de ataque está baseada em informações do *site* da CVE (*Common Vulnerabilities and Exposures*), uma organização que cataloga vulnerabilidades de segurança cibernética divulgadas publicamente. O trabalho de (MOUSTAFA, 2015) usa a categorização CVE como base com o propósito de representar, de forma realista, um ambiente de ameaças modernas.

Além do gerador de tráfego de rede *IXIA PerfectStorm*, o trabalho de (MOUSTAFA, 2015) envolve o uso da ferramenta *tcpdump*, que permite capturar, salvar e imprimir uma descrição do conteúdo dos pacotes em uma interface de rede (TCPDUMP, 2024). Dois períodos de captura foram utilizados, o primeiro de 16 horas de simulação no dia 22 de janeiro de 2015 e um segundo de 15 horas no dia 17 de fevereiro de 2015, capturando 100 GB. Na sequência, cada arquivo PCAP foi dividido em 1.000 MB. As estatísticas coletadas nos dois períodos de simulação são descritas na Tabela 1.

Tabela 2 – Estatísticas dos Recursos de Tráfego nos Períodos de Simulação do UNSW-NB15 (MOUSTAFA, 2015)

Estatísticas	**22/01/2015**	**17/02/2015**	**Total**
Horas de simulação	16	15	31
No. de fluxos (*No. of flows*)	987.627	976.882	1.964.509
Bytes enviados pela fonte (*Src bytes*)	4.860.168.866	5.940.523.728	10.800.692.594
Bytes enviados pelo destino (*Des bytes*)	44.743.560.943	44.303.195.509	89.046.756.452
Pacotes da fonte (*Src Pkts*)	41.168.425	41.129.810	82.298.235
Pacotes do destino (*Dst pkts*)	53.402.915	52.585.462	105.988.377
Fluxos que usam o Protocolo TCP	771.488	720.665	1.492.153
Fluxos que usam o Protocolo UDP	301.528	688.616	990.144
Fluxos que usam o Protocolo ICMP	150	374	524
Fluxos que usam outros Protocolos	150	374	524
Registros de Tráfego Normal	1.064.987	1.153.774	2.218.761
Registros de Comportamento Malicioso	22.215	299.068	321.283
no. de IPs únicos (fonte)	40	41	81
no. de IPs únicos (destino)	44	45	89

A ferramenta *Argus* (WEISBERG, 2022) processa pacotes oriundos da rede (no formato PCAP, por exemplo) e gera *features* dos pacotes do fluxo de rede. A ferramenta *Bro-IDS* (PAXSON, 1999), por outro lado, é um analisador de tráfego de rede de código aberto, predominantemente usada para inspecionar o tráfego de rede, em busca de atividades maliciosas. No experimento de (MOUSTAFA, 2015), as *features* do *dataset* UNSW-NB15 foram extraídas utilizando essas duas ferramentas (*Argus* e *Bro-IDS*), além de doze algoritmos desenvolvidos na linguagem de programação C#.

Após a geração dos arquivos PCAP para a criação do UNSW-NB15 (MOUSTAFA,

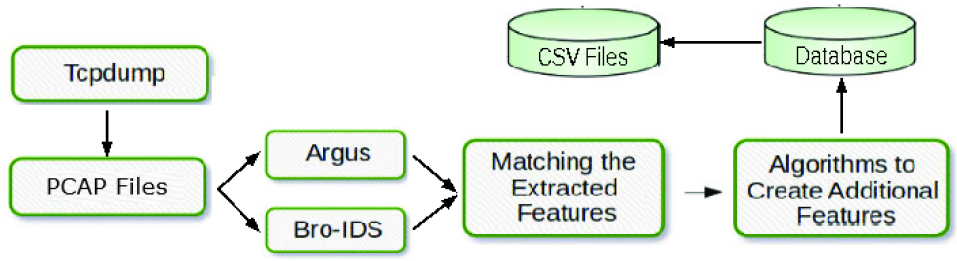


Figura 2 – Arquitetura da Estrutura de Geração do *Dataset* UNSW-NB15 (TUFAN; TEZCAN; ACARTURK, 2021)

2015), foi utilizada a estrutura descrita na Figura 1 para gerar os arquivos CSV que compõem o *dataset*. O formato de arquivo CSV ("*Comma Separated Value*") é comumente utilizado para troca de dados entre diferentes aplicações, sendo cada registro representado por uma linha do arquivo, com seus atributos separados por vírgulas (REPICI, 2002).

Além disso, o artigo de (MOUSTAFA, 2015) divide os atributos do UNSW-NB15 em 4 grupos: i) atributos 1 a 35 - informações básicas extraídas diretamente dos pacotes de dados e cabeçalhos; ii) atributos 36 a 40 - recursos gerais focados em monitoramento e defesa de redes; iii) atributos 41 a 47 - recursos específicos para análise de padrões de conexão em janelas de 100 conexões; e iv) atributos 48 e 49 - rótulo categorizando o registro como normal ou tráfego malicioso e o nome da categoria de ataque.

Os pesquisadores responsáveis pelo desenvolvimento do UNSW-NB15 disponibilizaram publicamente duas partições do *dataset* no site oficial da Universidade de *South Wales Canberra* (MOUSTAFA; SLAY, 2016), nos arquivos CSV de treino e de teste, possuindo 175.341 e 82.332 registros, respectivamente, e serão utilizados para os experimentos deste trabalho. Quanto à distribuição dos registros para a classificação de ataques, somando ambas partições, são 164.673 registros de ataques de rede e 93.000 registros de comportamento normal, dispostos na Tabela 3.

Tabela 3 – Distribuição das Classes de Ataque nos *Datasets* de Teste e Treino do UNSW-NB15 (THANH, 2018)

Tipo de ataque	**Registros *Dataset* de teste**	**Registros *Dataset* de treino**
Normal	56.000	37.000
Analysis	2.000	677
Backdoor	1.746	583
DoS	12.264	4.089
Exploits	33.393	11.132
Fuzzers	18.184	6.062
Generic	40.000	18.871
Reconnaissance	10.491	3.496
Shellcode	1.133	378
Worms	130	44
Total	**175.341**	**82.332**

Além da menor quantidade de registros, os *datasets* de treino e teste também possuem menos atributos. Nem todos os 49 atributos do *dataset* original do UNSW-NB15 são

necessariamente relevantes para o rótulo das classes e, portanto, não aparecem nesses conjuntos. Os atributos descrevendo a hora de início e final de um registro (*record_start_time* e *record_last_time*, respectivamente) não são necessários, devido à presença do atributo *duration* (duração total do registro), por exemplo. Além disso, alguns atributos preditivos não são relevantes por serem específicos da infraestrutura computacional utilizada para as simulações, como endereços IP e portas dos dispositivos fonte e destino (quatro atributos: *srcip*, *dstip*, *sport* e *dsport*) (ZOGHI; SERPEN, 2024).

Os atributos *label* e *attack_cat* são atributos alvos (*output/target features*), sendo o primeiro binário e o segundo do tipo nominal, contendo o nome das categorias de ataque. Dessa forma, restam apenas 42 atributos preditivos nos *datasets* de treino e teste do UNSW-NB15, dispostos na Tabela 4.

Tabela 4 – Lista de *Features* dos *Datasets* de Treino e Teste do UNSW-NB15 (MOUSTAFA; SLAY, 2016)

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

4.1.2 NSL-KDD

O NSL-KDD é amplamente utilizado como um *benchmark* para comparar a eficiência de métodos de Aprendizado de Máquina e Inteligência Artificial aplicados à detecção de intrusões. Esse *dataset* foi criado com o intuito de resolver alguns problemas identificados em um dos *datasets* mais amplamente utilizados para a avaliação de sistemas IDS, o KDD-CUP 99. Dentre esses problemas, destacam-se a grande quantidade de registros redundantes e o desequilíbrio entre a quantidade de registros de cada classe de ataque, além de uma quantidade total de registros grande demais para ser totalmente processada, o que exigia a seleção de subconjuntos menores do *dataset* para a realização de testes.

Esses problemas foram solucionados no NSL-KDD por meio da filtragem de registros e criação dos *datasets* de treino e teste, *NSL-KDDTrain+* e *NSL-KDDTest+*, respectivamente. Houve também uma grande diminuição no número total de registros do KDD-CUP para o NSL-KDD. Enquanto o *dataset* original possuía 5.209.458 registros to-

tais (sendo 4.898.431 deles do subconjunto de treino e 311.027 do subconjunto de teste), o NSL-KDD é composto por apenas 148.517 registros (125.973 no *NSL-KDDTrain+* e 22.544 no *NSL-KDDTest+*) (TAVALLAEE et al., 2009). Essa diminuição na quantidade de registros permite utilizar o *dataset* inteiro, ao invés de selecionar partes menores de um grande *dataset*, o que também facilita a comparação dos resultados encontrados por diferentes trabalhos sobre o NSL-KDD.

Tabela 5 – Lista de *Features* do *Dataset* NSL-KDD (HARB et al., 2011)

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

O NSL-KDD tem 41 *features* que descrevem os registros de tráfego de rede, apresentados na Tabela 5, além de um atributo alvo, "*label*", que define a classe do registro (normal ou uma das classes de comportamento malicioso). As *features* podem ser agrupadas em: i) características básicas - informações básicas do pacote ou conexão, como duração e tipo de protocolo; ii) características de conteúdo - informações extraídas do conteúdo do pacote, úteis para detectar ataques baseados em *payload*; iii) características baseadas em tráfego - estatísticas de tráfego relacionadas à conexão atual ou a um grupo de conexões em uma determinada janela de tempo.

4.1.3 Comparação dos *Datasets* UNSW-NB15 e NSL-KDD

Apesar do *dataset* original do UNSW-NB15 apresentar um maior número de atributos que o NSL-KDD (47 colunas), os subconjuntos de teste e treino do UNSW-NB15 analisados neste trabalho apresentam apenas 42 atributos, além de dois atributos alvos, definindo se o comportamento de rede é normal ou malicioso e qual o tipo de ataque. Já o NSL-KDD possui apenas 41 atributos e um atributo que descreve o tipo de ataque (que possui o valor "Normal"para o caso de tráfego normal de rede) (TAVALLAEE et al., 2009). Apesar dessa semelhança na quantidade de *features*, cada *dataset* ainda possui um conjunto distinto de classes de ataques de rede (9 classes no UNSW-NB15 e 4 no NSL-KDD, dispostas na Tabela 6).

Tabela 6 – Classes de Ataque dos *Datasets* UNSW-NB15 e NSL-KDD

UNSW-NB15	**NSL-KDD**
Fuzzers	DoS
Analysis	Probe
Backdoors	R2L
DoS	U2R
Exploits	
Generic	
Reconnaisance	
Shellcode	
Worms	

O UNSW-NB15 possui uma quantidade maior de registros em comparação ao NSL-KDD, como é possível observar na Tabela 7. No entanto, os dois *datasets* ainda possuem um número próximo de registros em seus conjuntos de treino e teste.

Tabela 7 – Comparação da Quantidade de Registros do UNSW-NB15 e NSL-KDD

Dataset	**UNSW-NB15**	**NSL-KDD**
Treino	175.341	125.973
Teste	82.332	22.544
Total	257.673	148.517

Além disso, o *dataset* NSL-KDD não possui um atributo binário indicando diretamente se o registro representa um comportamento normal ou malicioso de rede como no UNSW-NB15. Ao invés disso, o seu atributo alvo "*label*" indica a classe do registro. Essa diferença na representação dos registros foi tratada na seção 4.2, durante o pré-processamento dos dados.

4.1.4 Atributos Comuns entre os *Datasets*

Para verificar a viabilidade de construir uma Federação de Máquinas de Aprendizado treinadas sobre o UNSW-NB15 e o NSL-KDD, um passo importante é a comparação dos conjuntos de atributos dos dois *datasets*. A identificação de um conjunto de atributos em comum entre ambos pode facilitar a construção de um *dataset* que combine os seus dados, a fim de treinar Máquinas de Aprendizado capazes de identificar ataques de rede em mais de um *dataset*.

A Tabela 8 descreve os atributos em comum usados para representar cada registro do UNSW-NB15 e do NSL-KDD. Embora ambos os *datasets* possuam uma quantidade de atributos próxima, o conjunto de atributos equivalentes identificado entre o UNSW-NB15 e o NSL-KDD é pequeno, em comparação ao total de *features*.

Tabela 8 – Lista de *Features* Comuns entre os *Datasets* UNSW-NB15 e NSL-KDD.

Id	UNSW-NB15	NSL-KDD	Descrição
1	dur	duration	Duração da conexão em segundos.
2	proto	protocol_type	Tipo de protocolo utilizado (TCP, UDP, etc.).
3	service	service	Serviço de destino da conexão (HTTP, FTP, etc.).
4	sbytes	src_bytes	Bytes enviados da origem para o destino.
5	dbytes	dst_bytes	Bytes recebidos pelo destino.
6	state	flag	Estado da conexão.
7	is_sm_ips_ports	land	Se a conexão é entre hosts com mesmo IP e porta.
8	ct_srv_dst	srv_count	Número de conexões para o mesmo serviço e IP de destino

Esse cenário evidencia que a existência de atributos em comum entre os *datasets* é apenas o primeiro passo para a viabilização de uma federação. Contudo, nem todos os atributos compartilhados são necessariamente relevantes para a detecção de intrusões. Assim, torna-se essencial investigar se esses conjuntos de atributos coincidem com aqueles que apresentam maior importância na classificação nos *datasets* UNSW-NB15 e NSL-KDD. Caso os atributos comuns estejam entre os mais relevantes, a construção de modelos federados tende a preservar a capacidade de detecção. Por outro lado, se houver divergência significativa entre os atributos essenciais de cada conjunto, a federação pode resultar em perda de desempenho. As próximas seções aprofundam essa análise por meio da aplicação de técnicas de pré-processamento e redução de dimensionalidade.

4.2 Pré-Processamento dos Dados

Antes de aplicar as técnicas de redução de dimensionalidade e avaliar a compatibilidade dos *datasets*, é preciso aplicar algumas técnicas de pré-processamento, a fim de corrigir algumas de suas incongruências e preparar os dados para serem processados pelas máquinas de aprendizado de forma mais eficiente (UMAR et al., 2024). Dentre essas incongruências, destacam-se a presença de colunas vazias e a diferença de escala entre os valores dos atributos.

É preciso garantir que não há colunas vazias nos *datasets*, pois essas estruturas não fornecem nenhuma informação útil ao processo de modelagem e podem comprometer a integridade das análises. A presença de colunas compostas apenas por valores nulos ("*Not a number*", ou NaN) pode causar falhas em algoritmos de aprendizado de máquina, distorcer estatísticas descritivas e afetar negativamente o desempenho dos modelos, além de indicar possíveis falhas no pré-processamento dos dados.

Após verificar os arquivos com os conjuntos de treino e teste dos *datasets* UNSW-NB15 e NSL-KDD, não se encontram colunas totalmente vazias em nenhum dos dois conjuntos. Essa verificação foi realizada por meio do seguinte trecho de código:

```
# mesmo processo feito com os arquivos csv do NSL-KDD
treino = pd.read_csv(r"UNSW_NB15\UNSW_NB15_training-set.csv")
teste = pd.read_csv(r"UNSW_NB15\UNSW_NB15_testing-set.csv")
```

```
# ...
colunas_vazias_treino = treino.columns[treino.isnull().all()].tolist()
colunas_vazias_teste = teste.columns[teste.isnull().all()].tolist()

print("\nColunas VAZIAS:")
print("Treino:")
print(colunas_vazias_treino)
print("Teste:")
print(colunas_vazias_teste)

if(len(colunas_vazias_treino) > 0):
    treino = treino.drop(colunas_vazias_treino, axis=1, inplace=False)
if(len(colunas_vazias_teste) > 0):
    teste = teste.drop(colunas_vazias_teste, axis=1, inplace=False)
```

A princípio, considera-se a avaliação da capacidade dos modelos de aprendizado de máquina de classificar os registros apenas em duas classes (comportamento de rede normal e tráfego malicioso), a fim de explorar uma análise mais simples e direta do problema de identificação de ataques de rede. Nesse contexto, é feita uma alteração no *dataset* NSL-KDD. O atributo alvo "*label*", nesse *dataset*, descreve o tipo de comportamento de rede (que pode ser normal, U2R, R2L, etc.), enquanto no UNSW-NB15 ele representa um valor binário, indicando se o registro corresponde a uma conexão normal ou nociva. A fim de permitir essa análise inicial, adiciona-se a coluna "*target*" ao NSL-KDD, com o valor normal nos registros em que o tipo de conexão já possui o valor normal na coluna *label*, e *attack* nos demais registros.

```
    #...
    # Experimento inicial: classificacao binaria dos registros

    # Criar coluna 'target' com base na coluna 'label'
    treino['target'] = treino['label'].apply(lambda x: 'normal' if x == 'normal' else 'attack')
    teste['target'] = teste['label'].apply(lambda x: 'normal' if x == 'normal' else 'attack')
```

A normalização é uma técnica de transformação de dados usada para transformar valores numéricos em diferentes escalas para uma escala comum, sem distorcer as diferenças entre os valores. A normalização dos dados busca mapear os dados em um *dataset* para as mesmas faixas de valores. Esse processo acelera a etapa de treinamento do modelo e é especialmente útil para algoritmos de classificação que envolvem redes neurais ou medições de distância, como classificação por vizinho mais próximo (*nearest-neighbour classification*) e algoritmos de agrupamento (*clustering*). Existem vários métodos de normalização, sendo *Min-Max Normalization*, *Z-Score Normalization* e *Decimal Scaling* os métodos mais usados (UMAR et al., 2024).

$$x_{\text{new}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{4.1}$$

$$x_{\text{new}} = \frac{x - u}{s} \tag{4.2}$$

Os métodos de normalização aplicados neste trabalho são o *Min-Max Normalization* e o *Z-Score Normalization*. A técnica de normalização *Min-Max* utiliza a eq. (4.1), sendo i) x_{new}

- valor do atributo normalizado; ii) x - valor do atributo antes da normalização; iii) x_{max} - valor máximo do atributo no *dataset* e iv) x_{min} - valor mínimo do atributo no *dataset*. Já a normalização *Z-Score* utiliza a eq. (4.2), sendo i) x_{new} - valor do atributo normalizado; ii) x - valor do atributo antes da normalização; iii) u - média do atributo no *dataset* e iv) s - desvio padrão do atributo no *dataset*. A implementação desses métodos utiliza as classes *MinMaxScaler* e *StandardScaler*, da biblioteca *sci-kit learn* (PEDREGOSA et al., 2011).

Para avaliar a qualidade dos *datasets* UNSW-NB15 e NSL-KDD, bem como identificar o efeito que o pré-processamento de dados teve em cada *dataset*, é feita uma comparação entre o desempenho dos modelos de classificação *Random Forest* (RF) e *Multi Layer Perceptron* (MLP) sobre os *datasets* antes de aplicar qualquer técnica de normalização, após a aplicação da técnica *Min-Max* e após a aplicação da técnica *Z-Score*.

A escolha dos algoritmos de Aprendizado de Máquina avaliados se fundamenta em sua recorrência na literatura especializada em IDS (SIRISHA et al., 2021; KOCHER; KUMAR, 2021), além de suas características distintas e complementares: o MLP é indicado para modelagem de padrões complexos e não lineares, enquanto o RF combina robustez com interpretabilidade e oferece recursos adicionais de seleção de atributos via MDI.

Já a escolha das técnicas de normalização *Min-Max* e *Z-Score* fundamenta-se em suas propriedades complementares para tratar dados de tráfego de rede, que frequentemente apresentam atributos em escalas heterogêneas, como contagem de pacotes, duração de conexões e volume de bytes transmitidos (UMAR et al., 2024). Além disso, a aplicação e comparação dessas duas técnicas de normalização se baseia na implementação tanto do *Z-Score* quanto do *Min-Max* na literatura recente de pré-processamento para Aprendizado de Máquina (THANH, 2018; TUFAN; TEZCAN; ACARTURK, 2021). Assim, a aplicação e comparação dessas duas técnicas tornam possível avaliar de forma mais robusta o impacto da normalização sobre o desempenho dos modelos de ML considerados neste trabalho.

Embora a análise inicial tenha focado na classificação binária, uma reavaliação do escopo da pesquisa indicou que a identificação do tipo de ataque é igualmente relevante para sistemas IDS. Em vista disso, o desempenho dos modelos na classificação dos registros em apenas duas classes (ataque de rede e comportamento normal) é desconsiderado para as etapas posteriores, bem como a criação da coluna "*target*" do NSL-KDD. Apesar disso, os resultados obtidos nesses primeiros experimentos estão disponíveis no Apêndice F. Todos os códigos-fonte e scripts utilizados nas etapas descritas ao longo deste trabalho estão disponíveis publicamente em repositório no *GitHub* (ABREU, 2025).

Ao invés de observar a capacidade dos modelos em classificar os registros de forma binária, como foi feito nos primeiros experimentos, observa-se a capacidade dos modelos em classificar os registros de cada um dos *datasets* dentro dos seus próprios conjuntos de classes de ataque (descritos detalhadamente nas seções 4.1.1 e 4.1.2).

Nas próximas seções, avalia-se os valores de Acurácia, Precisão, *Recall*, *F1-Score*, *AUC-ROC Score* e Tempo de Execução (em segundos) alcançados pelas máquinas de aprendizado RF e MLP sobre os *datasets* antes e depois da normalização, a fim de mensurar o impacto das

técnicas *Min-Max* e *Z-Score* sobre a qualidade dos *datasets* para os modelos avaliados. O código executado para treinar e testar as máquinas de aprendizado no UNSW-NB15 está disponível no Apêndice D. Os resultados dos testes feitos com os modelos RF e MLP aparecem listados nas Tabelas 9 a 14 e 18 a 23, respectivamente. Os testes sobre o *dataset* UNSW-NB15 foram executados em um computador com as seguintes especificações: Sistema operacional *Windows* 11 *Home* 64 bits, Processador AMD *Ryzen* 5 3500U com 2,10 GHz, 8 GB de RAM, usando a versão 3.11.4 do *Python*. Já os testes sobre o NSL-KDD são originalmente da monografia do dicente Miguel Borges de Rezende Costa, que possui esse *dataset* como objeto de estudo.

4.2.1 *Random Forest* Antes e Depois da Normalização

Em média, nota-se que o classificador RF apresenta um desempenho semelhante sobre os *datasets* antes e depois das técnicas de normalização serem aplicadas. Apesar do desempenho do RF sobre o UNSW-NB15 ser levemente superior ao observado no NSL-KDD, na maioria dos experimentos, o tempo gasto pelo modelo para ser treinado e testado no UNSW-NB15 foi significativamente maior do que no NSL-KDD, o que pode ser explicado pela maior quantidade de registros presentes no primeiro *dataset*. Para a técnica *Random Forest*, os experimentos envolveram o treinamento e teste de diversos classificadores, variando o valor do parâmetro "*n_estimators*", que representa o número de árvores de decisão usadas para formar o modelo.

4.2.1.1 *Random Forest* Antes da Normalização

A Tabela 9 apresenta os resultados alcançados pelo classificador *Random Forest* (RF) sobre o *datasets* UNSW-NB15 antes de qualquer técnica de normalização ser aplicada.

Tabela 9 – Desempenho Antes da Normalização no UNSW-NB15.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo(s)**
1.1	10	0,743125	0,827959	0,743125	0,770632	0,950671	6,547040
1.2	25	0,751543	0,831838	0,751543	0,775681	0,960186	15,423853
1.3	50	0,752089	0,837015	0,752089	0,775600	0,962603	31,556545
1.4	75	0,754324	0,839771	0,754324	0,777815	0,963657	49,510802
1.5	100	0,754348	0,841144	0,754348	0,778033	0,964211	62,783272
1.6	125	0,754470	0,841006	0,754470	0,778252	0,964367	77,988338
1.7	150	0,755077	0,842156	0,755077	0,778912	0,964727	93,246143
Média	**-**	**0,752139**	**0,837270**	**0,752139**	**0,776132**	**0,961489**	**48,722285**

O experimento 1.7 (RF sobre o UNSW-NB15, usando 150 árvores), entre esse grupo de experimentos, possui os maiores valores em todas as métricas, apesar de também possuir o maior tempo de execução (aproximadamente 93 segundos). Os outros dois experimentos com as melhores métricas nessa etapa são o 1.6 e o 1.5, que ambos possuem valores com menos de 1% de diferença dos obtidos no melhor experimento e demandam um tempo de execução menor.

A Tabela 10 apresenta os resultados obtidos pelo RF sobre o *dataset* NSL-KDD, antes da aplicação de qualquer técnica de normalização.

Tabela 10 – Desempenho Antes da Normalização no NSL-KDD.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
2.1	10	0,753992	0,814923	0,753992	0,714875	0,892047	3,595366
2.2	25	0,750177	0,816268	0,750177	0,710827	0,916214	7,731620
2.3	50	0,750000	0,816345	0,750000	0,710235	0,924809	10,735732
2.4	75	0,752750	0,818767	0,752750	0,712726	0,926933	12,609293
2.5	100	0,751020	0,814428	0,751020	0,710103	0,932688	14,531148
2.6	125	0,747738	0,814669	0,747738	0,705959	0,937945	18,853442
2.7	150	0,745963	0,811960	0,745963	0,703436	0,939365	22,498096
Média	**-**	**0,750234**	**0,815337**	**0,750234**	**0,709451**	**0,924714**	**12,650100**

Sobre o NSL-KDD antes da normalização, o experimento 2.1 (modelo com 10 árvores de decisão) obteve os melhores valores de Acurácia, *Recall* (ambos foram de 75,3992%) e *F1-Score* (71,4875%), além de ter demandado o menor tempo de treinamento do conjunto e possuir uma Precisão próxima da média do conjunto. O AUC-ROC desse experimento, por outro lado, foi aproximadamente 3% pior que a média. Os experimentos que obtiveram a melhor Precisão e AUC-ROC foram o 2.4 (75 árvores) e o 2.7 (100 árvores), respectivamente.

No geral, as médias de Acurácia, Precisão e *Recall* foram muito próximas das alcançadas sobre o *dataset* UNSW-NB15. Já as métricas *F1-Score* e AUC-ROC são, em média, inferiores às obtidas sobre o primeiro *dataset*. Essa diferença indica que, embora o modelo tenha mantido uma boa capacidade de classificação geral no NSL-KDD, seu equilíbrio entre Precisão e *F1-Score* e sua capacidade de discriminar entre classes positivas e negativas (AUC-ROC) foram ligeiramente reduzidos.

4.2.1.2 *Random Forest* Após Normalização *Min-Max*

Nas Tabelas 11 e 12 apresentam os resultados obtidos pelo classificador RF sobre os *datasets* UNSW-NB15 e NSL-KDD, respectivamente. Ao contrário do que se percebe nos experimentos anteriores, é possível notar que os experimentos no NSL-KDD também se destacaram positivamente entre os feitos sobre os *datasets* normalizados usando a técnica *Min-Max*. O experimento 4.3 (RF composto por 50 árvores, sobre o NSL-KDD) obteve os maiores valores de acurácia e *recall* (aproximadamente 75,7% em ambas as métricas), enquanto o experimento 3.7 (RF com 150 árvores sobre o *dataset* UNSW-NB15) se destacou com os maiores valores de *F1-Score* e *AUC-ROC* (aproximadamente 77,7% e 97,2%, respectivamente). Além disso, o experimento 3.5 (RF com 100 árvores sobre o UNSW-NB15) apresentou o maior valor de precisão desse grupo, com aproximadamente 84,38%. Apesar disso, o desempenho geral do classificador não foi muito diferente do observado na subseção 4.2.1.1, apresentando apenas o AUC-ROC como estatística melhorada para os dois *datasets* (em todas as outras métricas, apenas um dos *datasets* melhorou com a normalização *Min-Max*).

Tabela 11 – Desempenho Após Normalização *Min-Max* no UNSW-NB15.

Experimento	Árvores	Acurácia	Precisão	Recall	*F1-score*	AUC-ROC	Tempo(s)
3.1	10	0,742263	0,822511	0,742263	0,766430	0,960515	5,234350
3.2	25	0,750814	0,835897	0,750818	0,773880	0,968015	13,237884
3.3	50	0,752174	0,840277	0,752174	0,774550	0,970171	26,622633
3.4	75	0,754045	0,842944	0,754045	0,776344	0,971329	40,448369
3.5	100	0,754288	0,843887	0,754288	0,776809	0,971706	53,863185
3.6	125	0,754543	0,842923	0,754543	0,777175	0,971866	68,354344
3.7	150	0,754543	0,843633	0,754543	0,777408	0,972006	82,229213
Média	**-**	**0,745810**	**0,838725**	**0,744667**	**0,774942**	**0,969658**	**41,141711**

Tabela 12 – Desempenho Após Normalização *Min-Max* no NSL-KDD.

Experimento	Árvores	Acurácia	Precisão	*Recall*	*F1-score*	AUC-ROC	Tempo (s)
4.1	10	0,753549	0,802089	0,753549	0,705902	0,885846	1,791345
4.2	25	0,743657	0,804704	0,743657	0,696039	0,903961	4,008810
4.3	50	0,756875	0,819376	0,756875	0,708145	0,912739	8,394212
4.4	75	0,750488	0,813309	0,750488	0,702167	0,915226	13,347669
4.5	100	0,752351	0,814611	0,752351	0,704169	0,921524	17,567128
4.6	125	0,749468	0,814594	0,749468	0,702172	0,928455	20,767237
4.7	150	0,749689	0,813381	0,749689	0,702502	0,928277	25,025297
Média	**-**	**0,750868**	**0,811437**	**0,750011**	**0,702156**	**0,913147**	**13,286528**

4.2.1.3 *Random Forest* Após Normalização *Z-Score*

Os resultados obtidos sobre os *datasets* após a normalização *Z-Score* estão dispostos nas Tabelas 13 e 14. Assim como foi observado antes das técnicas de normalização, os experimentos que se destacaram após a normalização *Z-Score* foram aqueles realizados sobre o *dataset* UNSW-NB15. O experimento com os maiores valores de acurácia, *recall* e *AUC-ROC* foi o 5.7 (RF com 150 árvores sobre o UNSW-NB15), enquanto o experimento 5.5 (RF com 100 árvores sobre o NSL-KDD) se destacou por apresentar os maiores valores de precisão e *f1-score*. Em comparação aos resultados obtidos antes da normalização, os resultados após a normalização *Z-Score* apresentaram maior precisão e *F1-Score* em ambos os *datasets*.

Tabela 13 – Desempenho Após Normalização *Z-Score* no UNSW-NB15.

Experimento	Árvores	Acurácia	Precisão	*Recall*	*F1-score*	AUC-ROC	Tempo (s)
5.1	10	0,742008	0,833112	0,742008	0,770578	0,948778	6,071144
5.2	25	0,752891	0,837842	0,752891	0,777615	0,959279	14,258145
5.3	50	0,751761	0,836956	0,751761	0,776448	0,962317	28,857221
5.4	75	0,753875	0,839430	0,753875	0,778078	0,963681	42,632549
5.5	100	0,754664	0,841833	0,754664	0,778658	0,963987	58,709273
5.6	125	0,754579	0,841534	0,754579	0,778538	0,964140	72,183542
5.7	150	0,754676	0,841235	0,754676	0,778524	0,964368	89,188376
Média	**-**	**0,744765**	**0,838277**	**0,744765**	**0,777205**	**0,960650**	**44,271893**

Tabela 14 – Desempenho Após Normalização *Z-Score* no NSL-KDD.

Experimento	Árvores	Acurácia	Precisão	*Recall*	*F1-score*	AUC-ROC	Tempo (s)
6.1	10	0,754213	0,815118	0,754214	0,715118	0,891954	1,625645
6.2	25	0,750976	0,816897	0,750976	0,711648	0,916551	3,863984
6.3	50	0,750577	0,816735	0,750577	0,710780	0,925182	7,785537
6.4	75	0,753903	0,819421	0,753903	0,713878	0,927282	11,659250
6.5	100	0,752262	0,816911	0,752262	0,711388	0,932801	15,354240
6.6	125	0,748137	0,814859	0,748137	0,706321	0,937996	19,780471
6.7	150	0,746496	0,812307	0,746496	0,704072	0,939392	23,530262
Média	**-**	**0,750366**	**0,816178**	**0,750366**	**0,710744**	**0,924451**	**11,657056**

4.2.1.4 Comparação dos Resultados do Classificador RF Após Normalização

Observa-se que, apesar de algumas melhorias nas métricas obtidas nos experimentos após as técnicas de normalização, em média, os valores de precisão e acurácia das máquinas de aprendizado testadas ainda se mantêm abaixo de 90% em todos os casos do UNSW-NB15, e com o *dataset* NSL-KDD, a acurácia continua abaixo de 80% e a precisão abaixo de 70%. Isso pode ser explicado pelo fato de esse tipo de técnica de ML ser baseado em árvores de decisão e, portanto, não ser tão sensível à escala dos atributos do conjunto de dados de entrada quanto outros métodos, como MLP, por exemplo (FILHO, 2025).

A Tabela 15 apresenta os melhores resultados encontrados por experimento nessa etapa, comparando os métodos de normalização. Percebe-se que, apesar do maior tempo de execução necessário, o uso de mais de 50 árvores de decisão para compor o classificador RF resultou nos melhores resultados para a maioria das métricas.

Tabela 15 – Resumo dos Melhores Resultados Obtidos com RF Após Normalização

Métrica	**Melhor Experimento**	**Valor**
Acurácia	4.3 (*Min-Max*, 50 árvores)	0,756875
Precisão	3.5 (*Min-Max*, 100 árvores)	0,843887
Recall	4.3 (*Min-Max*, 50 árvores)	0,756875
F1-Score	5.5 (*Z-Score*, 100 árvores)	0,778658
AUC-ROC	3.7 (*Min-Max*, 150 árvores)	0,972006
Menor Tempo(s)	6.1 (*Z-Score*, 10 árvores)	1,625645
Maior Tempo (s)	1.7 (Sem normalização, 150 árvores)	93,246143

As Tabelas 16 e 17 apresentam as médias dos resultados dos experimentos feitos usando RF, antes e após a normalização dos *datasets*, com os melhores valores destacados em negrito. A princípio, percebe-se que a normalização teve efeitos variados no *dataset* UNSW-NB15, com a técnica *Min-Max* aumentando a precisão e o AUC-ROC e diminuindo o tempo de execução, e a técnica *Z-Score* proporcionando o melhor F1-Score, apesar das técnicas de normalização terem diminuído a acurácia e o *Recall*. No *dataset* NSL-KDD, por outro lado, os valores obtidos após a normalização são todos superiores aos resultados encontrados antes da normalização, com o destaque para a técnica *Z-Score*, que proporcionou melhora em todas as métricas.

Tabela 16 – Média das Métricas Encontradas nos Experimentos (UNSW-NB15)

Métrica	**Antes**	***Min-Max***	***Z-Score***
Acurácia	**0,752139**	0,745810	0,744765
Precisão	0,837270	**0,838725**	0,838277
Recall	**0,752139**	0,744667	0,744765
F1-Score	0,776132	0,774942	**0,777205**
AUC-ROC	0,961489	**0,969658**	0,960650
Tempo(s)	48,722285	**41,141711**	44,271893

Tabela 17 – Média das Métricas Encontradas nos Experimentos (NSL-KDD)

Métrica	**Antes**	***Min-Max***	***Z-Score***
Acurácia	0,750234	**0,750868**	0,750366
Precisão	0,815337	0,811437	**0,816178**
Recall	0,750234	0,750011	**0,750366**
F1-Score	0,709451	0,702156	**0,710744**
AUC-ROC	0,924714	0,913147	**0,939392**
Tempo (s)	12,650100	13,286528	**11,657056**

4.2.2 MLP Antes e Depois da Normalização

Já para avaliar os efeitos da normalização sobre o MLP, os experimentos são feitos variando a quantidade de camadas e o número de neurônios em cada camada oculta da rede neural MLP, representados pelo parâmetro "*hidden_layer_sizes*". Apesar da alteração na organização das camadas ocultas, a camada de entrada ainda possui uma quantidade de neurônios equivalente ao número de colunas do *dataset* e a camada de saída possui um neurônio para cada classe do *dataset*, com a função de ativação "*sigmoid*" (padrão da biblioteca *sci-kit learn*)

4.2.2.1 *Multilayer Percepton* Antes da Normalização

As Tabelas 18 e 19 apresentam os resultados obtidos pelo modelo classificador MLP. Em comparação ao RF, esse modelo obteve resultados bastante inferiores nos *datasets* antes da normalização, principalmente no UNSW-NB15, no qual o modelo alcançou médias abaixo de 50% para acurácia, precisão e *recall*, e abaixo de 30% para *F1-Score*, sugerindo uma grande dificuldade em generalizar padrões relevantes nos dados brutos. A métrica AUC-ROC também manteve-se próxima de 50% para o *dataset* UNSW-NB15, o que indica que o modelo é incapaz de distinguir eficientemente entre as diferentes classes de ataque sobre o *dataset* não normalizado. Os resultados obtidos no NSL-KDD antes da normalização também são limitados, embora razoáveis quando comparados aos encontrados no UNSW-NB15. As médias de acurácia, precisão, *recall* e *F1-Score* próximas de 70% evidenciam a presença de falsos positivos e falsos negativos na classificação feita pelo modelo.

Tabela 18 – Desempenho Antes da Normalização no UNSW-NB15.

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo(s)**
7.1	(16, 8)	0,449473	0,266574	0,449473	0,279083	0,500828	23,046435
7.2	(32, 16)	0,449388	0,410597	0,449388	0,278676	0,506241	74,463093
7.3	(64, 32)	0,449400	0,512228	0,449400	0,278724	0,515533	92,016244
7.4	(10, 10, 20)	0,451003	0,479663	0,451003	0,283204	0,505480	72,835578
7.5	(20, 40, 20)	0,449546	0,391216	0,449546	0,279114	0,501922	35,133903
7.6	(40, 60, 40)	0,453809	0,316071	0,453809	0,289510	0,516315	193,361245
7.7	(60, 40, 60)	0,449534	0,383778	0,449534	0,279134	0,515613	163,169312
7.8	(60, 80, 60)	0,581208	0,601512	0,581208	0,473366	0,720994	185,921100
7.9	(80, 100, 80)	0,452521	0,422912	0,452521	0,286943	0,523889	213,542104
7.10	(100, 120, 100)	0,452631	0,309476	0,452631	0,287410	0,524783	240,982046
7.11	(128, 128, 64, 64)	0,449570	0,390108	0,449570	0,279304	0,502055	195,325478
7.12	(100, 80, 60, 40, 20)	0,449400	0,449400	0,449400	0,278681	0,500650	158,775967
Média	**-**	**0,463537**	**0,414353**	**0,463537**	**0,298859**	**0,530137**	**141,126060**

Tabela 19 – Desempenho Antes da Normalização no NSL-KDD

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
8.1	(16, 8)	0,738556	0,686934	0,738556	0,689659	0,875193	45,140759
8.2	(32, 16)	0,749601	0,710703	0,749601	0,700826	0,887066	59,059358
8.3	(64, 32)	0,728353	0,681346	0,728353	0,680911	0,815156	54,052127
8.4	(10, 10, 20)	0,747693	0,800105	0,747693	0,710175	0,924937	66,414326
8.5	(20, 40, 20)	0,744544	0,686008	0,744544	0,694611	0,908603	41,631726
8.6	(40, 60, 40)	0,739532	0,707515	0,739532	0,691549	0,820113	59,282957
8.7	(60, 40, 60)	0,706618	0,672460	0,706618	0,659152	0,801102	73,823635
8.8	(60, 80, 60)	0,783978	0,824183	0,783978	0,751463	0,858432	127,867878
8.9	(80, 100, 80)	0,749379	0,774036	0,749379	0,698538	0,799004	149,985320
8.10	(100, 120, 100)	0,741794	0,702415	0,741794	0,691752	0,802640	160,436199
8.11	(128, 128, 64, 64)	0,721655	0,684037	0,721655	0,674142	0,804742	467,912291
8.12	(100, 80, 60, 40, 20)	0,722321	0,730406	0,722321	0,673350	0,864178	80,536993
Média	**-**	**0,738503**	**0,717099**	**0,738503**	**0,692879**	**0,846527**	**115,348566**

4.2.2.2 *Multilayer Percepton* Após Normalização *Min-Max*

Nas Tabelas 20 e 21 estão dispostos os resultados obtidos pelo MLP após a normalização dos *datasets* por meio da técnica *Min-Max*. Ao contrário do que observa-se nas *Random Forests*, a aplicação de técnicas de normalização sobre os *datasets* representa uma notável melhoria nos resultados da rede neural MLP em ambos os conjuntos de dados, com destaque para o *dataset* UNSW-NB15. Apesar disso, o tempo necessário para a execução das máquinas de aprendizado nessa etapa foi muito maior do que para os *datasets* sem normalização.

Tabela 20 – Desempenho Após Normalização *Min-Max* no UNSW-NB15.

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
9.1	(16, 8)	0,720267	0,822416	0,720267	0,732673	0,965412	178,663420
9.2	(32, 16)	0,719587	0,812016	0,719587	0,732706	0,969865	352,323888
9.3	(64, 32)	0,758247	0,799167	0,758247	0,762238	0,964025	564,970213
9.4	(10, 20, 10)	0,744279	0,805597	0,744279	0,747064	0,968873	181,057046
9.5	(20, 40, 20)	0,723728	0,786433	0,723728	0,736658	0,950908	268,654675
9.6	(40, 60, 40)	0,742773	0,812393	0,742773	0,753646	0,965490	779,745810
9.7	(60, 40, 60)	0,737878	0,806627	0,737878	0,753953	0,968901	810,592807
9.8	(60, 80, 60)	0,759292	0,802477	0,759292	0,767684	0,963696	919,641296
9.9	(80, 100, 80)	0,761004	0,793018	0,761004	0,767697	0,963187	1083,070221
9.10	(100, 120, 100)	0,739834	0,790736	0,739834	0,757332	0,954332	1330,220600
9.11	(128, 128, 64, 64)	0,715639	0,789661	0,715639	0,737141	0,943208	1645,153229
9.12	(100, 80, 60, 40, 20)	0,755599	0,802024	0,755599	0,765017	0,964248	1296,435136
Média	**-**	**0,740265**	**0,801102**	**0,740265**	**0,751502**	**0,962457**	**813,311290**

Tabela 21 – Desempenho Após Normalização *Min-Max* no NSL-KDD

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
10.1	(16, 8)	0,764594	0,792877	0,764594	0,720107	0,927022	80,594
10.2	(32, 16)	0,785974	0,770794	0,785974	0,755071	0,936040	140,498
10.3	(64, 32)	0,773998	0,816437	0,773998	0,737861	0,933921	185,143
10.4	(10, 20, 10)	0,746363	0,783382	0,746363	0,700144	0,903158	71,599
10.5	(20, 40, 20)	0,771380	0,811975	0,771380	0,729146	0,939799	144,299
10.6	(40, 60, 40)	0,743302	0,782558	0,743302	0,706905	0,937764	334,410
10.7	(60, 40, 60)	0,779853	0,814577	0,779853	0,744005	0,910242	231,812
10.8	(60, 80, 60)	0,773598	0,803722	0,773598	0,735806	0,928189	256,311
10.9	(80, 100, 80)	0,779808	0,817701	0,779808	0,740256	0,945974	280,270
10.10	(100, 120, 100)	0,760912	0,815684	0,760912	0,720542	0,936234	254,798
10.11	(100, 80, 60, 40, 20)	0,758472	0,768412	0,758472	0,721461	0,938666	311,459
10.12	(128, 128, 64, 64)	0,770804	0,741271	0,770804	0,732928	0,928393	336,385
Média	**-**	**0,765093**	**0,791947**	**0,765093**	**0,727801**	**0,930452**	**220,642667**

4.2.2.3 *Multilayer Percepton* Após Normalização *Z-Score*

As Tabelas 22 e 23 apresentam os resultados obtidos após a técnica de normalização *Z-Score* sobre os *datasets* UNSW-NB15 e NSL-KDD, respectivamente. Assim como na normalização *Min-Max*, observa-se um aumento significativo nas médias de acurácia, precisão, *recall*, *F1-Score* e AUC-ROC, além de um maior tempo de execução também.

Tabela 22 – Desempenho Após Normalização *Z-Score* no UNSW-NB15

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
11.1	(16, 8)	0,733761	0,812814	0,733761	0,744033	0,967719	166,443
11.2	(32, 16)	0,735814	0,805855	0,735814	0,746746	0,967103	348,635
11.3	(64, 32)	0,760531	0,789546	0,760531	0,764107	0,966378	564,071
11.4	(10, 20, 10)	0,762535	0,814165	0,762535	0,762751	0,970229	209,073
11.5	(20, 40, 20)	0,758745	0,803391	0,758745	0,762600	0,966651	271,437
11.6	(40, 60, 40)	0,766506	0,811620	0,766506	0,772180	0,969136	782,089
11.7	(60, 40, 60)	0,738000	0,798527	0,738000	0,750648	0,960638	818,350
11.8	(60, 80, 60)	0,755818	0,806943	0,755818	0,766905	0,963620	914,773
11.9	(80, 100, 80)	0,728878	0,791119	0,728878	0,745409	0,954082	1056,249
11.10	(100, 120, 100)	0,734332	0,785882	0,734332	0,751004	0,946931	1304,383
11.11	(100, 80, 60, 40, 20)	0,731575	0,786381	0,731575	0,748217	0,953878	1288,136
11.12	(128, 128, 64, 64)	0,706688	0,788116	0,706688	0,731051	0,941839	1608,778
Média	**-**	**0,743005**	**0,800143**	**0,743005**	**0,753020**	**0,961781**	**777,701407**

Tabela 23 – Desempenho Após a Normalização *Z-Score* no NSL-KDD.

Experimento	**Camadas**	**Acurácia**	**Precisão**	**Recall**	***F1-score***	**AUC-ROC**	**Tempo(s)**
12.1	(16, 8)	0,758339	0,744337	0,758339	0,711439	0,894265	63,110778
12.2	(32, 16)	0,763707	0,786349	0,763707	0,722808	0,930288	135,978719
12.3	(64, 32)	0,768364	0,798253	0,768364	0,723785	0,932227	167,694143
12.4	(10, 20, 10)	0,775373	0,819236	0,775373	0,729747	0,862040	66,838931
12.5	(20, 40, 20)	0,761932	0,794991	0,761932	0,714378	0,937369	84,978763
12.6	(40, 60, 40)	0,749246	0,790733	0,749246	0,704268	0,937084	183,228182
12.7	(60, 40, 60)	0,775727	0,815634	0,775727	0,746994	0,923117	209,041571
12.8	(60, 80, 60)	0,757408	0,756823	0,757408	0,710805	0,918457	198,942044
12.9	(80, 100, 80)	0,774707	0,804894	0,774707	0,727490	0,907018	332,523676
12.10	(100, 120, 100)	0,753948	0,785872	0,753948	0,706696	0,926072	266,840984
12.11	(128, 128, 64, 64)	0,758295	0,810827	0,758295	0,713241	0,897813	501,220467
12.12	(100, 80, 60, 40, 20)	0,770005	0,774090	0,770005	0,722950	0,886710	387,358716
Média	**-**	**0,763939**	**0,789707**	**0,763939**	**0,719935**	**0,912432**	**222,637207**

4.2.2.4 Comparação dos Resultados do Classificador MLP Após Normalização

A Tabela 24 apresenta os melhores resultados encontrados nos experimentos realizados com o *MultiLayer Perceptron* sobre os *datasets* antes e depois das técnicas de normalização serem aplicadas, junto aos melhores resultados observados no classificador *Random Forests* (detalhados na Tabela 15). Apesar de demandar um tempo muito maior, o classificador MLP, nos melhores casos, obteve resultados superiores aos observados nos melhores testes usando RF.

Tabela 24 – Melhores Resultados do MLP em Comparação a RF

Métrica	**Experimento**	**Valor**	**Configuração**	Melhor resultado RF
Acurácia	9.2	0,888233	(32, 16)	0,756875
Precisão	9.2	0,846484	(32, 16)	0,843887
Recall	9.4	0,990272	(10, 20, 10)	0,756875
F1-Score	9.2	0,905591	(32, 16)	0,778658
AUC-ROC	9.5	0,979631	(20, 40, 20)	0,972006
Menor Tempo (s)	8.1	16,442896	(10, 10, 20)	1,625645
Maior Tempo (s)	9.11	1645,153229	(128, 128, 64, 64)	93,246143

Os melhores resultados do classificador, assim como no RF, são obtidos sobre o *dataset* UNSW-NB15. As Tabelas 25 e 26 apresentam as médias das métricas observadas em cada grupo de experimentos realizados com o classificador MLP, com os melhores valores de média destacados em negrito.

favorecem a ideia de que a normalização *Min-Max*, em média, contribui para um melhor desempenho do classificador MLP sobre os *datasets*, em comparação com a normalização *Z-Score*.

Tabela 25 – Média das Métricas Encontradas nos Experimentos (UNSW-NB15)

Métrica	**Antes**	***Min-Max***	***Z-Score***
Acurácia	0,463537	0,740265	**0,743005**
Precisão	0,414353	**0,801102**	0,800143
Recall	0,463537	0,740265	**0,743005**
F1-Score	0,298859	0,751502	**0,753020**
AUC-ROC	0,530137	**0,962457**	0,961781
Tempo(s)	**41,126060**	813,311290	777,701407

Tabela 26 – Média das Métricas Encontradas nos Experimentos (NSL-KDD)

Métrica	**Antes**	***Min-Max***	***Z-Score***
Acurácia	0,765126	**0,791010**	0,780096
Precisão	0,658330	**0,685016**	0,668126
Recall	0,950323	0,973666	**0,974978**
F1-Score	0,777284	**0,800468**	0,792982
AUC-ROC	0,836858	**0,948077**	0,930640
Tempo (s)	**106,419060**	248,647143	199,553726

Percebe-se que o MLP, em média, alcança resultados melhores no *dataset* NSL-KDD após a normalização *Min-Max*, em comparação à técnica *Z-Score*. No *dataset* UNSW-NB15, por outro lado, a diferença entre os resultados é mais sutil. Além dos valores serem mais próximos, a normalização *Min-Max* apresenta o menor tempo de execução (comparado ao *Z-Score*), melhor AUC-ROC e melhor Precisão, enquanto a normalização *Z-Score* apresenta os maiores valores de Acurácia, *Recall* e *F1-Score*.

Apesar dessa diferença no desempenho das Máquinas de Aprendizado após as técnicas *Min-Max* e *Z-Score* sobre os *datasets*, considera-se que a diferença observada nos resultados não é significativa o suficiente para justificar a adoção de apenas um método de normalização nas etapas seguintes. Dessa forma, considera-se os *datasets* normalizados tanto por meio da técnica *Min-Max* quanto por meio do *Z-Score* nos testes descritos nas próximas seções.

4.3 Redução de Dimensionalidade

Vale destacar que, até o momento, os *datasets* UNSW-NB15 e NSL-KDD não tiveram as suas quantidades de atributos (42 e 41, respectivamente) alteradas após a etapa inicial de limpeza e normalização dos dados, tendo em vista que o *dataset* não apresentou nenhuma coluna que precisasse ser removida (colunas compostas apenas por valores nulos). A partir da aplicação das técnicas de redução de dimensionalidade, busca-se uma redução no tamanho do conjunto de atributos dos *datasets*.

Como foi explicado na seção 2.3, é possível reduzir o número de atributos preditivos dos *datasets* por meio da *Feature Extraction* (FE), usando técnicas como a Análise de Componentes Principais (*Principal Component Analysis* - PCA), ou por meio da *Feature Selection* (FS), selecionando atributos pela avaliação da Redução Média de Impureza (*Mean Decrease in Impurity* -

MDI). Detalha-se a seguir os experimentos feitos usando as técnicas PCA e MDI para identificar as *features* mais importantes em cada um dos *datasets* UNSW-NB15 e NSL-KDD.

4.3.1 Análise dos Componentes Principais (PCA)

Após a normalização dos dados, usando as técnicas *Min-Max* e *Z-Score*, a técnica PCA é aplicada sobre os *datasets*, com o parâmetro de variância acumulada definido como 0,99, ou seja, nessa etapa são definidos o menor número de componentes necessários para alcançar 99% da variância encontrada em cada *dataset* original, a partir das suas *features* originais. Vale destacar que os conjuntos de *features* apresentados nas Tabelas não incluem os atributos alvo "*label*" e "*attack_cat*" dos *datasets*.

Cabe destacar que, ao contrário das técnicas de *Feature Selection*, como o MDI, o PCA não permite identificar diretamente os atributos mais importantes do *dataset*, uma vez que sua abordagem consiste em transformar o espaço original em novos componentes lineares que maximizam a variância (HASAN; ABDULAZEEZ, 2021). Assim, a análise de importância das *features* individuais torna-se indireta, sendo possível apenas inferir sua contribuição a partir dos coeficientes de cada atributo nos componentes principais.

A fim de identificar os atributos com maior influência na construção dos componentes principais gerados pelo PCA, realiza-se uma análise baseada na soma dos valores absolutos dos *loadings* (pesos) associados a cada atributo (ou seja, o quanto cada atributo original contribui para cada componente) em todos os componentes retidos. Assim, para estimar a "contribuição total" de cada atributo, é calculada a soma do valor absoluto de seu *loading* em cada componente principal. As Tabelas 27 e 28 apresentam os atributos do UNSW-NB15 (com exceção dos atributos alvos "*label*" e "*attack_cat*"), ordenados conforme sua contribuição nos componentes principais gerados pela análise PCA, após a normalização *Min-Max* e *Z-Score*, respectivamente.

Sobre o UNSW-NB15 normalizado com a técnica *Min-Max*, a técnica PCA resultou em uma redução de 42 *features* para 17 componentes, com uma variância total explicada de 99,0758%.

Tabela 27 – Contribuições das *Features* para o PCA no UNSW-NB15 (*Min-Max*)

Id	Atributo	Contribuição	Id	Atributo	Contribuição
1	swin	0,352426	22	is_sm_ips_ports	0,049526
2	dwin	0,349584	23	sinpkt	0,033170
3	dttl	0,331659	24	dur	0,029144
4	sttl	0,258536	25	tcprtt	0,019993
5	dtcpb	0,224900	26	ackdat	0,016274
6	stcpb	0,224624	27	sload	0,013785
7	dmean	0,119721	28	synack	0,012193
8	ct_srv_dst	0,118363	29	ct_flw_http_mthd	0,005337
9	ct_dst_src_ltm	0,118237	30	is_ftp_login	0,005036
10	ct_srv_src	0,115825	31	ct_ftp_cmd	0,005036
11	ct_state_ttl	0,113805	32	sjit	0,004653
12	service	0,107944	33	dpkts	0,002915
13	ct_src_dport_ltm	0,107194	34	spkts	0,002716
14	rate	0,103451	35	djit	0,002554
15	ct_dst_ltm	0,101862	36	dloss	0,002458
16	ct_src_ltm	0,087922	37	dbytes	0,002278
17	ct_dst_sport_ltm	0,085930	38	dinpkt	0,002080
18	state	0,070044	39	sloss	0,002010
19	proto	0,068810	40	sbytes	0,001816
20	smean	0,058464	41	response_body_len	0,000819
21	dload	0,056723	42	trans_depth	0,000750

Já no UNSW-NB15 normalizado com a técnica *Z-Score*, essa redução resultou em 29 componentes principais, com uma variância explicada de 99,3%.

Tabela 28 – Contribuições das *Features* para o PCA no UNSW-NB15 (*Z-Score*)

Id	Atributo	Contribuição	Id	Atributo	Contribuição
1	dmean	0,134475	22	ct_dst_src_ltm	0,113323
2	sttl	0,132802	23	dloss	0,111399
3	tcprtt	0,129771	24	dpkts	0,109048
4	synack	0,128097	25	dbytes	0,108585
5	ackdat	0,126591	26	smean	0,106177
6	ct_state_ttl	0,126033	27	service	0,105349
7	rate	0,124661	28	dur	0,105088
8	dload	0,122050	29	is_sm_ips_ports	0,102133
9	swin	0,121261	30	spkts	0,100726
10	dwin	0,121078	31	sinpkt	0,099499
11	stcpb	0,120264	32	djit	0,099396
12	dtcpb	0,119940	33	ct_flw_http_mthd	0,099297
13	sload	0,119672	34	proto	0,099134
14	ct_src_ltm	0,118978	35	sloss	0,093552
15	ct_srv_src	0,118863	36	response_body_len	0,092751
16	ct_dst_ltm	0,118104	37	dinpkt	0,092313
17	ct_srv_dst	0,117835	38	sbytes	0,091280
18	dttl	0,116908	39	sjit	0,090618
19	ct_src_dport_ltm	0,116013	40	trans_depth	0,074625
20	state	0,115241	41	is_ftp_login	0,070060
21	ct_dst_sport_ltm	0,113636	42	ct_ftp_cmd	0,070060

As Tabelas 29 e 30 apresentam os atributos do *dataset* NSL-KDD (com exceção do atributo alvo "*label*") após a normalização *Min-Max* e *Z-Score*, respectivamente, ordenados conforme sua importância para a criação dos componentes principais, oriundos do trabalho do discente Miguel Borges de Rezende Costa.

No *dataset* NSL-KDD, a técnica PCA resultou na redução de 41 *features* para 18 componentes principais após a normalização *Min-Max*, com uma variância explicada de 99,0095%.

Tabela 29 – Contribuições das *Features* para o PCA no NSL-KDD (*Min-Max*)

Id	Atributo	Contribuição	Id	Atributo	Contribuição
1	wrong_fragment	43,330170	22	dst_host_rerror_rate	0,056077
2	dst_host_srv_diff_host_rate	41,036910	23	is_guest_login	0,055294
3	dst_host_serror_rate	2,622880	24	dst_host_srv_rerror_rate	0,013213
4	dst_host_diff_srv_rate	2,570316	25	num_failed_logins	0,012420
5	dst_host_same_srv_rate	1,900708	26	land	0,007450
6	diff_srv_rate	1,790536	27	root_shell	0,004962
7	protocol_type	1,453855	28	rerror_rate	0,002730
8	dst_host_srv_count	1,234812	29	hot	0,001910
9	logged_in	0,934961	30	su_attempted	0,001553
10	srv_serror_rate	0,826232	31	num_file_creations	0,000129
11	srv_count	0,460873	32	num_access_files	0,000099
12	flag	0,405267	33	is_host_login	0,000074
13	same_srv_rate	0,275766	34	num_shells	0,000055
14	dst_host_same_src_port_rate	0,195084	35	service	0,000040
15	count	0,194738	36	src_bytes	0,000007
16	duration	0,181225	37	dst_bytes	0,000001
17	dst_host_srv_serror_rate	0,124154	38	num_root	0,000001
18	dst_host_count	0,091992	39	num_compromised	0,000001
19	srv_diff_host_rate	0,083163	40	urgent	0,000001
20	serror_rate	0,069154	41	num_outbound_cmds	$2,05 \times 10^{-27}$
21	srv_rerror_rate	0,061190	42		

No *dataset* NSL-KDD normalizado com a técnica *Z-Score*, por outro lado, foram gerados 30 componentes principais, com uma variância total explicada de 99,0887%.

Tabela 30 – Importância das *Features* para o PCA no NSL-KDD (*Z-Score*)

Id	Atributo	Contribuição	Id	Atributo	Contribuição
1	logged_in	43,58827	22	service	0,467497
2	srv_count	10,17572	23	num_failed_logins	0,436873
3	protocol_type	6,758706	24	is_guest_login	0,273051
4	dst_host_same_src_port_rate	5,023091	25	duration	0,221032
5	dst_host_rerror_rate	3,883350	26	srv_diff_host_rate	0,151338
6	srv_rerror_rate	3,260211	27	wrong_fragment	0,075050
7	dst_host_srv_count	2,961421	28	land	0,065774
8	dst_host_srv_diff_host_rate	2,771480	29	su_attempted	0,015583
9	rerror_rate	2,399092	30	num_shells	0,013042
10	dst_host_same_srv_rate	2,385199	31	num_file_creations	0,011656
11	count	2,189035	32	root_shell	0,003134
12	flag	2,015506	33	is_host_login	0,002547
13	same_srv_rate	1,635362	34	hot	0,002542
14	dst_host_serror_rate	1,608696	35	urgent	0,001582
15	dst_host_count	1,451791	36	num_compromised	0,000298
16	dst_host_diff_srv_rate	1,416745	37	src_bytes	0,000088
17	diff_srv_rate	1,135209	38	dst_bytes	0,000053
18	serror_rate	1,012202	39	num_root	0,000050
19	dst_host_srv_serror_rate	0,873142	40	num_access_files	0,0000001
20	srv_serror_rate	0,870023	41	num_outbound_cmds	$1,97 \times 10^{-29}$
21	dst_host_srv_rerror_rate	0,844553			

4.3.2 *Mean Decrease in Impurity* (MDI)

Com o objetivo de complementar a análise de redução de dimensionalidade baseada na extração de atributos (PCA), foi também adotada uma abordagem de seleção de atributos (*Feature Selection* – FS) por meio do critério *Mean Decrease Impurity* (MDI).

A técnica foi aplicada utilizando um modelo de *Random Forest* com 100 árvores, estimando a importância de cada *feature* a partir da redução média de impureza nos nós das árvores. No caso do *dataset* UNSW-NB15, foi definido um limiar (*threshold*) de 1% do valor máximo de

importância, resultando na exclusão das *features* cuja importância calculada fosse inferior a 0,01. Esse valor foi adotado como limiar por equilibrar a remoção de atributos pouco relevantes com a manutenção de um conjunto suficientemente informativo. Esse valor foi observado como eficaz em trabalhos anteriores, como o de (SIRISHA et al., 2021) e, em experimentos preliminares desta pesquisa, mostrou-se adequado para reduzir a dimensionalidade sem comprometer significativamente o desempenho dos modelos.

No *dataset* UNSW-NB15, após as técnicas de normalização *Min-Max* e *Z-Score*, houve a remoção das *features* dispostas na Tabela 31 em ordem crescente. Apesar de uma pequena alteração na ordem de importância, as *features* removidas foram as mesmas tanto no *dataset* normalizado usando *Min-Max* quanto *Z-Score*. Dessa forma, nos dois casos o UNSW-NB15 teve o seu número de *features* reduzido de 41 para 28.

Tabela 31 – Comparação das *Features* Selecionadas do UNSW-NB15 após o MDI

Min-Max			***Z-Score***		
Id	***Feature***	**Importância**	**Id**	***Feature***	**Importância**
1	sbytes	0,076469	1	sbytes	0,078061
2	sttl	0,075155	2	ct_dst_sport_ltm	0,072442
3	ct_dst_src_ltm	0,065488	3	sttl	0,063560
4	ct_dst_sport_ltm	0,063337	4	ct_dst_src_ltm	0,061370
5	ct_srv_dst	0,060738	5	ct_srv_dst	0,057590
6	smean	0,057712	6	smean	0,051952
7	ct_srv_src	0,050396	7	ct_state_ttl	0,045799
8	ct_state_ttl	0,043678	8	ct_srv_src	0,044947
9	ct_src_dport_ltm	0,040801	9	ct_src_dport_ltm	0,041166
10	dload	0,035617	10	service	0,040539
11	service	0,034509	11	dload	0,034762
12	dbytes	0,027389	12	sload	0,031599
13	rate	0,026400	13	rate	0,027966
14	sload	0,025156	14	proto	0,014899
15	ct_dst_ltm	0,023513	15	dur	0,024852
16	dur	0,022853	16	ct_src_ltm	0,023816
17	dmean	0,022573	17	dmean	0,023245
18	ackdat	0,021904	18	dttl	0,023040
19	synack	0,020817	19	dbytes	0,023515
20	dttl	0,020061	20	tcprtt	0,023413
21	ct_src_ltm	0,019785	21	ct_dst_ltm	0,022757
22	tcprtt	0,018641	22	ackdat	0,019203
23	sinpkt	0,015672	23	synack	0,017558
24	proto	0,015192	24	sinpkt	0,016857
25	sjit	0,014961	25	dinpkt	0,016549
26	dpkts	0,014284	26	djit	0,012492
27	dinpkt	0,012000	27	dpkts	0,012151
28	djit	0,011963	28	sjit	0,011344

Já no *dataset* NSL-KDD, analisado na pesquisa independente conduzida pelo discente Miguel Borges, a seleção foi realizada com o auxílio da função `SelectFromModel` da biblioteca *scikit-learn*, utilizando como limite o valor *'median'*. A Tabela 32 apresenta os atributos remanescentes no *dataset* NSL-KDD após a aplicação do MDI e as técnicas de normalização *Min-Max* e *Z-Score*, selecionando 21 atributos em ambos os casos.

Tabela 32 – Comparação das *Features* Selecionadas do NSL-KDD após o MDI

Min-Max			***Z-Score***		
Id	***Feature***	**Importância**	**Id**	***Feature***	**Importância**
1	flag	0.096856	1	src_bytes	0.160215
2	same_srv_rate	0.080183	2	same_srv_rate	0.071809
3	srv_serror_rate	0.060538	3	flag	0.067481
4	serror_rate	0.060163	4	dst_bytes	0.063868
5	diff_srv_rate	0.057226	5	count	0.061033
6	count	0.056727	6	serror_rate	0.053027
7	dst_host_same_src_port_rate	0.047284	7	srv_serror_rate	0.045415
8	dst_host_same_srv_rate	0.046264	8	dst_host_same_srv_rate	0.041877
9	dst_host_diff_srv_rate	0.041198	9	diff_srv_rate	0.039589
10	dst_host_srv_serror_rate	0.040190	10	dst_host_same_src_port_rate	0.038307
11	dst_host_rerror_rate	0.040130	11	dst_host_diff_srv_rate	0.034037
12	service	0.039284	12	dst_host_srv_count	0.032371
13	dst_host_srv_count	0.038753	13	dst_host_srv_serror_rate	0.030966
14	protocol_type	0.035665	14	protocol_type	0.029169
15	dst_host_srv_diff_host_rate	0.033543	15	service	0.028878
16	logged_in	0.031908	16	srv_count	0.025851
17	srv_count	0.026318	17	logged_in	0.023675
18	dst_host_count	0.022796	18	dst_host_rerror_rate	0.023514
19	dst_host_serror_rate	0.020770	19	dst_host_srv_diff_host_rate	0.022621
20	dst_host_srv_rerror_rate	0.020671	20	dst_host_serror_rate	0.015948
21	src_bytes	0.019744	21	dst_host_count	0.015853

Nesses experimentos, os conjuntos de atributos resultantes são muito próximos, com a diferença do atributo *"dst_host_srv_rerror_rate"*, que permaneceu no NSL-KDD normalizado com *Min-Max*, mas não após a normalização *Z-Score*, e o atributo *"dst_bytes"*, que permaneceu no *dataset* normalizado com *Z-Score* e foi removido após a normalização *Min-Max*.

4.3.3 Análise dos *Datasets* após a Redução de Dimensionalidade

Após a aplicação das técnicas de normalização e redução de dimensionalidade, existem quatro *datasets* diferentes gerados a partir do UNSW-NB15, a serem avaliados. A Tabela 33 apresenta o número de atributos presentes em cada conjunto de dados obtido após a aplicação das técnicas PCA e MDI estudadas, juntamente com as técnicas de normalização *Min-Max* e *Z-Score*.

Tabela 33 – Número de Atributos em cada Experimento no UNSW-NB15

Técnica de Redução de Dimensionalidade	**Normalização**	**Número de Atributos**
PCA	*Min-Max*	17
PCA	*Z-Score*	29
MDI	*Min-Max*	28
MDI	*Z-Score*	28

Já para o NSL-KDD, as técnicas de redução de dimensionalidade e os métodos de normalização também geram conjuntos de dados com diferentes quantidades de atributos. A Tabela 34 mostra o número de atributos de cada *dataset* obtido após a aplicação das técnicas de normalização e redução de dimensionalidade avaliadas.

Tabela 34 – Número de Atributos em cada Experimento no NSL-KDD

Técnica de Redução de Dimensionalidade	Normalização	Número de Atributos
PCA	*Min-Max*	18
PCA	*Z-Score*	30
MDI	*Min-Max*	21
MDI	*Z-Score*	21

4.4 Treinamento das Máquinas de Aprendizado

A fim de comparar a combinação das técnicas de normalização e as técnicas de redução de dimensionalidade, cada um dos *datasets* apresentados na Tabela 33 é submetido a uma máquina de aprendizado RF e a uma rede neural MLP. Assim como nos testes realizados nas seções 4.2.1 e 4.2.2, aplicam-se variados valores para o número de árvores de decisão e neurônios nas camadas ocultas nas técnicas de aprendizado de máquina RF e MLP, respectivamente.

Os dados dos testes realizados sobre os *datasets* gerados a partir do NSL-KDD são provenientes da pesquisa do discente Miguel Borges de Rezende Costa e são analisados a seguir em conjunto com os dados coletados nos testes feitos sobre os *datasets* gerados a partir do *dataset* UNSW-NB15, com as quantidades de atributos apresentadas na Tabela 34. Abaixo das médias dos resultados, obtidos nos experimentos com a redução de dimensionalidade, estão dispostas as médias dos resultados obtidos antes de qualquer pré-processamento (MLs sobre *datasets* sem normalização e sem redução de dimensionalidade).

4.4.1 Resultados após PCA

4.4.1.1 *Random Forest* Após *Min-Max* com PCA

A Tabela 35 apresenta o desempenho alcançado pelo algoritmo RF sobre o *dataset* UNSW-NB15 após a normalização *Min-Max* e submetido à redução de dimensionalidade do método PCA. Nesse conjunto de testes, observa-se uma queda geral nos desempenhos médios em relação ao cenário original (sem normalização ou redução). A acurácia e o *recall* médios diminuíram de 75,21% para 72,42% (queda de aproximadamente 2,79%), enquanto a precisão apresentou uma redução mais significativa, de cerca de 13,16% (de 83,73% para 70,56%).

Tabela 35 – Desempenho Após Normalização *Min-Max* e PCA no UNSW-NB15.

Experimento	Árvores	Acurácia	Precisão	*Recall*	*F1-score*	AUC-ROC	Tempo (s)
13.1	10	0,720687	0,691703	0,720687	0,692518	0,930875	5,457813
13.2	25	0,724212	0,704542	0,724212	0,694487	0,944143	12,783849
13.3	50	0,725466	0,708367	0,725466	0,696020	0,949542	25,939838
13.4	75	0,724788	0,707382	0,724788	0,694786	0,951508	37,851010
13.5	100	0,725073	0,708219	0,725073	0,695240	0,952429	50,288745
13.6	125	0,724388	0,707740	0,724388	0,694545	0,953127	62,174137
13.7	150	0,724890	0,709450	0,724890	0,694789	0,953796	75,155136
Média	**-**	**0,724215**	**0,705629**	**0,724215**	**0,694512**	**0,947346**	**38,807504**
Antes	**-**	**0,752139**	**0,837270**	**0,752139**	**0,776132**	**0,961489**	**48,722285**

O *F1-Score* também sofreu queda relevante, passando de 77,61% para 69,45% (redução de cerca de 8,16%). O *AUC-ROC*, por sua vez, foi levemente impactado, com uma diminuição

de 1,41% (de aproximadamente 0,961 para 0,947). O tempo médio de execução, em contrapartida, teve um leve ganho, reduzindo de aproximadamente 48,72 segundos para 38,60 segundos, representando uma queda de cerca de 20%. Ainda que exista um ganho moderado em desempenho computacional, a combinação das técnicas *Min-Max* com a redução de dimensionalidade PCA resulta em uma degradação perceptível nas métricas de qualidade da classificação sobre o UNSW-NB15, especialmente em precisão e *F1-Score*, indicando que essa estratégia de pré-processamento não se mostrou vantajosa para esse *dataset*.

A Tabela 36 apresenta o desempenho alcançado pela máquina de aprendizado RF sobre o *dataset* NSL-KDD após a normalização *Min-Max* e a técnica de redução de dimensionalidade PCA ao conjunto de dados NSL-KDD.

Tabela 36 – Desempenho Após Normalização *Min-Max* e PCA no NSL-KDD.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
14.1	10	0,997879	0,998039	0,997879	0,997869	0,999903	25,974637
14.2	25	0,998047	0,998037	0,998047	0,998038	0,999956	60,615771
14.3	50	0,998047	0,998034	0,998047	0,998038	0,999958	109,715909
14.4	75	0,998047	0,998002	0,998047	0,998034	0,999958	171,206318
14.5	100	0,998014	0,998034	0,998014	0,998003	0,999975	224,253235
14.6	125	0,998047	0,998068	0,998047	0,998035	0,999974	307,155616
14.7	150	0,998081	0,998068	0,998081	0,998068	0,999975	449,912189
Média	**-**	**0,998023**	**0,998040**	**0,998023**	**0,998012**	**0,999957**	**192,119383**
Antes	**-**	**0,750234**	**0,815337**	**0,750234**	**0,709451**	**0,924714**	**12,650100**

O algoritmo apresenta desempenho significativamente superior em todos os indicadores de avaliação, em comparação com os resultados obtidos antes do pré-processamento. A acurácia média alcançada foi de 0,998023, representando um aumento expressivo frente ao valor anterior de 0,750234. A *Precisão* e o *Recall* médios mantiveram-se praticamente equilibrados, com valores de 0,998040 e 0,998023, respectivamente, resultando em um *F1-score* de 0,998012. Esses números indicam uma excelente capacidade do modelo em identificar corretamente as classes, reduzindo a taxa de falsos positivos e negativos. Além disso, o valor médio da métrica AUC-ROC atingiu 0,999957, evidenciando uma separação quase perfeita entre as classes no espaço dimensional reduzido. O tempo médio de execução dos experimentos foi de 192,12 segundos, mostrando um aumento considerável em comparação com o tempo anterior (12,65 segundos), porém justificável diante dos ganhos significativos de desempenho. Esses resultados sugerem que a aplicação do PCA, combinada à normalização *Min-Max*, é altamente eficaz no aprimoramento do desempenho do *Random Forest* no NSL-KDD.

4.4.1.2 *Random Forest* Após *Z-Score* com PCA

A Tabela 37 apresenta o desempenho alcançado pela máquina de aprendizado RF sobre o *dataset* UNSW-NB15 após a normalização *Z-Score* e a técnica PCA. Observa-se, de forma geral, que essa combinação de métodos resultou em uma queda substancial no desempenho do classificador em todos os indicadores. A acurácia média dos experimentos foi de aproximadamente 36,24%, um valor significativamente inferior à acurácia obtida antes da aplicação do PCA (75,2139%).

Tabela 37 – Desempenho Após Normalização *Z-Score* e PCA no UNSW-NB15.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
15.1	10	0,297620	0,342718	0,297620	0,287980	0,578952	7,341148
15.2	25	0,353620	0,386913	0,353620	0,326371	0,604714	17,528607
15.3	50	0,383173	0,414848	0,383173	0,336222	0,626922	34,559171
15.4	75	0,382546	0,411976	0,382546	0,334403	0,648009	51,378444
15.5	100	0,378508	0,413615	0,378508	0,328181	0,640641	68,186289
15.6	125	0,383076	0,403414	0,383076	0,332876	0,643988	85,473976
15.7	150	0,359214	0,420524	0,359214	0,314510	0,633386	104,040238
Média	**-**	**0,362394**	**0,399001**	**0,362394**	**0,322220**	**0,625230**	**52,072553**
Antes	**-**	**0,752139**	**0,837270**	**0,752139**	**0,776132**	**0,961489**	**48,722285**

Os demais indicadores também sofreram reduções importantes: a precisão caiu de cerca de 83,73% para 39,90%, o *recall* de 75,21% para 36,24%, e o *F1-score* de 77,61% para 32,22%. A métrica AUC-ROC cai de 96,14% para 62,52%, evidenciando a perda de qualidade do classificador após a aplicação do PCA. Além da degradação no desempenho, nota-se também um aumento significativo no tempo de execução. A média do tempo de processamento dos experimentos com PCA é de aproximadamente 52,07 segundos, superando a média de 48,72 segundos observada nos experimentos anteriores às técnicas de pré-processamento. A análise dos experimentos individualmente revela que, mesmo com o aumento no número de árvores, os ganhos em desempenho são praticamente nulos após a aplicação do PCA.

A Tabela 38 apresenta o desempenho alcançado pela máquina de aprendizado RF sobre o *dataset* NSL-KDD após a normalização *Z-Score* e a técnica de redução de dimensionalidade PCA. De forma geral, os resultados obtidos foram significativamente superiores ao cenário base (sem normalização ou redução). A acurácia média dos experimentos com normalização e PCA foi de 99,80%, representando um aumento substancial de 24,78 pontos percentuais em relação à acurácia observada antes do pré-processamento (75,02%).

Tabela 38 – Desempenho Após Normalização *Z-Score* com PCA no NSL-KDD.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
16.1	10	0,997643	0,997639	0,997643	0,997639	0,999871	36,429653
16.2	25	0,997980	0,997975	0,997980	0,997974	0,999958	89,604537
16.3	50	0,998081	0,998076	0,998081	0,998076	0,999993	182,525601
16.4	75	0,998283	0,998282	0,998283	0,998280	0,999993	250,754532
16.5	100	0,998350	0,998348	0,998350	0,998346	0,999993	281,767321
16.6	125	0,998283	0,998280	0,998283	0,998280	0,999994	334,174672
16.7	150	0,998283	0,998280	0,998283	0,998278	0,999993	366,947593
Média	**-**	**0,998129**	**0,998126**	**0,998129**	**0,998125**	**0,999985**	**220,314273**
Antes	**-**	**0,750234**	**0,815337**	**0,750234**	**0,709451**	**0,924714**	**12,650100**

Todos os demais indicadores apresentaram evolução semelhante: a precisão aumentou de 81,53% para 99,81%, o *Recall* de 75,02% para 99,81%, e o F1-score de 70,95% para 99,81%. A métrica AUC-ROC teve crescimento notável, saltando de 92,47% para 99,99%, indicando quase total separabilidade entre as classes após o novo tratamento dos dados. O tempo médio de execução também foi impactado, passando de 12,65 segundos no cenário original para 220,31 segundos após a aplicação do Z-Score e PCA.

4.4.1.3 *Multi-Layer Perceptron* após *Min-Max* com PCA

Os resultados apresentados na Tabela 39 indicam uma melhora significativa no desempenho do classificador MLP sobre o *dataset* UNSW-NB15 após a aplicação conjunta da normalização Min-Max e da redução de dimensionalidade por PCA.

Tabela 39 – Desempenho do MLP Após *Min-Max* com PCA no UNSW-NB15.

Experimento	Camadas	Acurácia	Precisão	*Recall*	*F1-score*	AUC-ROC	Tempo (s)
17.1	(16, 8)	0,728660	0,726957	0,728660	0,700478	0,958914	53,333889
17.2	(32, 16)	0,745519	0,735334	0,745519	0,719776	0,960357	58,224357
17.3	(64, 32)	0,752357	0,754092	0,752357	0,730034	0,960308	148,338394
17.4	(10, 20, 10)	0,739297	0,731227	0,739297	0,710638	0,961985	67,177872
17.5	(20, 40, 20)	0,744258	0,737261	0,744258	0,719742	0,959353	81,601215
17.6	(40, 60, 40)	0,748370	0,731984	0,748370	0,718367	0,957931	225,599460
17.7	(60, 40, 60)	0,761140	0,753726	0,761140	0,745094	0,962752	249,724807
17.8	(60, 80, 60)	0,746095	0,750470	0,746095	0,717912	0,959659	320,132888
17.9	(80, 100, 80)	0,756115	0,758345	0,756115	0,728483	0,961536	417,304578
17.10	(100, 120, 100)	0,745479	0,754190	0,745479	0,718425	0,956328	507,543782
17.11	(100, 80, 60, 40, 20)	0,760535	0,768312	0,760535	0,740169	0,960124	505,704453
17.12	(128, 128, 64, 64)	0,752676	0,768398	0,752676	0,725603	0,956869	645,702021
Média	**-**	**0,749472**	**0,749821**	**0,749472**	**0,723731**	**0,959772**	**279,561233**
Antes	**-**	**0,463537**	**0,414353**	**0,463537**	**0,298859**	**0,530137**	**141,126060**

A Acurácia média saltou de 46,35% para 74,95%, acompanhada por ganhos expressivos em precisão (de 41,43% para 74,98%), *Recall* (de 46,35% para 74,95%) e *F1-score* (de 29,88% para 72,37%). O AUC-ROC também apresenta uma elevação notável, de aproximadamente 53,01% para 95,97%. Apesar do tempo de execução médio ter aumentado de 141 segundos para cerca de 280 segundos, os ganhos em desempenho superam essa desvantagem. Esses resultados sugerem que a combinação da normalização *Min-Max* com o método PCA teve um impacto positivo na eficácia do MLP ao lidar com o UNSW-NB15, especialmente quando comparado ao desempenho antes do pré-processamento.

Os resultados da Tabela 40 evidenciam uma melhoria expressiva no desempenho do classificador MLP quando aplicada a combinação da normalização *Min-Max* com a técnica de redução de dimensionalidade PCA sobre o *dataset* NSL-KDD. A acurácia média passou de 73,85% para 99,78%, representando um aumento de 25,93 pontos percentuais em relação ao cenário sem pré-processamento.

Tabela 40 – Desempenho do MLP Após *Min-Max* com PCA no NSL-KDD

Experimento	Camadas	Acurácia	Precisão	*Recall*	*F1-score*	AUC-ROC	Tempo (s)
18.1	(16, 8)	0,997172	0,997196	0,997172	0,997176	0,999843	64,214537
18.2	(32, 16)	0,997273	0,997285	0,997273	0,997275	0,999901	85,264370
18.3	(64, 32)	0,997778	0,997793	0,997778	0,997782	0,999952	109,569019
18.4	(10, 20, 10)	0,996263	0,996291	0,996263	0,996274	0,999831	110,385835
18.5	(20, 40, 20)	0,997408	0,997425	0,997408	0,997414	0,999835	75,153888
18.6	(40, 60, 40)	0,997946	0,997973	0,997946	0,997947	0,999874	128,212154
18.7	(60, 40, 60)	0,997778	0,997794	0,997778	0,997775	0,999936	103,631520
18.8	(60, 80, 60)	0,997610	0,997616	0,997610	0,997612	0,999933	129,891777
18.9	(80, 100, 80)	0,998317	0,998323	0,998317	0,998314	0,999918	119,385821
18.10	(100, 120, 100)	0,998014	0,998020	0,998014	0,998014	0,999906	214,150349
18.11	(100, 80, 60, 40, 20)	0,998418	0,998430	0,998418	0,998422	0,999891	226,031353
18.12	(128, 128, 64, 64)	0,997879	0,997887	0,997879	0,997880	0,999932	187,686586
Média	**-**	**0,997830**	**0,997838**	**0,997830**	**0,997831**	**0,999906**	**137,215179**
Antes	**-**	**0,738503**	**0,717099**	**0,738503**	**0,692879**	**0,846527**	**115,348566**

Todos os demais indicadores apresentaram elevações consistentes: a precisão passou de 71,71% para 99,78%, o *Recall* de 73,85% para 99,78% e o *F1-score* de 69,28% para 99,78%. A métrica AUC-ROC saltou de 84,65% para 99,99%, evidenciando capacidade quase perfeita de separação entre as classes. Apesar do aumento no tempo médio de execução de 115,35 segundos para 137,21 segundos (cerca de 19%), o ganho substancial em desempenho justifica o custo computacional adicional.

4.4.1.4 *Multi-Layer Protocol* Após *Z-Score* com PCA

Os resultados da Tabela 41 revelam que a aplicação conjunta da normalização *Z-Score* e da técnica de redução de dimensionalidade PCA sobre o *dataset* UNSW-NB15 provocou uma deterioração significativa no desempenho do classificador MLP, que já não havia sido muito eficiente, com todas as métricas abaixo de 50%, com exceção do AUC-ROC, que possui a média de 53,01%, aproximadamente, o que também indica uma capacidade discriminativa muito limitada do modelo.

Tabela 41 – Desempenho do MLP Após *Z-Score* com PCA no UNSW-NB15.

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
19.1	(16, 8)	0,306181	0,321281	0,306181	0,309740	0,582620	67,848197
19.2	(32, 16)	0,169225	0,199234	0,169225	0,177357	0,499946	78,973953
19.3	(64, 32)	0,189049	0,254656	0,189049	0,211078	0,498129	224,184340
19.4	(10, 20, 10)	0,500727	0,495545	0,500727	0,486618	0,778374	82,492319
19.5	(20, 40, 20)	0,359317	0,361188	0,359317	0,355104	0,657877	105,331981
19.6	(40, 60, 40)	0,247307	0,310767	0,247307	0,256174	0,569236	287,888101
19.7	(60, 40, 60)	0,167382	0,226305	0,167382	0,175860	0,502815	324,009536
19.8	(60, 80, 60)	0,283744	0,335521	0,283744	0,294402	0,597498	361,440811
19.9	(80, 100, 80)	0,166236	0,175417	0,166236	0,163716	0,499965	433,118971
19.10	(100, 120, 100)	0,317518	0,323950	0,317518	0,306337	0,598888	527,074211
19.11	(100, 80, 60, 40, 20)	0,258610	0,296362	0,258610	0,273080	0,578030	521,006257
19.12	(128, 128, 64, 64)	0,306665	0,303172	0,306665	0,302047	0,596286	654,801785
Média	**-**	**0,288792**	**0,305623**	**0,288792**	**0,289069**	**0,578769**	**294,510943**
Antes	**-**	**0,463537**	**0,414353**	**0,463537**	**0,298859**	**0,530137**	**141,126060**

A Acurácia média caiu de 46,35% para 28,87%, com reduções similares nos valores médios de Precisão (de 41,43% para 30,56%), *Recall* (de 46,35% para 28,87%). O *F1-score* foi a métrica que tem a menor diferença em relação aos experimentos anteriores (de 29,88% para 28,91%) e o AUC-ROC é o único parâmetro desse conjunto de experimentos com um valor superior ao observado antes do pré-processamento. Apesar disso, essas duas métricas ainda possuem valores pouco satisfatórios, em comparação aos encontrados no resto dos experimentos nessa seção. Além desses resultados, o tempo médio de execução também possui um aumento de cerca de 141,12 segundos para 294,51 segundos, mais do que o dobro do tempo nos primeiros experimentos.

A Tabela 42 evidencia os ganhos expressivos obtidos com a combinação da normalização *Z-Score* e da técnica de redução de dimensionalidade PCA sobre o *dataset* NSL-KDD.

Tabela 42 – Desempenho do MLP Após *Z-Score* com PCA no NSL-KDD

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
20.1	(16, 8)	0,997172	0,997196	0,997172	0,997176	0,999843	43,788188
20.2	(32, 16)	0,997273	0,997285	0,997273	0,997275	0,999901	64,202623
20.3	(64, 32)	0,997778	0,997793	0,997778	0,997782	0,999952	84,425123
20.4	(10, 20, 10)	0,996263	0,996291	0,996263	0,996274	0,999831	108,605467
20.5	(20, 40, 20)	0,997408	0,997425	0,997408	0,997414	0,999835	66,973010
20.6	(40, 60, 40)	0,997946	0,997973	0,997946	0,997947	0,999874	124,685399
20.7	(60, 40, 60)	0,997778	0,997794	0,997778	0,997775	0,999936	101,883419
20.8	(60, 80, 60)	0,997610	0,997616	0,997610	0,997612	0,999933	113,123671
20.9	(80, 100, 80)	0,998317	0,998323	0,998317	0,998314	0,999918	91,376976
20.10	(100, 120, 100)	0,998014	0,998020	0,998014	0,998014	0,999906	184,215965
20.11	(100, 80, 60, 40, 20)	0,998418	0,998430	0,998418	0,998422	0,999891	204,986679
20.12	(128, 128, 64, 64)	0,997879	0,997887	0,997879	0,997880	0,999932	232,071002
Média	**-**	**0,997666**	**0,997681**	**0,997666**	**0,997667**	**0,999908**	**118,778669**
Antes	**-**	**0,738503**	**0,717099**	**0,738503**	**0,692879**	**0,846527**	**115,348566**

Observa-se um aumento expressivo na média de todos os indicadores do classificador MLP: a Acurácia passou de 73,85% para 99,77%, a Precisão de 71,71% para 99,77%, o *Recall* de 73,85% para 99,77% e o *F1-score* de 69,28% para 99,77%. O valor médio de *AUC-ROC* também apresentou um salto marcante, de 0,846 para 0,9999, indicando praticamente total separabilidade entre as classes. O tempo médio de execução teve um leve aumento, de 115,35 segundos para 118,78 segundos, sugerindo que o ganho em desempenho preditivo ocorreu com um custo computacional praticamente estável. Esses resultados confirmam que a estratégia de pré-processamento com *Z-Score* e PCA é altamente eficaz no contexto do NSL-KDD para redes neurais MLP.

4.4.2 Resultados após MDI

4.4.2.1 *Random Forest* Após *Min-Max* com MDI

A Tabela 43 apresenta o desempenho alcançado pela máquina de aprendizado RF sobre o *dataset* UNSW-NB15 normalizado pela técnica *Min-Max* e após a remoção das *features* pela técnica de redução de dimensionalidade MDI.

Tabela 43 – Desempenho do RF Após *Min-Max* com MDI no UNSW-NB15.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
21.1	10	0,752659	0,747325	0,752659	0,725838	0,943426	1,858191
21.2	25	0,753218	0,760103	0,753218	0,723092	0,952721	4,411316
21.3	50	0,753640	0,760856	0,753640	0,722353	0,957471	9,131381
21.4	75	0,752859	0,760058	0,752859	0,721075	0,959444	14,705740
21.5	100	0,752671	0,759440	0,752671	0,720541	0,959877	17,886857
21.6	125	0,753247	0,760749	0,753247	0,721451	0,960554	21,439139
21.7	150	0,753406	0,761719	0,753406	0,721584	0,960801	26,118284
Média	**-**	**0,752814**	**0,758607**	**0,752814**	**0,722133**	**0,956899**	**13,078987**
Antes	**-**	**0,752139**	**0,837270**	**0,752139**	**0,776132**	**0,961489**	**48,722285**

Nesse conjunto de testes, observa-se um aumento pouco significativo nas médias de Acurácia e *Recall* (ambas foram de aproximadamente 75,21% para 75,28%, representando um ganho menor que 0,1%), uma queda notável no valor da Precisão (de cerca de 83,73% para 75,86%, 7,87% de diferença) e do *F1-Score* (de 77,61% para 72,21%, mais de 5% pior que antes) e uma leve diminuição no *AUC-ROC Score* (que foi de 96,15% para 95,68%). Apesar disso,

houve uma redução de mais de 72% no tempo de execução. Ainda que tenha se obtido um ganho perceptível em velocidade de execução dos testes, a combinação da técnica de normalização *Min-Max* aliada à filtragem de atributos pelo MDI não apresenta muitas melhorias em comparação ao desempenho sobre o UNSW-NB15 antes do pré-processamento.

A Tabela 44 apresenta o desempenho do algoritmo RF aplicado ao *dataset* NSL-KDD após normalização com *Min-Max* e redução de atributos utilizando o método MDI. Observa-se que todos os indicadores atingiram patamares extremamente elevados, com valores muito próximos a 100%. A acurácia média passou de 75,02% para 99,90%, a precisão de 81,53% para 99,90%, o *Recall* de 75,02% para 99,90% e o *F1-score* de 70,94% para 99,90%.

Tabela 44 – Desempenho do RF Após *Min-Max* com MDI no NSL-KDD.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
22.1	10	0,998620	0,998610	0,998620	0,998611	0,999963	3,005524
22.2	25	0,998889	0,998885	0,998889	0,998883	0,999963	7,319448
22.3	50	0,999091	0,999087	0,999091	0,999086	0,999998	14,030191
22.4	75	0,999024	0,999020	0,999024	0,999018	0,999998	21,170610
22.5	100	0,999024	0,999020	0,999024	0,999019	0,999998	39,699925
22.6	125	0,999024	0,999018	0,999024	0,999019	0,999999	46,713293
22.7	150	0,999057	0,999053	0,999057	0,999054	0,999998	52,651149
Média	**-**	**0,998964**	**0,998956**	**0,998964**	**0,998956**	**0,999988**	**26,084591**
Antes	**-**	**0,750234**	**0,815337**	**0,750234**	**0,709451**	**0,924714**	**12,650100**

O valor médio de AUC-ROC subiu de 92,47% para 99,99%, indicando separação praticamente perfeita entre as classes. Além disso, mesmo com a elevação drástica da performance, o tempo médio de execução aumentou de forma proporcional ao número de árvores, mas manteve-se em níveis bastante competitivos (média de 26,08 segundos), especialmente considerando a complexidade adicional do pré-processamento. Esses resultados mostram que a combinação de *Min-Max* e MDI no NSL-KDD é altamente eficaz para o RF, proporcionando desempenho quase máximo com custo computacional aceitável.

4.4.2.2 *Random Forest* Após *Z-Score* com MDI

A Tabela 45 apresenta o desempenho alcançado pela máquina de aprendizado RF sobre o *dataset* UNSW-NB15 após a normalização *Z-Score* e a filtragem de atributos por MDI.

Tabela 45 – Desempenho do RF Após *Z-Score* com MDI no UNSW-NB15.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
23.1	10	0,750988	0,747180	0,750988	0,722144	0,935051	2,533927
23.2	25	0,752117	0,756176	0,752117	0,720472	0,947266	4,741485
23.3	50	0,752009	0,755037	0,752009	0,720119	0,952884	9,515589
23.4	75	0,751296	0,759074	0,751296	0,718970	0,955242	13,935365
23.5	100	0,751404	0,759639	0,751404	0,719199	0,956461	18,509570
23.6	125	0,751524	0,758522	0,751524	0,719271	0,957327	23,122220
23.7	150	0,752003	0,762413	0,752003	0,719817	0,957886	28,481534
Média	**-**	**0,751763**	**0,756292**	**0,751763**	**0,719142**	**0,951445**	**14,691527**
Antes	**-**	**0,752139**	**0,837270**	**0,752139**	**0,776132**	**0,961489**	**48,722285**

Comparando com os resultados obtidos antes da normalização e do MDI, nota-se que a Acurácia se manteve praticamente inalterada (de 75,21% para 75,18%), assim como o , porém

houve redução expressiva na Precisão, que caiu de 83,73% para 75,63%, e também no *F1-score*, que passou de 77,61% para 71,91%. Além disso, o valor da AUC-ROC apresentou uma leve queda de 0,961 para 0,951, o que indica uma leve redução na capacidade discriminativa do classificador após o pré-processamento.

Apesar dessas reduções em desempenho, o tempo médio de execução sofreu uma queda significativa, indo de 48,72 segundos para 14,69 segundos, evidenciando o impacto positivo da seleção de atributos sobre a eficiência computacional do modelo.

A Tabela 46 apresenta o desempenho alcançado pelo algoritmo RF sobre o *dataset* NSL-KDD após a normalização *Z-Score* e a redução de atributos utilizando o método MDI. Observa-se uma performance extremamente elevada e consistente em todos os experimentos, com Acurácia média de 99,8687%, Precisão de 99,8680%, *Recall* de 99,8687% e *F1-score* de 99,8682%. O valor médio de AUC-ROC é de aproximadamente 99,991%, evidenciando a altíssima capacidade discriminativa do modelo mesmo após a filtragem de atributos. O tempo médio de execução foi de 13,49 segundos, mantendo boa eficiência computacional mesmo com o alto número de árvores em alguns experimentos.

Tabela 46 – Desempenho do *Random Forest* Após *Z-Score* com MDI no NSL-KDD.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
24.1	10	0,998687	0,998677	0,998687	0,998678	0,999963	2,938938
24.2	25	0,998586	0,998580	0,998586	0,998581	0,999857	5,040070
24.3	50	0,998687	0,998681	0,998687	0,998682	0,999953	8,924688
24.4	75	0,998687	0,998681	0,998687	0,998682	0,999953	12,615490
24.5	100	0,998687	0,998681	0,998687	0,998682	0,999953	17,103016
24.6	125	0,998687	0,998681	0,998687	0,998682	0,999953	21,176323
24.7	150	0,998687	0,998681	0,998687	0,998682	0,999953	24,640390
Média	**-**	**0,998687**	**0,998680**	**0,998687**	**0,998682**	**0,999911**	**13,491845**
Antes	**-**	**0,750234**	**0,815337**	**0,750234**	**0,709451**	**0,924714**	**12,650100**

Comparando com os resultados anteriores à normalização e aplicação do MDI, nota-se um salto expressivo na Acurácia e no *Recall* (de 75,02% para 99,87%), no *F1-score* (de 70,94% para 99,87%) e no AUC-ROC (de 92,47% para 99,99%). Apesar do tempo médio de execução ter aumentado levemente em relação ao cenário original, o ganho de desempenho justifica plenamente o uso combinado do *Z-Score* com a seleção de atributos via MDI no contexto do NSL-KDD.

4.4.2.3 *Multi-Layer Perceptron* com *Min-Max* Após MDI

A Tabela 47 apresenta o desempenho alcançado pela máquina de aprendizado MLP sobre o *dataset* UNSW-NB15 após a normalização *Min-Max*. O modelo apresentou acurácia média de 74,62%, precisão de 74,70%, *Recall* de 74,62%, F1-score de 71,32% e AUC-ROC de 0,956, com tempo médio de execução de 261,74 segundos.

Tabela 47 – Desempenho do MLP Após *Min-Max* com MDI no UNSW-NB15.

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
25.1	(16, 8)	0,744817	0,725794	0,744817	0,709896	0,956916	67,200434
25.2	(32, 16)	0,740335	0,727476	0,740335	0,703306	0,957330	76,888364
25.3	(64, 32)	0,742952	0,752142	0,742952	0,710508	0,957772	214,454087
25.4	(10, 20, 10)	0,747555	0,717615	0,747555	0,697493	0,958908	66,674235
25.5	(20, 40, 20)	0,740836	0,732725	0,740836	0,708380	0,956695	80,316067
25.6	(40, 60, 40)	0,750184	0,742401	0,750184	0,716288	0,958225	246,182333
25.7	(60, 40, 60)	0,748576	0,765379	0,748576	0,716442	0,958231	276,466099
25.8	(60, 80, 60)	0,741652	0,755975	0,741652	0,709206	0,955481	305,792710
25.9	(80, 100, 80)	0,746277	0,756135	0,746277	0,715953	0,956027	352,975880
25.10	(100, 120, 100)	0,750623	0,758225	0,750623	0,722996	0,954126	433,477165
25.11	(100, 80, 60, 40, 20)	0,748062	0,765529	0,748062	0,720161	0,955374	436,987128
25.12	(128, 128, 64, 64)	0,747281	0,768683	0,747281	0,720364	0,951966	515,886057
Média	**-**	**0,746201**	**0,747018**	**0,746201**	**0,713215**	**0,956186**	**261,742318**
Antes	**-**	**0,463537**	**0,414353**	**0,463537**	**0,298859**	**0,530137**	**141,126060**

Em comparação ao desempenho antes do pré-processamento, que obteve Acurácia de apenas 46,35% e *F1-score* de 29,89%, observa-se um aumento expressivo em todos os indicadores de desempenho, especialmente no AUC-ROC, que subiu de 0,530 para 0,956. Apesar do aumento significativo no tempo de execução (de aproximadamente 141,12 segundos passou para 261,74 segundos), o aprimoramento nos resultados evidencia o impacto positivo da normalização e da seleção de atributos sobre a eficácia do modelo MLP no UNSW-NB15.

A Tabela 48 apresenta o desempenho alcançado pela máquina de aprendizado MLP sobre o *dataset* NSL-KDD após a normalização *Min-Max* e a aplicação do método de redução de dimensionalidade MDI.

Tabela 48 – Desempenho do MLP Após *Min-Max* com MDI no NSL-KDD.

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
25.1	(16, 8)	0,997138	0,997164	0,997138	0,997142	0,999949	105,882082
25.2	(32, 16)	0,997845	0,997842	0,997845	0,997843	0,999957	101,776020
25.3	(64, 32)	0,998249	0,998264	0,998249	0,998253	0,999981	156,676089
25.4	(10, 20, 10)	0,997273	0,997307	0,997273	0,997281	0,999961	139,098604
25.5	(20, 40, 20)	0,997037	0,997008	0,997037	0,997003	0,999978	128,811056
25.6	(40, 60, 40)	0,997408	0,997404	0,997408	0,997392	0,999984	168,785677
25.7	(60, 40, 60)	0,998283	0,998287	0,998283	0,998282	0,999992	169,271150
25.8	(60, 80, 60)	0,998216	0,998217	0,998216	0,998216	0,999991	228,689764
25.9	(80, 100, 80)	0,998014	0,998020	0,998014	0,998012	0,999988	201,988957
25.10	(100, 120, 100)	0,998350	0,998348	0,998350	0,998347	0,999988	288,284347
25.11	(100, 80, 60, 40, 20)	0,997206	0,997213	0,997206	0,997204	0,999975	296,247591
25.12	(128, 128, 64, 64)	0,998721	0,998722	0,998721	0,998721	0,999993	341,517785
Média	**-**	**0,997758**	**0,997765**	**0,997758**	**0,997757**	**0,999982**	**193,169093**
Antes	**-**	**0,738503**	**0,717099**	**0,738503**	**0,692879**	**0,846527**	**115,348566**

Os resultados mostram um salto expressivo no desempenho do MLP. A acurácia e o *Recall* médios subiram de 73,85% para 99,78% (aumento de cerca de 35 pontos percentuais), a precisão passou de 71,71% para 99,78% (ganho de quase 28 pontos), e o *F1-score* de 69,28% para 99,78%. A métrica AUC-ROC praticamente atingiu o valor máximo, passando de 84,65% para 99,998%.

O tempo médio de execução aumentou de 115,35 segundos para 193,17 segundos, o que indica que, embora o custo computacional tenha crescido, o ganho em desempenho pode justificar a adoção dessa configuração para cenários que demandem alta precisão e baixa taxa de erro.

4.4.2.4 *Multi-Layer Perceptron* com *Z-Score* Após MDI

A Tabela 49 apresenta o desempenho alcançado pela máquina de aprendizado MLP sobre o *dataset* NSL-KDD após a normalização *Z-Score* e a aplicação do MDI. Os resultados indicam uma melhora expressiva em todas as métricas de desempenho em relação ao cenário anterior sem normalização e sem redução de dimensionalidade.

Tabela 49 – Desempenho do MLP Após *Z-Score* com MDI no UNSW-NB15.

Experimento	**Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
27.1	(16, 8)	0,729806	0,717791	0,729806	0,691830	0,956608	51,193596
27.2	(32, 16)	0,736690	0,745494	0,736690	0,702377	0,956899	60,749598
27.3	(64, 32)	0,752340	0,738227	0,752340	0,719139	0,957498	141,290095
27.4	(10, 20, 10)	0,749129	0,733215	0,749129	0,711723	0,959331	62,792493
27.5	(20, 40, 20)	0,745188	0,745124	0,745188	0,713190	0,957887	82,031563
27.6	(40, 60, 40)	0,748290	0,754352	0,748290	0,717971	0,956731	314,392876
27.7	(60, 40, 60)	0,755060	0,740131	0,755060	0,715593	0,957248	337,880575
27.8	(60, 80, 60)	0,751005	0,758349	0,751005	0,721877	0,956648	405,837706
27.9	(80, 100, 80)	0,745673	0,758989	0,745673	0,718092	0,954523	480,696351
27.10	(100, 120, 100)	0,749197	0,748398	0,749197	0,718128	0,952560	504,350030
27.11	(100, 80, 60, 40, 20)	0,747954	0,757977	0,747954	0,722288	0,952990	469,728602
27.12	(128, 128, 64, 64)	0,744749	0,746391	0,744749	0,712703	0,951066	463,521260
Média	**-**	**0,746641**	**0,745688**	**0,746641**	**0,714726**	**0,955556**	**251,483708**
Antes	**-**	**0,463537**	**0,414353**	**0,463537**	**0,298859**	**0,530137**	**141,126060**

A Acurácia média passou de cerca de 46,35% para 74,66%, a Precisão evoluiu de 41,43% para aproximadamente 74,57%. O *Recall* também aumentou consideravelmente (de 46,35% para 74,66%), demonstrando maior cobertura das amostras relevantes. O *F1-score* praticamente dobrou, passando de 29,88% para 71,47%. O AUC-ROC teve um salto notável de 0,5301 para 0,9556, evidenciando a eficácia do classificador mesmo sob diferentes limiares de decisão. O tempo médio de execução aumentou consideravelmente (de 141,1 para 251,5 segundos), especialmente em arquiteturas com mais camadas e neurônios. No geral, a combinação da normalização *Z-Score* com a técnica MDI proporcionou uma melhora robusta no desempenho do MLP, demonstrando sua eficácia para melhorar a capacidade discriminativa do modelo sobre o *dataset*.

A Tabela 50 apresenta o desempenho alcançado pela rede neural *Multi-Layer Perceptron* (MLP) sobre o *dataset* NSL-KDD após a aplicação da normalização *Z-Score* e da seleção de atributos via MDI. Observa-se que todos os cenários de configuração de camadas ocultas resultaram em métricas extremamente elevadas e consistentes, com Acurácia média de 99,779%, Precisão de 99,780%, *Recall* de 99,779% e *F1-score* de 99,780%. O valor médio de AUC-ROC foi de 99,979%, evidenciando a excelente capacidade discriminativa do modelo. Apesar do tempo de execução variar de 81 a 251 segundos dependendo da arquitetura, a média de aproximadamente 148,55 segundos ainda é aceitável considerando a complexidade das redes testadas.

Tabela 50 – Desempenho do MLP Após *Z-Score* com MDI no NSL-KDD.

Experimento	Camadas	Acurácia	Precisão	*Recall*	*F1-score*	AUC-ROC	Tempo (s)
28.1	(16, 8)	0,997340	0,997347	0,997340	0,997341	0,999911	82,151926
28.2	(32, 16)	0,997610	0,997609	0,997610	0,997607	0,999936	81,386486
28.3	(64, 32)	0,997845	0,997866	0,997845	0,997853	0,999944	103,854934
28.4	(10, 20, 10)	0,996364	0,996338	0,996364	0,996343	0,999853	99,774938
28.5	(20, 40, 20)	0,997273	0,997287	0,997273	0,997274	0,999957	105,889618
28.6	(40, 60, 40)	0,997576	0,997637	0,997576	0,997590	0,999909	125,911734
28.7	(60, 40, 60)	0,997812	0,997837	0,997812	0,997822	0,999928	144,479173
28.8	(60, 80, 60)	0,998014	0,998021	0,998014	0,998015	0,999957	197,654208
28.9	(80, 100, 80)	0,998014	0,998013	0,998014	0,998012	0,999978	186,062358
28.10	(100, 120, 100)	0,998418	0,998421	0,998418	0,998418	0,999945	251,707496
28.11	(100, 80, 60, 40, 20)	0,998249	0,998253	0,998249	0,998248	0,999952	148,475289
28.12	(128, 128, 64, 64)	0,998384	0,998395	0,998384	0,998389	0,999947	228,112511
Média	**-**	**0,997793**	**0,997800**	**0,997793**	**0,997800**	**0,999790**	**148,546404**
Antes	**-**	**0,738503**	**0,717099**	**0,738503**	**0,692879**	**0,846527**	**115,348566**

Comparando com o cenário antes das transformações, a Acurácia média saltou de aproximadamente 73,85% para 99,78%, a Precisão subiu de 71,71% para 99,78% e o *Recall* também passou de 73,85% para 99,78%, mostrando que o modelo passou a classificar corretamente praticamente todas as instâncias positivas. O *F1-score* evoluiu de cerca de 69,29% para 99,78%, reforçando o ganho expressivo de desempenho. O *AUC-ROC* apresentou um avanço significativo de 84,65% para 99,98%, evidenciando a altíssima capacidade de separação entre classes. Apesar do aumento no tempo médio de execução, a precisão obtida justifica plenamente o uso do *Z-Score* aliado ao MDI neste contexto.

4.5 Comparação dos Resultados

Com base nos resultados obtidos após a aplicação das técnicas de normalização e redução de dimensionalidade, é possível observar comportamentos distintos entre os modelos de classificação *Random Forest* e *Multi-Layer Perceptron* sobre os *datasets* UNSW-NB15 e NSL-KDD. A seguir, apresenta-se uma análise comparativa que sintetiza o impacto das combinações testadas de pré-processamento sobre o desempenho dos modelos em múltiplas métricas de avaliação.

No geral, os melhores resultados são obtidos sobre o *dataset* NSL-KDD, independentemente do algoritmo e da combinação de pré-processamentos utilizados. A aplicação das técnicas de normalização, combinadas com as estratégias de redução de dimensionalidade, resulta em aumentos expressivos nas métricas de desempenho para ambos os algoritmos testados.

Os experimentos sobre o UNSW-NB15, por outro lado, apresentam resultados mais heterogêneos. Observa-se uma queda acentuada nas métricas de desempenho em vários cenários, especialmente após a aplicação do PCA, tanto para o RF quanto para o MLP. Embora alguns ganhos computacionais sejam observados (como nos experimentos do MLP, por exemplo), muito desse ganho se deve simplesmente ao efeito da normalização dos dados (como foi visto na seção 4.2.2, as técnicas *Min-Max* e *Z-Score* por si só já contribuíram para que o modelo MLP obtivesse resultados semelhantes aos do RF).

Essas diferenças observadas entre os *datasets* analisados reforçam a necessidade de uma avaliação comparativa com trabalhos relacionados, como o de (SIRISHA et al., 2021), a fim de

situar os resultados deste estudo no contexto da literatura.

4.6 Comparação com o RF Usado no Artigo de Sirisha et al.

O artigo de (SIRISHA et al., 2021) apresenta uma comparação entre o desempenho de diferentes modelos de aprendizado supervisionado e não supervisionado sobre os *datasets* NSL-KDD e CICIDS-2017, normalizados com a técnica *Z-Score* (*Standard Scaler*). Esse trabalho também inclui um processo de seleção de atributos dos *datasets* por meio de um modelo RF com o limiar definido em 0,01. Esse processo de *Feature Selection* resultou em uma redução da quantidade de atributos do NSL-KDD de 41 para 26, e do CICIDS de 70 para 27. Além disso, ambos os *datasets* são usados no trabalho descrito pelo artigo com o intuito de treinar modelos para a identificação de ataques de rede, usando uma *feature* binária, que distingue comportamento normal de ataques de rede.

Em contraste, o presente trabalho também inclui experimentos com a normalização *Min-Max* e a redução de dimensionalidade usando PCA para o pré-processamento dos *datasets*, além do uso do *dataset* UNSW-NB15, ao invés do CICIDS-2017. Apesar disso, alguns pontos em comum são o uso do *dataset* NSL-KDD, do modelo *Random Forest* com um limiar mínimo de importância fixo em 0,01 para *Feature Selection* (MDI) e a avaliação de algoritmos de aprendizado supervisionado. Também vale destacar que os modelos apresentados no referente estudo foram treinados para classificar os registros apenas como ataque ou comportamento normal, enquanto os modelos do presente trabalho têm como objetivo distinguir cada uma das classes de ataque dos *datasets* usados.

Outra importante diferença nas duas metodologias é a forma com que o treinamento dos modelos é conduzido nos estudos, principalmente para a avaliação dos resultados sobre o *dataset* NSL-KDD. Enquanto os modelos apresentados no artigo de /(SIRISHA et al., 2021) foram treinados no subconjunto *KDDTest+* e testados sobre o subconjunto *KDDTest+*, o estudo conduzido por Miguel Borges sobre o NSL-KDD e referenciado nesse trabalho envolveu a concatenação desses dois subconjuntos em um único *dataset*, para depois dividir os dados aleatoriamente em treino e teste, usando uma distribuição 80/20. A seguir estão as Tabelas comparando os resultados obtidos entre os experimentos desse estudo e do trabalho relacionado, sobre o *dataset* NSL-KDD e sobre os *datasets* UNSW-NB15 e CICIDS-2017, respectivamente. Buscando melhor legibilidade e organização dos dados, as Tabelas 51 e 74 apresentam apenas os melhores resultados obtidos pelos modelos RF e MLP deste trabalho. As Tabelas com todos os modelos usados no artigo relacionado e os modelos usados no presente trabalho estão disponíveis no apêndice G.

A Tabela 51 apresenta uma comparação entre os resultados obtidos sobre o NSL-KDD no artigo relacionado e alguns dos melhores resultados obtidos no presente trabalho sobre o mesmo *dataset*. Tanto o modelo *Random Forest* quanto o MLP obtiveram resultados superiores em comparação aos obtidos pelo RF treinado no estudo de Sirisha et al., principalmente nos valores de Acurácia, Precisão e *F1-Score*. Essa diferença nos resultados pode estar associada à

forma como os subconjuntos do NSL-KDD foram separados para treinar e testar a capacidade de classificação dos modelos nos dois estudos.

Tabela 51 – Comparação dos Resultados Sobre o NSL-KDD

Experimento	Acurácia	Precisão	*Recall*	*F1-Score*	AUC-ROC
RF (SIRISHA et al., 2021)	0,765037	0,652543	0,972196	0,780925	0,948926
RF + MDI (*Min-Max*)	0,998964	0,998956	0,998964	0,998956	0,999988
MLP + PCA (*Min-Max*)	0,997830	0,997838	0,997830	0,997831	0,999906

Além disso, a seleção dos atributos também é outro aspecto a ser considerado na comparação desses resultados. Apesar do MDI também ter sido explorado no presente trabalho sobre o NSL-KDD, essa técnica de redução de dimensionalidade foi aplicada usando a mediana para remover as *features* (como foi detalhado na seção 4.3.2), ao invés do limiar mínimo de importância fixo em 0,01.

A Tabela 74 mostra os resultados obtidos no estudo referido sobre o *dataset* CICIDS-2017, em comparação com os resultados obtidos neste trabalho sobre o *dataset* UNSW-NB15. No geral, os resultados obtidos nesse trabalho sobre o UNSW não foram tão elevados quanto os apresentados pelo RF sobre o CICIDS, no artigo relacionado.

Tabela 52 – Comparação CICIDS e UNSW-NB15

Modelo	*Dataset*	Acurácia	Precisão	*Recall*	*F1-Score*	AUC-ROC
RF (SIRISHA et al., 2021)	CICIDS-2017	0,937743	0,841460	0,841509	0,841484	0,986115
RF + MDI (*Min-Max*)	UNSW-NB15	0,752814	0,758607	0,752814	0,722133	0,956899
MLP + PCA (*Min-Max*)	UNSW-NB15	0,749472	0,749821	0,749472	0,723731	0,959772

Vale destacar a complexidade de comparar os resultados na Tabela 74, já que, ao contrário dos dados na Tabela 51, eles foram obtidos sobre *datasets* distintos, com diferentes quantidades de registros, distribuições, conjuntos de atributos e classes de ataque. Além disso, assim como na Tabela anterior, os dados comparados são obtidos de modelos treinados para tipos diferentes de classificação (binária e multiclasse). Por isso, essa última comparação deve ser vista apenas como uma referência ilustrativa, e não como uma avaliação direta de desempenho.

Após a análise individual do desempenho dos modelos sobre os *datasets* UNSW-NB15 e NSL-KDD, foi conduzido um conjunto adicional de experimentos com o objetivo de avaliar a capacidade de generalização entre domínios e investigar a viabilidade de integração federada entre Sistemas IDS treinados sobre diferentes fontes de dados.

Essa etapa experimental, apresentada a seguir, representa uma extensão prática dos conceitos discutidos na seção 2.5, permitindo analisar de forma empírica se modelos treinados sobre um *dataset* são capazes de identificar ataques em outro, e se o treinamento conjunto pode melhorar essa capacidade de detecção.

4.7 Avaliação Cruzada e Treinamento Conjunto entre UNSW-NB15 e NSL-KDD

Além da comparação da quantidade de atributos selecionados entre os dois *datasets*, feita na seção 4.3.2, faz-se necessário comparar também os atributos selecionados em si, a fim de identificar possíveis semelhanças entre eles. A identificação dessas semelhanças é um ponto de grande importância, no contexto de Federação de Máquinas de Aprendizado. Destaca-se que os experimentos a seguir foram feitos utilizando apenas o MDI, devido à dificuldade em identificar com certeza os atributos exatos envolvidos na geração dos componentes principais obtidos a partir do PCA, sendo o MDI então uma técnica mais adequada para a análise e uso dos componentes em comum entre os *datasets* no treinamento das máquinas de aprendizado. A Tabela 53 apresenta quais dos atributos anteriormente apontados como semelhantes (Tabela 8) foram mantidos no UNSW-NB15 e no NSL-KDD.

Tabela 53 – *Features* Comuns Mantidas Após o MDI.

Id	UNSW-NB15	NSL-KDD	Descrição
1	proto	protocol_type	Tipo de protocolo utilizado (TCP, UDP, etc.).
2	service	service	Serviço de destino da conexão (HTTP, FTP, etc.).
3	sbytes	src_bytes	Bytes enviados da origem para o destino.
4	dbytes	dst_bytes (Z-Score)	Bytes recebidos pelo destino.
5	ct_srv_dst	srv_count	Número de conexões para o mesmo serviço e IP de destino

Vale destacar que, após a normalização *Z-Score* em combinação com o MDI, o atributo *dst_bytes* do NSL-KDD foi mantido, enquanto no MDI após a normalização *Min-Max* ele foi removido. Tendo esse cenário em vista, considera-se apenas os atributos em comum observados após o MDI combinado com a normalização *Z-Score*, ampliando assim o conjunto de variáveis compartilhadas utilizadas no treinamento e teste com os dois *datasets*.

Utilizando esses cinco atributos dos dois *datasets* após o MDI, foram realizados experimentos para investigar a possibilidade de treinar um modelo de aprendizado de máquina em um dos *datasets* e verificar como ele se comporta ao receber registros de outro *dataset*. A metodologia adotada baseou-se em ajustar o *dataset* UNSW-NB15 após o MDI para ter as mesmas *features* que o NSL-KDD e usá-lo para treinar uma máquina de aprendizado. Após o treinamento, os registros do NSL-KDD foram filtrados para conter apenas as colunas em comum com o UNSW-NB15 e enviados para a máquina de aprendizado para classificação. Devido à diferença nos tipos de ataques representados por cada *dataset*, este conjunto de experimentos buscou avaliar apenas se foi possível distinguir ataques de rede do comportamento normal.

Além disso, vale destacar que esses experimentos fazem parte de uma análise inicial para a área de Federação de Máquinas de Aprendizado em Sistemas IDS, com o objetivo de subsidiar pesquisas futuras nessa área de estudo. Por esse motivo, optou-se por utilizar configurações controladas e representativas dos modelos RF e MLP, empregados em etapas anteriores.

No caso do RF, foram consideradas apenas duas configurações de complexidade, com 50 e 100 árvores, de modo a manter um equilíbrio entre capacidade de generalização e custo computacional. Essas quantidades permitem capturar adequadamente a variação entre atributos

sem introduzir excesso de variância, além de fornecer uma base comparativa estável para verificar se padrões aprendidos em um *dataset* podem ser transferidos ou correlacionados com outro no contexto federado. Valores maiores tenderiam a apenas refinar o desempenho local, sem contribuir significativamente para o objetivo exploratório desta análise.

Para o MLP, são adotadas duas topologias representativas: uma mais simples (32,16) e outra mais profunda (100,80,60,40,20). A primeira reflete um modelo de baixa complexidade, útil para avaliar a capacidade mínima de generalização entre domínios distintos, enquanto a segunda busca explorar um cenário de maior capacidade de abstração, testando se redes com múltiplas camadas são capazes de aprender representações compatíveis entre os conjuntos. Assim como no RF, a intenção não é otimizar hiperparâmetros, mas observar o comportamento dos modelos sob condições controladas de aprendizado. A Tabela 54 apresenta os resultados obtidos pela máquina de aprendizado treinada com registros do UNSW-NB15 e testada em registros do NSL-KDD.

Tabela 54 – Modelos Treinados com UNSW-NB15 e Testados no NSL-KDD

Modelo	**Árvores/Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-Score***	**AUC-ROC**	**Tempo (s)**
RF	50	0,398527	0,376025	0,085794	0,139712	0,633485	5,918862
RF	100	0,458102	0,555817	0,239383	0,334641	0,628988	14,942019
MLP	(32, 16)	0,252584	0,355144	0,383620	0,368833	0,385333	109,392319
MLP	(100, 80, 60, 40, 20)	0,427006	0,488404	0,137848	0,215011	0,610197	447,374276

Ao treinar os modelos no UNSW-NB15 e avaliá-los sobre o NSL-KDD, observou-se baixo desempenho geral, com acurácias entre aproximadamente 25% e 46% e valores de *F1-Score* inferiores a 35% na maioria das execuções. O RF com 100 árvores apresentou o melhor desempenho dentro desse grupo, alcançando acurácia de 45,81% e AUC-ROC de 62,9%, aproximadamente. Esses resultados indicam que o modelo não conseguiu generalizar adequadamente o conhecimento adquirido a partir dos padrões do UNSW-NB15 quando aplicado ao domínio do NSL-KDD. Esse comportamento era esperado, considerando que os *datasets* foram gerados em contextos distintos de captura e categorização de tráfego, com diferenças tanto na estrutura dos pacotes quanto na representação das classes de ataque. O desempenho mais baixo da MLP, sobretudo em termos de AUC-ROC (por volta de 38,5%), sugere que a rede neural, ao depender de relações mais complexas entre as variáveis, sofre ainda mais com a falta de compatibilidade semântica entre as *features* equivalentes.

Tabela 55 – Modelos Treinados com NSL-KDD e Testados no UNSW-NB15

Modelo	**Árvores/Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-Score***	**AUC-ROC**	**Tempo (s)**
RF	50	0,398527	0,376025	0,085794	0,139712	0,633485	6,399918
RF	100	0,458102	0,555817	0,239383	0,334641	0,628988	12,398303
MLP	(32, 16)	0,461829	0,517529	0,806359	0,630437	0,225193	123,994584
MLP	(100, 80, 60, 40, 20)	0,436543	0,516478	0,159978	0,244288	0,317206	468,671718

No cenário inverso (treinamento no NSL-KDD e teste no UNSW-NB15), houve uma leve melhora no desempenho da MLP (principalmente no modelo com duas camadas ocultas), que apresentou F1-Score de 63%, ainda que o AUC-ROC tenha permanecido baixo (aproximadamente 22%). Essa discrepância entre F1-Score e AUC-ROC indica que, apesar da rede neural

ter identificado corretamente parte dos registros de ataque, sua capacidade discriminativa entre classes permaneceu limitada. A RF, por sua vez, mantêm desempenho similar ao observado no cenário anterior, reforçando a hipótese de que os padrões de ataque do NSL-KDD não são diretamente transferíveis ao domínio do UNSW-NB15. Isso evidencia a necessidade de estratégias de normalização de domínio (*domain adaptation*) ou de federação entre modelos locais, em vez de simples reuso direto de modelos treinados.

Quando os modelos foram treinados sobre a união dos dois *datasets* filtrados pelas *features* comuns, o desempenho aumentou consideravelmente. Os resultados desse experimento aparecem descritos na Tabela 56.

Tabela 56 – Modelos Treinados nos Dois *Datasets* e Testados no UNSW-NB15

Modelo	**Árvores/Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-Score***	**AUC-ROC**	**Tempo (s)**
RF	50	0,810451	0,956874	0,698512	0,807531	0,923567	20,008213
RF	100	0,809387	0,955108	0,697966	0,806537	0,924186	39,983265
MLP	(32, 16)	0,796034	0,950049	0,677316	0,790829	0,915435	302,952458
MLP	(100, 80, 60, 40, 20)	0,803221	0,959103	0,683472	0,798162	0,878189	1276,117517

As acurácias passaram para valores próximos de 80%, com AUC-ROC acima de 0,91 tanto para RF quanto para MLP. Esse comportamento demonstra que a inclusão de dados heterogêneos durante o treinamento ampliou a capacidade de generalização dos modelos, permitindo reconhecer de forma mais robusta os padrões presentes no UNSW-NB15. Entre os modelos, a RF com 100 árvores e a MLP mais profunda apresentaram desempenhos praticamente equivalentes, o que indica consistência nos resultados e estabilidade da representação aprendida quando os dados de ambas as origens são utilizados em conjunto

Tabela 57 – Modelos Treinados nos Dois *Datasets* e Testados no NSL-KDD

Modelo	**Árvores/Camadas**	**Acurácia**	**Precisão**	***Recall***	***F1-Score***	**AUC-ROC**	**Tempo (s)**
RF	50	0,810451	0,956874	0,698512	0,807531	0,923567	16,531618
RF	100	0,809387	0,955108	0,697966	0,806537	0,924186	30,421707
MLP	(32, 16)	0,812536	0,946282	0,711057	0,811977	0,925205	215,224644
MLP	(100, 80, 60, 40, 20)	0,832764	0,955564	0,740669	0,834504	0,927577	1465,794247

De forma análoga, os modelos treinados sobre o conjunto combinado e avaliados no NSL-KDD mantiveram altos índices de acurácia e AUC-ROC, alcançando até 83% de acurácia e 93% de AUC-ROC na MLP mais profunda. Esses resultados confirmam que o treinamento conjunto foi capaz de produzir modelos mais adaptáveis e generalistas, reduzindo o impacto das discrepâncias entre os dois *conjuntos*, apesar de ainda estarem limitados por valores não ideais de *Recall*. A leve superioridade da MLP nesse cenário sugere que a rede neural se beneficiou do aumento da diversidade de padrões durante o aprendizado, ajustando-se melhor às variações de comportamento de tráfego.

Os resultados obtidos evidenciam que, quando treinados isoladamente, os modelos apresentam baixa capacidade de generalização entre domínios, o que inviabiliza o reuso direto de um modelo treinado em um *dataset* para classificar registros de outro. Entretanto, o treinamento combinado, análogo a uma etapa de fusão de conhecimento federado, proporciona ganhos expressivos de desempenho, aproximando os modelos do comportamento ideal esperado em sistemas

federados, apesar do conjunto de atributos utilizados para o experimentos ser bastante limitado. Isso reforça a hipótese de que a federação de modelos locais, cada um treinado em diferentes domínios (como UNSW-NB15 e NSL-KDD), pode ser uma estratégia eficaz para melhorar a detecção de ataques em ambientes distribuídos, sem exigir a centralização dos dados.

5 Conclusão

Este trabalho apresentou uma análise comparativa detalhada entre os *datasets* UNSW-NB15 e NSL-KDD, avaliando o impacto de técnicas de normalização e redução de dimensionalidade na detecção de intrusões por meio de modelos de *Machine Learning*. Além de evidenciar os efeitos práticos dessas técnicas, os resultados obtidos oferecem subsídios diretos para pesquisas em Aprendizado Federado, ao demonstrar como diferentes estratégias de pré-processamento afetam o desempenho e a generalização dos modelos.

Os experimentos realizados com os algoritmos *Random Forest* (RF) e *Multilayer Perceptron* (MLP) mostraram que o impacto das técnicas varia conforme o conjunto de dados utilizado, ressaltando desafios e oportunidades para abordagens futuras de federação. A análise comparativa entre os *datasets* evidenciou diferenças estruturais significativas: o UNSW-NB15, mais recente e representativo do tráfego moderno, apresentou maior complexidade, múltiplas classes e desbalanceamento entre categorias; já o NSL-KDD, embora mais antigo e limitado, demonstrou maior estabilidade experimental, devido à sua simplicidade e distribuição mais uniforme.

Observou-se ainda que a divisão fixa entre treino e teste fornecida no UNSW-NB15 pode restringir a capacidade de generalização dos modelos, diferentemente do NSL-KDD, cuja separação aleatória em tempo de execução proporciona maior flexibilidade experimental. Os experimentos complementares indicam que a redistribuição aleatória das instâncias de treino e teste tende a melhorar o desempenho pós-pré-processamento, sugerindo que a estratégia de particionamento dos dados é um fator crítico a ser considerado em estudos futuros.

Além das análises individuais, foram conduzidos experimentos de avaliação cruzada e treinamento conjunto entre os *datasets* UNSW-NB15 e NSL-KDD. Esses testes demonstraram que modelos treinados isoladamente possuem baixa capacidade de generalização entre domínios, mas que o treinamento conjunto, análogo a uma etapa inicial de federação, aumentou significativamente a acurácia e o AUC-ROC. Esse resultado indica que a integração de dados heterogêneos pode fortalecer a robustez dos modelos em contextos distribuídos de IDS.

Outro aspecto relevante foi a análise detalhada dos atributos do UNSW-NB15, contemplando sua origem, forma de coleta e cálculo. Essa investigação mostrou que a forma de geração e descrição dos dados influencia diretamente a dificuldade de classificação, tornando o UNSW-NB15 um conjunto mais representativo da complexidade do tráfego real em comparação ao NSL-KDD. Tal constatação reforça a importância de uma seleção criteriosa de atributos em sistemas de detecção de intrusões, especialmente quando se busca compatibilidade entre diferentes *datasets*.

No que se refere às técnicas de pré-processamento, a normalização mostrou-se essencial para estabilizar o treinamento do MLP e aprimorar o desempenho do RF em alguns cenários. Já a redução de dimensionalidade, por meio do PCA e MDI, demonstrou potencial para otimizar a eficiência computacional e reduzir redundâncias. Contudo, observou-se que, no UNSW-NB15,

uma redução excessiva de atributos pode limitar a capacidade dos modelos de distinguir adequadamente as múltiplas classes de ataque — o que reforça a necessidade de aplicar tais técnicas de forma criteriosa e adaptada às características de cada conjunto.

A partir desses achados, conclui-se que a utilização de múltiplos *datasets* em conjunto, no contexto de Aprendizado Federado, requer estratégias de pré-processamento que respeitem as especificidades de cada fonte de dados. O UNSW-NB15 reflete melhor a complexidade de tráfegos modernos, demandando abordagens robustas de adaptação local e de transferência de aprendizado, enquanto o NSL-KDD pode atuar como uma base mais estável para testes de federação.

Embora os experimentos de federação tenham sido conduzidos de forma controlada e limitados a subconjuntos de atributos comuns, eles representam uma primeira etapa de avaliação empírica da transferência de conhecimento entre domínios distintos. Para a viabilização de uma federação entre IDSs treinados em bases diferentes, torna-se indispensável o desenvolvimento de métodos de harmonização de atributos e mecanismos de aprendizado capazes de lidar com a heterogeneidade dos dados.

Por fim, os resultados obtidos reforçam que a combinação entre normalização, redução de dimensionalidade e aprendizado supervisionado possui grande potencial para aprimorar o desempenho de sistemas IDS. Entretanto, também evidenciam que essas técnicas não devem ser aplicadas de forma genérica, mas ajustadas de acordo com os objetivos do sistema e as características particulares dos dados. Essa reflexão é especialmente relevante em cenários de aprendizado federado, nos quais a heterogeneidade entre participantes pode afetar diretamente a qualidade do modelo global. Assim, este estudo reforça a importância da preparação criteriosa dos dados em IDS baseados em ML e abre caminho para soluções federadas mais robustas e alinhadas à complexidade crescente dos tráfegos de rede modernos

5.1 Trabalhos Futuros

Embora os experimentos realizados neste trabalho possibilitem uma análise abrangente do impacto de técnicas de normalização e redução de dimensionalidade sobre os *datasets* UNSW-NB15 e NSL-KDD, diversas possibilidades de aprofundamento permanecem em aberto. Primeiramente, destaca-se a necessidade de explorar métodos de avaliação mais robustos. A técnica de *holdout*, utilizada neste trabalho, mostrou-se eficiente em cenários exploratórios para o UNSW-NB15, mas métodos de validação cruzada, como o *K-Fold Cross-Validation*, podem fornecer estimativas estatisticamente mais consistentes e reduzir a influência de variações decorrentes da escolha do particionamento. Além disso, a composição fixa dos conjuntos disponibilizados pelos autores do *dataset* pode limitar a generalização dos modelos. Resultados preliminares apresentados no Apêndice H indicam que a redistribuição aleatória das instâncias de treino e teste tende a melhorar o desempenho após o pré-processamento. Uma análise mais aprofundada desse aspecto pode elucidar melhor o real potencial do UNSW-NB15 em comparação ao NSL-KDD.

Outro ponto relevante de análise consiste na avaliação da detecção por classe de ataque,

considerando que este trabalho apresentou os resultados unificados em um cenário com muitas classes. A investigação detalhada do desempenho dos modelos para cada categoria pode fornecer informações importantes sobre quais tipos de ataques são mais difíceis de identificar, além de evidenciar possíveis vieses introduzidos pelo pré-processamento e pelas técnicas de redução de dimensionalidade.

Além disso, deve-se aprofundar a análise da sensibilidade dos algoritmos de aprendizado em relação à inicialização de parâmetros. Muitos modelos, como o MLP, podem iniciar com pesos aleatórios que influenciam diretamente os resultados finais. O uso de diferentes sementes de inicialização (*random seeds*) pode aumentar a reprodutibilidade dos experimentos e oferecer uma visão mais robusta do comportamento dos modelos.

No caso específico do *Random Forest*, este trabalho considerou o número de árvores como parâmetro de variação, mas existem outras dimensões relevantes a serem exploradas. A profundidade máxima das árvores, os critérios de divisão, o número mínimo de amostras por folha e parâmetros relacionados ao custo computacional ("parâmetro de energia") podem influenciar significativamente o desempenho e a eficiência do modelo. Investigar esses aspectos pode levar a um entendimento mais completo sobre o comportamento do algoritmo em diferentes contextos de tráfego de rede.

Outro ponto de investigação futura envolve técnicas de balanceamento de dados. O UNSW-NB15 apresenta uma distribuição desbalanceada entre suas classes de ataque, o que pode prejudicar a capacidade dos modelos em detectar ataques menos frequentes. Métodos como o *Synthetic Minority Oversampling Technique* (SMOTE), *Random Undersampling* e abordagens híbridas podem contribuir para mitigar esse problema e aumentar a eficácia dos classificadores.

No contexto da Federação de Máquinas de Aprendizado, este trabalho já realizou experimentos iniciais de avaliação cruzada e treinamento conjunto entre os *datasets* UNSW-NB15 e NSL-KDD, representando uma etapa preliminar de transferência de conhecimento entre domínios. Como avanço futuro, propõe-se a implementação de um ambiente de aprendizado federado completo, explorando tanto a federação horizontal (com atributos comuns) quanto abordagens de transferência federada, possivelmente utilizando um número maior de *features* para análise.

Além disso, um aspecto interessante a ser investigado em ambientes federados diz respeito ao desenvolvimento de modelos de aprendizado capazes de distinguir entre diferentes classes de ataque, o que exige um tratamento especial para a representação e compatibilização das diferentes classes de ataque em cada *dataset* envolvido.

Por fim, vislumbra-se a aplicação prática de sistemas federados de IDS em ambientes reais, indo além da mera detecção e classificação de ataques, de modo a evoluírem para sistemas capazes de prever e mitigar proativamente ameaças em redes. O desenvolvimento de sistemas com a capacidade de identificar e prevenir ataques em tempo real configura um caminho promissor para avanços significativos na área de segurança de redes.

Referências

ABDULGANIYU, O.; TCHAKOUCHT, T. A.; SAHEED, Y. K. A Systematic Literature Review for Network Intrusion Detection System (IDS). **International Journal of Information Security**, v. 22, p. 1125–1162, 2023. Citado 3 vezes nas páginas 12, 15 e 19.

ABREU, G. M. **Análise RF MLP UNSW-NB15**. 2025. <https://github.com/gabriel-abreu542/Analise_RF_MLP_UNSW_NB15>. Repositório GitHub. Acesso em: 10 out. 2025. Citado na página 36.

ANDERSON, J. P. **Computer Security Technology Planning Study**. [S.l.], 1972. Citado na página 16.

ANDERSON, J. P. **Computer Security Threat Monitoring and Surveillance**. [S.l.], 1980. Citado na página 16.

APARECIDO, A. **Inteligência Artificial: Transformando o Monitoramento de Redes**. 2024. Disponível em: <https://www.digitizeme.blog/intelig%C3%AAncia-artificial-transformando-o-monitoramento-de-redes>. Citado na página 13.

ARMELLINI, A. **Inteligência em Gestão de Redes e Segurança de Dados**. 2024. <https://tiinside.com.br/08/02/2024/inteligencia-em-gestao-de-redes-e-seguranca-de-dados/>. Acesso em: 2024-07-29. Citado na página 13.

ASHFAQ, R. A. R.; WANG, X.-Z.; HUANG, J. Z.; ABBAS, H.; HE, Y.-L. Fuzziness Based Semi-Supervised Learning Approach for Intrusion Detection System. **Information Sciences**, v. 378, p. 484–497, 2017. Citado na página 12.

ASHOOR, A. S.; GORE, S. Importance of Intrusion Detection System. **International Journal of Scientific Engineering Research**, v. 2, 2010. Citado 2 vezes nas páginas 15 e 16.

BREIMAN, L. Random Forests. **Machine Learning**, v. 45, p. 5–32, 2001. Citado 2 vezes nas páginas 18 e 19.

CHUMERIN, N.; HULLE, M. V. 2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing. In: **Comparison of Two Feature Extraction Methods Based on Maximization of Mutual Information**. [S.l.: s.n.], 2006. Citado na página 18.

CUNNINGHAM, R. K.; LIPPMANN, R. P.; FRIED, D. J.; GARFINKEL, S. L.; GRAF, I.; KENDALL, K. R.; WEBSTER, S. E.; WYSCHOGROD, D.; ZISSMAN, M. A. **Evaluating Intrusion Detection Systems Without Attacking Your Friends: The 1998 DARPA Intrusion Detection Evaluation**. [S.l.], 1999. Citado na página 17.

DENNING, D.; NEUMANN, P. G. Requirements and Model for IDES - A Real-time Intrusion-detection Expert System. **SRI International**, 1985. Citado na página 16.

FILHO, M. 2025. Citado na página 40.

GAO, D.; LIU, Y.; HUANG, A.; JU, C.; YU, H.; YANG, Q. 2019 IEEE International Conference on Big Data (Big Data). In: **2019 IEEE International Conference on Big Data (Big Data)**. [S.l.: s.n.], 2019. Citado na página 24.

HAMID, Y.; RANGANATHAN, B.; JOURNAUX, L.; MUTHUKUMARASAMY, S. Benchmark Datasets for Network Intrusion Detection: A Review. **International Journal of Network Security**, 2018. Citado 2 vezes nas páginas 6 e 77.

HARB, H.; ZAGHROT, A.; GOMAA, M.; DESUKY, A. S. Selecting Optimal Subset of Features for Intrusion Detection Systems. **Advances in Computational Sciences and Technology**, 2011. Citado 2 vezes nas páginas 5 e 32.

HASAN, B. M. S.; ABDULAZEEZ, A. M. A Review of Principal Component Analysis Algorithm for Dimensionality Reduction. **Journal of Soft Computing and Data Mining**, v. 2, p. 20–30, 2021. Citado 2 vezes nas páginas 18 e 45.

KAIROUZ, P. et al. Advances and Open Problems in Federated Learning. **Foundations and Trends® in Machine Learning**, 2021. Citado na página 23.

KOCHER, G.; KUMAR, G. Analysis of Machine Learning Algorithms with Feature Selection for Intrusion Detection using UNSW-NB15 Dataset. **International Journal of Network Security Its Applications**, v. 13, 2021. Citado 3 vezes nas páginas 19, 26 e 36.

KONEčNý, J.; MCMAHAN, H. B.; RAMAGE, D.; RICHTáRIK, P. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. **University of Edinburgh**, 2016. Citado na página 23.

KUNHARE, N.; TIWARI, R. Study of the Attributes Using Four Class Labels on KDD99 and NSL-KDD Datasets with Machine Learning Techniques. In: **2018 8th International Conference on Communication Systems and Network Technologies (CSNT)**. [S.l.: s.n.], 2018. p. 127–131. Citado 2 vezes nas páginas 6 e 78.

MARKOVIC, T.; LEON, M.; BUFFONI, D.; PUNNEKKAT, S. Random Forest Based on Federated Learning for Intrusion Detection. In: **Artificial Intelligence Applications and Innovations**. Cham: Springer International Publishing, 2022. p. 132–144. Citado 2 vezes nas páginas 12 e 26.

MEDINA-ARCO, J. G.; MAGAN-CARRIÓN, R.; RODRÍGUEZ-GÓMEZ, R. A.; GARCÍA-TEODORO, P. M. Methodology for the Detection of Contaminated Training Datasets for Machine Learning-Based Network Intrusion-Detection Systems. **Sensors**, v. 24, 2024. Citado 3 vezes nas páginas 12, 16 e 17.

MOUSTAFA, N. UNSW-NB15: A Comprehensive Data set for Network Intrusion Detection systems. In: **Proceedings of the Military Communications and Information Systems Conference (MilCIS)**. [S.l.: s.n.], 2015. Citado 7 vezes nas páginas 5, 12, 17, 25, 28, 29 e 30.

MOUSTAFA, N.; SLAY, J. The Evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Data Set and the Comparison with the KDD99 Data Set. **Information Security Journal: A Global Perspective**, 2016. Citado 3 vezes nas páginas 5, 30 e 31.

NETO, F. Q. D. S. **Sistema para Detecção de Intrusão de Botnets Utilizando Aplicações de Machine Learning**. Tese (Doutorado) — Universidade Federal Rural de Pernambuco, 2021. Citado na página 13.

OJHA, V. K.; ABRAHAM, A.; SNášEL, V. Metaheuristic Design of Feedforward Neural Networks: A Review of Two Decades of Research. **Engineering Applications of Artificial Intelligence**, v. 60, p. 97–116, 2017. Citado 2 vezes nas páginas 12 e 18.

OTOUM, Y.; NAYAK, A. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. **Journal of Network and Systems Management**, v. 29, 2021. Citado na página 12.

PAXSON, V. Bro: A System for Detecting Network Intruders in Real-Time. **Computer Networks**, v. 31, 1999. Citado na página 29.

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. Citado 2 vezes nas páginas 14 e 36.

PHONG, L. T.; AONO, Y.; HAYASHI, T.; WANG, L.; MORIAI, S. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. **National Institute of Information and Communications Technology (NICT),**, 2017. Citado na página 24.

POWERS, D. M. W. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness Correlation. **International Journal of Machine Learning Technology**, v. 2, 2008. Citado 2 vezes nas páginas 21 e 22.

REPICI, D. J. **The Comma Separated Value (CSV) File Format: Create or Parse Data in This Popular Pseudo-Standard Format**. 2002. Acesso em 22/01/2025. Disponível em: <https://creativyst.com/Doc/Articles/CSV/CSV01.shtml#Overview>. Citado na página 30.

RING, M.; WUNDERLICH, S.; SCHEURING, D.; LANDES, D.; HOTHO, A. A Survey of Network-Based Intrusion Detection Data Sets. **Computers security**, 2019. Citado na página 17.

SARHAN, M.; LAYEGHY, S.; PORTMANN, M. Towards a Standard Feature Set for Network Intrusion Detection System Datasets. **Mobile Networks and Applications**, v. 27, 2022. Citado na página 17.

SARKER, I. H. Machine Learning: Algorithms, Real-World Applications and Research Directions. **SN Computer Science**, 2021. Citado na página 12.

SIRISHA; ASWADATI; CHAITANYA, K.; KRISHNA, K. V. S. S. R.; KANUMALLI, S. S. Intrusion Detection Models Using Supervised and Unsupervised Algorithms – A Comparative Estimation. **International Journal of Safety and Security Engineering**, v. 11, n. 1, p. 51–58, 2021. Citado 7 vezes nas páginas 25, 36, 48, 60, 61, 62 e 91.

SOKOLOVA, M.; LAPALME, G. A Systematic Analysis of Performance Measures for Classification Tasks. **Information Processing Management**, v. 45, 2009. Citado na página 19.

SPASOJEVIC, A. **O que é PCAP (captura de pacotes)?** 2023. Acesso em 15/01/2025. Disponível em: <https://phoenixnap.pt/gloss%C3%A1rio/captura-de-pacote-pcap>. Citado na página 28.

TAUD, H.; MAS, J.-F. Multilayer Perceptron (MLP). **Geomatic approaches for modeling land change scenarios**, 2018. Citado na página 18.

TAVALLAEE, M.; BAGHER, E.; LU, W.; GHORBANI, A. A. A Detailed Analysis of the KDD CUP 99 Data Set. **Computational Intelligence in Security and Defense Applications**, 2009. Citado 2 vezes nas páginas 17 e 32.

TCPDUMP. **TCPdump and Libpcap**. 2024. Disponível em: <"https://www.tcpdump.org/manpages/tcpdump.1.html">. Citado na página 29.

TEAM, T. pandas development. **Pandas: Python Data Analysis Library**. 2024. <https://pandas.pydata.org/>. Acessadi em: 16/09/2024. Citado na página 14.

THAKKAR, A.; LOHIYA, R. A Review of the Advancement in Intrusion Detection Datasets. **Procedia Computer Science**, v. 167, p. 636–645, 2020. Citado 3 vezes nas páginas 13, 16 e 17.

THANH, H. An approach to reduce data dimension in building effective network intrusion detection systems. **EAI Endorsed Transactions on Context-aware Systems and Applications**, 2018. Citado 3 vezes nas páginas 5, 30 e 36.

TREVISAN, V. **Multiclass classification evaluation with ROC Curves and ROC AUC**. 2022. Disponível em: <https://medium.com/data-science/multiclass-classification-evaluation-with-roc-curves-and-roc-auc-294fd4617e3a>. Citado na página 23.

TRIFONOVA, O.; LOKHOV, P.; ARCHAKOV, A. Metabolic profiling of human blood. **Biomeditsinskaya Khimiya**, v. 60, p. 281–294, 05 2014. Citado 2 vezes nas páginas 4 e 22.

TUFAN, E.; TEZCAN, C.; ACARTURK, C. Anomaly-Based Intrusion Detection by Machine Learning: A Case Study on Probing Attacks to an Institutional Network. **IEEE Access**, v. 9, p. 50078–50092, 01 2021. Citado 3 vezes nas páginas 4, 30 e 36.

TURCATO, A. C. **Desenvolvimento de Método para Detecção de Intrusão em Redes PROFINET Baseado em Técnicas de Aprendizado de Máquina**. Dissertação (Mestrado) — Universidade de São Paulo, 2020. Citado na página 13.

UMAR, M. A.; CHEN, Z.; SHUAIB, K.; LIU, Y. Effects of Feature Selection and Normalization on Network Intrusion Detection. **TechRxiv**, 2024. Citado 3 vezes nas páginas 34, 35 e 36.

WEISBERG, J. **Argus - System and Network Monitoring Software**. 2022. Acesso em 22/01/2025. Disponível em: <http://argus.tcp4me.com/>. Citado na página 29.

WOLD, S.; ESBENSEN, K.; GELADI, P. Principal Component Analysis. **Chemometrics and Intelligent Laboratory Systems**, 1987. Citado na página 18.

YANG, Q.; LIU, Y.; CHEN, T.; TONG, Y. Federated Machine Learning: Concept and Applications. **ACM Trans. Intell. Syst. Technol.**, 2019. Citado 2 vezes nas páginas 23 e 24.

ZOGHI, Z.; SERPEN, G. UNSW-NB15 Computer Security Dataset: Analysis through Visualization. **SECURITY AND PRIVACY**, 2024. Citado na página 31.

Apêndices

APÊNDICE A – Detalhamento do Cálculo de Métricas em Classificação Multiclasse

Em problemas de classificação multiclasse, como os tratados neste trabalho com os *datasets* UNSW-NB15 e NSL-KDD, a Matriz de Confusão apresenta uma estrutura expandida, com dimensões $k \times k$, onde k é o número total de classes.

Por isso, também é preciso compreender como a Matriz de Confusão e os valores TN, FN, FP e TP são calculados em um contexto com mais de duas classes, como é o caso dos *datasets* UNSW-NB15 e NSL-KDD.

Para representar o desempenho do modelo classificador sobre cada classe i do conjunto, usa-se a seguinte notação: i. TP_i: Verdadeiros Positivos da classe i (registros corretamente classificados em sua classe real); ii. FP_i: Falsos Positivos da classe i (registros de outras classes classificados incorretamente como pertencentes à classe i); iii. FN_i: Falsos Negativos da classe i (registros da classe i classificados incorretamente como pertencentes a outra classe); iv. TN_i: Verdadeiros Negativos da classe i (registros de todas as demais classes corretamente classificados como não pertencentes à classe i).

A Tabela 58 ilustra uma Matriz de Confusão genérica com três classes, usada para exemplificar o cálculo de TP_i, FP_i, FN_i e TN_i para cada classe i.

Tabela 58 – Exemplo de Matriz de Confusão para Várias Classes.

Previsto/Real	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	(Célula 1)	(Célula 2)	(Célula 3)
Classe 2	(Célula 4)	(Célula 5)	(Célula 6)
Classe 3	(Célula 7)	(Célula 8)	(Célula 9)

Para cada classe, os valores de Verdadeiros Positivos, Falsos Positivos, Falsos Negativos e Verdadeiros Negativos podem ser compreendidos com base nas posições da Matriz de Confusão. Por exemplo:

Para a Classe 1, o valor de TP_1 corresponde à Célula 1, que representa os registros da Classe 1 corretamente previstos como Classe 1. O valor de FP_1 equivale à soma das Células 2 e 3, que representam os registros das Classes 2 e 3 incorretamente previstos como Classe 1. O valor de FN_1 corresponde à soma das Células 4 e 7, ou seja, registros da Classe 1 incorretamente previstos como Classes 2 e 3. Já o valor de TN_1 corresponde à soma das Células 5, 6, 8 e 9, que são os registros das Classes 2 e 3 corretamente previstos como não pertencentes à Classe 1.

Seguindo a mesma lógica, para a Classe 2, TP_2 é a Célula 5, FP_2 corresponde às Células 4 e 6, FN_2 às Células 2 e 8, e TN_2 às Células 1, 3, 7 e 9. Já para a Classe 3, TP_3 é a Célula 9, FP_3 equivale às Células 7 e 8, FN_3 às Células 3 e 6, e TN_3 às Células 1, 2, 4 e 5.

Esses valores são utilizados para calcular as métricas individuais por classe, conforme descrito na Seção 2.4.1. Posteriormente, os resultados são agregados por meio de uma média ponderada, atribuindo a cada classe um peso proporcional à sua frequência no conjunto de dados.

Essa abordagem assegura uma avaliação mais justa e equilibrada do desempenho dos modelos, sobretudo em cenários com desbalanceamento entre classes, comuns em tarefas de detecção de intrusão.

APÊNDICE B – Descrição Detalhada dos Atributos do *Dataset* UNSW-NB15

Tabela 59 – Descrição dos atributos do *dataset* UNSW-NB15(HAMID et al., 2018).

ID	Nome	Descrição
1	srcip	Endereço IP do dispositivo fonte
2	sport	Número da porta de conexão do dispositivo fonte
3	dstip	Endereço IP do dispositivo destino
4	dsport	Número da porta de conexão do dispositivo destino
5	proto	Protocolo de comunicação (TCP, UDP, ICMP, ...)
6	state	Indica o estado da conexão e o protocolo associado ou (-), se o protocolo não usa estado
7	dur	Duração total do registro da comunicação
8	sbytes	Bytes enviados do dispositivo origem para o destino
9	dbytes	Bytes enviados do destino para o dispositivo origem
10	sttl	Valor "time to live"dos pacotes enviados pela origem
11	dttl	Valor "time to live"dos pacotes recebidos pelo destino
12	sloss	Pacotes enviados pelo disp. origem que foram retransmitidos ou descartados
13	dloss	Pacotes enviados pelo disp. destino que foram retransmitidos ou descartados
14	service	Serviço utilizado (http, ftp, smtp, ssh, dns, ftp-data, irc ou (-), se não usa nenhum serviço)
15	Sload	Taxa de bits/segundo da origem
16	Dload	Taxa de bits/segundo do destino
17	Spkts	Número de pacotes enviados da origem para o destino
18	Dpkts	Número de pacotes enviados do destino para a origem
19	swin	Valor da janela TCP anunciada pela origem (usado para controle de fluxo)
20	dwin	Valor da janela TCP anunciada pelo destino
21	stcpb	Número sequencial TCP base do disp. de origem
22	dtcpb	Número sequencial TCP base do disp. de destino
23	smeansz	Tamanho médio do pacote transmitido pelo disp. de origem
24	dmeansz	Tamanho médio do pacote transmitido pelo disp. de destino
25	trans_depth	Profundidade do pipeline em uma conexão HTTP (quantidade de requisições/respostas HTTP em uma mesma conexão)
26	res_bdy_len	Tamanho dos dados transferidos sem compressão pelo serviço HTTP do servidor
27	Sjit	Jitter* da origem (em milissegundos)
28	Djit	Jitter* do destino (em milissegundos)
29	Stime	Horário de início do registro
30	Ltime	Horário do término do registro
31	Sintpkt	Tempo médio entre o envio de pacotes enviados pela origem
32	Dintpkt	Tempo médio entre o envio de pacotes enviados pelo destino
33	tcprtt	RTT* da conexão TCP (soma dos atributos synack e ackdat)
34	synack	Tempo entre o envio do pacote SYN e recebimento do pacote SYN_ACK pela origem
35	ackdat	Tempo de estabelecimento da conexão TCP, tempo entre o envio do SYN_ACK e o recebimento do pacote ACK pelo destino
36	is_sm_ips_ports	Indica se os IPs e portas de origem e destino são iguais (1) ou não (0)
37	ct_state_ttl	Número de ocorrências do estado (atributo state) em intervalos específicos de TTL de origem/destino
38	ct_flw_http_mthd	Número de fluxos com métodos do serviço HTTP ("GET"e "POST", por exemplo)
39	is_ftp_login	Indica se a sessão FTP é acessada por usuário e senha (1) ou não (0)
40	ct_ftp_cmd	Número de fluxos que possuem um comando na sessão FTP
41	ct_srv_src	Número de conexões com o mesmo serviço e endereço de origem nas últimas 100 conexões
42	ct_srv_dst	Número de conexões com o mesmo serviço e endereço de destino nas últimas 100 conexões
43	ct_dst_ltm	Número de conexões com o mesmo endereço de destino nas últimas 100 conexões
44	ct_src_ltm	Número de conexões com o mesmo endereço de origem nas últimas 100 conexões
45	ct_src_dport_ltm	Número de conexões com o mesmo endereço de origem e porta de destino nas últimas 100 conexões
46	ct_dst_sport_ltm	Número de conexões com o mesmo endereço de destino e porta de origem nas últimas 100 conexões
47	ct_dst_src_ltm	Número de conexões com o mesmo endereço de origem e endereço de destino nas últimas 100 conexões
48	attack_cat	O nome da categoria de ataque (Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode e Worms)
49	Label	Identifica se é um comportamento normal de rede (0) ou um registro de ataque (1)

APÊNDICE C – Descrição Detalhada dos Atributos do *Dataset* NSL-KDD

Tabela 60 – Descrição dos atributos do *dataset* NSL-KDD (KUNHARE; TIWARI, 2018).

ID	Atributo	Descrição
1	duration	Duração da conexão em segundos.
2	protocol_type	Tipo de protocolo utilizado (TCP, UDP, etc.).
3	service	Serviço de destino da conexão (http, ftp, etc.).
4	flag	Status da conexão (SF, S0, REJ, etc.).
5	src_bytes	Bytes enviados da origem para o destino.
6	dst_bytes	Bytes recebidos pelo destino.
7	land	Se a conexão é entre hosts com mesmo IP e porta.
8	wrong_fragment	Número de fragmentos de pacotes incorretos.
9	urgent	Número de pacotes urgentes enviados.
10	hot	Número de acessos a arquivos do sistema.
11	num_failed_logins	Número de tentativas de login falhas.
12	logged_in	Se a sessão foi autenticada com sucesso.
13	num_compromised	Número de acessos comprometidos.
14	root_shell	Se foi obtido acesso root.
15	su_attempted	Número de tentativas de comando 'su'.
16	num_root	Número de comandos executados como root.
17	num_file_creations	Número de operações de criação de arquivos.
18	num_shells	Número de acessos a shells abertos.
19	num_access_files	Número de operações em arquivos sensíveis.
20	num_outbound_cmds	Número de comandos outbound executados.
21	is_host_login	Se é um login em um host específico.
22	is_guest_login	Se é um login de convidado.
23	count	Número de conexões feitas para o mesmo host.
24	srv_count	Número de conexões feitas para o mesmo serviço.
25	serror_rate	Taxa de conexões com erros SYN.
26	srv_serror_rate	Taxa de conexões com erros SYN para o mesmo serviço.
27	rerror_rate	Taxa de conexões com erros RST.
28	srv_rerror_rate	Taxa de conexões com erros RST para o mesmo serviço.
29	same_srv_rate	Taxa de conexões para o mesmo serviço.
30	diff_srv_rate	Taxa de conexões para serviços diferentes.
31	srv_diff_host_rate	Taxa de conexões para diferentes hosts dentro do mesmo serviço.
32	dst_host_count	Número de conexões para o mesmo host de destino.
33	dst_host_srv_count	Número de conexões para o mesmo serviço no host de destino.
34	dst_host_same_srv_rate	Taxa de conexões para o mesmo serviço no host de destino.
35	dst_host_diff_srv_rate	Taxa de conexões para serviços diferentes no host de destino.
36	dst_host_same_src_port_rate	Taxa de conexões para a mesma porta de origem no host de destino.
37	dst_host_srv_diff_host_rate	Taxa de conexões para diferentes hosts dentro do mesmo serviço de destino.
38	dst_host_serror_rate	Taxa de conexões com erros SYN para o host de destino.
39	dst_host_srv_serror_rate	Taxa de conexões com erros SYN para o mesmo serviço no host de destino.
40	dst_host_rerror_rate	Taxa de conexões com erros RST para o host de destino.
41	dst_host_srv_rerror_rate	Taxa de conexões com erros RST para o mesmo serviço no host de destino.

APÊNDICE D – Código: Análise do Desempenho das Máquinas RF e MLP sobre o *Dataset* UNSW-NB15 Antes e Depois da Normalização

```
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import roc_auc_score
from imblearn.over_sampling import SMOTE
from sklearn import metrics
import numpy as np
import pandas as pd
import time
import sys

def label_encode(treino, teste):
    categorical_columns = treino.select_dtypes(include=['object']).columns
    for col in categorical_columns:
        le = LabelEncoder()
        le.fit(list(treino[col].astype(str).values) + list(teste[col].astype(str).values))
        treino[col] = le.transform(list(treino[col].astype(str).values))
        teste[col] = le.transform(list(teste[col].astype(str).values))
    return treino, teste

def normalizacao(treino, teste, scaler):
    # Separar os dados em X e y
    X_treino = treino.drop('attack_cat', axis=1)
    y_treino = treino['attack_cat']

    X_teste = teste.drop('attack_cat', axis=1)
    y_teste = teste['attack_cat']

    # Aplicar o MinMaxScaler nas variaveis de entrada
    X_treino_normalizado = scaler.fit_transform(X_treino)
    X_teste_normalizado = scaler.transform(X_teste)

    # Retornar os dados normalizados, mantendo as classes de ataque originais
    treino_normalizado = pd.DataFrame(X_treino_normalizado, columns=X_treino.columns)
    teste_normalizado = pd.DataFrame(X_teste_normalizado, columns=X_teste.columns)

    # Adicionar as colunas de attack_cat de volta aos datasets normalizados
    treino_normalizado['attack_cat'] = y_treino
    teste_normalizado['attack_cat'] = y_teste

    return treino_normalizado, teste_normalizado

def avaliar_dataset_RF(treino, teste, n_arvores):
```

```
    X_treino = treino.drop('attack_cat', axis=1)
    y_treino = treino['attack_cat']
    X_teste = teste.drop('attack_cat', axis=1)
    y_teste = teste['attack_cat']

    print("Classes no treino:", np.unique(y_treino))
    print("Classes no teste :", np.unique(y_teste))

    modelo = RandomForestClassifier(random_state=42, n_estimators=n_arvores)

    # Treinar o modelo
    inicio = time.time()
    modelo.fit(X_treino, y_treino)
    fim = time.time()

    tempo = fim - inicio

    # Fazer previsoes
    y_pred = modelo.predict(X_teste)

    acc = metrics.accuracy_score(y_teste, y_pred)
    pre = metrics.precision_score(y_teste, y_pred, average='weighted', zero_division=np.nan)
    rec = metrics.recall_score(y_teste, y_pred, average='weighted', zero_division=np.nan)
    f1 = metrics.f1_score(y_teste, y_pred, average='weighted')

    print("\nRANDOM FOREST:")
    print(f"({n_arvores} Árvores)")
    print(f"Resultados:")

    print(f"Accuracy: {acc}")
    print(f"Precision: {pre}")
    print(f"Recall: {rec}")
    print(f"F1-score: {f1}")

    auc_roc = roc_auc_score(y_teste, modelo.predict_proba(X_teste), average='weighted', multi_class='ovr')
    print("AUC-ROC: ", auc_roc)

    print(f"Tempo de treino: {tempo} segundos")


def avaliar_dataset_MLP(treino, teste, camadas):
    X_treino = treino.drop('attack_cat', axis=1)
    y_treino = treino['attack_cat']
    X_teste = teste.drop('attack_cat', axis=1)
    y_teste = teste['attack_cat']

    modelo = MLPClassifier(hidden_layer_sizes=(camadas))

    inicio = time.time()
    # Treinar o modelo
    modelo.fit(X_treino, y_treino)
    fim = time.time()

    tempo = fim - inicio

    # Fazer previsoes
    y_pred = modelo.predict(X_teste)

    acc = metrics.accuracy_score(y_teste, y_pred)
    pre = metrics.precision_score(y_teste, y_pred, average='weighted', zero_division=np.nan)
```

```
    rec = metrics.recall_score(y_teste, y_pred, average='weighted', zero_division=np.nan)
    f1 = metrics.f1_score(y_teste, y_pred, average='weighted')

    print("\nMULTI-LAYER PERCEPTRON:")
    print(f"(Camadas: {camadas})")
    print(f"Resultados :")

    print(f"Accuracy: {acc}")
    print(f"Precision: {pre}")
    print(f"Recall: {rec}")
    print(f"F1-score: {f1}")

    auc_roc = roc_auc_score(y_teste, modelo.predict_proba(X_teste), average='weighted', multi_class='ovr')
    print("AUC-ROC: ", auc_roc)
    print(f"Tempo de treino: {tempo} segundos")

# Carregar os dados
treino = pd.read_csv(r"UNSW_NB15\UNSW_NB15_training-set.csv")
teste = pd.read_csv(r"UNSW_NB15\UNSW_NB15_testing-set.csv")

# Remover as colunas 'label' e 'id'
drop_columns = ['label', 'id']
treino_orig = treino.drop(drop_columns, axis=1, inplace=False)
teste_orig = teste.drop(drop_columns, axis=1, inplace=False)

colunas_vazias_treino = treino.columns[treino.isnull().all()].tolist()
colunas_vazias_teste = teste.columns[teste.isnull().all()].tolist()

print("\nColunas VAZIAS:")
print("Treino:")
print(colunas_vazias_treino)
print("Teste:")
print(colunas_vazias_teste)

if(len(colunas_vazias_treino) > 0):
    treino = treino.drop(colunas_vazias_treino, axis=1, inplace=False)
if(len(colunas_vazias_teste) > 0):
    teste = teste.drop(colunas_vazias_teste, axis=1, inplace=False)


# Aplicar Label Encoding
treino_orig, teste_orig = label_encode(treino_orig, teste_orig)

print("\nRESULTADOS ANTES DA NORMALIZAÇÃO:")

avaliar_dataset_MLP(treino_orig, teste_orig, (10,10,20))
avaliar_dataset_MLP(treino_orig, teste_orig, (20,40,20))
avaliar_dataset_MLP(treino_orig, teste_orig, (40,60,40))
avaliar_dataset_MLP(treino_orig, teste_orig, (60,40,60))
avaliar_dataset_MLP(treino_orig, teste_orig, (60,80,60))
avaliar_dataset_MLP(treino_orig, teste_orig, (80,100,80))
avaliar_dataset_MLP(treino_orig, teste_orig, (100,120,100))
avaliar_dataset_MLP(treino_orig, teste_orig, (16,8))
avaliar_dataset_MLP(treino_orig, teste_orig, (32,16))
avaliar_dataset_MLP(treino_orig, teste_orig, (64,32))
avaliar_dataset_MLP(treino_orig, teste_orig, (100, 80, 60, 40, 20))
avaliar_dataset_MLP(treino_orig, teste_orig, (128, 128, 64, 64))


avaliar_dataset_RF(treino_orig, teste_orig, 10)
```

```
avaliar_dataset_RF(treino_orig, teste_orig, 25)
avaliar_dataset_RF(treino_orig, teste_orig, 50)
avaliar_dataset_RF(treino_orig, teste_orig, 75)
avaliar_dataset_RF(treino_orig, teste_orig, 100)
avaliar_dataset_RF(treino_orig, teste_orig, 125)
avaliar_dataset_RF(treino_orig, teste_orig, 150)

print("=============================================================================================================")
print("RESULTADOS APÓS A NORMALIZAÇÃO MINMAX")
# Aplicar normalização MinMaxScaler
treino_normalizado, teste_normalizado = normalizacao(treino_orig, teste_orig, MinMaxScaler())

avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (16,8))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (32,16))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (64,32))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (10,20,10))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (20,40,20))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (40,60,40))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (60,40,60))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (60,80,60))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (80,100,80))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (100,120,100))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (100, 80, 60, 40, 20))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (128, 128, 64, 64))

avaliar_dataset_RF(treino_normalizado, teste_normalizado, 10)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 25)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 50)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 75)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 100)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 125)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 150)

print("=============================================================================================================")
print("RESULTADOS APÓS A NORMALIZAÇÃO STANDARD (Z-SCORE)")
# Aplicar normalização StandardScaler
treino_normalizado, teste_normalizado = normalizacao(treino_orig, teste_orig, StandardScaler())

avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (10,20,10))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (20,40,20))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (40,60,40))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (60,40,60))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (60,80,60))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (80,100,80))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (100,120,100))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (16,8))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (32,16))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (64,32))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (100, 80, 60, 40, 20))
avaliar_dataset_MLP(treino_normalizado, teste_normalizado, (128, 128, 64, 64))

avaliar_dataset_RF(treino_normalizado, teste_normalizado, 10)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 25)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 50)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 75)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 100)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 125)
avaliar_dataset_RF(treino_normalizado, teste_normalizado, 150)
```

APÊNDICE E – Código: Análise Final do Desempenho das Máquinas RF e MLP Sobre o *Dataset* UNSW-NB15 após MDI e PCA

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import roc_auc_score
from sklearn import metrics
import numpy as np
import pandas as pd
import time
import sys

def avaliar_dataset_RF(treino, teste, n_arvores):
    X_treino = treino.drop('attack_cat', axis=1)
    y_treino = treino['attack_cat']
    X_teste = teste.drop('attack_cat', axis=1)
    y_teste = teste['attack_cat']

    modelo = RandomForestClassifier(random_state=42, n_estimators=n_arvores)

    inicio = time.time()
    modelo.fit(X_treino, y_treino)
    fim = time.time()

    tempo = fim - inicio

    y_pred = modelo.predict(X_teste)

    acc = metrics.accuracy_score(y_teste, y_pred)
    pre = metrics.precision_score(y_teste, y_pred, average='weighted', zero_division=np.nan)
    rec = metrics.recall_score(y_teste, y_pred, average='weighted', zero_division=np.nan)
    f1 = metrics.f1_score(y_teste, y_pred, average='weighted')

    print("\nRANDOM FOREST:")
    print(f"({n_arvores} Árvores)")
    print(f"Resultados:")

    print(f"Accuracy: {acc}")
    print(f"Precision: {pre}")
    print(f"Recall: {rec}")
    print(f"F1-score: {f1}")

    auc_roc = roc_auc_score(y_teste, modelo.predict_proba(X_teste), average='weighted', multi_class='ovr')
    print("AUC-ROC: ", auc_roc)

    print(f"Tempo de treino: {tempo} segundos")


def avaliar_dataset_MLP(treino, teste, camadas):
    X_treino = treino.drop('attack_cat', axis=1)
    y_treino = treino['attack_cat']
```

```
    X_teste = teste.drop('attack_cat', axis=1)
    y_teste = teste['attack_cat']

    modelo = MLPClassifier(hidden_layer_sizes=(camadas))

    inicio = time.time()
    modelo.fit(X_treino, y_treino)
    fim = time.time()

    tempo = fim - inicio

    y_pred = modelo.predict(X_teste)

    acc = metrics.accuracy_score(y_teste, y_pred)
    pre = metrics.precision_score(y_teste, y_pred, average='weighted', zero_division=np.nan)
    rec = metrics.recall_score(y_teste, y_pred, average='weighted', zero_division=np.nan)
    f1 = metrics.f1_score(y_teste, y_pred, average='weighted')

    print("\nMULTI-LAYER PERCEPTRON:")
    print(f"(Camadas: {camadas})")
    print(f"Resultados :")

    print(f"Accuracy: {acc}")
    print(f"Precision: {pre}")
    print(f"Recall: {rec}")
    print(f"F1-score: {f1}")

    auc_roc = roc_auc_score(y_teste, modelo.predict_proba(X_teste), average='weighted', multi_class='ovr')
    print("AUC-ROC: ", auc_roc)
    print(f"Tempo de treino: {tempo} segundos")

def testes_MLP(treino, teste):
    avaliar_dataset_MLP(treino, teste, (16,8))
    avaliar_dataset_MLP(treino, teste, (32,16))
    avaliar_dataset_MLP(treino, teste, (64,32))
    avaliar_dataset_MLP(treino, teste, (10,20,10))
    avaliar_dataset_MLP(treino, teste, (20,40,20))
    avaliar_dataset_MLP(treino, teste, (40,60,40))
    avaliar_dataset_MLP(treino, teste, (60,40,60))
    avaliar_dataset_MLP(treino, teste, (60,80,60))
    avaliar_dataset_MLP(treino, teste, (80,100,80))
    avaliar_dataset_MLP(treino, teste, (100,120,100))
    avaliar_dataset_MLP(treino, teste, (100, 80, 60, 40, 20))
    avaliar_dataset_MLP(treino, teste, (128, 128, 64, 64))

def testes_RF(treino, teste):
    avaliar_dataset_RF(treino, teste, 10)
    avaliar_dataset_RF(treino, teste, 25)
    avaliar_dataset_RF(treino, teste, 50)
    avaliar_dataset_RF(treino, teste, 75)
    avaliar_dataset_RF(treino, teste, 100)
    avaliar_dataset_RF(treino, teste, 125)
    avaliar_dataset_RF(treino, teste, 150)


UNSW_MinMax_PCA_treino = pd.read_csv(r"UNSW_NB15\UNSW_NB15_test_PCA_MinMax.csv")
UNSW_MinMax_PCA_teste = pd.read_csv(r"UNSW_NB15\UNSW_NB15_train_PCA_MinMax.csv")

UNSW_Z_Score_PCA_treino = pd.read_csv(r"UNSW_NB15\UNSW_NB15_test_PCA_Z_Score.csv")
UNSW_Z_Score_PCA_teste = pd.read_csv(r"UNSW_NB15\UNSW_NB15_train_PCA_Z_Score.csv")
```

```
UNSW_MinMax_MDI_treino = pd.read_csv(r"UNSW_NB15\UNSW_NB15_test_MDI_MinMax.csv")
UNSW_MinMax_MDI_teste = pd.read_csv(r"UNSW_NB15\UNSW_NB15_train_MDI_MinMax.csv")

UNSW_Z_Score_MDI_treino = pd.read_csv(r"UNSW_NB15\UNSW_NB15_test_MDI_Z_Score.csv")
UNSW_Z_Score_MDI_teste = pd.read_csv(r"UNSW_NB15\UNSW_NB15_train_MDI_Z_Score.csv")

datasets_UNSW = [UNSW_MinMax_PCA_treino, UNSW_MinMax_PCA_teste,
                 UNSW_Z_Score_PCA_treino, UNSW_Z_Score_PCA_teste,
                 UNSW_MinMax_MDI_treino, UNSW_MinMax_MDI_teste,
                 UNSW_Z_Score_MDI_treino, UNSW_Z_Score_MDI_teste]

stdout_orig = sys.stdout
with open('resultados/Resultados_Finais/MLP/UNSW_NB15_MinMax_PCA.txt', 'w') as f:
    sys.stdout = f
    print("=============================================================================================================")
    print("RESULTADOS UNSW_NB15-NB15 + MINMAX + PCA")

    testes_MLP(UNSW_MinMax_PCA_treino, UNSW_MinMax_PCA_teste)

sys.stdout = stdout_orig
print("Execução finalizada. Resultados salvos em 'resultados/Resultados_Finais/MLP/UNSW_NB15_MinMax_PCA.txt'")

with open('resultados/Resultados_Finais/RF/UNSW_NB15_MinMax_PCA.txt', 'w') as f:
    sys.stdout = f
    print("=============================================================================================================")
    print("RESULTADOS UNSW_NB15-NB15 + MINMAX + PCA")
    testes_RF(UNSW_MinMax_PCA_treino, UNSW_MinMax_PCA_teste)

sys.stdout = stdout_orig
print("Execução finalizada. Resultados salvos em 'resultados/Resultados_Finais/RF/UNSW_NB15_MinMax_PCA.txt'")

with open('resultados/Resultados_Finais/MLP/UNSW_NB15_Z_Score_PCA.txt', 'w') as f:
    sys.stdout = f
    print("=============================================================================================================")
    print("RESULTADOS UNSW_NB15-NB15 + Z-SCORE + PCA")
    testes_MLP(UNSW_Z_Score_PCA_treino, UNSW_Z_Score_PCA_teste)

sys.stdout = stdout_orig
print("Execução finalizada. Resultados salvos em 'resultados/Resultados_Finais/MLP/UNSW_NB15_Z_Score_PCA.txt'")

with open('resultados/Resultados_Finais/RF/UNSW_NB15_Z_Score_PCA.txt', 'w') as f:
    sys.stdout = f
    print("=============================================================================================================")
    print("RESULTADOS UNSW_NB15-NB15 + Z-SCORE + PCA")
    testes_RF(UNSW_Z_Score_PCA_treino, UNSW_Z_Score_PCA_teste)

sys.stdout = stdout_orig
print("Execução finalizada. Resultados salvos em 'resultados/Resultados_Finais/RF/UNSW_NB15_Z_Score_PCA.txt'")

with open('resultados/Resultados_Finais/MLP/UNSW_NB15_MinMax_MDI.txt', 'w') as f:
    sys.stdout = f
    print("=============================================================================================================")
    print("RESULTADOS UNSW_NB15-NB15 + MINMAX + MDI")
    testes_MLP(UNSW_MinMax_MDI_treino, UNSW_MinMax_MDI_teste)

sys.stdout = stdout_orig
print("Execução finalizada. Resultados salvos em 'resultados/Resultados_Finais/MLP/UNSW_NB15_MinMax_MDI.txt'")

with open('resultados/Resultados_Finais/RF/UNSW_NB15_MinMax_MDI.txt', 'w') as f:
```

```
    sys.stdout = f
    print("=============================================================================================================")
    print("RESULTADOS UNSW_NB15-NB15 + MINMAX + MDI")
    testes_RF(UNSW_MinMax_MDI_treino, UNSW_MinMax_MDI_teste)

sys.stdout = stdout_orig
print("Execução finalizada. Resultados salvos em 'resultados/Resultados_Finais/RF/UNSW_NB15_MinMax_MDI.txt'")

with open('resultados/Resultados_Finais/MLP/UNSW_NB15_Z_Score_MDI.txt', 'w') as f:
    sys.stdout = f
    print("=============================================================================================================")
    print("RESULTADOS UNSW_NB15-NB15 + Z-SCORE + MDI")
    testes_MLP(UNSW_Z_Score_MDI_treino, UNSW_Z_Score_MDI_teste)

sys.stdout = stdout_orig
print("Execução finalizada. Resultados salvos em 'resultados/Resultados_Finais/MLP/UNSW_NB15_Z_Score_MDI.txt'")

with open('resultados/Resultados_Finais/RF/UNSW_NB15_Z_Score_MDI.txt', 'w') as f:
    sys.stdout = f
    print("=============================================================================================================")
    print("RESULTADOS UNSW_NB15-NB15 + Z-SCORE + MDI")
    testes_RF(UNSW_Z_Score_MDI_treino, UNSW_Z_Score_MDI_teste)

sys.stdout = stdout_orig
print("Execução finalizada. Resultados salvos em 'resultados/Resultados_Finais/RF/UNSW_NB15_Z_Score_MDI.txt'")
```

APÊNDICE F – Resultados dos Experimentos com a Classificação Binária

Tabela 61 – RF Antes da Normalização no *Dataset* UNSW-NB15.

Experimento	Árvores	Acurácia	Precisão	Recall	*F1-score*	AUC-ROC	Tempo(s)
1.1	10	0,877860	0,832943	0,973396	0,897709	0,963279	4,154402
1.2	25	0,870561	0,818324	0,983190	0,893213	0,973564	9,256054
1.3	50	0,874022	0,821737	0,984845	0,895928	0,976919	18,551580
1.4	75	0,871472	0,818002	0,985926	0,894148	0,977950	28,863676
1.5	100	0,872686	0,819748	0,985463	0,894999	0,978410	45,225527
1.6	125	0,871981	0,818631	0,985926	0,894524	0,978687	48,048322
1.7	150	0,872613	0,819400	0,985948	0,894992	0,979102	57,263251
Média	**-**	**0,873028**	**0,821255**	**0,983528**	**0,895073**	**0,975416**	**30,194687**

Tabela 62 – RF Antes da Normalização no *Dataset* NSL-KDD.

Experimento	Árvores	Acurácia	Precisão	*Recall*	*F1-score*	AUC-ROC	Tempo (s)
2.1	10	0,792140	0,681762	0,970446	0,800884	0,907510	2,033560
2.2	25	0,781405	0,669911	0,970961	0,792819	0,936525	4,373226
2.3	50	0,787349	0,676249	0,971373	0,797380	0,956905	8,975479
2.4	75	0,781139	0,669481	0,971579	0,792724	0,961249	13,873384
2.5	100	0,772667	0,660597	0,971270	0,786360	0,962019	17,458292
2.6	125	0,772312	0,660204	0,971373	0,786116	0,962174	21,345275
2.7	150	0,772445	0,660388	0,971167	0,786179	0,963249	26,012827
Média	**-**	**0,779922**	**0,668370**	**0,971167**	**0,791780**	**0,949947**	**13,438863**

Tabela 63 – RF Após Normalização *Min-Max* no UNSW-NB15.

Experimento	Árvores	Acurácia	Precisão	Recall	*F1-score*	AUC-ROC	Tempo(s)
3.1	10	0,879136	0,813718	0,978138	0,899114	0,967508	3,427556
3.2	25	0,875079	0,817692	0,984471	0,893364	0,977216	8,402236
3.3	50	0,872730	0,819697	0,984283	0,894076	0,979944	17,068799
3.4	75	0,871265	0,817852	0,985727	0,893977	0,980140	26,134182
3.5	100	0,871263	0,818161	0,985992	0,894027	0,980331	33,566460
3.6	125	0,870949	0,817196	0,986238	0,893794	0,980536	57,132757
3.7	150	0,871156	0,817504	0,986238	0,893945	0,980752	70,355694
Média	**-**	**0,873368**	**0,817403**	**0,984727**	**0,895471**	**0,978918**	**30,869669**

Tabela 64 – RF Após Normalização *Min-Max* no NSL-KDD.

Experimento	Árvores	Acurácia	Precisão	*Recall*	*F1-score*	AUC-ROC	Tempo (s)
4.1	10	0,789434	0,678664	0,970858	0,798881	0,891905	1,689614
4.2	25	0,768941	0,656624	0,971785	0,783706	0,916800	4,238980
4.3	50	0,782825	0,671462	0,970858	0,793870	0,937586	8,230529
4.4	75	0,778345	0,666643	0,970961	0,790526	0,949844	11,902346
4.5	100	0,778966	0,667351	0,970755	0,790955	0,950046	15,988517
4.6	125	0,769695	0,657665	0,970549	0,784045	0,952844	19,842518
4.7	150	0,775861	0,664107	0,970549	0,788604	0,954566	24,706421
Média	**-**	**0,777581**	**0,666274**	**0,970888**	**0,790227**	**0,936799**	**12,942275**

Tabela 65 – RF Após Normalização *Z-Score* no UNSW-NB15.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
5.1	10	0.879585	0.835322	0.973154	0.898985	0.963818	6.988893
5.2	25	0,872662	0,820612	0,983786	0.894821	0,973657	13,528304
5.3	50	0,874132	0,822587	0,936308	0.895989	0,976799	30,801627
5.4	75	0,872091	0,818182	0,985693	0.894577	0,977828	46,054802
5.5	100	0,873294	0,825481	0,985375	0.895435	0,978346	74,058023
5.6	125	0,871931	0,818466	0,986215	0.894527	0,978704	77,878097
5.7	150	0,872990	0,819632	0,986389	0,895331	0,978746	86,392024
Média	**-**	0,873812	0,822040	0,976426	0,895381	0,975128	48,528253

Tabela 66 – RF Após Normalização *Z-Score* no NSL-KDD.

Experimento	**Árvores**	**Acurácia**	**Precisão**	***Recall***	***F1-score***	**AUC-ROC**	**Tempo (s)**
6.1	10	0,836490	0,734945	0,970340	0,836396	0,954770	1,159709
6.2	25	0,821452	0,715488	0,971988	0,824243	0,980675	2,695368
6.3	50	0,827086	0,722409	0,972091	0,828855	0,982281	5,331836
6.4	75	0,821674	0,715628	0,972400	0,824485	0,982687	8,033527
6.5	100	0,826199	0,721170	0,972503	0,828188	0,982161	13,573171
6.6	125	0,820964	0,714729	0,972503	0,823925	0,982237	12,947133
6.7	150	0,825489	0,720290	0,972503	0,827607	0,982577	15,615734
Média	**-**	**0,825622**	**0,720094**	**0,972047**	**0,827386**	**0,966770**	**8,765068**

Tabela 67 – MLP Antes da Normalização no *Dataset* UNSW-NB15.

Experimento	**Camadas**	**Acurácia**	**Precisão**	**Recall**	***F1-score***	**AUC-ROC**	**Tempo(s)**
7.1	(16, 8)	0,758939	0,864979	0,666174	0,752670	0,766958	31,115091
7.2	(32, 16)	0,758769	0,861527	0,669483	0,753460	0,765729	44,958867
7.3	(64, 32)	0,628225	0,597021	0,999272	0,747465	0,596550	182,025707
7.4	(10, 20, 10)	0,755271	0,860493	0,658328	0,747616	0,785014	42,769724
7.5	(20, 40, 20)	0,624017	0,594778	0,995610	0,744683	0,509937	94,608596
7.6	(40, 60, 40)	0,622374	0,593276	0,999092	0,744458	0,589508	114,068118
7.7	(60, 40, 60)	0,623124	0,594487	0,992596	0,743602	0,588105	112,226995
7.8	(60, 80, 60)	0,569512	0,562182	0,999087	0,718744	0,532141	135,924961
7.9	(80, 100, 80)	0,624702	0,594079	0,999961	0,745760	0,592283	122,801164
7.10	(100, 120, 100)	0,558373	0,554913	1.000000	0,713758	0,519054	184.095670
7.11	(128, 128, 64, 64)	0.569268	0.561091	0.999757	0.718782	0.526578	231,131472
7.12	(100, 80, 60, 40, 20)	0,623731	0,594109	0,999404	0,745215	0,590643	220,219262
Média	**-**	**0.643009**	**0.638671**	**0.914934**	**0,739690**	**0,613517**	**126,328838**

Tabela 68 – MLP Antes da Normalização no *Dataset* NSL-KDD.

Experimento	**Camadas**	**Acurácia**	**Precisão**	**Recall**	***F1-score***	**AUC-ROC**	**Tempo(s)**
8.1	(16, 8)	0,775683	0,690177	0,869632	0,769581	0,863270	36,477833
8.2	(32, 16)	0,778211	0,671446	0,949954	0,786780	0,863776	42,175799
8.3	(64, 32)	0,780962	0,674083	0,951601	0,789155	0,815685	38,302904
8.4	(10, 10, 20)	0,744677	0,632407	0,972608	0,766453	0,817539	16,442896
8.5	(20, 40, 20)	0,766812	0,656787	0,960663	0,780180	0,880675	20,145366
8.6	(40, 60, 40)	0,773731	0,662759	0,966533	0,786328	0,807200	56,997005
8.7	(60, 40, 60)	0,723740	0,619273	0,931109	0,743830	0,761566	70,574463
8.8	(60, 80, 60)	0,811125	0,707260	0,958089	0,813785	0,840252	218,446278
8.9	(80, 100, 80)	0,764727	0,651850	0,974050	0,781026	0,801649	93,905518
8.10	(100, 120, 100)	0,738822	0,635079	0,925445	0,753248	0,766748	152,245070
8.11	(100, 80, 60, 40, 20)	0,777236	0,665537	0,970652	0,789646	0,905479	295,250371
8.12	(128, 128, 64, 64)	0,745786	0,633307	0,973535	0,767401	0,918457	236,065214
Média	**-**	**0,765126**	**0,658330**	**0,950323**	**0,777284**	**0,836858**	**106,419060**

Tabela 69 – MLP Após Normalização *Min-Max* no UNSW-NB15.

Experimento	**Camadas**	**Acurácia**	**Precisão**	**Recall**	***F1-score***	**AUC-ROC**	**Tempo(s)**
9.1	(16, 8)	0,873730	0,819938	0,987536	0,895967	0,978905	150,500319
9.2	(32, 16)	0,888233	0,846484	0,973573	0,905591	0,978493	226,735884
9.3	(64, 32)	0,867633	0,825094	0,963933	0,889126	0,971649	579,000652
9.4	(10,20,10)	0,857030	0,798474	0,990272	0,884090	0,979618	106,608185
9.5	(20,40,20)	0,856921	0,800096	0,986654	0,883636	0,979631	222,842960
9.6	(40,60,40)	0,860261	0,810095	0,974698	0,884806	0,972913	562,459052
9.7	(60,40,60)	0,862289	0,815448	0,969249	0,885722	0,971978	664,429004
9.8	(60,80,60)	0,857795	0,808228	0,972470	0,882775	0,970274	700,854806
9.9	(80,100,80)	0,859811	0,810228	0,973374	0,884339	0,965692	794,976184
9.10	(100,120,100)	0,854504	0,803911	0,973109	0,880455	0,967560	934,918990
9.11	(128, 128, 64, 64)	0,858985	0,815513	0,961374	0,882456	0,963664	1.397,509286
9.12	(100, 80, 60, 40, 20)	0,866212	0,822571	0,965212	0,888201	0,971649	579,000652
Média	**-**	**0,858783**	**0,814748**	**0,963867**	**0,880238**	**0,965468**	**859,595792**

Tabela 70 – MLP Após Normalização *Min-Max* no NSL-KDD.

Experimento	**Camadas**	**Acurácia**	**Precisão**	**Recall**	***F1-score***	**AUC-ROC**	**Tempo(s)**
10.1	(16, 8)	0,855350	0,759704	0,971476	0,852637	0,944480	74,461663
10.2	(32, 16)	0,785397	0,673256	0,974977	0,796500	0,956356	175,246999
10.3	(64, 32)	0,799060	0,687758	0,977139	0,807300	0,949061	250,151746
10.4	(10, 20, 10)	0,765969	0,652963	0,974771	0,782056	0,912973	98,048902
10.5	(20, 40, 20)	0,781938	0,671091	0,968386	0,792784	0,946959	76,862791
10.6	(40, 60, 40)	0,794846	0,683557	0,975183	0,803734	0,956204	257,567816
10.7	(60, 40, 60)	0,810016	0,701545	0,972814	0,815205	0,950313	298,933619
10.8	(60, 80, 60)	0,777014	0,664167	0,975698	0,790341	0,949809	300,936646
10.9	(80, 100, 80)	0,796886	0,685834	0,975183	0,805306	0,948161	323,215275
10.10	(100, 120, 100)	0,783357	0,671426	0,973432	0,794704	0,941455	373,662092
10.11	(100, 80, 60, 40, 20)	0,804649	0,695383	0,972505	0,810922	0,937399	364,744460
10.12	(128, 128, 64, 64)	0,795467	0,684116	0,975698	0,804295	0,962704	481,855382
Média	**-**	**0,791010**	**0,685016**	**0,973666**	**0,800468**	**0,948077**	**248,647143**

Tabela 71 – MLP Após Normalização *Z-Score* no UNSW-NB15.

Experimento	**Camadas**	**Acurácia**	**Precisão**	**Recall**	***F1-score***	**AUC-ROC**	**Tempo(s)**
11.1	(16, 8)	0,869188	0,816212	0,983984	0,892280	0,977562	102,081064
11.2	(32, 16)	0,873002	0,826726	0,973352	0,894067	0,974795	200,031525
11.3	(64, 32)	0,846329	0,798208	0,964815	0,873639	0,966425	417,807137
11.4	(10,20,10)	0,878140	0,834467	0,971367	0,897728	0,976458	157,485999
11.5	(20,40,20)	0,864670	0,814953	0,975779	0,888144	0,973533	291,897717
11.6	(40,60,40)	0,848564	0,800512	0,965587	0,875335	0,961093	776,404880
11.7	(60,40,60)	0,856690	0,809556	0,967264	0,881411	0,967214	1.097,332408
11.8	(60,80,60)	0,857297	0,814854	0,958638	0,880918	0,960510	1.122,196092
11.9	(80,100,80)	0,851637	0,809164	0,956013	0,876480	0,961435	1.686,912738
11.10	(100,120,100)	0,853641	0,812542	0,954359	0,877759	0,958898	1.501,780393
11.11	(128, 128, 64, 64)	0,835775	0,806565	0,923123	0,860916	0,943929	1325,731760
11.12	(100, 80, 60, 40, 20)	0,857151	0,817336	0,953697	0,880267	0,960148	1026,096791
Média	**-**	**0,857674**	**0,813425**	**0,962331**	**0,881579**	**0,965167**	**1.617,626417**

Tabela 72 – MLP Após Normalização Z-Score no NSL-KDD.

Experimento	**Camadas**	**Acurácia**	**Precisão**	**Recall**	***F1-score***	**AUC-ROC**	**Tempo(s)**
12.1	(16, 8)	0,780784	0,668456	0,974359	0,792927	0,937309	66,804333
12.2	(32, 16)	0,768142	0,655673	0,972402	0,783229	0,924833	125,151084
12.3	(64, 32)	0,777768	0,665284	0,974256	0,790657	0,921539	149,293077
12.4	(10, 20, 10)	0,777058	0,664328	0,975183	0,790286	0,925202	62,060606
12.5	(20, 40, 20)	0,805625	0,695761	0,975183	0,812109	0,952380	57,176162
12.6	(40, 60, 40)	0,779764	0,667042	0,975801	0,792407	0,920559	185,219241
12.7	(60, 40, 60)	0,767477	0,653606	0,979096	0,783906	0,931376	201,053957
12.8	(60, 80, 60)	0,784865	0,672193	0,977036	0,796441	0,918233	210,504811
12.9	(80, 100, 80)	0,799592	0,688576	0,976316	0,807581	0,914333	305,098648
12.10	(100, 120, 100)	0,773022	0,660562	0,973123	0,786943	0,904467	341,662947
12.11	(128, 128, 64, 64)	0,767921	0,654736	0,975801	0,783659	0,931492	404,682168
12.12	(100, 80, 60, 40, 20)	0,788680	0,677198	0,973432	0,798733	0,947422	307,638964
Média	**-**	**0,780096**	**0,668126**	**0,974978**	**0,792982**	**0,930640**	**199,553726**

APÊNDICE G – Comparação Completa com o Artigo de Sirisha et al.

Tabela 73 – Comparação dos Resultados Sobre o NSL-KDD

Experimento	Acurácia	Precisão	*Recall*	*F1-Score*	AUC-ROC
RF (SIRISHA et al., 2021)	0,765037	0,652543	0,972196	0,780925	0,948926
RF + PCA (*Min-Max*)	0,998023	0,998040	0,998023	0,998012	0,999957
RF + PCA (*Z-Score*)	0,998129	0,998126	0,998129	0,998125	0,999985
RF + MDI (*Min-Max*)	0,998964	0,998956	0,998964	0,998956	0,999988
RF + MDI (*Z-Score*)	0,998687	0,998680	0,998687	0,998682	0,999911
MLP + PCA (*Min-Max*)	0,997830	0,997838	0,997830	0,997831	0,999906
MLP + PCA (*Z-Score*)	0,997666	0,997681	0,997666	0,997667	0,999908
MLP + MDI (*Min-Max*)	0,997758	0,997765	0,997758	0,997757	0,999982
MLP + MDI (*Z-Score*)	0,997793	0,997800	0,997793	0,997800	0,999790

Tabela 74 – Comparação CICIDS e UNSW-NB15.

Modelo	*Dataset*	Acurácia	Precisão	*Recall*	*F1-Score*	AUC-ROC
Logistic Regression (SIRISHA et al., 2021)	CICIDS-2017	0.823021	0.540815	0.654184	0.592122	0.897242
Decision Tree (SIRISHA et al., 2021)	CICIDS-2017	0.891597	0.654829	0.947296	0.774368	0.910645
RF (SIRISHA et al., 2021)	CICIDS-2017	0,937743	0,841509	0,841484	0,841460	0,986115
Gaussian NB (SIRISHA et al., 2021)	CICIDS-2017	0.696664	0.317034	0.471939	0.3792802	0.766184
K-Nearest Neighbors (SIRISHA et al., 2021)	CICIDS-2017	0.906897	0.682306	0.984089	0.805871	0.950408
RF + PCA (*Min-Max*)	UNSW-NB15	0,724215	0,705629	0,724215	0,694512	0,947346
RF + PCA (*Z-Score*)	UNSW-NB15	0,362394	0,399001	0,362394	0,322220	0,625230
RF + MDI (*Min-Max*)	UNSW-NB15	0,752814	0,758607	0,752814	0,722133	0,956899
RF + MDI (*Z-Score*)	UNSW-NB15	0,751763	0,756292	0,751763	0,719142	0,951445
MLP + PCA (*Min-Max*)	UNSW-NB15	0,749472	0,749821	0,749472	0,723731	0,959772
MLP + PCA (*Z-Score*)	UNSW-NB15	0,288792	0,305623	0,288792	0,289069	0,578769
MLP + MDI (*Min-Max*)	UNSW-NB15	0,746201	0,747018	0,746201	0,713215	0,956186
MLP + MDI (*Z-Score*)	UNSW-NB15	0,746641	0,745688	0,746641	0,714726	0,955556

APÊNDICE H – Experimentos Adicionais sobre o *Dataset* UNSW-NB15.

A fim de explorar divisões alternativas dos *datasets* de treino e teste do UNSW-NB15, foram realizados experimentos adicionais com os modelos RF e MLP, usando as mesmas técnicas de pré-processamento descritas no trabalho (normalização *Min-Max* e *Z-Score*, redução de dimensionalidade por PCA e MDI), com os resultados descritos nas Tabelas 75 a 80.

Tabela 75 – RF Sobre o UNSW-NB15 Após Normalização *Min-Max* e PCA.

Árvores	Acurácia	Precisão	Recall	*F1-score*	AUC-ROC	Tempo(s)
10	0.789483	0.786911	0.789483	0.783402	0.954057	15.846498
25	0.795828	0.790356	0.795828	0.787371	0.959633	40.301466
50	0.798681	0.794204	0.798681	0.789791	0.961470	91.715772
75	0.798060	0.792117	0.798060	0.788628	0.962136	128.298303
100	0.798991	0.792532	0.798991	0.789708	0.962535	274.086900
125	0.798894	0.792553	0.798894	0.789293	0.962697	203.034873
150	0.799146	0.792748	0.799146	0.789584	0.962887	255.069467
Média	**0.797583**	**0.792489**	**0.797583**	**0.788825**	**0.960773**	**144.621611**

Tabela 76 – RF Sobre o UNSW-NB15 Após Normalização *Z-Score* e PCA.

Árvores	Acurácia	Precisão	Recall	*F1-score*	AUC-ROC	Tempo(s)
10	0.781896	0.777350	0.781896	0.775095	0.952797	22.094796
25	0.788047	0.781818	0.788047	0.779365	0.957917	56.718719
50	0.789968	0.784971	0.789968	0.780801	0.959943	119.411491
75	0.790822	0.784635	0.790822	0.781101	0.960559	172.879030
100	0.791714	0.784649	0.791714	0.781930	0.960890	225.532969
125	0.791676	0.783879	0.791676	0.781645	0.961072	284.893136
150	0.791326	0.783500	0.791326	0.781179	0.961185	375.019047
Média	**0.789636**	**0.782686**	**0.789636**	**0.780445**	**0.959766**	**179.507884**

Tabela 77 – RF Sobre o UNSW-NB15 Após Normalização *Min-Max* e MDI.

Árvores	Acurácia	Precisão	Recall	*F1-score*	AUC-ROC	Tempo(s)
10	0.824837	0.823296	0.824837	0.817435	0.973282	25.649488
25	0.828214	0.827470	0.828214	0.818436	0.976328	41.646204
50	0.830368	0.829922	0.830368	0.820470	0.977288	91.920192
75	0.833434	0.831139	0.833434	0.821539	0.977791	142.093840
100	0.833608	0.832805	0.833608	0.821491	0.978067	237.107846
125	0.833511	0.831529	0.833511	0.821501	0.978187	196.607498
150	0.833395	0.832071	0.833395	0.821123	0.978265	186.904570
Média	**0.829981**	**0.829176**	**0.829981**	**0.820285**	**0.977601**	**131.132091**

Tabela 78 – RF Sobre o UNSW-NB15 Após Normalização *Z-Score* e MDI.

Árvores	Acurácia	Precisão	Recall	*F1-score*	AUC-ROC	Tempo(s)
10	0.823033	0.823977	0.823033	0.817194	0.963813	13.470850
25	0.823615	0.824646	0.823615	0.817969	0.967733	25.319566
50	0.822121	0.823454	0.822121	0.817379	0.968667	45.392251
75	0.819618	0.820735	0.819618	0.814941	0.969019	66.718329
100	0.822994	0.823348	0.822994	0.817210	0.969195	95.814715
125	0.821345	0.822279	0.821345	0.815797	0.969245	162.444071
150	0.821345	0.822240	0.821345	0.815461	0.969311	187.607442
Média	**0.822296**	**0.822954**	**0.822296**	**0.816850**	**0.968426**	**85.823889**

Tabela 79 – MLP Sobre o UNSW-NB15 Após Normalização *Min-Max* e MDI.

Camadas	**Acurácia**	**Precisão**	**Recall**	***F1-score***	**AUC-ROC**	**Tempo(s)**
(16,8)	0.797109	0.787854	0.797109	0.772006	0.978067	451.479847
(32,16)	0.808324	0.794120	0.808324	0.785088	0.978067	545.071655
(64,32)	0.817852	0.809038	0.817852	0.798307	0.978067	2342.962889
(10,20,10)	0.794703	0.780952	0.794703	0.771278	0.978067	795.356484
(20,40,20)	0.812846	0.805249	0.812846	0.791883	0.978067	641.677051
(40,60,40)	0.821966	0.824312	0.821966	0.797472	0.978067	652.519912
(60,40,60)	0.822431	0.814491	0.822431	0.801241	0.978067	2081.640940
(60,80,60)	0.821500	0.815025	0.821500	0.807909	0.978067	1663.787078
(80,100,80)	0.823925	0.816004	0.823925	0.803750	0.978067	1777.978064
(100,120,100)	0.826914	0.827993	0.826914	0.805045	0.978067	2567.564899
(100,80,60,40,20)	0.823130	0.817773	0.823130	0.805001	0.978067	2892.771608
(128,128,64,64)	0.823576	0.818712	0.823576	0.808690	0.978067	4209.871912
Média	**0.817441**	**0.808460**	**0.817441**	**0.795722**	**0.978067**	**1743.556111**

Tabela 80 – MLP Sobre o UNSW-NB15 Após Normalização *Min-Max* e PCA.

Camadas	**Acurácia**	**Precisão**	**Recall**	***F1-score***	**AUC-ROC**	**Tempo(s)**
(16,8)	0.793829	0.784035	0.793829	0.762355	0.971189	182.189934
(32,16)	0.809120	0.802649	0.809120	0.784623	0.975294	404.335598
(64,32)	0.815038	0.811639	0.815038	0.800982	0.977001	940.171304
(10,20,10)	0.800369	0.795813	0.800369	0.774466	0.973085	336.259823
(20,40,20)	0.814243	0.804611	0.814243	0.792953	0.976839	325.393569
(40,60,40)	0.817270	0.816213	0.817270	0.803523	0.977641	746.909222
(60,40,60)	0.814883	0.811595	0.814883	0.806399	0.977859	1109.522521
(60,80,60)	0.816144	0.809045	0.816144	0.804862	0.977305	956.160329
(80,100,80)	0.816513	0.812137	0.816513	0.807069	0.977070	1333.486091
(100,120,100)	0.816144	0.804974	0.816144	0.796276	0.975891	1672.583733
(100,80,60,40,20)	0.815135	0.817006	0.815135	0.809621	0.977424	1599.338620
(128,128,64,64)	0.812496	0.807912	0.812496	0.796830	0.975692	1807.215433
Média	**0.812348**	**0.807220**	**0.812348**	**0.794997**	**0.976524**	**992.796968**