Classificação de Ataques de Rede com Random Forest e SVM Utilizando o Dataset CIC-IDS2017

1. Introdução

A segurança de redes é uma preocupação crescente no cenário tecnológico mundial. A crescente proliferação de dispositivos conectados e a sofisticação de ameaças tecnológicas impulsionam ainda mais a necessidade de mais segurança tecnológica. Um sistema de detecção de intrusões (IDS) eficaz é necessário para identificar atividades maliciosas na rede, garantindo integridade e segurança.

Com isso, a utilização de técnicas de aprendizado de máquina surge como desafio importante para aprimorar a capacidade dos IDS's em reconhecer padrões de ataque de rede. Para enfrentar esses desafios, o dataset CIC-CIDS2017 foi escolhido devido à sua enorme abrangência e realismo, pois oferece uma representação realista dos tráfegos de rede, incluindo atividades benignas quanto a uma enorme variedade de ataques.

Este artigo tem como objetivo explorar a aplicação de dois algoritmos de aprendizado de máquina amplamente utilizados para tarefas de classificação em cibersegurança: Random Forest e Support Vector Machine (SVM). Através da utilização do dataset CID-CIDS2017, buscaremos classificar diferentes tipos de ataques de rede, avaliando o desempenho individual de cada modelo, e comparando-os.

2. Background

2.1 Dataset CIC-CIDS2017

O CIC-CIDS2017 *(Canadian Institute for Cybersecurity - Intrusion Detection System 2017)* é um dataset de detecção de intrusões reconhecido por ser bastante completo e realista. Ele foi construído com base em onze critérios essenciais para um dataset *benchmark* confiável, e se destaca por incluir tráfego de rede benigno e ataques atualizados, simulando dados do mundo real.

A coleta de dados ocorreu de 3 a 7 de julho de 2017, durante 5 dias úteis, com tráfego de segunda-feira sendo as atividades reais. Para simular tráfego real, usou-se o sistema B-Profile, que imita o comportamento humano para gerar tráfego baseado em protocolos de rede. Os ataques incluíram Brute Force (FTP e SSH), DoS, Heartbleed, Web Attack, Infiltration, Botnet e DDoS, que foram executados em diferentes momentos durante a semana.

2.2 Algoritmos de Classificação

Random Forest: É um algoritmo de aprendizado de máquina baseado em ensembles que constrói múltiplas árvores de decisão. Para classificação, o Random Forest combina previsões de árvores individuais para determinar a classe final, resultando em um modelo robusto. Além disso, o Random Forest pemite a análise da importância de diferentes

características, que no contexto atual é valioso para entender quais atributos do tráfego de rede são mais relevantes para a detecção de ataques.

Support Vector Machine(SVM): É um algoritmo de aprendizagem de máquina usado para tarefas de classificação e regressão, buscando encontrar um hiperplano para separar classes em um espaço de alta dimensionalidade. O SVM é eficaz em dados de rede que tem alta dimensionalidade e classes balanceadas.

LinearSVC: É uma implementação de Support Vector Classifier (SVC) para o caso linear. É recomendado para datasets grandes, sendo uma alternativa mais rápida que o SVM padrão. Ele é flexível pois consegue separar os dados mesmo quando eles não estão organizados em linha reta.

2.3 Técnicas de Pré-Processamento e Balanceamento

Pré-Processamento de Dados: Dados de tráfego de rede são complexos e precisam de etapas de pré processamento para garantir a qualidade e adequação dos dados. Isso inclui limpeza de valores infinitos e ausentes, padronização dos nomes das colunas e remoção de linhas duplicadas.

Seleção de Características (*Feature Selection*): Devido à alta dimensionalidade dos dados, a seleção de características é crucial para reduzir o número de atributos e melhorar o desempenho dos modelos. Essa técnica ajuda a identificar as características mais relevantes.

SMOTE(Synthetic Minority Over-sampling Technique): O dataset utilizado apresenta classes desbalanceadas, onde o número de ataques (*classes minoritárias*) é significativamente menor que a classe de tráfego normal (*classe majoritária*). O SMOTE é uma técnica que cria amostras sintéticas para classes minoritárias, ajudando a balancear o conjunto de dados de treinamento e permitindo que o modelo aprenda de forma eficaz a identificar os tipos de ataques.

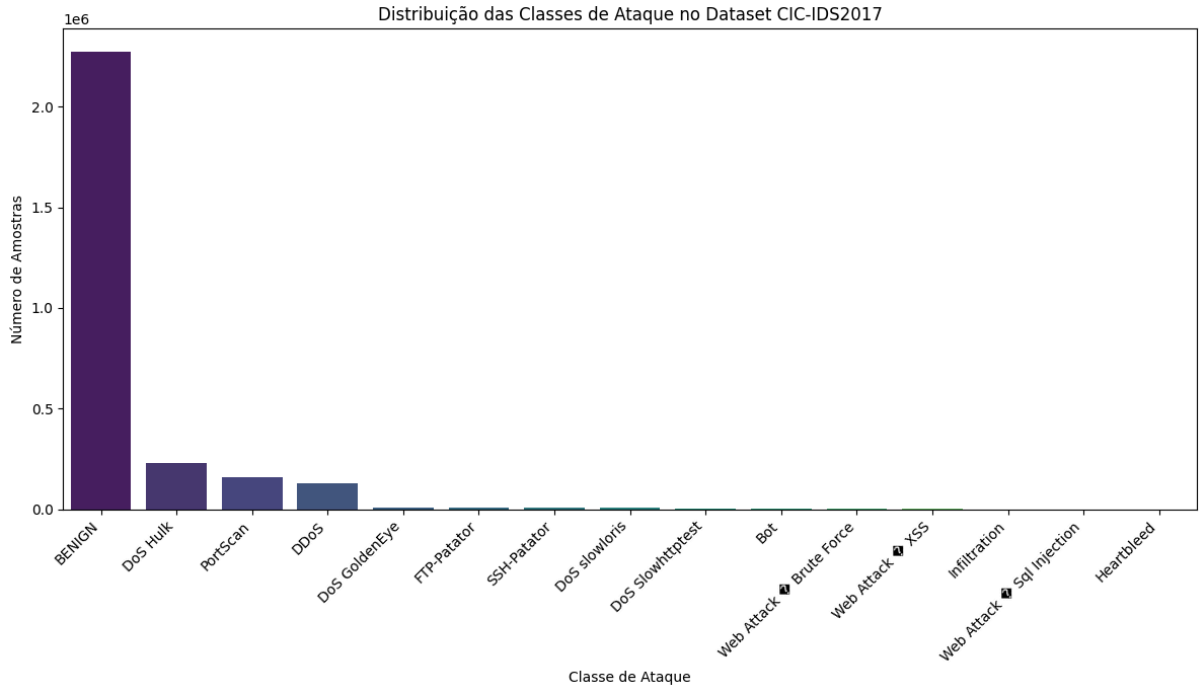
3. Métodos

3.1 Carregamento e Visão dos Dados

O dataset é composto por oito arquivos csv. Os arquivos são carregados e concatenados em um único DataFrame para formar o *dataset* completo. O *dataset* completo resultou em 2.830.743 linhas e 79 colunas.

3.2 Análise Exploratória dos Dados

Uma análise inicial da distribuição de classes revelou um grande desbalanceamento



3.3 Pré-Processamento dos Dados

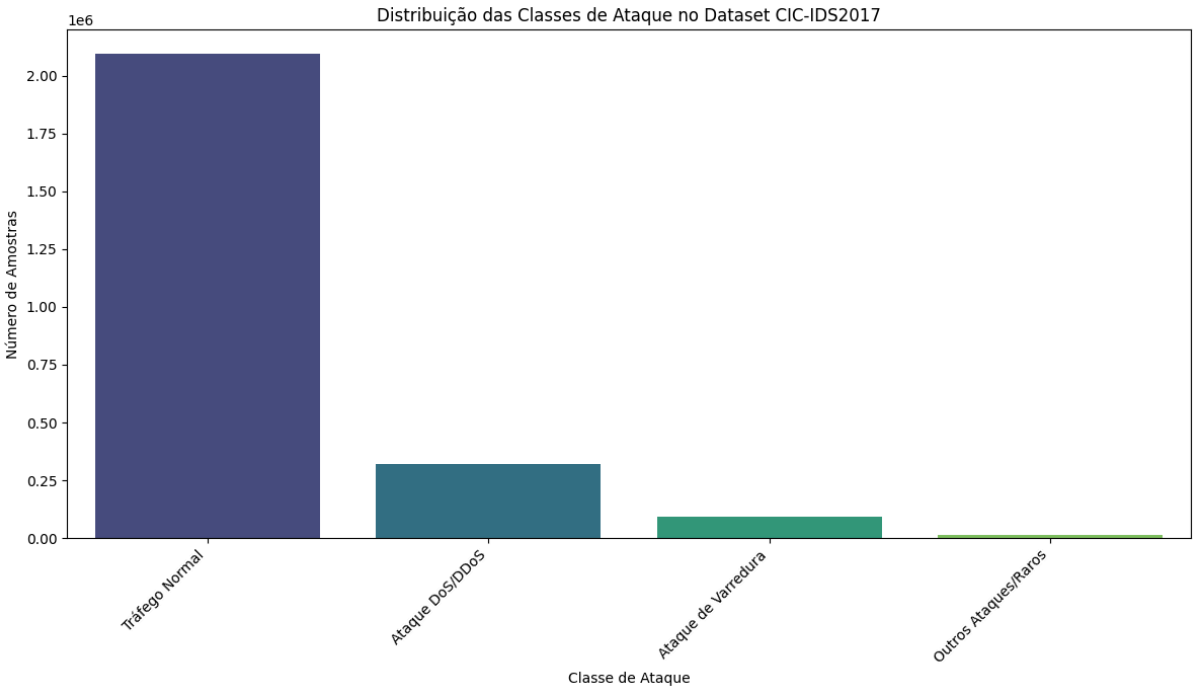
As etapas de Pré-Processamento foram essenciais para preparar os dados para treinamento:

- **Padronização dos Nomes das Colunas:** Os nomes das colunas foram convertidos para minúsculas e traduzidas para português, espaços e caracteres especiais (como pontos, barras e hífens) foram substituídos por sublinhados.

- **Tratamento de Valores Infinitos e Ausentes:** Valores infinitos (positivos e negativos) foram substituídos por NaN e, posteriormente, todas as linhas que continham NaN foram removidas do *dataset*.

- **Remoção de Duplicatas:** Todas as linhas duplicadas foram identificadas e removidas do *dataset*, garantindo que cada amostra fosse única.

- **Limpeza e Agrupamento dos Rótulos (Labels):** A coluna 'rotulo' foi limpa, removendo caracteres indesejados. Para minimizar o desbalanceamento das classes e facilitar a classificação, algumas classes de ataque menos frequentes foram agrupadas em categorias mais amplas.



Assim ficou o balanceamento das classes após o agrupamento.

3.4 Divisão e Escalonamento dos Dados

- **Separação de Features e Target:** O *dataset* foi dividido em características (X) e rótulos (y).
- **Codificação dos Rótulos:** A variável 'rotulo', que estava em formato de texto, foi codificada para valores numéricos inteiros utilizando LabelEncoder.
- **Remoção de Colunas Não Numéricas e Diminuição da Precisão:** As colunas não numéricas de X, exceto o rótulo, foram removidas, e a precisão dos tipos de dados numéricos (float64 para float32 e int64 para int32) foi reduzida para otimizar o uso da memória e a velocidade de processamento.
- **Divisão Treino-Teste:** Os dados foram divididos em conjuntos de treino e teste na proporção de 70% e 30%, utilizando train_test_split com *stratify* para garantir que a distribuição das classes fosse mantida em ambos os conjuntos.
- **Escalonamento das Features:** As características numéricas foram padronizadas utilizando StandardScaler. Este passo garante que todas as *features* tenham média zero e desvio padrão um, prevenindo que características com grandes escalas dominem o processo de treinamento do modelo.
- **Remoção de Colunas sem Variância:** Colunas com variância zero (ou seja, valores constantes) foram identificadas e removidas dos conjuntos de treino e teste escalonados, pois não fornecem informações relevantes para o modelo.

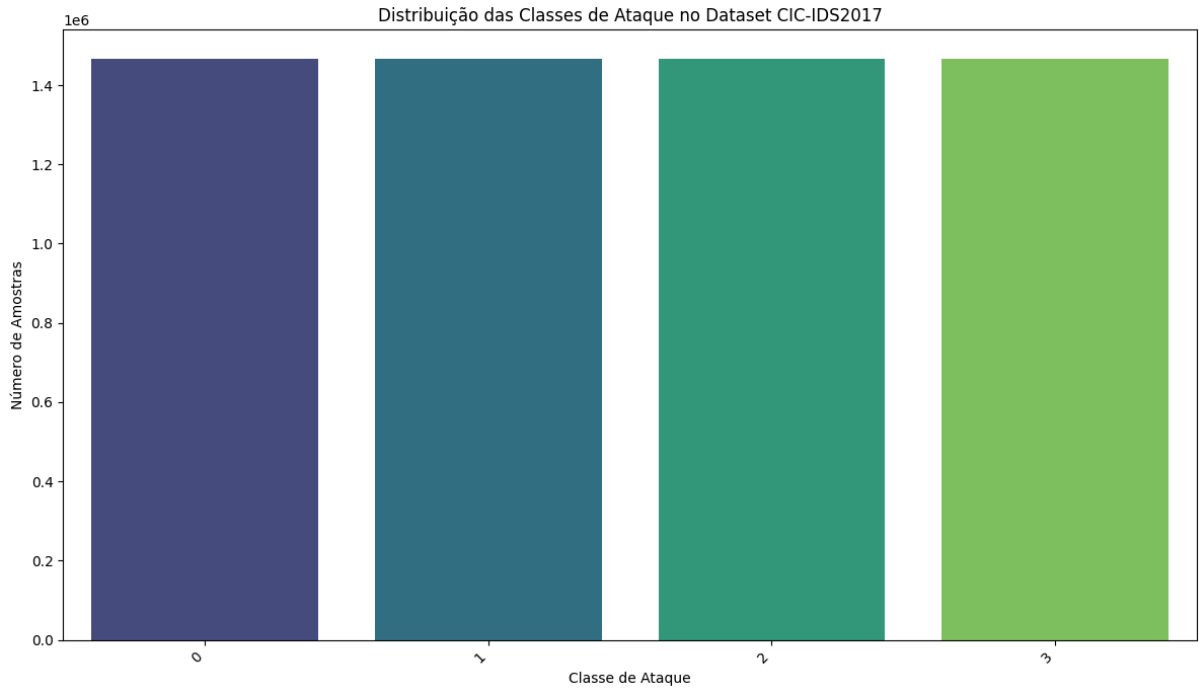
3.5 Seleção de Características (*Feature Selection*)

A dimensionalidade dos dados foi reduzida utilizando o *SelectKBest* para selecionar as 30 características mais relevantes. Este método ajuda a melhorar a eficiência computacional e desempenho do modelo focando apenas nos atributos mais importantes.

3.6 Balanceamento de Classes com SMOTE

Para resolver o problema das classes desbalanceadas no conjunto de treino, foi aplicada a técnica de SMOTE. O SMOTE gerou amostras sintéticas para classes minoritárias, garantindo que todas as classes tivessem a mesma quantidade de amostras.

Após a aplicação do SMOTE o balanceamento ficou da seguinte forma:



3.7 Treinamento e Avaliação dos Modelos

Dois modelos de classificação, Random Forest e SVM(*LinearSVC*), foram implementados e avaliados.

3.7.1 Random Forest:

- **Instanciação:** Um modelo *RandomForestClassifier* foi instanciado com *random_state=42* para reprodutibilidade e *n_jobs=-1* para utilizar todos os núcleos da CPU, acelerando o treinamento.
- **Treinamento:** O modelo foi treinado com os dados balanceados e com as características selecionadas.
- **Inferência:** As previsões foram realizadas no conjunto de teste, e o tempo de inferência foi calculado para avaliar a agilidade do modelo em cenários reais.
- **Avaliação:** O desempenho do Random Forest foi avaliado utilizando as seguintes

métricas: acurácia, precisão, recall e F1-score (por classe e ponderado), além da matriz de confusão. A importância das características também foi analisada para identificar os atributos mais influentes na classificação.

3.7.2 SVM(*LinearSVC*):

- **Instanciação:** Um modelo *LinearSVC* foi instanciado com *random_state=42* e *max_iter=10000*.
- **Treinamento:** O modelo SVM foi treinado com os mesmos dados de treino balanceados e com características selecionadas.
- **Inferência:** As previsões foram geradas no conjunto de teste para avaliar o tempo de inferência.
- **Avaliação:** O desempenho do SVM foi analisado pelas mesmas métricas do Random Forest: acurácia, precisão, recall, F1-score e matriz de confusão. Além disso, os coeficientes do modelo foram utilizados para inferir a importância das características.

A comparação entre os dois modelos foi realizada com base em todas as métricas de desempenho e nos tempos de treinamento e inferência.

4. Resultados e Discussões

Os resultados obtidos com os modelos Random Forest e SVM para a classificação são apresentados:

4.1 Resultados do Random Forest

O modelo Random Forest demonstrou um desempenho notável na classificação de ataques de rede.

Métricas de Classificação:

- Acurácia: 0.9968
- F1-Score (ponderado): 0.9968
- Precisão (ponderado): 0.9970
- Recall (ponderado): 0.9968

O relatório de classificação detalha o desempenho por classe:

Classe de Ataque	Precisão	Recall	F1-Score

Ataque DoS/DDoS	1.00	1.00	1.00
Ataque de Varredura	0.99	0.99	0.99
Outros Ataques/Raros	0.79	0.95	0.87
Tráfego Normal	1.00	1.00	1.00

Eficiência Computacional:

- Tempo de Treinamento: 10.50 minutos
- Tempo de Inferência: 2.32 segundos

4.2 Resultados do SVM(*LinearSVC*)

O modelo SVM apresentou desempenho inferior ao Random Forest, principalmente em classes minoritárias.

Métricas de Classificação:

- Acurácia: 0.8615
- F1-Score (ponderado): 0.9044
- Precisão (ponderado): 0.9644
- Recall (ponderado): 0.8615

O relatório de classificação detalha o desempenho por classe:

Classe de Ataque	**Precisão**	**Recall**	**F1-Score**
Ataque DoS/DDoS	0.87	0.90	0.88

Ataque de Varredura	0.65	1.00	0.79
Outros Ataques/Raros	0.05	0.93	0.09
Tráfego Normal	1.00	0.85	0.92

Eficiência Computacional:

- Tempo de Treinamento: 23.84 minutos
- Tempo de Inferência: 0.23 segundos

4.3 Discussão Comparativa

A comparação entre o Random Forest e o SVM(LinearSVC) revela que o Random Forest superou o SVM em todas as métricas de classificação. O Random Forest alcançou uma acurácia próxima de 100%, com pontuações de precisão, recall e F1-score de 1.00 para as classes "Ataque DoS/DDoS" e "Tráfego Normal". Para a classe "Outros Ataques/Raros" o Random Forest obteve um F1-score expressivo de 0.87, o que indica a eficácia da técnica SMOTE no balanceamento dos dados de treinamento e a capacidade do modelo em aprender padrões de ataques menos comuns.

Já o SVM (LinearSVC) demonstrou um desempenho inferior, com acurácia geral de 0.86. O F1-score para "Outros Ataques/Raros" foi de apenas 0.09, e para "Ataque de Varredura", foi de 0.79. Esses resultados sugerem que o SVM, apesar do pré-processamento e balanceamento, não conseguiu generalizar tão bem quanto o Random Forest.

Em eficiência computacional, o Random Forest teve um tempo de treinamento mais rápido (10.50 minutos contra 23.84 minutos do SVM). No entanto, o SVM (LinearSVC) foi significativamente mais rápido na fase de inferência (0.23 segundos contra 2.32 segundos do Random Forest). Essa diferença na velocidade de inferência é um fator crítico em aplicações de detecção de intrusão em tempo real, onde a agilidade no reconhecimento é essencial.

5. Conclusão

Este trabalho explorou a aplicação de algoritmos de machine learning, Random Forest e SVM, para a classificação de ataques de rede utilizando o dataset CIC-IDS2017. Os resultados obtidos demonstram a eficácia desses modelos na detecção de intrusões.

O modelo Random Forest se destacou como a melhor escolha para esta tarefa,

alcançando uma acurácia geral de 0.99, e pontuações de precisão, recall e F1-score próximas de 1.00 para as classes de "Tráfego Normal" e "Ataque DoS/DDoS". Sua capacidade de generalização foi evidente, mesmo com a presença de classes minoritárias, em grande parte devido à aplicação da técnica SMOTE, que balanceou o conjunto de treinamento, e à seleção de características relevantes, que otimizou o processo de aprendizado. A análise da importância das características no Random Forest também forneceu *insights* valiosos sobre quais atributos do tráfego de rede são mais críticos para a detecção de ataques.

Por outro lado, o SVM(*LinearSVC*), embora eficaz em classes majoritárias, demonstrou limitações ao lidar com as classes minoritárias e ataques específicos, com um F1-score de 0.09 para "Outros Ataques/Raros". Apesar de ser mais rápido na inferência, seu tempo de treinamento foi superior ao do Random Forest, e seu desempenho em termos de acurácia, precisão, recall e F1-score geral foi inferior.

A implementação e comparação dos modelos Random Forest e SVM no *dataset* CIC-IDS2017 ressaltam a importância de um pré-processamento adequado, seleção de características e balanceamento de classes para o desenvolvimento de sistemas de detecção de intrusões eficazes. Para futuras melhorias, poderiam ser exploradas diferentes estratégias de *kernel* em SVM, outras técnicas de seleção de características, e a aplicação de modelos de *deep learning*, que poderiam oferecer ainda mais insights e um desempenho aprimorado na identificação de ataques de rede.

Referências

Coutinho, B. (2019, 28 de julho). ***Modelos de Predição | SVM. Aprenda a criar seu primeiro algoritmo de Classificação com SVM.*** Medium. Disponível em: https://medium.com/turing-talks/turing-talks-12-classificação-por-svm-f4598094a3f1

Pessanha, C. (2019, 20 de novembro). ***Random Forest: como funciona um dos algoritmos mais populares de ML.*** Medium. Disponível em: https://medium.com/cinthiabpessanha/random-forest-como-funciona-um-dos-algoritmos-mais-populares-de-ml-cc1b8a58b3b4

Singh, R. (2024, 17 de outubro). ***Support Vector Machine(SVM).*** Medium. Disponível em: https://medium.com/@RobuRishabh/support-vector-machines-svm-27cd45b74fbb

Soni, A. (2023, 29 de outubro). ***Linear SVM Classification.*** Medium. Disponível em: https://medium.com/@akhil0435/linear-svm-classification-40dde297c931