UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Pedro Henrique Domeneghi da Silva

Análise Comparativa de Técnicas de Aprendizado de Máquina para Detecção de Intrusões em Redes no Conjunto de Dados CICIDS2017

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Pedro Henrique Domeneghi da Silva

Análise Comparativa de Técnicas de Aprendizado de Máquina para Detecção de Intrusões em Redes no Conjunto de Dados CICIDS2017

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

Pedro Henrique Domeneghi da Silva

Análise Comparativa de Técnicas de Aprendizado de Máquina para Detecção de Intrusões em Redes no Conjunto de Dados CICIDS2017

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, 04 de abril de 2025:

Luís Fernando Faina
Orientador

Márcia Aparecida Fernandes
Coorientador

Paulo Rodolfo da Silva Leite Coelho
Membro da Banca

Paulo Henrique Ribeiro Gabriel
Membro da Banca

Uberlândia, Brasil
2025

Resumo

Este trabalho apresenta a análise e comparação de modelos de Aprendizado de Máquina no contexto de um Sistema de Detecção de Intrusões em Redes, utilizando o conjunto de dados CICIDS2017. Para isso, foram avaliados os modelos *Random Forest* (RF) e *Multi-Layer Perceptron* (MLP), combinados com técnicas de redução de dimensionalidade, como *Principal Component Analysis* (PCA) e *Mean Decrease in Impurity* (MDI). A técnica MDI resultou em uma redução de 70 para 51 características, enquanto a técnica PCA resultou em uma redução de 70 para 33 características. A validação dos modelos foi realizada por meio da técnica *Stratified K-Fold Cross-Validation*, e as métricas de desempenho, como precisão ponderada, *recall* ponderado e F1-Score ponderado, foram utilizadas para avaliar os resultados. Testes com diferentes valores de *n_estimators* no RF revelaram que 25 árvores proporcionam o melhor equilíbrio entre desempenho e eficiência computacional, com um F1-Score ponderado aproximado de 0,9981 e levando apenas 8,7 minutos para treinar o modelo. Já o MLP atingiu um F1-Score ponderado aproximado de 0,9959 e levando 13,1 minutos para treinar o modelo.

Palavras-chave: Aprendizado de máquina, detecção de intrusões, redução de dimensionalidade, CICIDS2017, redes neurais.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IDC	International Data Corporation
ML	*Machine Learning*
NIDS	*Network Intrusion Detection System*
CIC	Canadian Institute for Cybersecurity
IDS	*Intrusion Detection System*
RF	*Random Forest*
MLP	*Multi-Layer Perceptron*
PCA	*Principal Component Analisys*
MDI	*Mean Decrease in Impurity*
SHARP	*Shapley Additive Explanations*
LIME	*Local Interpretable Model-agnostic Explanations*
ReLU	*Rectified Linear Unit*
DoS	*Denial of Service*
DDoS	*Distributed Denial of Service*
XSS	*Cross-Site Scripting*
SQL	*Structured Query Language*

Sumário

1 Introdução

A segurança cibernética tornou-se uma das principais preocupações na era digital, dado o crescimento exponencial do uso de dispositivos conectados à Internet e a complexidade das infraestruturas de rede. Nesse contexto, os Sistemas de Detecção de Intrusão em Redes (NIDS - *Network Intrusion Detection System*) se destacam como ferramentas cruciais para identificar atividades maliciosas e proteger as redes digitais contra ataques, com os modelos baseados em assinaturas e em anomalias sendo os mais comuns e amplamente utilizados. Com o avanço das técnicas de Aprendizado de Máquina (ML - *Machine Learning*), a integração de modelos inteligentes aos NIDS se tornou uma abordagem promissora para melhorar a precisão e a eficiência na detecção de ameaças (LABONNE, 2020).

Um Sistema de Detecção de Intrusão baseado em assinaturas funciona comparando cada pacote que trafega pela rede com um conjunto de padrões pré-definidos, conseguindo detectar somente ataques conhecidos e previamente definidos. Assim, se um pacote corresponder a uma assinatura armazenada, o sistema emite um alerta. Por outro lado, um sistema baseado em anomalias monitora o comportamento habitual do tráfego da rede, criando um perfil de referência. Assim, qualquer desvio significativo em relação ao padrão esperado é tratado como uma possível ameaça, permitindo a identificação de novos ataques que ainda não foram documentados (LABONNE, 2020).

Com a crescente sofisticação dos ataques cibernéticos, métodos tradicionais de detecção, como os baseados em assinatura (LABONNE, 2020), tornam-se inadequados. Segundo um relatório da Cybersecurity Ventures, espera-se que os custos globais com crimes cibernéticos atinjam US$ 10,5 trilhões anualmente até 2025 (Cybersecurity Ventures, 2021). Além disso, a International Data Corporation (IDC) prevê que os gastos mundiais com soluções de segurança cibernética alcançarão US$ 174,7 bilhões em 2024 (International Data Corporation, 2021). Nesse cenário, a inovação trazida pelo uso de ML nos NIDS permite a análise de grandes volumes de dados de rede em tempo real, identificando padrões anômalos que podem indicar tentativas de intrusão.

O Aprendizado de Máquina pode ser dividido em dois grupos: o supervisionado e o não-supervisionado. As técnicas supervisionadas são amplamente exploradas e necessitam de um conjunto de dados rotulados para treinar um modelo de decisão, utilizado para classificar novos dados de rede. No entanto, essas soluções dependem de novos dados rotulados para atualizar o modelo, o que representa um desafio, pois requer a intervenção de especialistas para rotular os dados continuamente. Já os métodos não-supervisionados são menos explorados na detecção de intrusões, não precisando de uma base de dados

rotulada e utilizando apenas os dados brutos e suas relações para identificar os padrões anômalos.

Diversos estudos têm investigado a aplicação de técnicas de Aprendizado de Máquina supervisionado em NIDS. Labonne et al. (LABONNE, 2020) e Neto et al. (NETO; GOMES, 2019) fornecem uma visão abrangente sobre a importância de combinar ML com detecção de intrusões, mostrando resultados promissores na identificação de anomalias em redes, comparando diferentes modelos de *Machine Learning* e usando conjuntos de dados complexos e diversificados, como o CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018), demonstrando alta precisão e baixa taxa de falsos positivos.

Nesse contexto, este trabalho analisa e compara diferentes técnicas de Aprendizado de Máquina supervisionado para Sistemas de Detecção de Intrusões, testando diferentes modelos como *Random Forest* (RF) e *Multi-Layer Perceptron* (MLP). Com isso, foi possível melhorar os resultados de metodologias baseadas em Aprendizado de Máquina e a detecção de várias categorias de ataques cibernéticos presentes no conjunto de dados CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Além disso, este trabalho analisa a importância das diferentes características do conjunto de dados na eficácia dos modelos, utilizando técnicas de redução de dimensionalidade como *Mean Decrease in Impurity* (MDI) e Análise de Componentes Principais (PCA - *Principal Component Analysis*).

Foram utilizados diversos algoritmos, ferramentas e técnicas de processamento de dados:

1. **Scikit-Learn e Pandas**: Bibliotecas essenciais para a manipulação de dados e a implementação das técnicas e modelos de Aprendizado de Máquina.

2. **Técnicas de pré-processamento**: O pré-processamento dos dados é uma fase crítica que inclui as seguintes subetapas: limpeza dos dados, normalização e redução de dimensionalidade.

3. **Técnicas de redução de dimensionalidade**: MDI e PCA são técnicas utilizadas para reduzir a dimensionalidade do conjunto de dados e identificar as características mais relevantes, ajudando a melhorar a eficácia dos modelos.

4. **Modelos de Machine Learning**: *Random Forest* (RF) é um algoritmo de aprendizado supervisionado amplamente utilizado por sua robustez e capacidade de lidar com grandes volumes de dados e classes desbalanceadas. Já o *Multi-Layer Perceptron* (MLP) é uma rede neural artificial usada para classificação complexa. Sua capacidade de capturar padrões não lineares é especialmente útil na detecção de intrusões.

5. **Técnicas de validação de modelos**: *Stratified K-Fold Cross-Validation* é uma técnica utilizada para validar os modelos de forma robusta, garantindo que todas as classes sejam representadas de maneira equilibrada em cada *fold*.

6. **Métricas de avaliação de desempenho**: A avaliação de desempenho dos modelos será feita utilizando as métricas precisão, *recall* e F1-Score. O tipo de métrica ponderada (*Weighted*), que leva em consideração o peso de cada classe na avaliação, será utilizado.

A monografia está dividida em mais quatro capítulos. O Capítulo 2 aborda a fundamentação teórica que serve de base para os métodos propostos, além de apresentar os trabalhos relacionados. No Capítulo 3, é detalhado o método para alcançar os objetivos estabelecidos. O Capítulo 4 discute os resultados obtidos. Por fim, o Capítulo 5 traz as conclusões, além de oferecer sugestões para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os conceitos teóricos fundamentais para o desenvolvimento deste trabalho. Inicialmente, discute-se o conceito de NIDS, descrevendo sua evolução e importância, com a finalidade de apresentar o contexto em que o trabalho está inserido. Em seguida, abordam-se as técnicas de redução de dimensionalidade aplicadas (MDI e PCA). Na sequência, são discutidos os modelos de Aprendizado de Máquina utilizados neste trabalho, especificamente o RF (*Random Forest*) e o MLP (*Multi-Layer Perceptron*). Em seguida, é descrita a técnica de validação de modelos utilizada (*Stratified K-Fold Cross-Validation*) e são apresentadas as métricas de avaliação de desempenho de modelos. Por fim, são discutidos os trabalhos relacionados.

2.1 Sistemas de Detecção de Intrusão em Redes

Os Sistemas de Detecção de Intrusão em Redes (NIDS - *Network Intrusion Detection System*) são componentes cruciais na segurança cibernética, responsáveis por monitorar e analisar o tráfego de rede em tempo real para identificar atividades suspeitas (BHUYAN; BHATTACHARYYA; KALITA, 2020). Estes sistemas funcionam como uma camada de defesa, alertando sobre possíveis ameaças e, em alguns casos, tomando medidas para mitigar os riscos.

A concepção dos NIDS remonta às décadas de 1980 e 1990, com os primeiros sistemas baseados em assinaturas que funcionam comparando cada pacote que trafega pela rede com um conjunto de padrões pré-definidos, conseguindo detectar somente ataques conhecidos e previamente definidos. Assim, se um pacote corresponder à uma assinatura armazenada, o sistema emite um alerta (LABONNE, 2020). Com a evolução das redes e o aumento da complexidade dos ataques, surgiram sistemas baseados em anomalias, que monitoram o comportamento habitual do tráfego da rede, criando um perfil de referência. Assim, qualquer desvio significativo em relação ao padrão esperado é tratado como uma possível ameaça, permitindo a identificação de novos ataques que ainda não foram documentados (KHAN, 2016).

Nos últimos anos, os NIDS evoluíram significativamente, incorporando técnicas de Aprendizado de Máquina para melhorar a detecção de ameaças e reduzir falsos positivos. A integração com sistemas de *Big Data* e análise em tempo real permite uma detecção mais rápida e precisa de intrusões (WANG; JONES, 2017). Além disso, o uso de redes neurais e outros modelos avançados tem permitido identificar padrões complexos de ataques que antes não eram possivelmente detectados.

Exemplos populares de NIDS incluem Snort (ROESCH, 1998), Suricata (OISF (Open Information Security Foundation), 2009) e Zeek (PAXSON, 1995), amplamente utilizados em ambientes corporativos e acadêmicos para proteger infraestruturas de rede contra uma ampla gama de ameaças cibernéticas (BHUYAN; BHATTACHARYYA; KALITA, 2020). A Figura 1 mostra a estrutura geral de um NIDS e seu funcionamento.

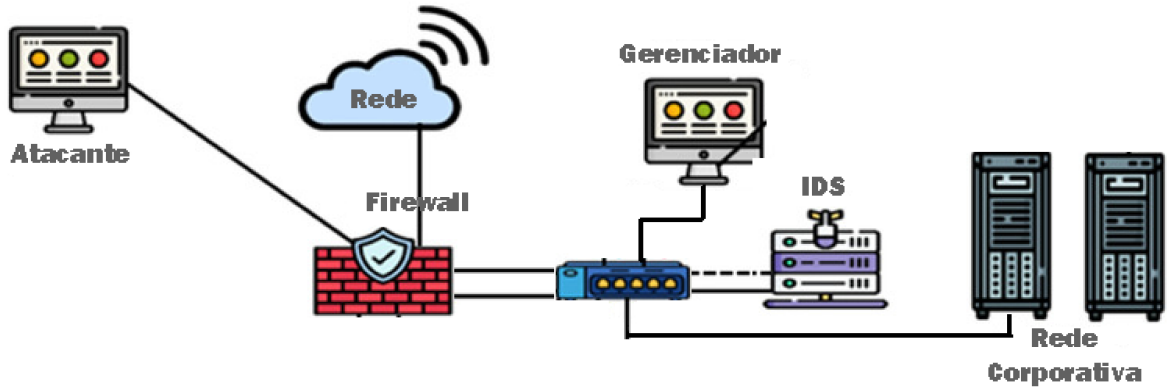

Figura 1 – Estrutura de um NIDS. Adaptado de (ABDEL-BASSET et al., 2022).

2.2 Técnicas de Redução de Dimensionalidade

Antes de realizar a redução de dimensionalidade, a limpeza e a normalização dos dados é essencial para assegurar que os dados estejam em um formato adequado para os modelos de ML. A redução de dimensionalidade é um processo fundamental em Aprendizado de Máquina, especialmente quando trabalhamos com grandes volumes de dados que podem conter redundâncias ou características irrelevantes. Simplificar o conjunto de dados não só melhora o tempo de processamento dos modelos de ML, como também pode aumentar sua precisão, removendo ruídos que prejudicam a qualidade da predição. Neste trabalho, são apresentadas duas técnicas para redução de dimensionalidade: *Mean Decrease in Impurity* (MDI) e *Principal Component Analysis* (PCA).

2.2.1 *Mean Decrease in Impurity* (MDI)

A técnica *Mean Decrease in Impurity* (MDI) é uma abordagem comumente utilizada em modelos de *Random Forest* para avaliar a importância das características no processo de classificação. A MDI calcula a redução média de impureza causada por cada característica ao longo das árvores que compõem a floresta. Quanto maior a redução de impureza atribuída a uma característica, mais importante ela é considerada para a predição (BREIMAN, 2001).

A impureza de um nó em uma árvore de decisão é uma medida usada para avaliar a qualidade das divisões em cada nó da árvore. A impureza reflete o grau de mistura das classes dentro de um nó: quanto mais homogêneo for o nó em relação às classes, menor será sua impureza. Quando uma característica é usada para fazer uma divisão que separa bem as classes, a impureza diminui. A MDI calcula a importância de cada característica somando a redução de impureza causada pela característica em todas as árvores da floresta e depois tirando uma média. Assim, a característica que causa uma maior redução da impureza ao longo das árvores é considerada mais relevante para o modelo (GAO; WEN; ZHANG, 2019).

A principal vantagem da MDI é que ela não apenas identifica as características mais relevantes, mas também oferece uma interpretação intuitiva da importância de cada uma. A técnica é particularmente útil em modelos de *Random Forest*, onde múltiplas árvores são criadas, o que possibilita uma estimativa robusta da contribuição de cada característica para o modelo.

Apesar de sua eficácia, a MDI possui limitações, como a tendência de favorecer características contínuas ou com mais níveis de variação. Para lidar com essa limitação, técnicas complementares, como SHARP (*Shapley Additive Explanations*) e LIME (*Local Interpretable Model-agnostic Explanations*) foram desenvolvidas, proporcionando uma visão mais detalhada e global sobre a importância das características e como elas afetam a decisão dos modelos (LUNDBERG; LEE, 2017; RIBEIRO; SINGH; GUESTRIN, 2016).

2.2.2 *Principal Component Analysis* (PCA)

A Análise de Componentes Principais (PCA - *Principal Component Analysis*) é uma técnica estatística de redução de dimensionalidade amplamente utilizada em problemas de Aprendizado de Máquina e análise de dados. A PCA transforma um conjunto de características possivelmente correlacionadas em um novo conjunto de características linearmente independentes chamadas de componentes principais. O objetivo da PCA é identificar as direções (componentes principais) onde a variabilidade dos dados é maior, projetando-os em um espaço de menor dimensão, mas que ainda retém a maior parte da variabilidade original (VIDAL et al., 2016).

A técnica foi introduzida por Karl Pearson em 1901 (PEARSON, 1901) e, posteriormente, aprimorada por Harold Hotelling nos anos 1930. A PCA tornou-se uma ferramenta essencial para lidar com dados de alta dimensionalidade, onde um grande número de características pode aumentar a complexidade computacional dos modelos de ML e diminuir seu desempenho. Ao reduzir a dimensionalidade do conjunto de dados, a PCA ajuda a simplificar os modelos de ML, tornando-os mais rápidos e menos propensos ao *overfitting* (quando o modelo se ajusta demais aos dados de treino e tem dificuldade de generalizar para novos dados), além de facilitar a visualização e interpretação dos dados.

A PCA funciona transformando o conjunto de dados original em uma nova base ortogonal de eixos (componentes principais, que são combinações lineares das características originais), onde o primeiro componente principal explica a maior parte da variabilidade dos dados, o segundo componente explica a segunda maior parte, e assim por diante. As características com menor variabilidade podem ser descartadas, resultando em um conjunto de dados com menos dimensões, mas que ainda contém as informações mais relevantes. A Figura 2 ilustra a porcentagem da variância explicada por cada componente principal em um exemplo com 10 dimensões.

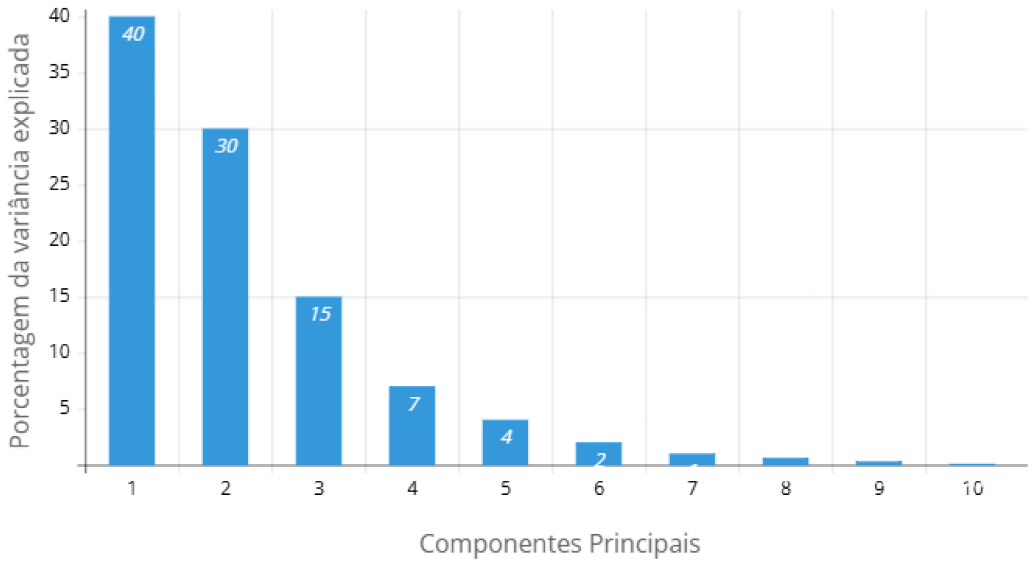

Figura 2 – Porcentagem da variância explicada por cada componente principal.

O funcionamento geral da PCA é descrito a seguir.

- A primeira etapa envolve uma normalização dos dados por média e desvio padrão para garantir que todas as características estejam na mesma escala;
- Logo após, é calculada a matriz de covariância para analisar a correlação entre as características do conjunto;
- A próxima etapa envolve a descoberta dos componentes principais e a porcentagem da variância explicada por cada um;
- Por fim, as características menos relevantes são descartadas.

A PCA é amplamente aplicada em áreas como compressão de imagens, processamento de sinais e bioinformática, e sua utilização em Aprendizado de Máquina é valiosa quando se lida com conjuntos de dados complexos e de grande escada (BHASKAR; HOYLE; SINGH, 2006). No contexto deste trabalho, a PCA é utilizada para reduzir

a dimensionalidade do conjunto de dados CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018), melhorando a eficiência e precisão dos modelos.

2.3 Técnicas de Aprendizado de Máquina

O Aprendizado de Máquina é um subcampo da inteligência artificial que permite que sistemas computacionais aprendam a partir de dados e façam previsões ou decisões sem serem explicitamente programados (FABIAN, 2011). Nos últimos anos, o ML tem se destacado como uma ferramenta poderosa em várias áreas, incluindo a detecção de intrusões em redes, devido à sua capacidade de lidar com grandes volumes de dados e identificar padrões complexos. Entre os principais algoritmos estão: *Random Forest* e *Multi-Layer Perceptron*, que serão discutidos a seguir.

2.3.1 *Random Forest*

O *Random Forest* (RF) é um algoritmo de aprendizado supervisionado que faz parte da classe de *ensemble learning*, onde múltiplos modelos fracos (árvores de decisão, neste caso) são combinados para produzir uma predição final mais robusta e precisa. Ele foi proposto por Leo Breiman em 2001 (BREIMAN, 2001) e é amplamente utilizado devido à sua simplicidade, robustez e precisão, especialmente em problemas com grandes volumes de dados e classes desbalanceadas, como a detecção de intrusões em redes.

O RF é baseado em árvores de decisão, que são modelos que particionam os dados em subconjuntos menores, de acordo com regras de decisão em cada nó da árvore, até chegar a uma classe final no nó folha. Cada árvore de decisão é criada com base em um subconjunto aleatório de instâncias e características do conjunto de dados de treinamento, garantindo que cada árvore seja ligeiramente diferente das demais. A principal inovação do *Random Forest* é combinar várias dessas árvores para criar um modelo mais robusto e menos propenso ao *overfitting*. O funcionamento geral do RF é descrito a seguir e também é ilustrado na Figura 3.

- O RF constrói um conjunto de árvores de decisão (chamado "floresta"), cada uma treinada em um subconjunto de instâncias do conjunto de dados de treinamento. Para cada árvore, o RF seleciona aleatoriamente um subconjunto de instâncias, com reposição. Isso significa que algumas instâncias podem aparecer mais de uma vez, enquanto outras podem não ser usadas para aquela árvore;
- Diferente de uma árvore de decisão tradicional, onde todas as características são consideradas em cada divisão do nó da árvore, o RF seleciona aleatoriamente um subconjunto de características em cada nó da árvore para avaliar qual delas deve

ser usada para dividir os dados. Isso gera diversidade entre as árvores, uma vez que cada uma delas usa uma parte diferente dos dados e das características;

- Cada árvore é construída até atingir sua profundidade máxima ou até que todas as folhas estejam puras (contendo apenas uma classe);
- Após todas as árvores terem sido construídas, o RF realiza a predição final por meio de votação, em que cada árvore individual vota na classe que ela considera mais provável, e a classe com o maior número de votos é escolhida como a predição final do modelo.

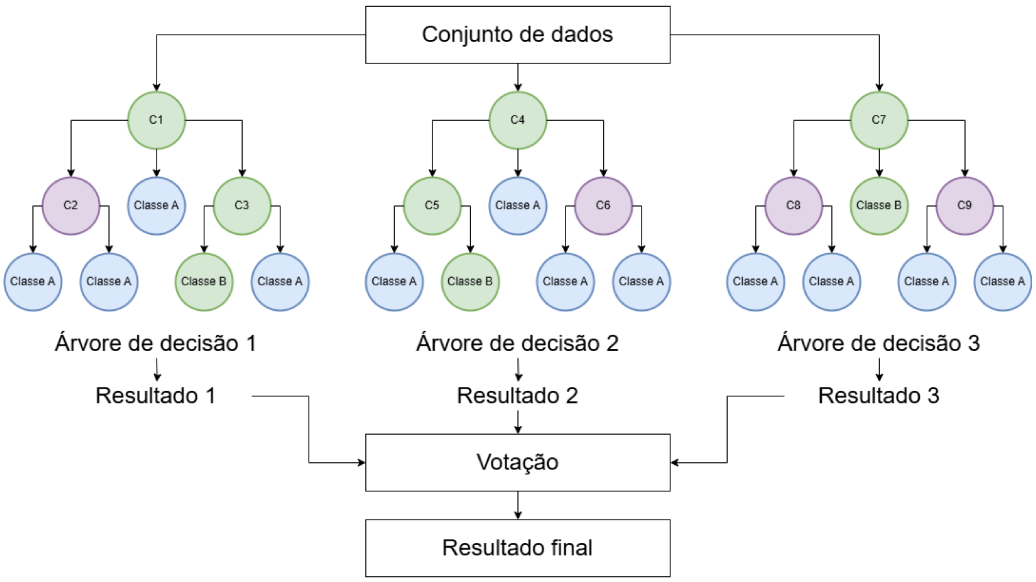

Figura 3 – Funcionamento geral do *Random Forest*.

Uma das grandes vantagens do RF é sua robustez contra *overfitting*. Ao combinar várias árvores, ele reduz a dependência de um único conjunto de dados ou de uma árvore específica, o que resulta em um modelo mais generalizável. Além disso, o RF também é resistente a dados ruidosos, sendo menos sensível a flutuações nos dados, já que ele faz uma média das predições de várias árvores.

2.3.2 *Multi-Layer Perceptron*

O *Multi-Layer Perceptron* (MLP) é um tipo de rede neural artificial, que pertence à categoria dos algoritmos de aprendizado profundo (*Deep Learning*), amplamente utilizado para resolver problemas de classificação complexos. Diferente de modelos lineares, o MLP é capaz de capturar padrões não lineares nos dados, tornando-o ideal para problemas complexos, como a detecção de intrusões em redes, onde os padrões de ataque são muitas vezes sutis e não lineares. O MLP é composto por três tipos principais de camadas que serão descritos a seguir.

- **Camada de entrada**: Esta camada recebe os dados brutos, com cada nó da camada representando uma característica de entrada.
- **Camadas ocultas**: Entre a camada de entrada e a camada de saída, há uma ou mais camadas intermediárias (chamadas camadas ocultas) compostas por nós (neurônios). Cada neurônio recebe as entradas ponderadas da camada anterior e aplica uma função de ativação, como *ReLU* (*Rectified Linear Unit*) ou função sigmoide, que introduz não linearidade ao modelo, permitindo que ele aprenda padrões complexos.
- **Camada de saída**: A última camada contém os neurônios que correspondem às classes ou predições. Para problemas de classificação, ela usa uma função para transformar os valores em probabilidades para cada classe.

O MLP aprende a partir dos dados utilizando um processo conhecido como retropropagação (*backpropagation*) do erro, que ajusta os pesos das conexões entre os neurônios de modo a minimizar o erro nas predições. O funcionamento geral do MLP é descrito a seguir e também é ilustrado na Figura 4:

- Os dados de entrada são propagados através da rede, passando pelas camadas ocultas até atingir a camada de saída, onde uma predição é feita. Cada neurônio em uma camada aplica uma função de ativação sobre as entradas que recebe da camada anterior, processando a informação e passando-a adiante;
- O erro é calculado comparando as predições da rede com os valores de classes reais do conjunto de dados;
- Após calcular o erro, o algoritmo de retropropagação ajusta os pesos das conexões entre os neurônios com base no gradiente do erro em relação a esses pesos. O objetivo é minimizar o erro iterativamente, ajustando os pesos de forma que a predição melhore com o tempo.

Uma das grandes vantagens do MLP é a capacidade de aplicar funções de ativação não lineares, permitindo que o modelo capture padrões complexos e não lineares que seriam difíceis de modelar com outros algoritmos. Além disso, o MLP pode ser adaptado para resolver uma ampla variedade de problemas, podendo ser configurado com diferentes números de camadas ocultas e neurônios para aumentar sua capacidade de aprendizado. No entanto, essas redes requerem grande poder computacional, especialmente em problemas complexos com muitos dados.

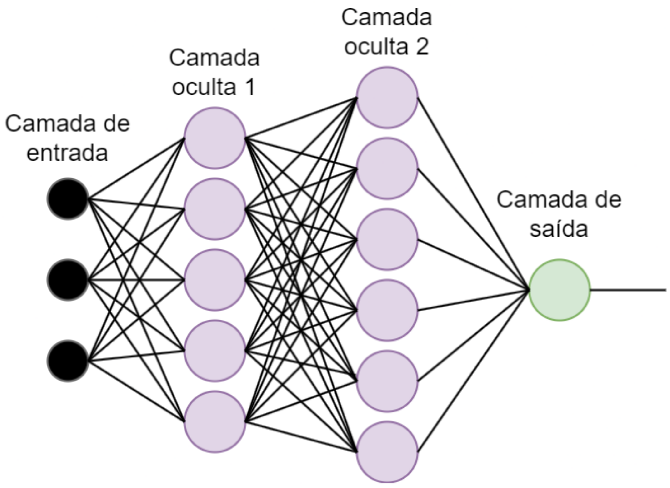

Figura 4 – Funcionamento geral do *Multi-Layer Perceptron*.

2.4 Técnicas de Validação de Modelos

A validação cruzada é uma técnica essencial em Aprendizado de Máquina para avaliar o desempenho de um modelo de forma mais robusta e evitar *overfitting* nos dados de treino, surgindo como uma forma de avaliar a generalização de um modelo, especialmente quando a quantidade de dados é limitada. O *Stratified K-Fold Cross-Validation* é uma variação da validação cruzada comum, que leva em consideração a proporção de instâncias de cada classe no conjunto de dados (KOHAVI et al., 1995).

No método padrão de *K-Fold Cross-Validation*, o conjunto de dados é dividido em K subconjuntos (*folds*). Em cada iteração, 1 (um) subconjunto é usado como conjunto de teste, enquanto os outros K - *1* são usados para treinar o modelo. Isso é repetido K vezes, e o desempenho do modelo é avaliado pela média dos resultados em todas as iterações. O funcionamento do *K-Fold Cross-Validation* padrão é também ilustrado na Figura 5.

No *Stratified K-Fold*, os dados são divididos em *folds* de modo que cada subconjunto mantenha aproximadamente a mesma proporção de classes presente no conjunto de dados original (KOHAVI et al., 1995). Isso é especialmente importante em problemas onde há classes desbalanceadas, como na detecção de intrusões, onde a quantidade de tráfego benigno (normal) é muito maior do que a de ataques.

Recentemente, técnicas de validação cruzada têm sido combinadas com métodos de busca em hiperparâmetros, como *Grid Search* e *Random Search*, para otimizar o desempenho dos modelos de Aprendizado de Máquina. A técnica é amplamente utilizada em problemas de classificação desequilibrada, onde a representatividade das classes é crucial para uma avaliação precisa do modelo (LIU; WU; WANG, 2019).

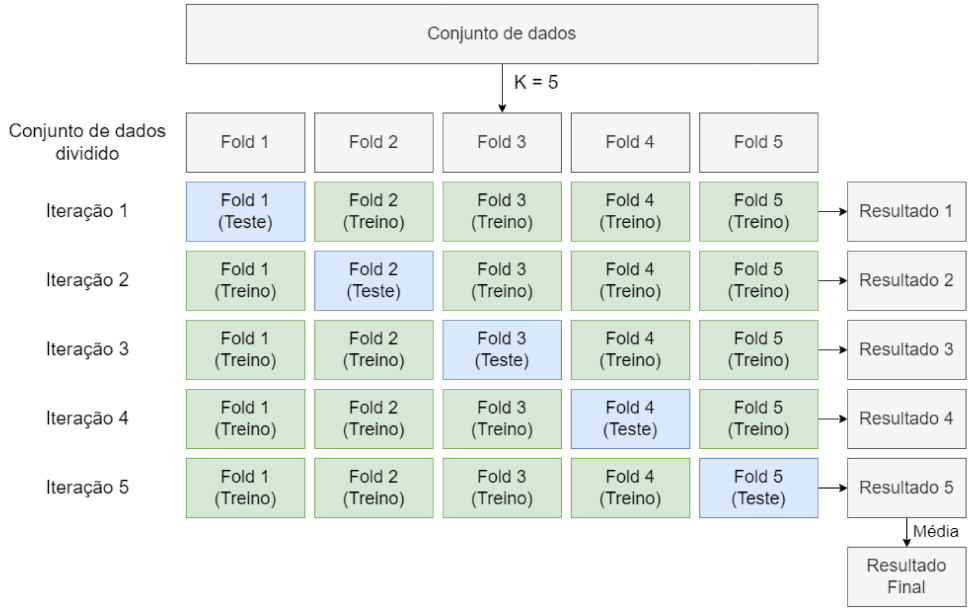

Figura 5 – Funcionamento geral do *K-Fold Cross-Validation*.

2.5 Métricas de Avaliação de Desempenho dos Modelos

A avaliação de desempenho dos modelos de ML é essencial para garantir que os modelos desenvolvidos sejam eficazes e precisos na detecção de intrusões em redes. As métricas de avaliação fornecem informações críticas sobre como os modelos se comportam em diferentes situações, ajudando a identificar suas forças e limitações. As métricas utilizadas são calculadas na forma ponderada (*Weighted*), o que ajusta as avaliações conforme a distribuição das classes no conjunto de dados. A Figura 6 ilustra a matriz de confusão que é usada como base para cálculo das métricas.

		Predito	
Real		**Ataque**	**Normal**
	Ataque	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Normal	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Figura 6 – Matriz de Confusão.

A precisão é uma das métricas mais utilizadas para avaliar o desempenho de modelos, representando a proporção de instâncias positivas que foram corretamente identificadas pelo modelo em relação ao total de instâncias positivas preditas pelo modelo. Matematicamente, a precisão é definida como:

$$Precisão = \frac{TP}{TP + FP}$$

Onde:

- TP: Verdadeiros positivos (instâncias corretamente classificadas como positivas)
- TN: Verdadeiros negativos (instâncias corretamente classificadas como negativas)
- FP: Falsos positivos (instâncias incorretamente classificadas como positivas)
- FN: Falsos negativos (instâncias incorretamente classificadas como negativas)

O *recall* mede a capacidade do modelo de identificar corretamente as instâncias positivas, avaliando a proporção de instâncias positivas que foram corretamente identificadas pelo modelo em relação ao total de instâncias positivas reais. Matematicamente, o *recall* é definido como:

$$Recall = \frac{TP}{TP + FN}$$

O F1-Score é uma métrica que combina precisão e *recall* em uma única métrica, calculando a média harmônica entre essas duas métricas. O F1-Score é particularmente útil quando há um desbalanceamento entre classes, pois oferece uma avaliação mais equilibrada do desempenho do modelo. Matematicamente, o F1-Score é definido como:

$$F1 - Score = 2 * \frac{Precisão * Recall}{Precisão + Recall}$$

As métricas ponderadas são particularmente importantes quando o conjunto de dados apresenta um desequilíbrio entre as classes, como é comum em problemas de detecção de intrusões. Nesse contexto, as classes de ataques podem ser muito menos representadas do que as classes benignas (normais), o que pode distorcer os resultados das métricas se não forem ponderadas. Matematicamente, elas são definidas como:

$$Precisão_{weighted} = \sum_{i=1}^{n} (\frac{N_i}{N}) * Precisão_i$$

$$Recall_{weighted} = \sum_{i=1}^{n} (\frac{N_i}{N}) * Recall_i$$

$$F1 - Score_{weighted} = \sum_{i=1}^{n} (\frac{N_i}{N}) * F1 - Score_i$$

Onde:

- n: Número de classes

- N_i: Número de instâncias da classe i
- N: Número total de instâncias
- $Precisão_i$: Precisão calculada para a classe i
- $Recall_i$: *Recall* calculado para a classe i
- $F1 - Score_i$: F1-Score calculado para a classe i

2.6 Trabalhos relacionados

O trabalho de Manuel G. da Silva Neto e Danielo G. Gomes (NETO; GOMES, 2019) apresenta uma abordagem sistemática para a seleção de algoritmos no desenvolvimento de Sistemas de Detecção de Intrusão em Redes utilizando Aprendizado de Máquina. Utilizando o conjunto de dados CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018), os autores realizaram uma análise abrangente de características, reduzindo-as de 78 para 51 através da técnica de seleção de características *Mean Decrease in Impurity* (MDI), sendo fundamental para otimizar o desempenho dos modelos de ML. Além disso, os autores realizaram a normalização dos dados utilizando a técnica de mínimo e máximo (*Min-Max*) e avaliaram algoritmos como árvores de decisão, *Random Forest* (RF) e *Multi-Layer Perceptron* (MLP), aplicando técnicas de validação cruzada como o *Stratified K-Fold Cross-Validation* para garantir a robustez dos resultados, alcançando métricas de precisão e F1-Score entre 0,98 e 0,99, além de tempos de teste reduzidos. Em contraste, este trabalho normalizou os dados utilizando média e desvio padrão (*StardardScaler*) e utilizou a técnica PCA para reduzir a dimensionalidade do conjunto de dados de 78 para 33 características.

O trabalho de Maxime Labonne (LABONNE, 2020) tem como objetivo desenvolver um NIDS baseado em anomalias utilizando técnicas de ML. Labonne propõe uma metodologia que utiliza a Análise de Componentes Principais (PCA) para a redução de dimensionalidade e algoritmos como *Random Forest* e *Multi-Layer Perceptron* para a detecção de anomalias. Os principais resultados indicam que a abordagem é eficaz na identificação de ataques desconhecidos, alcançando uma taxa de detecção elevada e uma baixa taxa de falsos positivos. Além disso, Labonne destaca que a utilização de PCA ajudou a melhorar significativamente o desempenho do modelo ao reduzir o número de características, o que também contribuiu para a diminuição do tempo de processamento.

O trabalho de Arnaud Rosay, Florent Carlier e Pascal Leroux (ROSAY; CARLIER; LEROUX, 2020) propõe o uso de uma rede neural artificial para a detecção de intrusões em rede utilizando o algoritmo MLP e o conjunto de dados CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018). A metodologia adotada pelos autores envolve a

aplicação de técnicas de pré-processamento de dados, incluindo a normalização e a redução de dimensionalidade, para otimizar o desempenho da rede neural. Como resultado, a rede neural alcançou um F1-Score de 99,95%, demonstrando a eficácia das redes neurais em detectar padrões complexos de ataques.

3 Método

Este capítulo descreve as etapas e procedimentos metodológicos adotados. A metodologia foi cuidadosamente estruturada para garantir que a avaliação seja eficiente e capaz de identificar diferentes tipos de ataques cibernéticos com alta precisão, mesmo em cenários de classes desbalanceadas.

Primeiramente, aborda-se o processo de seleção e preparação do conjunto de dados. Em seguida, são detalhadas as etapas de pré-processamento dos dados, essenciais para garantir a qualidade e consistência das informações utilizadas nos modelos de ML. Na sequência, são descritas as técnicas de redução de dimensionalidade aplicadas, que simplificam o conjunto de dados sem perda significativa de informação. Posteriormente, são apresentados os modelos de Aprendizado de Máquina utilizados. Em seguida, é detalhado o uso do *Stratified K-Fold Cross-Validation* para a validação dos modelos, garantindo uma avaliação mais equilibrada e robusta. Por fim, são abordadas as métricas de avaliação utilizadas para analisar o desempenho dos modelos.

Uma visão geral da metodologia adotada é ilustrada na Figura 7.

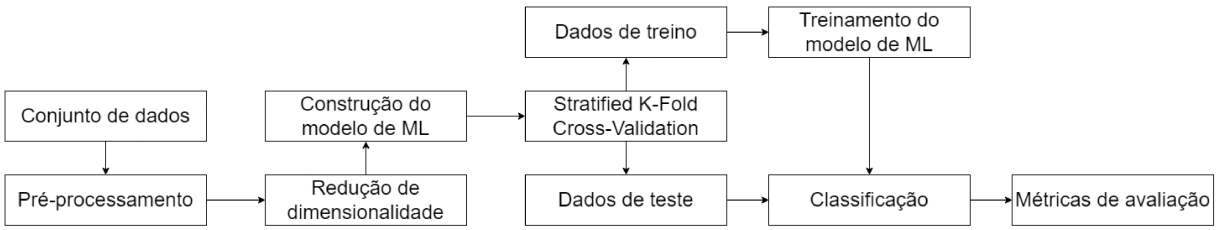

Figura 7 – Visão geral da metodologia utilizada.

A biblioteca Pandas (MCKINNEY, 2010) é utilizada para a leitura dos arquivos do conjunto de dados e em algumas etapas do pré-processamento, como a criação de matrizes para representar os dados, sendo uma das ferramentas mais robustas e amplamente utilizadas para manipulação de dados em Python.

A biblioteca Scikit-Learn (FABIAN, 2011) é utilizada para a implementação dos modelos de ML e o processamento de dados, sendo uma das ferramentas mais robustas e amplamente utilizadas para Aprendizado de Máquina em Python. Esta biblioteca fornece uma ampla gama de modelos de ML, técnicas de redução de dimensionalidade, técnicas de validação, ferramentas de pré-processamento e normalização e métricas de avaliação de desempenho, facilitando a execução das tarefas descritas.

3.1 Seleção do *dataset* CICIDS2017

A seleção do conjunto de dados é uma etapa crítica, pois os dados influenciam diretamente a capacidade do modelo de identificar corretamente padrões de comportamento normal e anômalo. O conjunto de dados CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018), que é amplamente utilizado para a avaliação de Sistemas de Detecção de Intrusão em Redes, foi escolhido devido à sua abrangência e relevância na área de detecção de intrusões, incluindo diversas categorias de ataques cibernéticos.

O conjunto de dados CICIDS2017, desenvolvido pelo Instituto Canadense de Cibersegurança, é amplamente utilizado em pesquisas de cibersegurança e Aprendizado de Máquina, pois foi projetado para simular um ambiente de rede corporativa real, com atividades normais e uma diversidade de ataques comuns e complexos. Entre as categorias de ataques representadas no conjunto de dados estão: *Denial of Service* (DoS), *Distributed Denial of Service* (DDoS), *Brute Force*, *Infiltration*, *Botnet*, *Heartbleed*, *PortScan* e ataques baseados em vulnerabilidades da Web, como *SQL Injection* e *Cross-Site Scripting* (XSS). Além disso, o CICIDS2017 inclui um grande volume de tráfego benigno (normal), permitindo a criação de um modelo de Aprendizado de Máquina que consegue diferenciar adequadamente tráfego legítimo e malicioso.

O conjunto de dados CICIDS2017 foi coletado ao longo de cinco dias de captura de tráfego de rede, de segunda-feira (03/07/2017) até sexta-feira (07/07/2017), com cada dia representando diferentes cenários de ataque e atividades normais. Ele contém 78 características, além de uma que representa a classe a qual a instância pertence (*Label*), e 2.830.743 fluxos de rede rotulados (instâncias), capturados pelo *CICFlowMeter*. A inclusão de uma variedade de características fornece uma base robusta para modelos de ML treinarem e identificarem padrões específicos de cada tipo de ataque, porém é importante notar que o conjunto de dados é altamente desbalanceado, com aproximadamente 80% dos fluxos sendo normais, enquanto apenas 0,000003% são rotulados como *Heartbleed*. A Tabela 1 mostra o percentual aproximado de instâncias de cada classe no conjunto de dados antes do pré-processamento.

Tabela 1 – Percentual aproximado de instâncias de cada classe no conjunto de dados antes do pré-processamento.

Classe	**Em relação ao total**	**Em relação aos ataques**
Benign	80,30%	-
Bot	0,07%	0,35%
DDoS	4,52%	22,95%
DoS GoldenEye	0,36%	1,84%
DoS Hulk	8,16%	41,44%
DoS Slowhttptest	0,19%	0,99%
DoS Slowloris	0,20%	1,04%

FTP-Patator	0,28%	1,42%
Heartbleed	0,0000039%	0,0019%
Infiltration	0,000013%	0,0064%
PortScan	5,61%	28,50%
SSH-Patator	0,21%	1,06%
Web Attack Brute Force	0,05%	0,27%
Web Attack Sql Injection	0,0000074%	0,0038%
Web Attack XSS	0,02%	0,12%

Um dos desafios associados ao uso do CICIDS2017 é a necessidade de pré-processamento extensivo. O conjunto de dados contém dados redundantes, valores ausentes e, em alguns casos, inconsistências que precisam ser corrigidas antes do treinamento dos modelos.

3.2 Pré-processamento do *dataset*

O pré-processamento do conjunto de dados é uma etapa essencial para garantir a qualidade e a consistência das informações utilizadas nos modelos de ML. No caso do CICIDS2017, essa etapa é especialmente importante devido à complexidade e ao volume de dados, que incluem dezenas de características e milhões de instâncias. O pré-processamento visa limpar, transformar e padronizar os dados, facilitando o aprendizado dos modelos e melhorando a precisão e a eficiência dos resultados. As principais etapas de pré-processamento aplicadas ao conjunto de dados incluem a limpeza dos dados (remoção de valores ausentes, infinitos e duplicados e remoção de colunas irrelevantes), normalização e redução de dimensionalidade.

Abaixo estão descritas as principais etapas de pré-processamento realizadas.

- **Tratamento de valores ausentes, infinitos e duplicados**: Valores ausentes e infinitos podem impactar negativamente o desempenho dos modelos, uma vez que não fornecem informações úteis e podem introduzir ruído nos dados. Para tratar esses valores, as instâncias que possuem valores desse tipo são removidas;
- **Remoção de colunas constantes e baixa variabilidade**: Colunas com valores constantes ou baixa variabilidade não contribuem para a identificação de padrões no tráfego de rede, pois não diferenciam instâncias de tráfego benigno (normal) de instâncias de ataque. É calculada a variância de cada coluna numérica e aquelas com variância zero são removidas, simplificando o conjunto de dados ao eliminar características irrelevantes para o treinamento dos modelos;
- **Normalização**: No CICIDS2017, as características possuem escalas diferentes, como taxas de pacotes por segundo, contagem de bytes e intervalos de tempo entre pacotes. A normalização foi realizada utilizando a técnica *StandardScaler*, que transforma

as características para uma distribuição normal com média zero e desvio padrão de 1. A normalização é definida pela fórmula

$$Z = \frac{X - u}{s}, onde$$

- Z é o valor normalizado
- X é o valor original da característica
- u é a média da característica
- s é o desvio padrão da característica

Essa transformação permite que os modelos tratem todas as características com o mesmo peso, evitando que aquelas com valores mais altos influenciem desproporcionalmente o processo de aprendizado.

- **Redução de dimensionalidade**: Para simplificar o conjunto de dados e melhorar o desempenho dos modelos de ML, foi implementada uma redução de dimensionalidade com duas técnicas: *Mean Decrease in Impurity* (MDI) e *Principal Component Analysis* (PCA).

Após a limpeza dos dados, o conjunto de dados foi reduzido para 70 características, além de uma que representa a classe a qual a instância pertence (*Label*), e 2.520.798 fluxos de rede rotulados (instâncias). Com isso, as características *Bwd PSH Flags*, *Bwd URG Flags*, *Fwd Avg Bytes/Bulk*, *Fwd Avg Packets/Bulk*, *Fwd Avg Bulk Rate*, *Bwd Avg Bytes/Bulk*, *Bwd Avg Packets/Bulk* e *Bwd Avg Bulk Rate* foram removidas do conjunto de dados. As Tabelas 2 e 3 mostram o percentual aproximado de instâncias de cada classe no conjunto de dados após a limpeza dos dados e as características do conjunto de dados após a limpeza dos dados, respectivamente.

Tabela 2 – Percentual aproximado de instâncias de cada classe no conjunto de dados após a limpeza dos dados.

Classe	**Em relação ao total**	**Em relação aos ataques**
Benign	83,11%	-
Bot	0,08%	0,46%
DDoS	5,08%	30,07%
DoS GoldenEye	0,41%	2,41%
DoS Hulk	6,85%	40,60%
DoS Slowhttptest	0,20%	1,23%
DoS Slowloris	0,21%	1,26%
FTP-Patator	0,23%	1,39%
Heartbleed	0,00044%	0,0025%
Infiltration	0,0014%	0,0084%
PortScan	3,60%	21,30%

SSH-Patator	0,12%	0,75%
Web Attack Brute Force	0,06%	0,34%
Web Attack Sql Injection	0,00083%	0,0049%
Web Attack XSS	0,025%	0,15%

Tabela 3 – Características após a limpeza dos dados.

Destination Port	Flow Duration	Total Fwd Packets
Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets
Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean
Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min
Bwd Packet Length Mean	Bwd Packet Length Std	Flow Bytes/s
Flow Packets/s	Flow IAT Mean	Flow IAT Std
Flow IAT Max	Flow IAT Min	Fwd IAT Total
Fwd IAT Mean	Fwd IAT Std	Fwd IAT Max
Fwd IAT Min	Bwd IAT Total	Bwd IAT Mean
Bwd IAT Std	Bwd IAT Max	Bwd IAT Min
Fwd PSH Flags	Fwd URG Flags	Fwd Header Length
Bwd Header Length	Fwd Packets/s	Bwd Packets/s
Min Packet Length	Max Packet Length	Packet Length Mean
Packet Length Std	Packet Length Variance	FIN Flag Count
SYN Flag Count	RST Flag Count	PSH Flag Count
ACK Flag Count	URG Flag Count	CWE Flag Count
ECE Flag Count	Down/Up Ratio	Average Packet Size
Avg Fwd Segment Size	Avg Bwd Segment Size	Fwd Header Length.1
Subflow Fwd Packets	Subflow Fwd Bytes	Subflow Bwd Packets
Subflow Bwd Bytes	Init_Win_bytes_forward	Init_Win_bytes_backward
act_data_pkt_fwd	min_seg_size_forward	Active Mean
Active Std	Active Max	Idle Max
Idle Mean	Idle Std	
Idle Min	Active Min	

3.3 Redução de dimensionalidade do *dataset*

Para simplificar o conjunto de dados e melhorar o desempenho dos modelos de ML, foi implementada uma redução de dimensionalidade com duas técnicas: *Mean Decrease in Impurity* (MDI) e *Principal Component Analysis* (PCA). A redução de dimensionalidade foi necessária devido à quantidade elevada de características no conjunto de dados CICIDS2017, o que não apenas impacta o tempo de processamento, mas também aumenta a complexidade do modelo e o risco de *overfitting*. A seguir, são descritas as implementações e os procedimentos adotados para reduzir a dimensionalidade do conjunto de dados, mantendo as informações mais relevantes para a tarefa de detecção de intrusões.

3.3.1 Redução com MDI

A técnica MDI foi utilizada para avaliar a importância das características originais diretamente, sem transformá-las em um novo espaço. A MDI é uma técnica dependente de

modelo, pois calcula a importância das características com base na redução de impureza causada por cada uma no modelo de *Random Forest*. A MDI ajudou a identificar as características mais relevantes para a tarefa de classificação, permitindo manter apenas aquelas que mais contribuíram para a detecção de intrusões.

A técnica foi aplicada em um modelo de *Random Forest* com 100 árvores, sendo selecionadas as características com importância diferente de zero. Como resultado, o conjunto de dados foi reduzido para 51 características, além de uma que representa a classe a qual a instância pertence (*Label*). A Tabela 4 mostra as características do conjunto de dados após a aplicação da técnica MDI.

Tabela 4 – Características após a aplicação da técnica MDI.

Flow Bytes/s	Total Length of Fwd Packets	Bwd Packet Length Std
Subflow Fwd Bytes	Destination Port	PSH Flag Count
Bwd Packets/s	Flow IAT Mean	Fwd Packet Length Std
Init_Win_bytes_forward	Fwd IAT Min	Init_Win_bytes_backward
min_seg_size_forward	Average Packet Size	Fwd Header Length.1
Fwd Header Length	Bwd Packet Length Mean	Idle Min
Fwd Packet Length Max	Total Length of Bwd Packets	Subflow Bwd Bytes
Active Std	Bwd IAT Mean	Bwd IAT Std
Packet Length Mean	Bwd Packet Length Max	Bwd IAT Min
Avg Bwd Segment Size	Flow IAT Std	Flow IAT Min
URG Flag Count	Min Packet Length	Max Packet Length
Fwd Packet Length Min	Flow Packets/s	Flow IAT Max
Flow Duration	Bwd Header Length	Fwd Packets/s
Fwd Packet Length Mean	Fwd IAT Total	Fwd IAT Std
Fwd IAT Mean	Fwd IAT Max	act_data_pkt_fwd
Total Fwd Packets	Fwd PSH Flags	Down/Up Ratio
Bwd IAT Max	Avg Fwd Segment Size	ACK Flag Count

3.3.2 Redução com PCA

A técnica PCA foi escolhida por sua capacidade de projetar dados em um espaço de menor dimensão, preservando a variância original ao máximo. Essa abordagem permite que os dados sejam representados de forma compacta, mas ainda mantendo as informações mais relevantes para a classificação.

A técnica foi aplicada sobre o conjunto de dados normalizado, após a limpeza dos dados e normalização, com o parâmetro de variância acumulada definido como 0,99. Assim, foram selecionadas as características que retivessem 0.99 da variabilidade dos dados originais. Como resultado, o conjunto de dados foi reduzido significativamente para 33 características, que são combinações lineares das 70 características originais do conjunto de dados, além de uma que representa a classe a qual a instância pertence (*Label*).

Ambas as técnicas, PCA e MDI, foram avaliadas no conjunto de dados. A técnica PCA é útil para reduzir dimensionalidade sem uma dependência direta do modelo de ML, enquanto a MDI proporciona uma seleção de características baseada no impacto direto sobre o desempenho do *Random Forest*. Para a construção final do modelo, foram escolhidas as características selecionadas pela técnica PCA, pois resultou em um conjunto de dados com menor complexidade, preservando as características mais relevantes, além de não depender do modelo RF.

3.4 Análise de Modelos de Aprendizado de Máquina

Foram implementados e analisados dois modelos de Aprendizado de Máquina: *Random Forest* (RF) e *Multi-Layer Perceptron* (MLP). A escolha desses modelos se deu com base em suas características adequadas para classificação em um cenário de tráfego de rede, com ambos apresentando capacidades complementares. Nesta seção, são detalhadas as implementações específicas, incluindo os parâmetros utilizados em cada modelo.

O modelo *Random Forest* foi configurado para explorar diferentes números de árvores (*n_estimators*), buscando identificar o melhor equilíbrio entre desempenho e eficiência computacional. A seguir estão os melhores parâmetros encontrados e que foram utilizados no modelo final.

- **class_weight='balanced'** - define automaticamente o peso de cada classe de acordo com sua frequência no conjunto de dados, equilibrando a influência das classes menos representadas;
- **n_estimators=25** - o número de árvores na floresta foi ajustado após testes com diferentes valores, tendo sido identificado que 25 árvores proporcionam um bom equilíbrio entre precisão e eficiência computacional;
- **outros parâmetros** - não foram alterados e, portanto, foram utilizados os valores padrões da biblioteca Scikit-Learn para o modelo.

O modelo *Multi-Layer Perceptron* foi configurado com uma estrutura de camadas ajustada para equilibrar a profundidade da rede e a eficiência computacional. A seguir estão os melhores parâmetros encontrados e que foram utilizados no modelo final.

- **solver='adam'** - o otimizador *Adam* foi utilizado para ajustar os pesos da rede neural, devido à sua rápida convergência e adaptabilidade;
- **activation='relu'** - a função de ativação *ReLU* foi escolhida devido à sua eficiência computacional;

- **hidden_layer_sizes=(20, 40, 20)** - define o número de camadas ocultas e o número de neurônios em cada camada. O parâmetro foi ajustado após testes com diferentes valores, tendo sido identificado que a melhor configuração nos testes realizados foi com 3 camadas ocultas e com o número de neurônios em cada camada sendo respectivamente de 20, 40 e 20. Essa configuração permitiu ao modelo capturar padrões complexos sem comprometer a eficiência computacional;
- **outros parâmetros** - não foram alterados e, portanto, foram utilizados os valores padrões da biblioteca Scikit-Learn para o modelo.

3.5 Aprendizado e Validação com Stratified K-Fold Cross-Validation

Para garantir uma avaliação robusta e consistente dos modelos de ML, foi utilizada a técnica *Stratified K-Fold Cross-Validation* com 10 *folds*. Essa abordagem divide o conjunto de dados em 10 subconjuntos (*folds*), preservando a proporção de classes em cada *fold*. Em cada iteração, 1 *fold* foi utilizado para validação (teste) e os outros 9 para treinamento. Esse processo foi repetido 10 vezes para obter uma média de desempenho dos resultados, garantindo que cada *fold* fosse usado tanto para treinamento quanto para validação (teste).

3.6 Avaliação das Métricas de Desempenho dos Modelos de ML

Os modelos de ML foram avaliados com métricas de desempenho ponderadas para garantir uma análise equilibrada das classes, devido ao desbalanceamento presente no conjunto de dados CICIDS2017. As métricas utilizadas são descritas a seguir.

- $Precisão_{weighted}$: representa a proporção de instâncias positivas que foram corretamente identificadas pelo modelo em relação ao total de instâncias positivas preditas pelo modelo
- $Recall_{weighted}$: mede a capacidade do modelo de identificar corretamente as instâncias positivas, avaliando a proporção de instâncias positivas que foram corretamente identificadas pelo modelo em relação ao total de instâncias positivas reais
- $F1 - Score_{weighted}$: combina precisão e *recall* em uma única métrica, destacando-se como a métrica principal para a otimização dos modelos

Essas métricas foram calculadas para cada iteração da técnica *Stratified K-Fold Cross-Validation* e, ao final, o valor médio foi calculado como resultado final.

4 Resultados

Este capítulo apresenta os resultados obtidos a partir do treinamento e validação dos modelos *Random Forest* (RF) e *Multi-Layer Perceptron* (MLP) no conjunto de dados CICIDS2017. Além disso, são discutidas as comparações com outros trabalhos relevantes, como o de Neto et al. (NETO; GOMES, 2019). A escolha do F1-Score como métrica principal reflete a importância do equilíbrio entre precisão e *recall*, já que o F1-Score é a média harmônica entre essas duas outras métricas. Além disso, durante os experimentos, apenas um parâmetro foi variado em cada teste, visando isolar o impacto de cada configuração e facilitar a análise comparativa. Além disso, todos os testes foram realizados em um computador com o processador **Intel Core i5-13600K (14 núcleos e 20 threads)** com velocidade de **3,50 GHz** e memória RAM de **16 GB** com velocidade de **2400 MHz**.

4.1 Comparação das Características e Impacto da Redução de Dimensionalidade

Para entender o impacto das técnicas de redução de dimensionalidade, foi realizada uma análise comparativa das características do conjunto de dados CICIDS2017. Inicialmente, o conjunto de dados contém 78 características, que podem ser divididas em duas categorias: aquelas extraídas diretamente dos pacotes (originais) e aquelas calculadas a partir desses dados. A Tabela 5 apresenta todas as 70 características após a limpeza dos dados, indicando para cada uma se é Original ou Calculada. Das 70 características, 18 (25,7%) são originais e 52 (74,3%) são calculadas.

Tabela 5 – Tipos das características após a limpeza dos dados.

Característica	**Tipo**	**Característica**	**Tipo**
Destination Port	Original	Flow Duration	Calculada
Total Fwd Packets	Original	Total Backward Packets	Original
Total Length of Fwd Packets	Calculada	Total Length of Bwd Packets	Calculada
Fwd Packet Length Max	Calculada	Fwd Packet Length Min	Calculada
Fwd Packet Length Mean	Calculada	Fwd Packet Length Std	Calculada
Bwd Packet Length Max	Calculada	Bwd Packet Length Min	Calculada
Bwd Packet Length Mean	Calculada	Bwd Packet Length Std	Calculada
Flow Bytes/s	Calculada	Flow Packets/s	Calculada
Flow IAT Mean	Calculada	Flow IAT Std	Calculada
Flow IAT Max	Calculada	Flow IAT Min	Calculada
Fwd IAT Total	Calculada	Fwd IAT Mean	Calculada
Fwd IAT Std	Calculada	Fwd IAT Max	Calculada
Fwd IAT Min	Calculada	Bwd IAT Total	Calculada
Bwd IAT Mean	Calculada	Bwd IAT Std	Calculada
Bwd IAT Max	Calculada	Bwd IAT Min	Calculada
Fwd PSH Flags	Original	Fwd URG Flags	Original
Fwd Header Length	Original	Bwd Header Length	Original

Fwd Packets/s	Calculada	Bwd Packets/s	Calculada
Min Packet Length	Calculada	Max Packet Length	Calculada
Packet Length Mean	Calculada	Packet Length Std	Calculada
Packet Length Variance	Calculada	FIN Flag Count	Original
SYN Flag Count	Original	RST Flag Count	Original
PSH Flag Count	Original	ACK Flag Count	Original
URG Flag Count	Original	CWE Flag Count	Original
ECE Flag Count	Original	Down/Up Ratio	Calculada
Average Packet Size	Calculada	Avg Fwd Segment Size	Calculada
Avg Bwd Segment Size	Calculada	Fwd Header Length.1	Original
Subflow Fwd Packets	Calculada	Subflow Fwd Bytes	Calculada
Subflow Bwd Packets	Calculada	Subflow Bwd Bytes	Calculada
Init_Win_bytes_forward	Original	Init_Win_bytes_backward	Original
act_data_pkt_fwd	Calculada	min_seg_size_forward	Calculada
Active Mean	Calculada	Active Std	Calculada
Active Max	Calculada	Idle Max	Calculada
Idle Mean	Calculada	Idle Std	Calculada
Idle Min	Calculada	Active Min	Calculada

A Tabela 6 apresenta todas as 51 características após a aplicação da técnica de redução de dimensionalidade MDI, indicando para cada uma se é Original ou Calculada. Das 51 características, 11 (21,6%) são originais e 40 (78,4%) são calculadas.

Tabela 6 – Tipos das características após a técnica MDI.

Característica	**Tipo**	**Característica**	**Tipo**
Destination Port	Original	Flow Duration	Calculada
Total Fwd Packets	Original	Total Length of Fwd Packets	Calculada
Total Length of Bwd Packets	Calculada	Fwd Packet Length Max	Calculada
Fwd Packet Length Min	Calculada	Fwd Packet Length Mean	Calculada
Fwd Packet Length Std	Calculada	Bwd Packet Length Max	Calculada
Bwd Packet Length Mean	Calculada	Flow Bytes/s	Calculada
Flow Packets/s	Calculada	Flow IAT Mean	Calculada
Flow IAT Std	Calculada	Flow IAT Max	Calculada
Flow IAT Min	Calculada	Fwd IAT Total	Calculada
Fwd IAT Mean	Calculada	Fwd IAT Std	Calculada
Fwd IAT Max	Calculada	Fwd IAT Min	Calculada
Bwd IAT Mean	Calculada	Bwd IAT Std	Calculada
Bwd IAT Max	Calculada	Bwd IAT Min	Calculada
Fwd PSH Flags	Original	Fwd Header Length	Original
Bwd Header Length	Original	Fwd Packets/s	Calculada
Bwd Packets/s	Calculada	Min Packet Length	Calculada
Max Packet Length	Calculada	Packet Length Mean	Calculada
PSH Flag Count	Original	ACK Flag Count	Original
URG Flag Count	Original	Down/Up Ratio	Calculada
Average Packet Size	Calculada	Avg Fwd Segment Size	Calculada
Avg Bwd Segment Size	Calculada	Fwd Header Length.1	Original
Subflow Fwd Bytes	Calculada	Subflow Bwd Bytes	Calculada
Init_Win_bytes_forward	Original	Init_Win_bytes_backward	Original
act_data_pkt_fwd	Calculada	min_seg_size_forward	Calculada
Active Std	Calculada	Idle Min	Calculada
Bwd Packet Length Std	Calculada		

Essa análise não é possível de ser realizada após a aplicação da técnica de redução de dimensionalidade PCA, pois as 33 características resultantes são combinações lineares

das características originais. Para o caso da técnica PCA, A Tabela 7 apresenta todas as 70 características do conjunto de dados após a limpeza dos dados com suas respectivas contribuições na construção dos componentes principais.

Tabela 7 – Contribuições das características para a técnica PCA.

Característica	**Contribuição**	**Característica**	**Contribuição**
Down/Up Ratio	3.843140	Init_Win_bytes_forward	3.834994
Bwd Packet Length Min	3.741707	FIN Flag Count	3.718683
URG Flag Count	3.531956	Fwd Packet Length Min	3.509398
Bwd Packets/s	3.438141	Flow Bytes/s	3.413622
Min Packet Length	3.408757	PSH Flag Count	3.293585
Init_Win_bytes_backward	3.188131	Destination Port	3.066490
Active Std	3.041965	Idle Std	2.991479
Bwd IAT Std	2.927588	Bwd IAT Total	2.927552
Active Min	2.816585	Fwd Packets/s	2.736683
Fwd PSH Flags	2.724999	SYN Flag Count	2.724999
ACK Flag Count	2.718111	Packet Length Variance	2.709899
Flow Packets/s	2.676960	Bwd IAT Max	2.491558
Flow IAT Min	2.448110	Flow Duration	2.178210
Fwd IAT Total	2.168164	Active Max	2.078096
Active Mean	2.001116	Fwd Packet Length Std	1.989283
Bwd IAT Min	1.962055	Average Packet Size	1.868765
Bwd IAT Mean	1.862927	Packet Length Mean	1.830861
min_seg_size_forward	1.809929	Bwd Header Length	1.801170
Flow IAT Mean	1.781202	Fwd Packet Length Max	1.749011
Idle Min	1.710370	Bwd Packet Length Std	1.646642
Total Length of Fwd Packets	1.631204	Subflow Fwd Bytes	1.629979
Fwd IAT Min	1.579582	Fwd IAT Std	1.556741
act_data_pkt_fwd	1.530459	Avg Bwd Segment Size	1.497598
Bwd Packet Length Mean	1.497598	Idle Mean	1.490062
Fwd Packet Length Mean	1.477602	Avg Fwd Segment Size	1.477602
Fwd IAT Max	1.380256	Flow IAT Max	1.378500
Flow IAT Std	1.375239	Idle Max	1.374867
Bwd Packet Length Max	1.297105	Fwd IAT Mean	1.289883
Max Packet Length	1.280916	Fwd Header Length	1.278254
Fwd Header Length.1	1.278254	Fwd URG Flags	1.259003
CWE Flag Count	1.259003	Packet Length Std	1.255297
RST Flag Count	1.237812	ECE Flag Count	1.237516
Subflow Bwd Packets	0.840267	Total Backward Packets	0.840267
Total Length of Bwd Packets	0.833129	Subflow Bwd Bytes	0.832998
Total Fwd Packets	0.808512	Subflow Fwd Packets	0.808512

4.2 Implementação com *Random Forest*

4.2.1 Implementação Proposta

Os resultados obtidos a partir do treinamento e validação do modelo RF com diferentes valores do parâmetro *n_estimators* (número de árvores) no conjunto de dados CICIDS2017 e utilizando as 33 características selecionadas pela técnica de redução de dimensionalidade PCA estão apresentados na Tabela 8. Mostram-se as métricas Precisão, *Recall* e F1-Score, além do tempo necessário (em minutos) para treinar o modelo.

Tabela 8 – Resultados obtidos com diferentes parâmetros no RF.

n_estimators	***Precisão***	***Recall***	**F1-Score**	**Tempo de treino**
5	0,9979	0,9980	0,9980	2,3 minutos
10	0,9980	0,9981	0,9980	3,8 minutos
20	0,9980	0,9981	0,9980	7,4 minutos
25	0,9981	0,9981	0,9981	8,7 minutos
50	0,9981	0,9981	0,9981	16,7 minutos
75	0,9981	0,9981	0,9981	24,8 minutos
100	0,9981	0,9981	0,9981	33,1 minutos
125	0,9981	0,9981	0,9981	41 minutos
150	0,9981	0,9981	0,9981	49,7 minutos

A melhor configuração encontrada foi com 25 árvores, que apresentou um excelente desempenho, alcançando um F1-Score ponderado aproximado de 0,9981 e levando apenas 8,7 minutos para treinar o modelo. Outras configurações mostraram que aumentar o número de árvores aumenta consideravelmente o tempo de treinamento e não possui um ganho significativo de desempenho.

4.2.2 Comparação com o RF do Artigo de Neto et al.

No artigo de Neto et al. (NETO; GOMES, 2019), o RF alcançou um F1-Score ponderado aproximado de 0,98 utilizando o parâmetro *n_estimators* fixado em 100, com 51 características selecionadas pela técnica de redução de dimensionalidade MDI e o conjunto de dados CICIDS2017, que foi pré-processado e aplicada a normalização por mínimo e máximo (*Min-Max*), que é definida pela fórmula

$$Z = \frac{X - min(X)}{max(X) - min(X)}, onde$$

- Z é o valor normalizado
- X é o valor original da característica
- $min(X)$ é o valor mínimo que a característica assume
- $max(X)$ é o valor máximo que a característica assume

Em contraste, este trabalho aplicou a normalização por média e desvio padrão (*StandardScaler*) e a técnica de redução de dimensionalidade PCA para reduzir o conjunto de dados para 33 características, além de uma que representa a classe a qual a instância pertence (*Label*). o RF proposto atingiu um F1-Score ponderado aproximado de 0,9981 com apenas 25 árvores. Essa melhoria pode ser atribuída à utilização da normalização *StandardScaler* e da técnica de redução de dimensionalidade PCA, que são técnicas mais robustas e eficientes no contexto em que o trabalho está inserido.

4.3 Implementação com *Multi-Layer Perceptron*

4.3.1 Implementação Proposta

Os resultados obtidos a partir do treinamento e validação do modelo *Multi-Layer Perceptron* (MLP) com diferentes valores do parâmetro *hidden_layer_sizes* (camadas ocultas) no conjunto de dados CICIDS2017 e utilizando as 33 características selecionadas pela técnica de redução de dimensionalidade PCA estão apresentados na Tabela 9. Mostram-se as métricas Precisão, *Recall* e F1-Score, além do tempo necessário (em minutos) para treinar o modelo MLP.

Tabela 9 – Resultados obtidos com diferentes parâmetros no MLP.

hidden_layer_sizes	***Precisão***	***Recall***	**F1-Score**	**Tempo de treino**
(51, 102, 51, 15)	0,9962	0,9962	0,9958	36,5 minutos
(33, 66, 33, 15)	0,9958	0,9960	0,9955	21,6 minutos
(33, 66, 33)	0,9962	0,9963	0,9959	38,4 minutos
(10, 20, 10)	0,9940	0,9940	0,9936	11,4 minutos
(20, 40, 20)	0,9962	0,9963	0,9959	13,1 minutos

A melhor configuração encontrada foi com 3 camadas ocultas e com o número de neurônios em cada camada oculta sendo respectivamente de 20, 40 e 20. O modelo apresentou um desempenho satisfatório, alcançando um F1-Score ponderado aproximado de 0,9959 e levando 13,1 minutos para treinar o modelo. Outras configurações alcançaram um F1-Score ponderado ligeiramente maior do que essa configuração, porém o tempo de treinamento é consideravelmente maior, justificando a escolha da configuração mencionada.

4.3.2 Comparação com o MLP do Artigo de Neto et al.

No artigo de Neto et al. (NETO; GOMES, 2019), o MLP alcançou um F1-Score ponderado aproximado de 0,98 utilizando o parâmetro *hidden_layer_sizes* de (51, 102, 51, 15), 51 características selecionadas pelo MDI e o conjunto de dados CICIDS2017, que foi pré-processado e aplicada a normalização por mínimo e máximo (*Min-Max*).

Em contraste, este trabalho aplicou a normalização por média e desvio padrão (*StandardScaler*) e a técnica de redução de dimensionalidade PCA para reduzir o conjunto de dados para 33 características, além de uma que representa a classe a qual a instância pertence (*Label*). o MLP proposto atingiu um F1-Score ponderado aproximado de 0,9959 com apenas 3 camadas ocultas. Essa melhoria pode ser atribuída à utilização da normalização *StandardScaler* e da técnica de redução de dimensionalidade PCA, que são técnicas mais robustas e eficientes no contexto em que o trabalho está inserido.

4.4 *Random Forest* x *Multi-Layer Perceptron*

Os melhores resultados obtidos a partir do treinamento e validação dos modelos RF e MLP no conjunto de dados CICIDS2017 estão apresentados na Tabela 10. Mostra-se as métricas Precisão, *Recall* e F1-Score, além do tempo necessário (em minutos) para treinar cada modelo.

Tabela 10 – Resultados obtidos.

Modelo	***Precisão***	***Recall***	**F1-Score**	**Tempo de treino**
RF	0,9981	0,9981	0,9981	8,7 minutos
MLP	0,9962	0,9963	0,9959	13,1 minutos

Com um F1-Score aproximado de 0,9981, o RF proposto apresentou o melhor desempenho entre os modelos avaliados. Esse resultado destaca a eficácia do RF em lidar com conjuntos de dados desbalanceados, uma característica essencial no contexto de detecção de intrusões. O tempo de treinamento de 8,7 minutos reflete o custo computacional menor em comparação com o MLP proposto.

O MLP proposto também demonstrou um desempenho robusto, com F1-Score aproximado de 0,9959. Embora ligeiramente inferior ao RF, o modelo mostrou-se eficaz na detecção dos ataques. Com um tempo de treinamento de 13,1 minutos, o MLP apresentou uma desvantagem em eficiência computacional, sendo mais adequado para cenários onde o tempo de treinamento não é um fator crítico.

A aplicação do PCA contribuiu para simplificar o conjunto de dados, reduzindo significativamente o tempo de treinamento dos modelos sem prejudicar o desempenho. A aplicação dessa técnica foi crucial para melhor o desempenho e reduzir o custo computacional comparado aos resultados obtidos no artigo de Neto et al. (NETO; GOMES, 2019).

5 Conclusão

Este trabalho apresenta uma análise de modelos de Aprendizado de Máquina no contexto de um Sistema de Detecção de Intrusões em Redes, utilizando o conjunto de dados CICIDS2017. A análise e comparação dos modelos de Aprendizado de Máquina *Random Forest* e *Multi-Layer Perceptron*, aplicando técnicas de redução de dimensionalidade como *Principal Component Analysis* (PCA) e a técnica de validação *Stratified K-Fold Cross-Validation*, que assegurou uma avaliação balanceada e confiável dos modelos, mostrou que a metodologia é eficiente para identificar tráfego malicioso, alcançando um desempenho satisfatório e com boa eficiência computacional.

Os resultados obtidos demonstraram que o modelo RF, com uma configuração de 25 árvores, alcançou um F1-Score ponderado de aproximadamente 0,9981 em apenas 8,7 minutos de treinamento, enquanto o MLP, com uma configuração de 3 camadas ocultas (20, 40, 20), obteve um F1-Score ponderado de aproximadamente 0,9959 em 13,1 minutos de treinamento. Esses resultados mostram a eficácia da metodologia utilizada, destacando o RF como o modelo com o melhor desempenho geral e o MLP como uma alternativa viável em cenários onde o tempo de treinamento não é um fator crítico.

A comparação com o trabalho de Neto et al. (NETO; GOMES, 2019) destacou avanços significativos. No artigo dos autores, o RF e o MLP alcançaram um F1-Score ponderado de aproximadamente 0,98, utilizando a técnica MDI para selecionar 51 características dentre as 78 características originais do conjunto de dados e a técnica de normalização *Min-Max*. Em contraste, neste trabalho, a aplicação da técnica PCA reduziu o número de características de 70 para 33. Essa abordagem simplificou o conjunto de dados e contribuiu para alcançar melhores resultados e uma diminuição do custo computacional.

Nesse contexto, o RF demonstrou ser a melhor escolha para a detecção de intrusões devido ao seu ótimo equilíbrio entre precisão e eficiência computacional. No entanto, o MLP continua sendo uma alternativa promissora.

5.1 Trabalhos futuros

Para trabalhos futuros, sugere-se avaliar o desempenho dos modelos em outros conjuntos de dados para verificar a generalização; explorar buscas em hiperparâmetros, como *Grid Search* e *Random Search*; explorar modelos de ML mais avançados, como redes neurais convolucionais; verificar e validar a eficácia das técnicas de normalização e redução de dimensionalidade aplicadas em outros modelos de ML.

Referências

ABDEL-BASSET, M.; GAMAL, A.; SALLAM, K. M.; ELGENDI, I.; MUNASINGHE, K.; JAMALIPOUR, A. An Optimization Model for Appraising Intrusion-Detection Systems for Network Security Communications: Applications, Challenges, and Solutions. **Sensors**, v. 22, n. 11, 2022. ISSN 1424-8220. Citado 2 vezes nas páginas 4 e 13.

BHASKAR, H.; HOYLE, D. C.; SINGH, S. Machine Learning in Bioinformatics: A Brief Survey and Recommendations for Practitioners. **Computers in Biology and Medicine**, v. 36, n. 10, p. 1104–1125, 2006. ISSN 0010-4825. Intelligent Technologies in Medicine and Bioinformatics. Citado na página 15.

BHUYAN, M. H.; BHATTACHARYYA, D. K.; KALITA, J. K. An Empirical Evaluation of Machine Learning Algorithms for the Detection of Intrusive Activity. **Pattern Recognition Letters**, v. 34, p. 307–315, 2020. Citado 2 vezes nas páginas 12 e 13.

BREIMAN, L. Random Forests. **Machine learning**, Springer, v. 45, p. 5–32, 2001. Citado 2 vezes nas páginas 13 e 16.

Cybersecurity Ventures. **Cybercrime to Cost the World $10.5 Trillion Annually by 2025**. 2021. Citado na página 9.

FABIAN, P. Scikit-Learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825, 2011. Citado 2 vezes nas páginas 16 e 24.

GAO, X.; WEN, J.; ZHANG, C. An Improved Random Forest Algorithm for Predicting Employee Turnover. **Mathematical Problems in Engineering**, Wiley Online Library, v. 2019, n. 1, p. 4140707, 2019. Citado na página 14.

International Data Corporation. **Worldwide Spending on Security Solutions Forecast to Reach $174.7 Billion in 2024**. 2021. Citado na página 9.

KHAN, M. A. A Survey of Security Issues for Cloud Computing. **Journal of Network and Computer Applications**, v. 71, p. 11–29, 2016. ISSN 1084-8045. Citado na página 12.

KOHAVI, R. et al. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: MONTREAL, CANADA. **Ijcai**. [S.l.], 1995. v. 14, n. 2, p. 1137–1145. Citado na página 19.

LABONNE, M. **Anomaly-Based Network Intrusion Detection Using Machine Learning**. Tese (Theses) — Institut Polytechnique de Paris, out. 2020. Citado 4 vezes nas páginas 9, 10, 12 e 22.

LIU, X.; WU, J.; WANG, L. Fraud Detection on Bank Transactions Using Random Forest and Support Vector Machine. **Procedia Computer Science**, v. 162, p. 221–227, 2019. Citado na página 19.

LUNDBERG, S. M.; LEE, S.-I. A Unified Approach to Interpreting Model Predictions. In: GUYON, I.; LUXBURG, U. V.; BENGIO, S.; WALLACH, H.; FERGUS, R.;

VISHWANATHAN, S.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: Curran Associates, Inc., 2017. v. 30. Citado na página 14.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). **Proceedings of the 9th Python in Science Conference**. [S.l.: s.n.], 2010. p. 56 – 61. Citado na página 24.

NETO, M. S.; GOMES, D. G. Network Intrusion Detection Systems Design: A Machine Learning Approach. In: **Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. Porto Alegre, RS, Brasil: SBC, 2019. p. 932–945. ISSN 2177-9384. Citado 7 vezes nas páginas 10, 22, 32, 35, 36, 37 e 38.

OISF (Open Information Security Foundation). **Suricata: Open Source Threat Detection Engine**. [S.l.], 2009. Accessed: 2024-09-21. Citado na página 13.

PAXSON, V. **Zeek: Network Security Monitor**. [S.l.], 1995. Accessed: 2024-09-21. Citado na página 13.

PEARSON, K. On Lines and Planes of Closest Fit to Systems of Points in Space. **The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science**, Taylor & Francis, v. 2, n. 11, p. 559–572, 1901. Citado na página 14.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 1135–1144. ISBN 9781450342322. Citado na página 14.

ROESCH, M. **Snort: The Open Source Network Intrusion Detection System**. [S.l.], 1998. Accessed: 2024-09-21. Citado na página 13.

ROSAY, A.; CARLIER, F.; LEROUX, P. MLP4NIDS: An Efficient MLP-Based Network Intrusion Detection for CICIDS2017 Dataset. In: BOUMERDASSI, S.; RENAULT, É.; MÜHLETHALER, P. (Ed.). **Machine Learning for Networking**. Cham: Springer International Publishing, 2020. p. 240–254. ISBN 978-3-030-45778-5. Citado na página 22.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. **ICISSp**, v. 1, n. 2018, p. 108–116, 2018. Citado 4 vezes nas páginas 10, 16, 22 e 25.

VIDAL, R.; MA, Y.; SASTRY, S. S.; VIDAL, R.; MA, Y.; SASTRY, S. S. **Principal Component Analysis**. [S.l.]: Springer, 2016. Citado na página 14.

WANG, L.; JONES, R. Big Data Analytics for Network Intrusion Detection: A Survey. **International Journal of Networks and communications**, v. 7, n. 1, p. 24–31, 2017. Citado na página 12.