UNIVERSIDADE DE ITAÚNA

Curso de Graduação em Ciência da Computação

Davi Ventura Cardoso Perdigão

Análise de Atividades e Detecção de Intrusão em Rede utilizando o Conjunto de Dados UNSW-NB15

Itaúna

2023

SUMÁRIO

1. Introdução

1.1. Contextualização

A **segurança de rede** é um desafio **crítico** na era da tecnologia, em que ameaças cibernéticas estão em constante evolução. Segundo um relatório da **Verizon (2020)**, os ataques cibernéticos são responsáveis por mais de **40%** dos **incidentes de segurança** em empresas de todo o mundo. As consequências desses ataques podem ser desastrosas, como **perda de dados**, **interrupção dos negócios**, **perda de reputação** e **prejuízos financeiros**. Para **proteger** sistemas e dados contra ataques maliciosos, é essencial ter uma **compreensão** profunda das atividades de rede e ser capaz de **detectar** comportamentos suspeitos. Nesse contexto, o conjunto de dados **UNSW-NB15** se destaca como uma valiosa fonte de informações, pois foi criado especificamente para **simular** uma variedade de **atividades normais** e **comportamentos de ataques contemporâneos**.

1.2. O problema proposto

O problema proposto neste projeto é desenvolver uma **análise detalhada** do conjunto de dados **UNSW-NB15**, com o objetivo de **identificar** e **detectar** possíveis **atividades maliciosas** e **ataques de rede**. Especificamente, iremos explorar os pacotes de rede brutos criados pela ferramenta **IXIA PerfectStorm**, que oferecem um híbrido de atividades normais e comportamentos sintéticos de ataque.

- **Por que esse problema é importante?**

A segurança da rede é crucial para garantir a **integridade**, **confidencialidade** e **disponibilidade dos dados**. Ao analisar o conjunto de dados **UNSW-NB15**, podemos obter **insights** valiosos sobre os padrões e comportamentos que podem

indicar **atividades maliciosas**, fortalecendo as **medidas de segurança** e **auxiliando na prevenção** e **resposta a ataques**.

- **De quem são os dados analisados?**

Os dados são provenientes do **Cyber Range Lab do Australian Centre for Cyber Security (ACCS)**, capturados usando a ferramenta **Tcpdump**.

- **Quais os objetivos dessa análise?**

O objetivo é identificar e detectar diferentes tipos de ataques presentes nos dados, como ***Fuzzers***, ***Analysis***, ***Backdoors***, ***DoS***, ***Exploits***, ***Generic***, ***Reconnaissance***, ***Shellcode*** e ***Worms***. Pretendemos desenvolver algoritmos específicos para extrair características relevantes do conjunto de dados.

- **Qual o período está sendo analisado?**

Os dados são provenientes da captura de **tráfego bruto**, resultando em um **conjunto de dados** com informações de atividades de rede ao longo de um determinado período de tempo. Infelizmente a ferramenta utilizada **não disponibiliza** a informação do período de tempo exato.

1.3. Objetivos

Os objetivos do trabalho em questão são:

- Realizar uma **análise aprofundada** dos pacotes de rede brutos do conjunto de dados **UNSW-NB15**;
- Desenvolver **algoritmos** e **técnicas** de **detecção de ataques** com base nas características extraídas do conjunto de dados;

- Avaliar o **desempenho** de diferentes modelos de detecção de ataques em termos de **acurácia**, **precisão**, **recall** e **F1-Score**, além de analisar o **tempo** e **pontuação** de treinamento dos mesmos;
- Fornecer **insights** e **recomendações** para fortalecer as **medidas de segurança da rede** com base nos **padrões** e **comportamentos** identificados nos dados.

Com esses objetivos, pretende-se **contribuir** para a área de **segurança de rede**, bem como para a área de **aprendizagem de máquina**, buscando **aprimorar** as técnicas e ferramentas de **detecção de ataques**, aumentando a **eficiência** e **eficácia** na **proteção de sistemas e dados** contra ameaças cibernéticas.

2. Coleta de Dados

Os dados do database **UNSW-NB15** foram obtidos no repositório do **Kaggle**. O conjunto de dados é fornecido em formato **CSV** e está organizado em diversas colunas que representam diferentes atributos e características das conexões de rede.

Ao trabalhar com um conjunto de dados, é comum dividir as informações em conjuntos de **treinamento** e **teste** para fins de **desenvolvimento** e **avaliação** de modelos de aprendizado de máquina. No contexto deste projeto, foram selecionadas **duas tabelas**, uma destinada à **treinamento** e outra à **teste**, que contêm **informações relevantes** para a **análise** e **detecção de ataques**.

Para criar um **DataFrame** abrangente, as duas tabelas foram **mescladas**, **combinando** as informações do **conjunto** de **treinamento** e do conjunto de teste em **uma única estrutura de dados**.

A mesclagem das tabelas de treinamento e teste em um único DataFrame proporciona uma **visão mais completa** dos dados e permite uma **análise abrangente** e consistente das **características** das conexões de rede e dos **rótulos** de classe associados a elas. Essa abordagem integrada ajuda a **construir modelos** de **detecção de ataques** mais **robustos** e **confiáveis**.

A seguir, é possível visualizar uma tabela com a **descrição** de cada **campo/coluna**:

Figura 1 - Descrição de cada campo/coluna do conjunto de dados UNSW-NB15.

Nome da coluna/campo	Descrição	Tipo
srcip	Endereço IP de origem	Categórico
sport	Porta de origem	Numérico
dstip	Endereço IP de destino	Categórico
dsport	Porta de destino	Numérico
proto	Protocolo de transporte	Categórico
state	Estado da conexão	Categórico
dur	Duração da conexão	Numérico
sbytes	Bytes enviados da origem para o destino	Numérico
dbytes	Bytes recebidos do destino para a origem	Numérico
sttl	Time to Live do estado de rastreamento da conexão	Numérico
dttl	Time to Live do destino	Numérico
sload	Taxa de carga de origem	Numérico
dload	Taxa de carga de destino	Numérico
sloss	Pacotes perdidos na origem	Numérico
dloss	Pacotes perdidos no destino	Numérico
service	Tipo de serviço	Categórico
serror	Erro na origem	Categórico
derror	Erro no destino	Categórico
srv_serror	Erro de serviço na origem	Categórico
srv_error	Erro de serviço no destino	Categórico
rerror	Erro na resposta	Categórico
srv_rerror	Erro de serviço na resposta	Categórico
same_srv_rate	Taxa de conexões para o mesmo serviço	Numérico
diff_srv_rate	Taxa de conexões para diferentes serviços	Numérico
srv_diff_host_rate	Taxa de conexões para hosts diferentes	Numérico
dst_host_count	Número de hosts de destino	Numérico
dst_host_srv_count	Número de serviços do host de destino	Numérico
dst_host_same_srv_rate	Taxa de conexões para o mesmo serviço no host de destino	Numérico
dst_host_diff_srv_rate	Taxa de conexões para diferentes serviços no host de destino	Numérico
dst_host_same_src_port_rate	Taxa de conexões para a mesma porta de origem no host de destino	Numérico
dst_host_srv_diff_host_rate	Taxa de conexões para hosts diferentes a partir do mesmo serviço do host de destino	Numérico
label	Rótulo de classe (ataque ou normal)	Categórico

Fonte: Compilação dos Autores, 2023.

3. Processamento/Tratamento de Dados

Antes de iniciarmos a análise, conforme a própria detentora dos dados, os conjuntos de treinamento e teste foram **invertidos**, portanto, alteramos os nomes antes de carregá-los dos arquivos CSV.

Para garantir o equilíbrio entre os conjuntos de treinamento e teste e evitar o processamento duplo, decidimos **concatená-los**, gerando assim um **novo dataframe**. Após isso, dividiu-os com uma proporção diferente utilizando **sklearn.model_selection.train_test_split().** Logo, esse novo dataframe ficou com 94823664 bytes, ou seja, aproximadamente **1 gigabyte**.

Para **diminuir a complexidade** em manipular e analisar esses dados, realizou-se uma redução de dimensionalidade da base. Para tal, utilizamos os seguintes critérios: **descarte de features desnecessárias,** sendo elas ***id*****,** pois serve apenas para identificação, e ***attack_cat*****,** pois este é um problema de classificação binária, ou seja, utilizaremos apenas rótulos para classificar ataque (1) ou normal (0), não necessitando dos detalhes do ataque. **Identificação e descarte das colunas altamente correlacionadas** (acima de 98%)**,** sendo elas ***sbytes sloss*** (0.995771577240686)**,** ***dbytes dloss*** (0.9967111338314709)**,** ***swin dwin*** (0.9804584289136614) e ***is_ftp_login ct_ftp_cmd*** (0.9988554882935945). É importante ressaltar que, para identificar a correlação citada anteriormente, realizou-se a **codificação** de recursos **categóricos** (***proto***, ***service*** e ***state***) usando **LabelEncoder**.

4. Análise e Exploração dos Dados

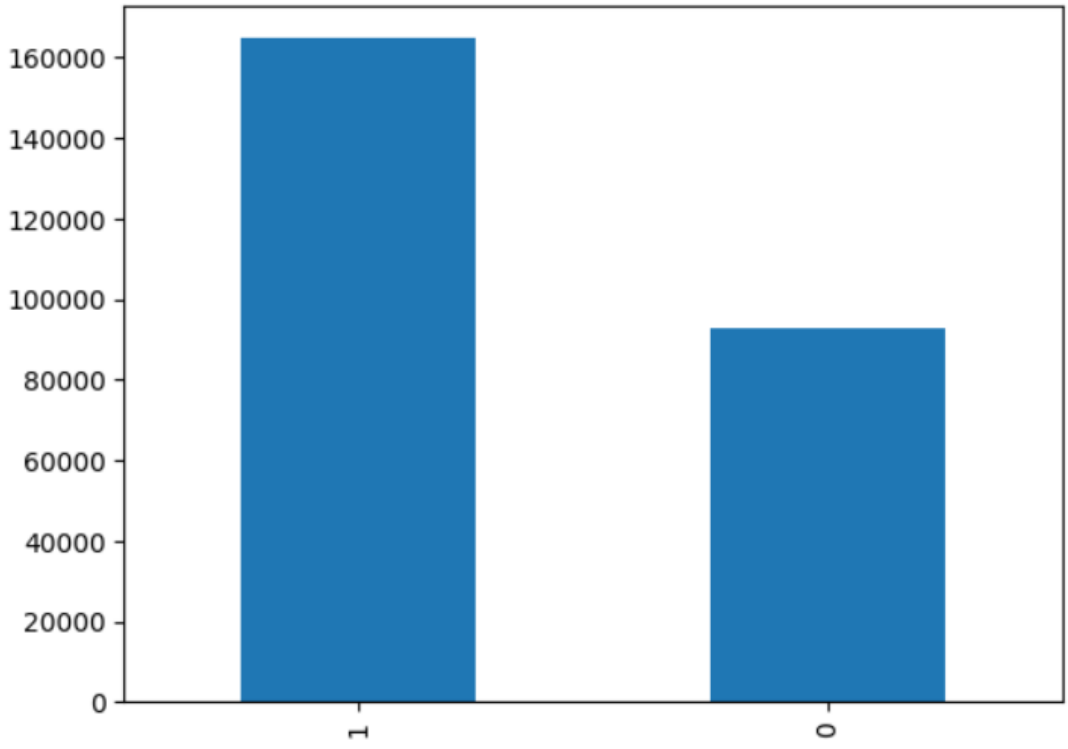
Além dos processos citados no capítulo anterior, realizou-se procedimentos padrões quando se trata da análise exploratória dos dados, sendo eles: ***df.head()*** (verificando o cabeçalho do dataset), ***df.tail()*** (verificando o rodapé do dataset), ***df.shape*** (verificando o formato do dataset, no caso, 257673 linhas e 45 colunas antes da redução de dimensionalidade e 257673 linhas e 39 colunas após a redução

de dimensionalidade), ***df.info()*** (obtendo Informações do dataset, incluindo o tipo de dados de cada coluna), ***df.describe(include="all")*** (obtendo estatísticas descritivas do dataframe), ***df.duplicated().sum()*** (verificando se existem linhas duplicadas, no caso, não existem), ***df.isnull().sum()*** (verificando se existem valores nulos, no caso, não existem),

Uma modificação importante realizada foi a de renomear todas as colunas para nomes mais descritivos, a fim de facilitar na interpretação dos dados, resultando no seguinte: 'dur': ***'duração'***, 'proto': ***'protocolo'***, 'service': ***'serviço'***, 'state': ***'estado'***, 'spkts': ***'cont. pacotes'***, 'dpkts': ***'destino'***, 'sbytes': ***'Bytes orig/dest'***, 'dbytes': ***'destino bytes'***, 'rate': ***'taxa transferência'***, 'sttl': ***'tempo orig/dest'***, 'dttl': ***'tempo dest/fonte'***, 'sload': ***'Bits orig p/ segundo'***, 'dload': ***'Bits dest p/ segundo'***, 'sloss': ***'perda pacotes orig'***, 'dloss': ***'perda pacotes dest'***, 'sinpkt': ***'tempo IP orig'***, 'dinpkt': ***'tempo IP dest'***, 'sjit': ***'tremulação da fonte (mSec)'***, 'djit': ***'tremulação de destino (mSec)'***, 'swin': ***'janela TCP orig'***, 'stcpb': ***'seq TCP orig'***, 'dtcpb': ***'seq TCP dest'***, 'dwin': ***'janela TCP dest'***, 'tcprtt': ***'tempo ida/volta TCP'***, 'synack': ***'tempo config TCP SYN/SYN_ACK'***, 'ackdat': ***'tempo config TCP SYN_ACK/ACK'***, 'smean': ***'média tam pacote src'***, 'dmean': ***'média tam pacote dst'***, 'trans_depth': ***'solic/resp http'***, 'response_body_len': ***'tam real conteúdo'***, 'ct_srv_src': ***'serv (14) orig (3)'***, 'ct_state_ttl': ***'tempo de vida orig/dest'***, 'ct_dst_ltm': ***'dest (3)'***, 'ct_src_dport_ltm': ***'orig (1) dest (4)'***, 'ct_dst_sport_ltm': ***'dest (3) orig (2)'***, 'ct_dst_src_ltm': ***'orig (1) dest (3)'***, 'is_ftp_login': ***'login user/senha'***, 'ct_ftp_cmd': ***'fluxos ftp'***, 'ct_flw_http_mthd': ***'fluxos com métodos http'***, 'ct_src_ltm': ***'orig (1)'***, 'ct_srv_dst': ***'serv (14) dest (3)'***, 'is_sm_ips_ports': ***'IP orig/dest ou porta iguais'***, 'label': ***'rótulo'***.

Logo após, realizou-se a verificação da proporção entre os resultados da target (tráfego em **ataque** ou **normal**). A proporção entre o ataque (**1**) e normal (**0**) não é igual, **64%** e **36%** respectivamente, porém apenas ligeiramente desequilibrada. Portanto, não realizamos nenhuma correção na amostragem.

Figura 2 - Gráfico ilustrando a proporção entre resultados da target.

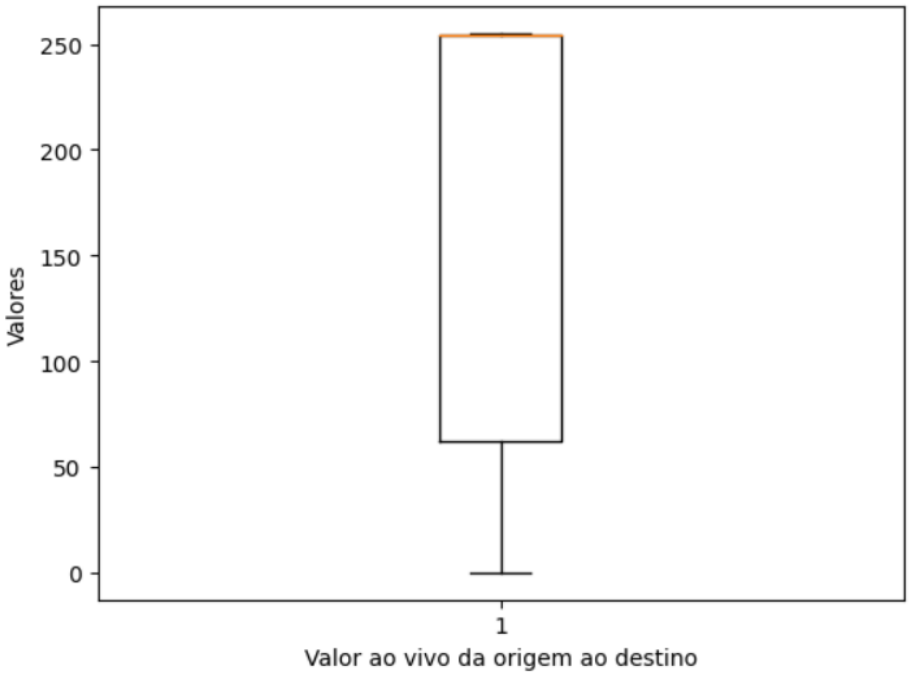


Fonte: Compilação dos Autores, 2023.

Outra análise realizada foi a de correlação entre as colunas do dataframe e a target de rótulo. Filtrou-se pelas **5** colunas com **maior correlação**, e foram elas: ***'tempo orig/dest'*** (62%), ***'tempo de vida orig/dest'*** (47%), ***'estado'*** (46%), ***'dest (3) orig (2)'*** (37%) e ***'janela TCP orig'*** (36%). A coluna ***'tempo orig/dest',*** também conhecida como **Source Time to Live,** refere-se ao tempo de vida da comunicação entre a origem e o destino em uma conexão de rede, e apesar dela apresentar uma **correlação** relativamente **alta** com a target, optamos por **não removê-la**. Isso pois, esse fator pode estar relacionado a **outros problemas** para além da rede estar sob ataque ou não, como por exemplo, a **velocidade da rede**. Segue abaixo os gráficos **boxplot** dessas colunas que foram gerados na exploração de dados, bem como uma **interpretação** acerca dos mesmos e possíveis **hipóteses** da alta correlação dessas colunas com a target:

- **tempo orig/dest:**

Figura 3 - Gráfico boxplot da coluna '*tempo orig/des*'.



Fonte: Compilação dos Autores, 2023.

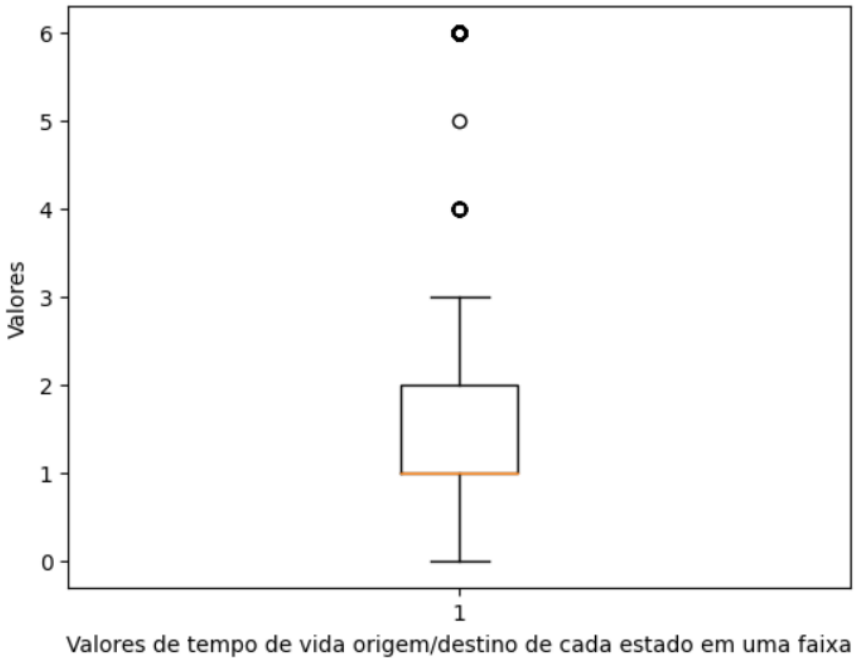
Como já citado anteriormente, essa coluna refere-se ao tempo de vida da comunicação entre a origem e o destino em uma conexão de rede. Considerando a alta correlação com a target, isso pode estar relacionado à persistência dos pacotes enviados pela fonte, caso muito comum em atividades maliciosas ou a determinados tipos de tráfego de rede. Além disso, observa-se as seguintes estatísticas no gráfico:

1. O valor **mínimo** é 0 e o valor **máximo** é 255;
2. O primeiro quartil (**Q1**) é 62, a mediana (**Q2**) é 254 e o terceiro quartil (**Q3**) também é 254;
3. O boxplot mostra que a mediana (**Q2**) está localizada aproximadamente no meio do boxplot, indicando que há uma **concentração** significativa de valores ao redor desse ponto;

4. A **caixa (box)** do boxplot é relativamente **pequena** em relação à faixa total dos dados, sugerindo que a maior parte dos valores se **concentra** em um **intervalo estreito**;
5. **Não há outliers** além das hastes do boxplot.

- **tempo de vida orig/dest:**

Figura 4 - Gráfico boxplot da coluna '*tempo de vida orig/des*'.



Fonte: Compilação dos Autores, 2023.

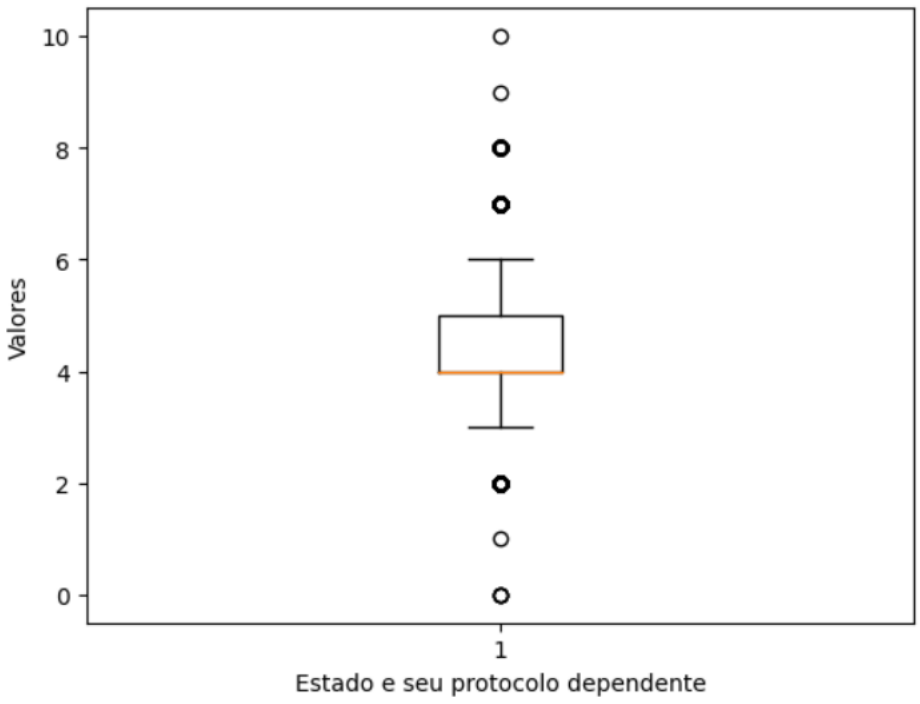
Essa coluna **(Connection Tracking State Time to Live)** indica o tempo de vida do estado de conexão, ou seja, essa medida controla por quanto tempo uma conexão é considerada ativa antes de ser descartada ou expirada. Ela pode estar associada a conexões maliciosas ou atividades de intrusão devido à sua relevância na análise de tráfego de rede e detecção de comportamentos anômalos registrados em um firewall ou sistema de segurança. Além disso, observa-se as seguintes estatísticas no gráfico:

1. O valor **mínimo** é 0 e o valor **máximo** é 6;

2. O primeiro quartil (**Q1**) é 1, a mediana (**Q2**) também é 1 e o terceiro quartil (**Q3**) é 2;
3. O boxplot mostra que a mediana (**Q2**) está localizada no meio do boxplot, indicando que há uma **concentração** significativa de valores ao redor desse ponto;
4. A **caixa (box)** do boxplot é relativamente **pequena** em relação à faixa total dos dados, sugerindo que a maior parte dos valores se **concentra** em um **intervalo estreito**;
5. Os **outliers** encontrados são os seguintes: 6 (**52 registros**), 4 (**458 registros**) e 5 (**49080 registros**). Esses valores estão **acima** do **limite superior** do boxplot e são considerados **discrepantes** em relação aos demais, porém eles são **relevantes** para a análise em questão, tendo em conta que os rótulos associados às linhas dos mesmos potencialmente classificam 'ataque' (1).

- **estado:**

Figura 5 - Gráfico boxplot da coluna '*estado*'.



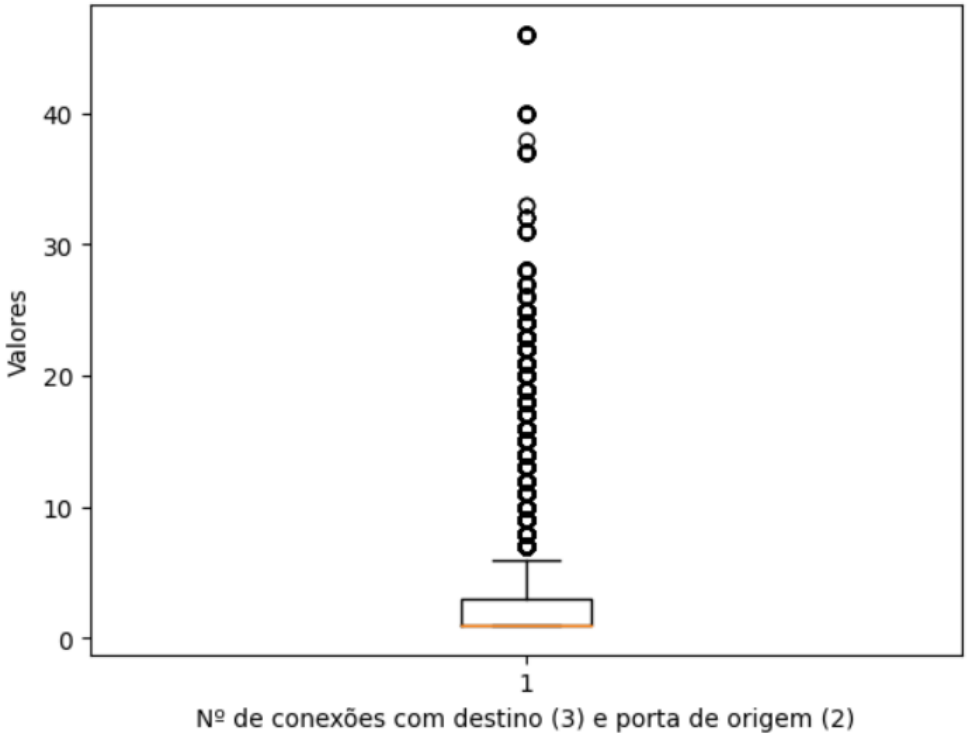
Fonte: Compilação dos Autores, 2023.

Nesse caso, a correlação entre a coluna 'estado' e a target pode ser significativa devido à natureza da análise de conexões. O estado de uma conexão é uma informação crucial para determinar se a conexão é legítima ou suspeita, especialmente em contextos de segurança de rede ou detecção de intrusões. A correlação pode surgir porque certos estados de conexão estão associados a padrões de comportamento específicos ou tipos de atividade de rede. Além disso, observa-se as seguintes estatísticas no gráfico:

1. O valor **mínimo** é 0 e o valor **máximo** é 10, indicando que existem **11 estados diferentes**;
2. O primeiro quartil (**Q1**) é 4, a mediana (**Q2**) também é 4 e o terceiro quartil (**Q3**) é 5;
3. O boxplot mostra que a mediana (**Q2**) está localizada em **4**, indicando que a **maioria** dos valores se **concentra** nesse ponto;
4. A caixa (box) do boxplot é relativamente **pequena**, sugerindo que a maioria dos valores está **concentrada** em um **intervalo estreito**;
5. Existem vários **outliers** representados, indicando que **7 dos 11 estados são discrepantes** com relação a outros **4**, que são encontrados com **maior recorrência**. Isso ocorre devido a essa grande **variedade** de estados, identificando um comportamento **incomum** nos rótulos associados às linhas dos mesmos (ataque).

- **dest (3) orig (2):**

Figura 6 - Gráfico boxplot da coluna ‘*dest (3) orig (2)*’.



Fonte: Compilação dos Autores, 2023.

Essa coluna indica o número de vezes que um determinado par de porta de destino e porta de origem foi observado nas conexões de rede. Essa informação pode ser útil para identificar padrões incomuns ou suspeitos de comunicação entre uma fonte e um destino específico.

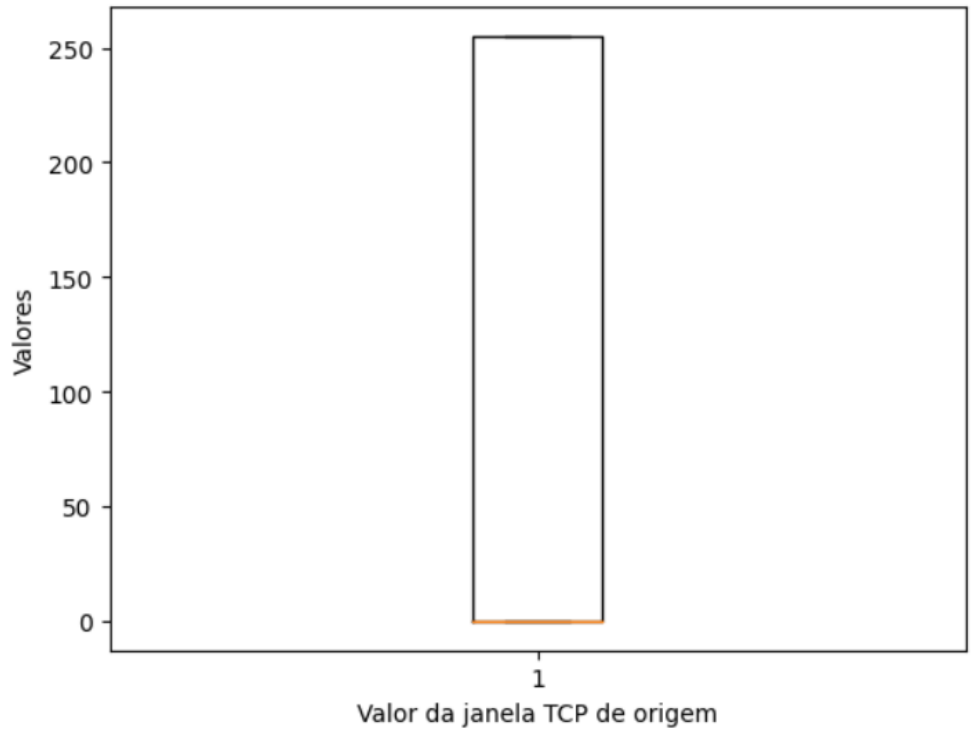
A correlação entre essa coluna 'dest (3) orig (2)' e a target pode ser grande pois, em redes de computadores, é comum que determinadas portas de destino e portas de origem sejam usadas para comunicações específicas e legítimas. Por exemplo, a porta 80 é frequentemente utilizada para comunicação **HTTP**. Se a coluna 'dest (3) orig (2)' revelar uma alta contagem para um par específico de porta de destino e porta de origem que geralmente está associado a comunicações legítimas, isso pode indicar uma conexão legítima. Por outro lado, certas atividades maliciosas ou suspeitas podem ocorrer em pares específicos

de porta de destino e porta de origem, podendo ser identificadas através de uma contagem anormalmente alta. Por exemplo, um alto número de conexões entre uma porta de origem obscura e uma porta de destino comumente usada por malwares pode indicar tráfego suspeito.

1. O valor **mínimo** é 1 e o valor **máximo** é 46, indicando que existentes **47 pares de portas diferentes**;
2. O primeiro quartil (**Q1**) é 1, a mediana (**Q2**) também é 1 e o terceiro quartil (**Q3**) é 3;
3. O boxplot mostra que a mediana (**Q2**) está localizada em **1**, indicando que a **maioria** dos valores se **concentra** nesse ponto;
4. O boxplot mostra que a caixa (box) é relativamente **pequena** em relação à faixa total dos dados, sugerindo que a maioria das observações têm **valores baixos (1, 2 ou 3)**. Isso pode indicar que existem pares de porta de destino e porta de origem **comuns** que são **amplamente utilizados** nas conexões de rede;
5. Há uma quantidade significativa de **outliers**, indicando a presença de pares de porta de destino e porta de origem **incomuns** ou **pouco frequentes** nas conexões de rede. Essas conexões podem ser resultado de atividades **maliciosas**, **tentativas de violação de segurança** ou **comportamentos não usuais** na rede.

- **janela TCP orig:**

Figura 7 - Gráfico boxplot da coluna '*janela TCP orig*'.



Fonte: Compilação dos Autores, 2023.

Essa coluna **(Source TCP Window Size)** representa o tamanho da janela TCP na fonte, ou seja, o tamanho máximo de dados que um dispositivo de origem pode enviar em uma conexão TCP antes de aguardar uma confirmação do dispositivo de destino, sendo um indicador também do protocolo e do desempenho. Essa medida pode ser útil na detecção de atividades maliciosas. Por exemplo, tamanhos de janela TCP extremamente grandes podem ser usados em ataques de negação de serviço **(DoS)**, onde um invasor tenta sobrecarregar um sistema enviando um grande volume de dados. Além disso, variações incomuns ou inconsistentes no tamanho da janela TCP podem ser um sinal de comportamento anômalo.

1. O valor **mínimo** é 0 e o valor **máximo** é 255;
2. O primeiro quartil (**Q1**) é 0, a mediana (**Q2**) também é 0 e o terceiro quartil (**Q3**) é 255;
3. O boxplot mostra que a mediana (**Q2**) está localizada em **0**, indicando que a **maioria** dos valores se **concentra** nesse ponto;
4. A caixa (box) do boxplot abrange **todo o intervalo de valores possíveis**, desde 0 até 255. Isso indica uma **distribuição ampla** dos valores;
5. **Não há outliers** além das hastes do boxplot.

5. Criação de Modelos de Machine Learning

5.1. Dividindo dataframe entre X e y

Nessa etapa, separamos o dataframe original em duas partes. A variável **X** contém **todas as colunas** do dataframe, exceto a coluna de rótulo. A variável **y** contém apenas a coluna **'rótulo'**. Essa separação é feita para podermos treinar os modelos utilizando as **features (X)** e realizar as previsões utilizando a **target (y)**;

5.2. Dividindo dados para treino e teste

Aqui, **dividimos** os dados em conjuntos de **treinamento** e **teste**. O conjunto de treinamento é usado para **treinar** os modelos, enquanto o conjunto de teste é usado para **avaliar o desempenho** dos modelos. O parâmetro ***'test_size=0.3'*** indica que queremos reservar **30% dos dados para teste**, enquanto os **70%** restantes serão utilizados para **treinamento**. O resultado foi que **180.371** dados ficaram para **treino** e **77.302** para **teste**;

5.3. Padronização dos dados usando o StandardScaler

Padronização dos dados usando o StandardScaler: Nessa etapa, aplicamos a **padronização** aos dados de **treinamento** e **teste** utilizando o objeto **StandardScaler**. A padronização é uma técnica que **transforma** os dados de forma que eles tenham **média zero** e **desvio padrão igual a um**. Isso é feito para garantir que **todas** as **features** tenham a **mesma escala** e **evitar** que uma feature com valores **muito grandes influencie** excessivamente o modelo;

5.4. Escolha dos modelos

- **Árvore de Decisão:** modelo que utiliza uma estrutura em forma de **árvore** para tomar decisões com base nas features dos dados. Ela é **intuitiva**, **fácil de interpretar** e pode **capturar relações não lineares nos dados**. Nesse contexto, a Árvore de Decisão pode ser **útil** para **identificar padrões e regras** de classificação com base nas diferentes **features** presentes nos dados de **conexão de rede**;

- **Random Forest:** é um **conjunto** de **Árvores de Decisão** que trabalham de forma **conjunta** para realizar a classificação. A Random Forest é conhecida por **lidar bem** com **dados complexos** e possui uma **boa capacidade de generalização**. No contexto de dados de **conexão de rede**, a Random Forest pode capturar a **interação** entre diferentes **features** e fornecer um **desempenho** de classificação **mais preciso**;

- **Gaussian Naive Bayes:** modelo **probabilístico** baseado no **'teorema de Bayes'.** Ele assume que as **features** são **independentes** entre si e segue uma **distribuição gaussiana**. Apesar dessa simplificação, o Naive Bayes é **rápido**, **fácil de implementar** e **eficaz** em muitos cenários. Nesse contexto, o Gaussian Naive Bayes pode ser utilizado para **calcular**

a probabilidade de uma **conexão de rede** ser **legítima** ou **maliciosa** com base nas **features** observadas.

Segue abaixo uma imagem dos códigos relativos às etapas descritas até o presente momento:

Figura 8 - Códigos relativos às etapas 1, 2, 3 e 4.

```
[ ]  # Dividindo dataframe entre X e y
     X = df.drop(columns=['rótulo'])
     feature_list = list(X.columns)
     X = np.array(X)
     y = df['rótulo']
```

```
[ ]  # Dividindo dados para treino e teste
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

```
[ ]  print("Dados para treino:", len(X_train))
     print("Dados para teste:", len(X_test))
```

```
Dados para treino: 180371
Dados para teste: 77302
```

```
[ ]  # Padronização dos dados usando o StandardScaler
     scaler = StandardScaler().fit(X_train)
     X_train = scaler.transform(X_train)
     X_test = scaler.transform(X_test)
```

```
[ ]  # Definindo os modelos de classificação que serão utilizados e os armazenando em um dicionário
     models = {}
     models['Árvore de Decisão'] = DecisionTreeClassifier()
     models['Random Forest'] = RandomForestClassifier()
     models['Gaussian Naive Bayes'] = GaussianNB()
```

Fonte: Compilação dos Autores, 2023.

5.5. Realizando o treinamento e avaliação de diferentes modelos de classificação

Nessa etapa, percorremos cada modelo definido anteriormente e realizamos o **treinamento** e **avaliação** de cada um deles. Para cada modelo, registramos o **tempo de treinamento**, realizamos as **previsões nos dados de teste** e calculamos diversas **métricas de avaliação**, como **pontuação de treinamento**, **acurácia**, **precisão**, **recall** e **F1-Score**:

Figura 9 - Código relativo à etapa 5.

```
[ ]  # Realizando o treinamento e avaliação de diferentes modelos de classificação
     train_score, accuracy, precision, recall, f1, training_time, y_pred = {}, {}, {}, {}, {}, {}, {}
     for key in models.keys():
         start_time = time.time() # Registrando o tempo de início do treinamento do modelo
         models[key].fit(X_train, y_train) # Realizando o treinamento do modelo específico associado à
         training_time[key] = time.time() - start_time

         y_pred[key] = models[key].predict(X_test) # Realizando a previsão do modelo específico nos dad

         train_score[key] = models[key].score(X_train, y_train)
         accuracy[key] = models[key].score(X_test, y_test)
         precision[key] = precision_score(y_test, y_pred[key])
         recall[key] = recall_score(y_test, y_pred[key])
         f1[key] = f1_score(y_test, y_pred[key])
```

Fonte: Compilação dos Autores, 2023.

6. Interpretação dos Resultados

Para facilitar a **compreensão** dos resultados para cada modelo, criamos um **dataframe** chamado **df_models**. Cada **linha** do dataframe representa um **modelo**, e as **colunas** incluem as **métricas** de **desempenho** já citadas anteriormente. Os **resultados obtidos** foram os seguintes:

Figura 10 - Dataframe de resultados obtidos.

	Pontuação de Treinamento	Acurácia	Precisão	Recall	F1-Score	Tempo de Treinamento
Árvore de Decisão	0.997882	0.935254	0.949299	0.949472	0.949386	4.377215
Random Forest	0.997882	0.950480	0.963215	0.959201	0.961204	56.668614
Gaussian Naive Bayes	0.833355	0.835593	0.841113	0.915955	0.876940	0.167843

Fonte: Compilação dos Autores, 2023.

Os resultados obtidos fornecem **insights** importantes para a **previsão de ataques ou tráfego normal de rede**. A seguir analisaremos cada uma das **métricas** e o que elas indicam:

- **Acurácia:** A acurácia representa a **porcentagem** de **previsões corretas** do **modelo** em relação ao **total de amostras de teste**. No caso dos resultados apresentados, a acurácia varia de **83%** a **95%**. Isso significa que os modelos estão acertando a classificação da **maioria** das amostras de teste. Essa métrica é útil para avaliar a **eficiência geral** do modelo;

- **Precisão:** A precisão mede a **proporção** de **verdadeiros positivos** (amostras classificadas como positivas corretamente) em relação ao **total de amostras classificadas como positivas.** Os valores de precisão para os modelos variam de **84%** a **96%**. Isso indica que os modelos têm uma **alta taxa de precisão** na identificação de conexões de rede maliciosas ou normais, **minimizando falsos positivos**;

- **Recall:** O recall, também conhecido como **taxa de sensibilidade** ou **taxa de verdadeiros positivos**, mede a proporção de **amostras positivas corretamente classificadas** em relação ao **total de amostras positivas**. Os valores de recall para os modelos variam de **91%** a **95%**. Isso indica que os modelos têm uma **boa capacidade de identificar corretamente** as amostras de tráfego malicioso ou normal, **minimizando falsos negativos;**

- **F1-Score:** O F1-Score é uma métrica que **combina** a **precisão** e o **recall** em uma **única medida**. Ele fornece uma **visão geral** do desempenho do modelo considerando tanto os **falsos positivos** quanto os **falsos negativos**. Os valores de F1-Score para os modelos variam de **87%** a **96%**. Quanto mais próximo de **1**, **melhor** é o desempenho do modelo.

Além disso, é importante considerar o **tempo de treinamento** dos modelos. O tempo de treinamento pode ser **relevante** em cenários onde a **eficiência computacional** é um **fator crítico**. Com base nesses resultados, podemos concluir que o modelo **Random Forest** apresenta o **melhor desempenho** geral, com **alta acurácia**, **precisão** e **F1-Score**. No entanto, é importante considerar o **tempo de treinamento**, pois o **Random Forest** leva consideravelmente **mais tempo** para treinar. Se a **velocidade de treinamento** for um fator **crítico**, o modelo **Gaussian Naive Bayes** pode ser uma opção mais **rápida**, apesar de ter um **desempenho** um pouco **inferior**. A **Árvore de Decisão** também oferece um **desempenho sólido**, ficando **entre os outros dois modelos** em termos de **métricas de desempenho** e **tempo de treinamento**.

Esses resultados fornecem **informações valiosas** para a **identificação de ataques** ou **tráfego normal** de rede. Os modelos apresentaram uma **boa capacidade de previsão, com altas taxas de acurácia, precisão e recall.** Isso indica que eles têm potencial para **identificar corretamente** conexões de rede **maliciosas** ou **normais**. Ao utilizar esses modelos em um **sistema de detecção de intrusões**, por exemplo, pode-se tomar **medidas apropriadas** com base nas **previsões** para **mitigar riscos de segurança** e **proteger a rede** contra possíveis ataques.

7. Apresentação dos Resultados

Figura 11 - Quadro contendo o resumo do fluxo de trabalho.

<table>
<tr><td colspan="3">Titulo: Análise de Atividades e Detecção de Rede utilizando o Conjunto de Dados UNSW-NB15</td></tr>
<tr>
<td>1 Declaração do Problema
O objetivo deste projeto é realizar uma análise detalhada do conjunto de dados UNSW-NB15, a fim de identificar possíveis atividades maliciosas e ataques de rede. Por meio da exploração dos pacotes de rede brutos capturados pela ferramenta IXIA PerfectStorm, buscamos extrair insights valiosos sobre os padrões e comportamentos que podem indicar ameaças, fortalecendo assim as medidas de segurança e contribuindo para a prevenção e resposta eficaz a esses ataques.</td>
<td>2 Resultados/Previsões
Utilizando os modelos Árvore de Decisão, Random Forest e Gaussian Naive Bayes, os resultados foram os seguintes: acurácia entre 83% e 95%, precisão entre 84% e 96%, recall entre 91% e 95%, F1-Score entre 87% e 96%. Ao utilizar esses modelos em um sistema de detecção de intrusões, por exemplo, pode-se tomar medidas apropriadas com base nas previsões para mitigar riscos de segurança e proteger a rede contra possíveis ataques.</td>
<td>3 Aquisição de Dados
Os dados do database UNSW-NB15 foram obtidos no repositório do Kaggle, (https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15?select=NUSW-NB15_features.csv). Para garantir o equilíbrio entre os conjuntos de treinamento e teste e evitar o processamento duplo, decidimos concatená-los, gerando assim um novo dataframe. Após isso, dividiu-os com uma proporção diferente utilizando sklearn.model_selection.train_test_split().</td>
</tr>
<tr>
<td>4 Modelagem
• Árvore de Decisão
• Random Forest
• Gaussian Naive Bayes</td>
<td>5 Avaliação do Modelo
O modelo Random Forest apresenta o melhor desempenho geral. No entanto, é importante considerar o tempo de treinamento, pois o Random Forest leva consideravelmente mais tempo para treinar. Se a velocidade de treinamento for um fator crítico, o modelo Gaussian Naive Bayes pode ser uma opção mais rápida, apesar de ter um desempenho um pouco inferior. A Árvore de Decisão também oferece um desempenho sólido, ficando entre os outros dois modelos</td>
<td>6 Preparação de Dados
• Inversão e concatenação dos conjuntos de treinamento e teste
• Redução de dimensionalidade
• Uso do train_test_split</td>
</tr>
</table>

Fonte: Compilação dos Autores, 2023.

8. Links

- Link para repositório com os códigos desenvolvidos para análise e exploração dos dados.
- Link para o repositório no Kaggle do database UNSW-NB15, onde foram obtidos os dados.