

# Pipeline de Dados UNSW-NB15

**Yago Aquino**

**Link do projeto:** [https://github.com/Yago-Pacheco/ufmg/tree/main/especializacao\\_ciencia\\_de\\_dados/topicos\\_em\\_ciencia\\_de\\_dados/tp\\_2](https://github.com/Yago-Pacheco/ufmg/tree/main/especializacao_ciencia_de_dados/topicos_em_ciencia_de_dados/tp_2)

# Introdução

Para este estudo, utilizamos o dataset UNSW-NB15, uma base de dados de referência na comunidade de cibersegurança. Ele foi gerado em um ambiente de laboratório que simula uma rede real, contendo tanto tráfego benigno quanto uma variedade de nove tipos de ataques modernos. O conjunto de dados é dividido em partições de treino e teste, o que nos permitiu desenvolver e validar nosso modelo de forma eficaz.

# Objetivos

1. Coleta de Dados
2. Análise Exploratória
3. Engenharia de Atributos
4. Construção do Modelo
5. Avaliação dos Resultados
6. Visualização dos Resultados

# 1. Coleta dos Dados

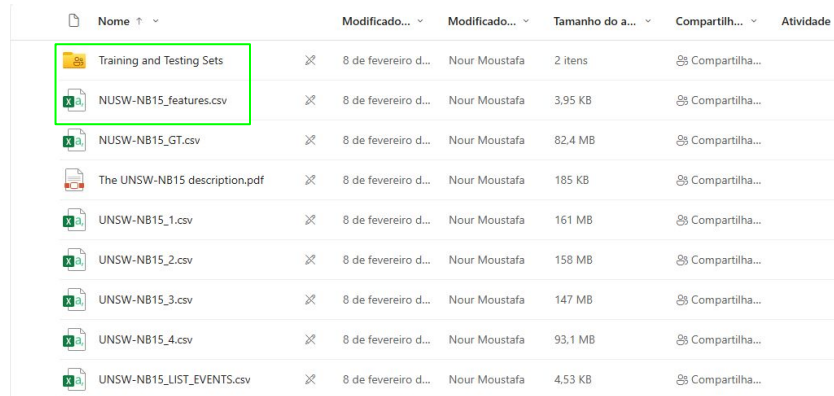
O conjunto de dados utilizado neste trabalho pode ser acessado através do seguinte link: <https://research.unsw.edu.au/projects/unsw-nb15-dataset>

As imagens a seguir ilustram como foi realizada a coleta dos dados no site. Ao todo, foram selecionados três arquivos: os conjuntos de treino e teste, localizados na pasta 'Training and Testing Sets', e o arquivo 'UNSW-NB15.csv'.

The UNSW-NB15 source files (pcap files, BRO files, Argus Files, CSV files and the reports) can be downloaded from [HERE](#). You can also use our new datasets created the [TON\\_IoT](#).



Nome	Modificado...	Modificado...	Tamanho do a...	Compartilh...	Atividade
Argus Files	8 de fevereiro d...	Nour Moustafa	2 itens	Compartilha...	
BRO Files	8 de fevereiro d...	Nour Moustafa	2 itens	Compartilha...	
CSV Files	8 de fevereiro d...	Nour Moustafa	9 itens	Compartilha...	
pcap files	8 de fevereiro d...	Nour Moustafa	2 itens	Compartilha...	
Reports	8 de fevereiro d...	Nour Moustafa	2 itens	Compartilha...	
ReadMe.pdf	8 de fevereiro d...	Nour Moustafa	37,9 KB	Compartilha...	



Nome	Modificado...	Modificado...	Tamanho do a...	Compartilh...	Atividade
Training and Testing Sets	8 de fevereiro d...	Nour Moustafa	2 itens	Compartilha...	
NUSW-NB15_features.csv	8 de fevereiro d...	Nour Moustafa	3,95 KB	Compartilha...	
NUSW-NB15_GT.csv	8 de fevereiro d...	Nour Moustafa	82,4 MB	Compartilha...	
The UNSW-NB15 description.pdf	8 de fevereiro d...	Nour Moustafa	185 KB	Compartilha...	
UNSW-NB15_1.csv	8 de fevereiro d...	Nour Moustafa	161 MB	Compartilha...	
UNSW-NB15_2.csv	8 de fevereiro d...	Nour Moustafa	158 MB	Compartilha...	
UNSW-NB15_3.csv	8 de fevereiro d...	Nour Moustafa	147 MB	Compartilha...	
UNSW-NB15_4.csv	8 de fevereiro d...	Nour Moustafa	93,1 MB	Compartilha...	
UNSW-NB15_LIST_EVENTS.csv	8 de fevereiro d...	Nour Moustafa	4,53 KB	Compartilha...	

## 2. Análise Exploratória

Os dados foram divididos em dois conjuntos: o de treino, com 175.341 registros, e o de teste, com 82.332. Ambos compartilham a mesma estrutura de 45 colunas, que se distribuem nos seguintes tipos de dados:

- 11 colunas do tipo float64
- 30 colunas do tipo int64
- 4 colunas do tipo string

Devido à sua extensão, a tabela com a descrição detalhada de cada coluna foi omitida desta apresentação. No entanto, essas informações podem ser consultadas no arquivo 'UNSW-NB15.csv', disponível no repositório do projeto, e também ao longo do notebook de Análise Exploratória de Dados (EDA).

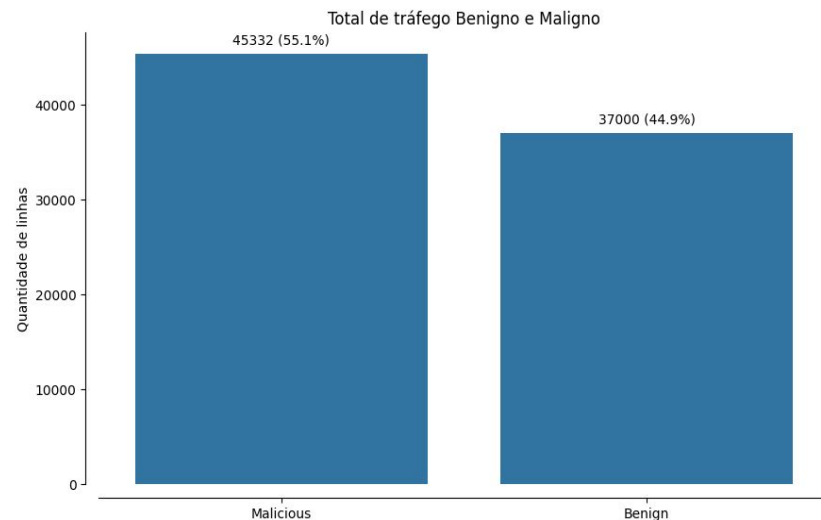
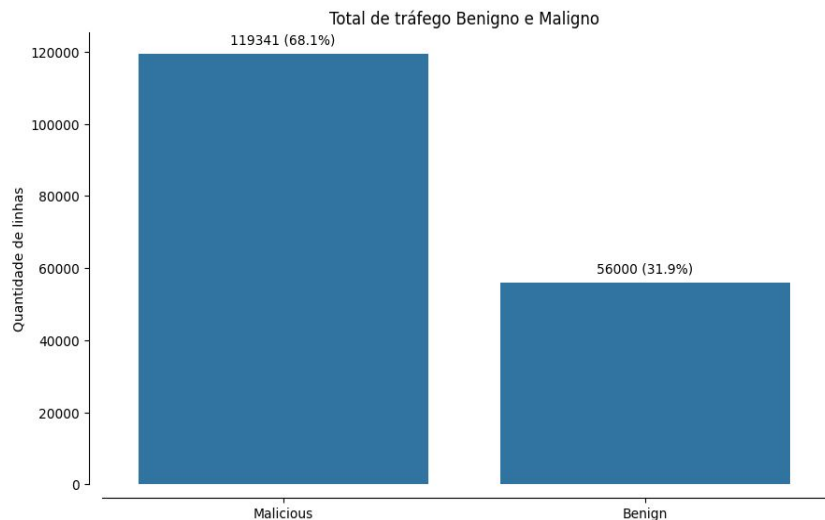
## 2. Análise Exploratória

Antes de iniciar a Análise Exploratória de Dados (EDA), foi realizada uma verificação preliminar da qualidade e integridade dos dados em ambos os conjuntos (treino e teste). Esta etapa, também conhecida como Sanity Check, confirmou que os dados são robustos e adequados para a análise, não apresentando valores nulos ou duplicados.

- Análise estatística descritiva das features numéricas;
- Distribuição das variáveis categóricas;
- Verificação de duplicatas (não se tem duplicatas)
- Distribuição da variável-alvo (label);
- Análise da composição dos diferentes tipos de ataque presentes no dataset.

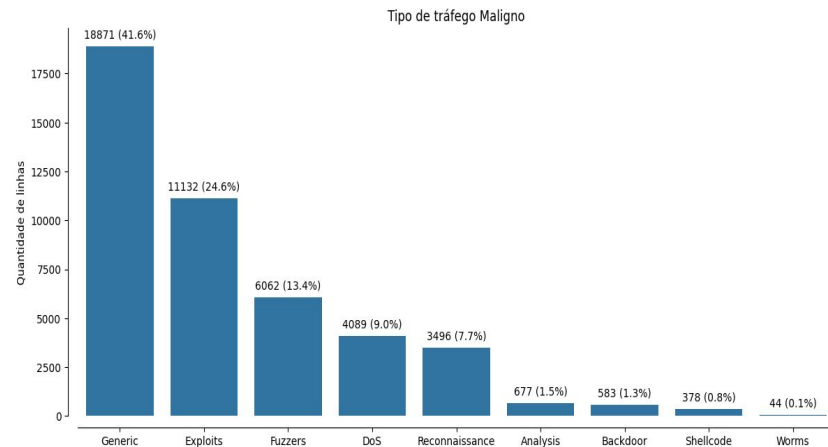
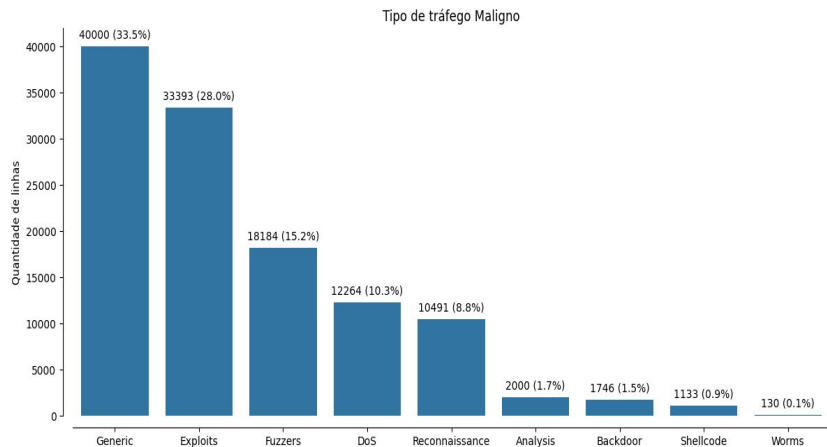
## 2. Análise Exploratória

A distribuição da variável-alvo, apresentada nas figuras a seguir, revela um claro desbalanceamento de classes tanto no conjunto de treino quanto no de teste. Essa característica é ainda mais forte nos dados de treino, nos quais o tráfego malicioso representa a classe majoritária.



## 2. Análise Exploratória

Ao analisar a distribuição dos tipos de ataque, observamos que a maioria do tráfego malicioso se concentra em quatro categorias principais: 'Generic', 'Exploits', 'Fuzzers' e 'DoS'. Conforme ilustrado nas figuras, essa concentração reflete os tipos de ameaças mais comuns em cenários de rede reais.





## 2. Análise Exploratória

A análise individual da tabela de estatísticas descritivas, com mais de 30 variáveis, seria impraticável. Diante disso, adotou-se uma abordagem estratégica, focando a análise em um subconjunto de 9 features-chave. Estas colunas foram selecionadas por representarem diferentes 'pilares' do tráfego de rede, oferecendo uma visão holística e eficiente do comportamento dos dados. As features escolhidas foram:

- **dur** (duração): Em geral conexões curtas podem ser scans e longas malware
- **sbytes e dbytes** (volume de dados): Picos de volume indicam floods, exploits ou data exfiltration.
- **rate** (pacotes por segundo): A métrica principal para detectar ataques DDoS.
- **slood** (bits por segundo): Mede a "agressividade" e intensidade da transmissão.
- **sttl** (tempo de vida do pacote): Valores anômalos podem indicar spoofing ou fingerprinting.
- **smean** (tamanho médio do pacote): Diferencia ataques de pacotes pequenos (floods) de grandes (exploits)
- **ct\_srv\_dst** (conexões recentes ao mesmo serviço): Detecta DoS em serviços e port Scanning.
- **ct\_dst\_src\_ltm** (conexões recentes entre o mesmo par): Detecta força bruta e atividade repetitivas de um atacante.

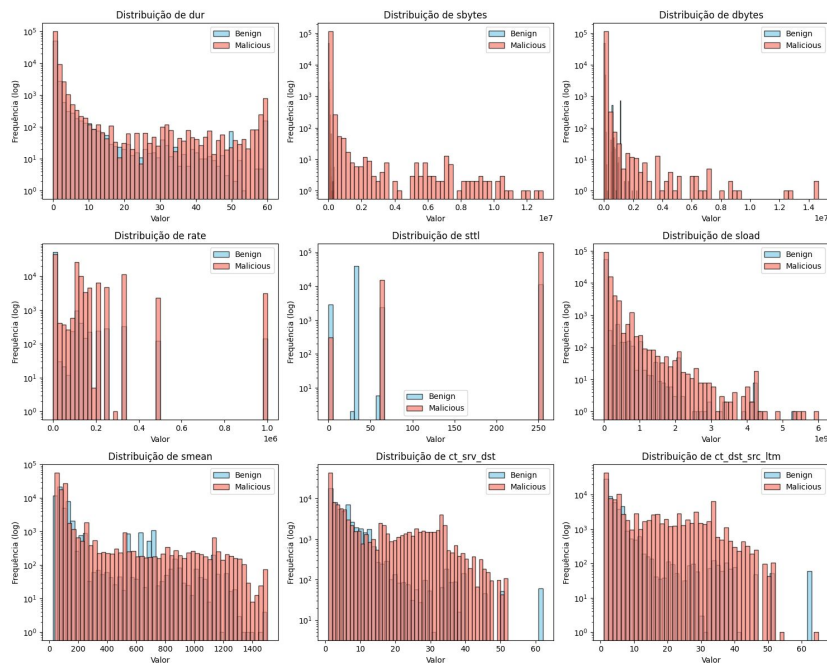
## 2. Análise Exploratória

Com essas 9 colunas se fez a análise da estatística descritiva e visualização da distribuição do tráfego maligno e benigno a fim de verificar diferenças entre eles. Um ponto importante aqui é que as análises a partir daqui se concentraram totalmente no conjunto de treino.

	count	mean	std	min	25%	50%	75%	max
dur	175341.000000	1.359389	6.480249	0.000000	0.000008	0.001582	0.668069	59.999989
sbytes	175341.000000	8844.843836	174765.644309	28.000000	114.000000	430.000000	1418.000000	12965233.000000
dbytes	175341.000000	14928.918564	143654.217718	0.000000	0.000000	164.000000	1102.000000	14655550.000000
rate	175341.000000	95406.187105	165400.978457	0.000000	32.786140	3225.806520	125000.000300	1000000.003000
sttl	175341.000000	179.546997	102.940011	0.000000	62.000000	254.000000	254.000000	255.000000
sload	175341.000000	73454033.194063	188357447.000193	0.000000	13053.338870	879674.750000	88888888.000000	5988000256.000000
smean	175341.000000	136.751769	204.677360	28.000000	57.000000	73.000000	100.000000	1504.000000
ct_srv_dst	175341.000000	9.100758	10.756952	1.000000	2.000000	4.000000	12.000000	62.000000
ct_dst_src_ltm	175341.000000	8.729881	10.956186	1.000000	1.000000	3.000000	12.000000	65.000000

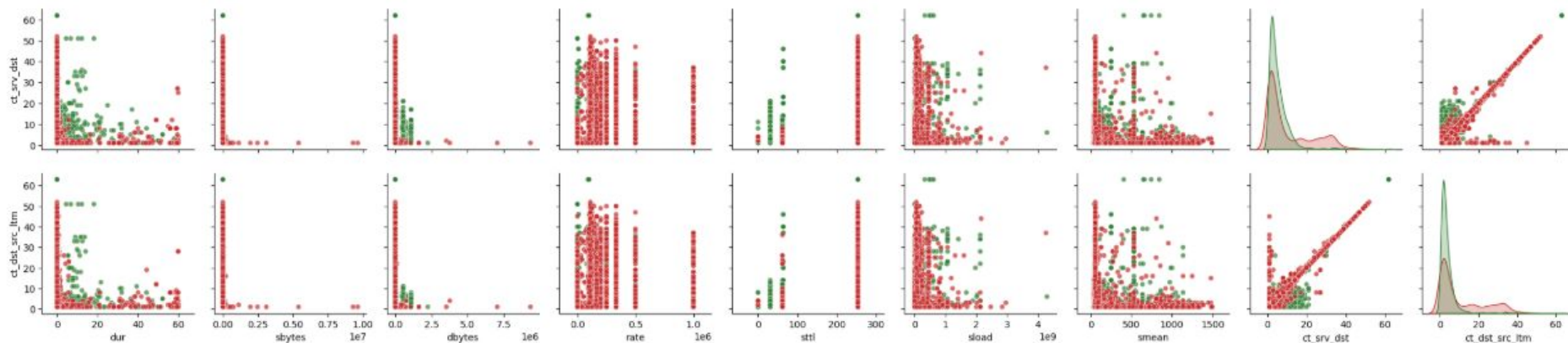
## 2. Análise Exploratória

A figura a seguir compara a distribuição do tráfego benigno e malicioso. A análise visual, no entanto, não revelou diferenças significativas, mostrando que as distribuições para ambas as classes são bastante semelhantes.



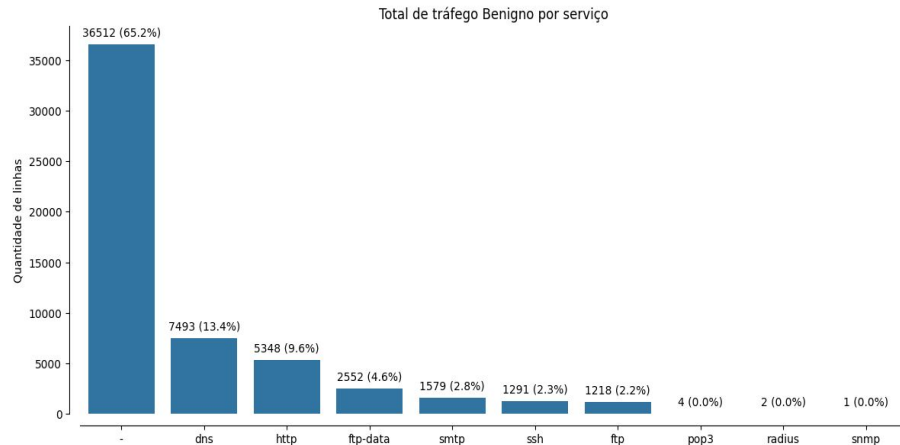
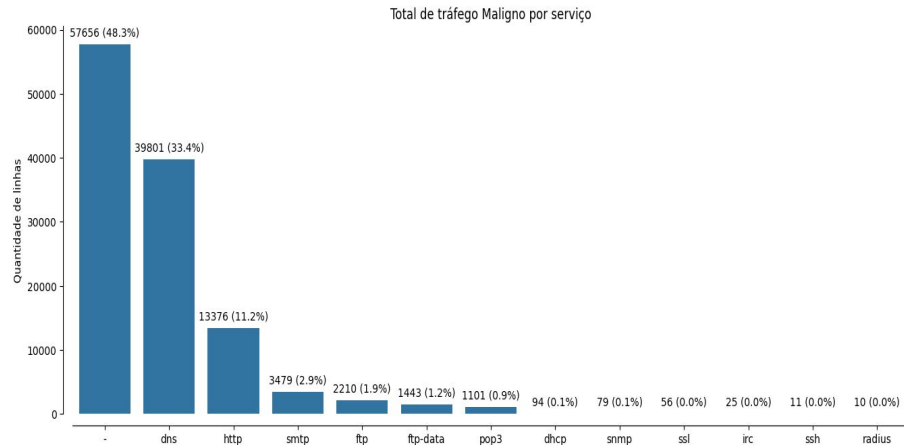
## 2. Análise Exploratória

Para investigar a correlação entre as features selecionadas, foi utilizado um pairplot. Devido à alta dimensionalidade e complexidade visual do gráfico completo, optou-se por apresentar apenas o recorte que ilustra a relação de interesse entre as variáveis `ct_srv_dst` e `ct_dst_src_ltm`.



## 2. Análise Exploratória

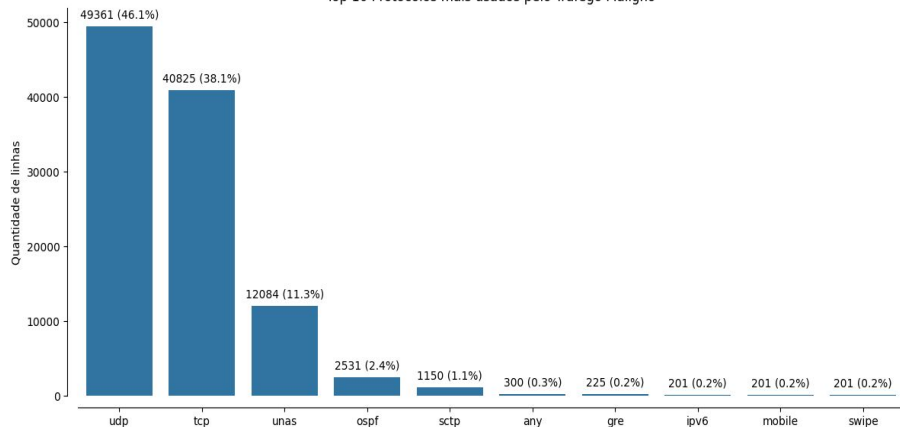
A análise da distribuição de serviços (service) revelou um insight crucial. Em ambos os tráfegos, benigno e malicioso, a categoria mais comum é a '-', que agrupa serviços não-padrão ou menos frequentes. O ponto mais relevante, no entanto, é que as três categorias de serviços mais utilizadas são exatamente as mesmas para ambos os tipos de tráfego. Essa sobreposição sugere uma tática de evasão, onde o tráfego malicioso simula o comportamento do tráfego legítimo para evitar a detecção.



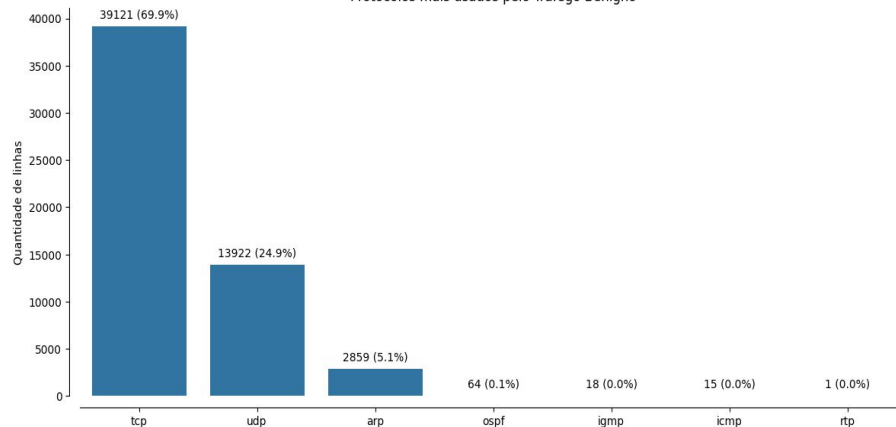
## 2. Análise Exploratória

Diferente dos serviços, a análise dos protocolos (proto) revela distinções marcantes entre o tráfego. Observa-se uma inversão nas duas categorias mais utilizadas entre o tráfego benigno e o malicioso. A diferença mais expressiva, no entanto, está na variedade: enquanto o tráfego legítimo opera com apenas 7 protocolos distintos, o tráfego de ataque utiliza um leque de 129 protocolos. Essa diversidade pode indicar tentativas de exploração de vulnerabilidades em diferentes camadas da rede. Nota-se também a presença do protocolo 'unas', que sinaliza tráfego ao qual nenhum protocolo padrão foi atribuído.

Top 10 Protocolos mais usados pelo Tráfego Maligno

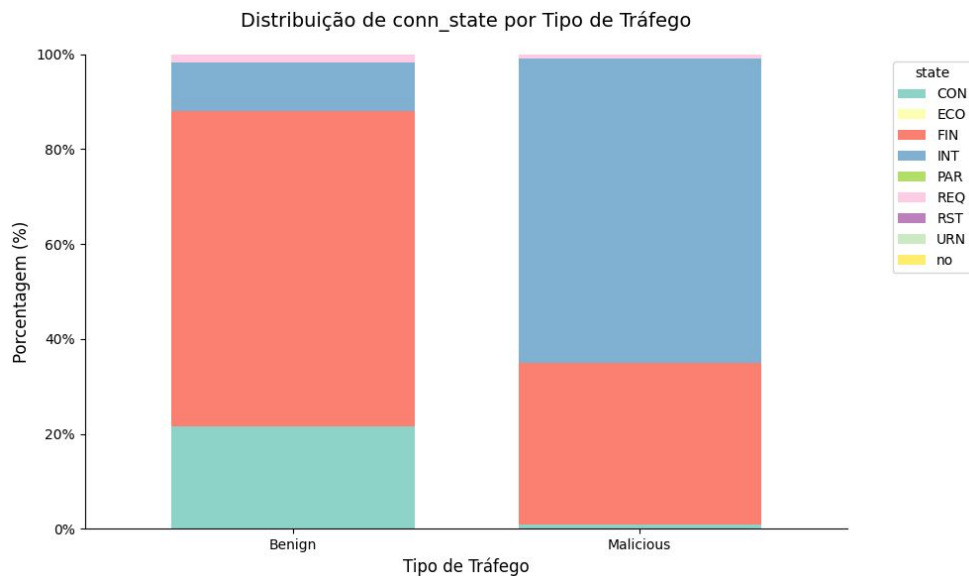


Protocolos mais usados pelo Tráfego Benigno



## 2. Análise Exploratória

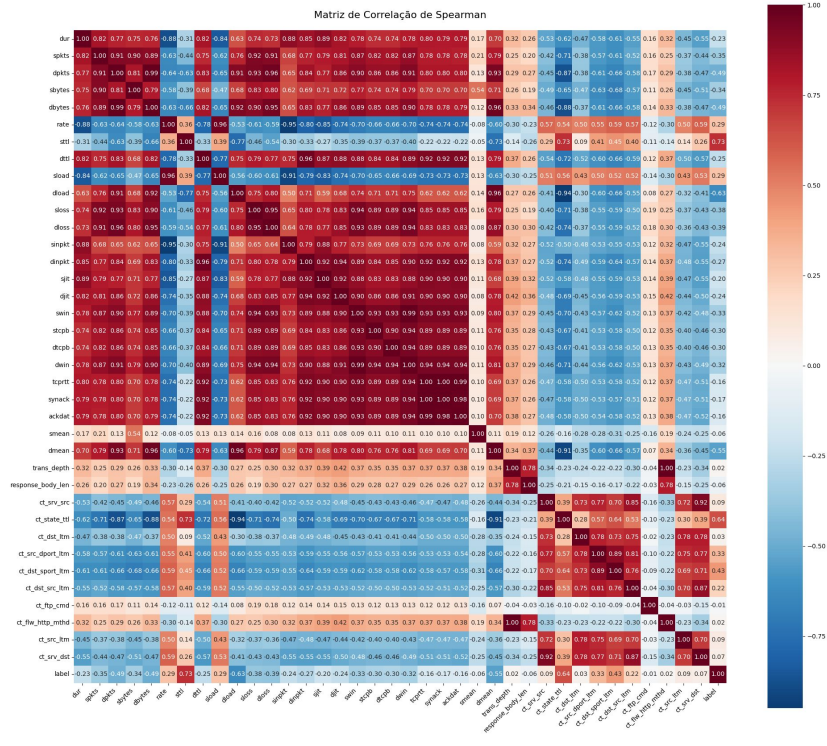
Uma análise fundamental foi a do estado da conexão (conn\_state), que revelou um comportamento distintivo entre as classes. Conforme detalhado nos gráficos e tabelas a seguir, estados como 'FIN' (conexão finalizada) são predominantes no tráfego benigno. Em contrapartida, o estado 'INT' (conexão interrompida) é significativamente mais frequente no tráfego malicioso. Este padrão é um forte indicador de atividades anômalas, como varreduras de rede ou tentativas de ataque que não completam um handshake de conexão normal.



state	CON	ECO	FIN	INT	PAR	REQ	RST	URN	no
class_name									
Benign	21.605357	0.021429	66.383929	10.205357	0.001786	1.651786	0.126786	0.001786	0.001786
Malicious	0.882346	0.000000	34.062057	64.152303	0.000000	0.893239	0.010055	0.000000	0.000000

## 2. Análise Exploratória

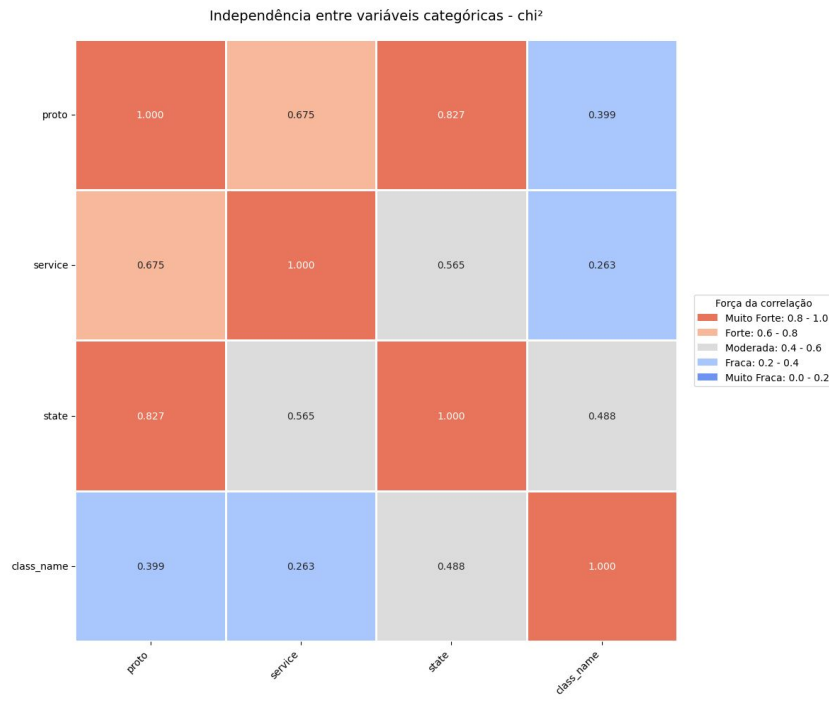
Esta análise de correlação expõe a redundância entre features e o conhecido data leakage da variável ttl, justificando a seleção e remoção de colunas para a modelagem.





## 2. Análise Exploratória

Para finalizar a Análise Exploratória, foi realizado um teste de independência para avaliar a relação entre as features categóricas e a variável-alvo. Os resultados confirmaram que o estado da conexão (conn\_state) é a variável categórica com o maior poder preditivo. Este achado é consistente com as análises anteriores e solidifica a importância desta feature para a distinção entre o tráfego.



### 3. Engenharia de Atributos

A etapa de Engenharia de Features focou exclusivamente na redução de dimensionalidade, sem a criação de novas variáveis. A seleção foi guiada por dois critérios principais:

- Multicolinearidade: Com base na Análise Exploratória, foram removidas 21 features que possuíam correlação maior ou igual a 0.91 com outra variável.
- Data Leakage e Viabilidade: A decisão de remover features que causam vazamento de dados (data leakage) ou que não podem ser obtidas em tempo real foi influenciada pela leitura do artigo '*UNSW-NB15: a comprehensive data set for network intrusion detection systems*', que discute essas questões, e também visando um cenário de produção.

Ao todo, 34 features foram removidas, resultando no dataset final ilustrado na figura. É crucial ressaltar que todo o processo foi replicado de forma idêntica nos conjuntos de treino e teste. Por fim, a normalização dos dados não foi necessária, uma vez que o modelo LightGBM (LGBM) não é sensível a diferentes escalas de variáveis.

	id	dur	proto	service	state	sbytes	sload	smean	is_ftp_login	is_sm_ips_ports	label
0	1	0.121478	tcp	-	FIN	258	14158.942380	43	0	0	0
1	2	0.649902	tcp	-	FIN	734	8395.112305	52	0	0	0
2	3	1.623129	tcp	-	FIN	364	1572.271851	46	0	0	0
3	4	1.681642	tcp	ftp	FIN	628	2740.178955	52	1	0	0
4	5	0.449454	tcp	-	FIN	534	8561.499023	53	0	0	0

### 3. Engenharia de Atributos

A etapa final do pré-processamento consistiu na codificação das variáveis categóricas remanescentes: proto, service e state. Para evitar a alta dimensionalidade que um One-Hot Encoding direto causaria, a variável proto, por exemplo, geraria mais de 120 colunas, foi aplicada uma estratégia de agrupamento. As categorias menos frequentes de cada feature foram consolidadas e, em seguida, o One-Hot Encoding foi aplicado sobre essa versão simplificada. Este processo otimizado resultou em um dataset final com 25 colunas. Por último, os dados foram divididos entre features (atributos) e a variável-alvo (label) para a etapa de modelagem.

```
1 # Reduzindo as categorias das colunas
2 protocolos_principais = ['tcp', 'udp', 'unas', 'arp', 'ospf']
3 servicos_principais = ['- ', 'dns', 'http', 'ftp-data', 'smtp', 'ftp', 'ssh']
4 state_principais = ['FIN', 'INT', 'CON', 'REQ', 'RST', ]
5
6 df_modi['proto_agg'] = df_modi['proto'].where(df_modi['proto'].isin(protocolos_principais), 'outro_proto')
7 df_modi['service_agg'] = df_modi['service'].where(df_modi['service'].isin(servicos_principais), 'outro_service')
8 df_modi['state_agg'] = df_modi['state'].where(df_modi['state'].isin(state_principais), 'outro_state')
```

```
1 X = df_final.drop(['label', 'id'], axis=1)
2 y = df_final['label']
3
4 X.shape, y.shape
((175341, 23), (175341,))
```

```
1 colunas_one_hot = ['proto_agg', 'service_agg', 'state_agg']
2
3 encoder = OneHotEncoder(
4     sparse_output=False,
5     drop='first',
6     handle_unknown='ignore'
7 )
8
9 X_categorical_encoded = encoder.fit_transform(df_modi[colunas_one_hot])
10
11 # Manter os nomes
12 feature_names = encoder.get_feature_names_out(colunas_one_hot)
13 print("\nFeatures criadas:", feature_names)
14
15 # Convertendo as mudanças para DF
16 df_encoded = pd.DataFrame(
17     X_categorical_encoded.astype('int8'),
18     columns=feature_names,
19     index=df_modi.index
20 )
21
22 # Fazendo a concatenação do meu DF
23 numeric_cols = df_modi.select_dtypes(include=[np.number]).columns
24 df_final = pd.concat([df_modi[numeric_cols], df_encoded], axis=1)
25
26 print("\n", df_final.shape)
27 df_final.head()
```

## 4. Modelagem

Com os dados pré-processados, iniciou-se a etapa de modelagem utilizando a divisão de treino e teste já fornecida pelo dataset. O processo seguiu uma abordagem robusta: primeiramente, o modelo foi instanciado e parametrizado; em seguida, foi treinado e avaliado no conjunto de treino utilizando validação cruzada para garantir a consistência dos resultados. Apenas após essa validação interna, o modelo foi aplicado ao conjunto de teste para a avaliação final de performance.

A escolha do modelo recaiu sobre o LightGBM (LGBM), um framework de gradient boosting que se destaca pela alta velocidade, baixo consumo de memória e precisão superior. Para este projeto, suas vantagens no tratamento de dados desbalanceados foram decisivas, como os parâmetros nativos de balanceamento (`is_unbalance=True`), a otimização para métricas robustas e o crescimento leaf-wise, que aprimora a identificação de padrões. Essas características o tornam a escolha ideal para problemas de classificação com classes desproporcionais, aliando eficiência computacional e alto poder preditivo.

## 4. Modelagem

As Figuras abaixo apresentam o processo da modelagem.

```
1 model_lgb = lgb.LGBMClassifier(
2     objective='binary',
3     is_unbalance=True,
4     n_estimators=100,
5     learning_rate=0.1,
6     max_depth=6,
7     num_leaves=31,
8     min_child_samples=20,
9     random_state=42,
10    verbose=-1,
11    n_jobs=-1
12 )
```

```
1 cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
2
3 scoring = {
4     'recall': 'recall',
5     'precision': 'precision',
6     'f1': 'f1',
7     'roc_auc': 'roc_auc',
8     'accuracy': 'accuracy'
9 }
10
11 scores = cross_validate(
12     model_lgb,
13     X, y,
14     cv=cv,
15     scoring=scoring,
16     return_train_score=False,
17     n_jobs=-1
18 )
```

```
1 # Visualização simples dos resultados de treino
2 print("RESULTADOS DA VALIDAÇÃO CRUZADA:")
3 print("=" * 80)
4
5 for metric, values in scores.items():
6     if metric.startswith('test_'):
7         metric_name = metric[5:].upper() # Remove 'test_' do nome
8         mean_score = values.mean()
9         std_score = values.std()
10
11     print(f"{metric_name:10}: {mean_score:.4f} (+/- {std_score * 2:.4f})")
12
13 print("\n" + "=" * 80)
14
15 recall_scores = scores['test_recall']
16 print(f"RECALL por fold: {recall_scores}")
17 print(f"RECALL médio: {recall_scores.mean():.4f}")
18
19 print("\nInformações do modelo:")
20 print(f"- Número de features: {X.shape[1]}")
21 print(f"- Número de amostras: {X.shape[0]}")
22 print(f"- Balanceamento das classes: {y.value_counts().to_dict()}")
```

```
1 model_lgb.fit(X, y)
2 prev = model_lgb.predict(X_test)
3
4 y_proba_test = model_lgb.predict_proba(X_test)[: , 1]
5
6 print(classification_report(y_test, prev))
```

## 5. Avaliação de Resultados

A etapa de validação com o conjunto de teste é essencial para garantir que nosso modelo é generalizável. Os resultados mostram uma performance distinta entre as duas etapas:

- O modelo alcançou uma performance quase perfeita, com acurácia de 93% e métricas de precisão e recall acima de 93%.
- Houve uma queda de performance, com a acurácia geral caindo para 87%. A principal diferença foi na capacidade de identificar corretamente o tráfego benigno, cujo recall caiu 10 pontos percentuais.
- A diferença entre os resultados de treino e teste indica que o modelo apresenta um leve overfitting. Embora ainda entregue um resultado sólido no teste, ele não conseguiu generalizar perfeitamente o aprendizado para dados não vistos, um desafio comum e importante a ser destacado.

COMPARAÇÃO: Validação Cruza vs Teste			
MÉTRICA	CV (MÉDIA±2σ)	TESTE	DIFERENÇA
ACCURACY	0.9300±0.0035	0.8735	-0.0565
PRECISION	0.9608±0.0024	0.8561	-0.1047
RECALL	0.9354±0.0054	0.9260	-0.0095
F1	0.9479±0.0027	0.8896	-0.0583
ROC_AUC	0.9863±0.0012	0.9690	-0.0173

## 5. Avaliação de Resultados

Ao analisar a performance do modelo no conjunto de teste, observamos um ponto crucial: o recall da classe 0 (tráfego benigno) foi de 81%, consideravelmente inferior ao recall de 93% da classe 1 (ataque). Isso indica que, embora o modelo seja excelente em identificar ataques, ele classifica incorretamente uma porção maior do tráfego legítimo como malicioso.

Este comportamento é um reflexo direto do desbalanceamento do dataset. Mesmo com o uso do balanceamento de classes nativo do LightGBM, que se mostrou eficaz, a dificuldade em modelar a classe minoritária (benigna) persiste. Um ponto de atenção, e que também foi levantado pelos criadores do dataset, é o potencial de data leakage em certas features, o que pode ter influenciado o aprendizado do modelo e reforçado esse viés.

	precision	recall	f1-score	support
Benigno	0.90	0.81	0.85	37000
Malicioso	0.86	0.93	0.89	45332
accuracy			0.87	82332
macro avg	0.88	0.87	0.87	82332
weighted avg	0.88	0.87	0.87	82332

ANÁLISE POR CLASSE:

=====

Matriz de Confusão:

	Pred	
Real	Benigno	Malicioso
Benigno	29942	7058
Malicioso	3356	41976

Métricas por Classe:

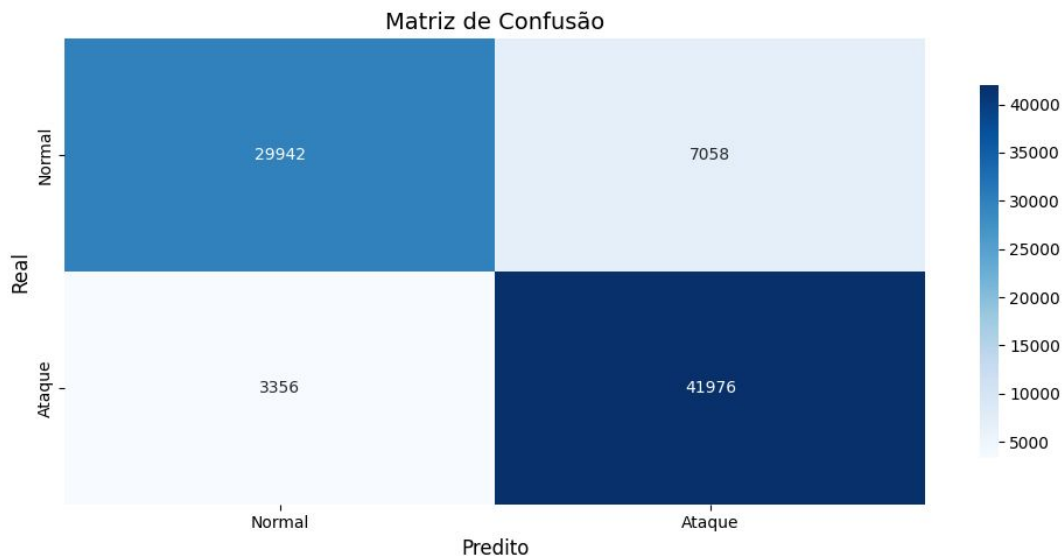
Classe	Precision	Recall	F1-Score	Support
Benigno	0.8992	0.8092	0.8519	37000
Malicioso	0.8561	0.9260	0.8896	45332

MÉTRICAS ESPECÍFICAS:

- True Positives: 41,976
- True Negatives: 29,942
- False Positives: 7,058
- False Negatives: 3,356
- Taxa de Falsos Positivos: 19.08%
- Taxa de Falsos Negativos: 7.40%

## 6. Visualização dos Resultados

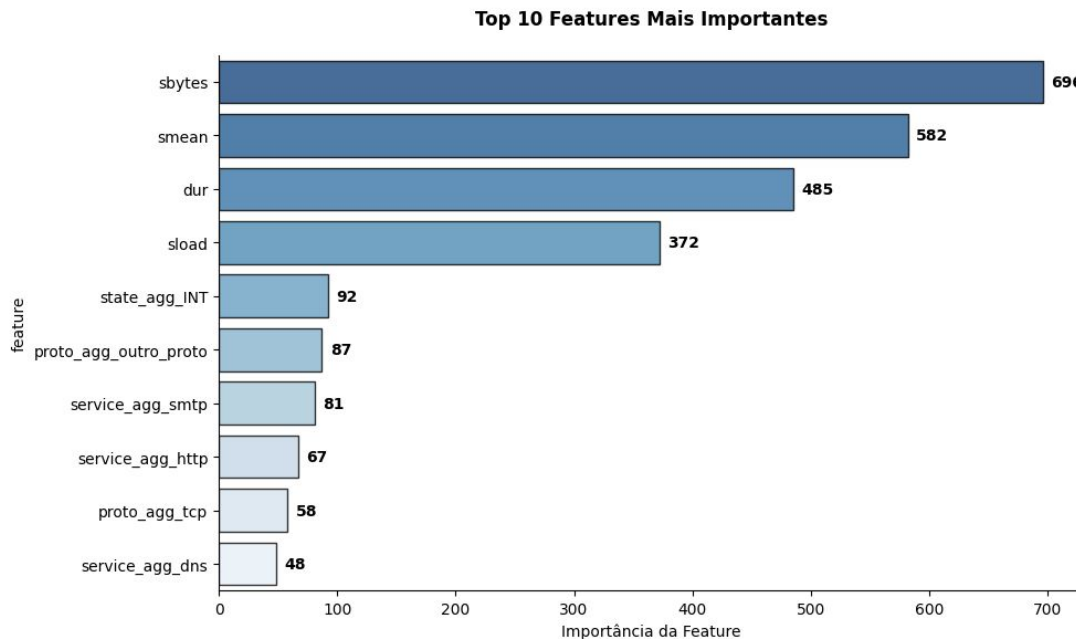
A matriz de confusão oferece uma visualização clara da performance do modelo, especialmente dos seus erros. A análise da matriz do conjunto de teste evidencia a principal dificuldade do nosso modelo: a correta classificação do tráfego benigno. O resultado mostra um índice de Falsos Positivos acima de 19%, o que significa que uma parcela considerável do tráfego legítimo foi incorretamente classificada como um ataque. Este é um ponto de atenção crítico em sistemas de detecção reais.





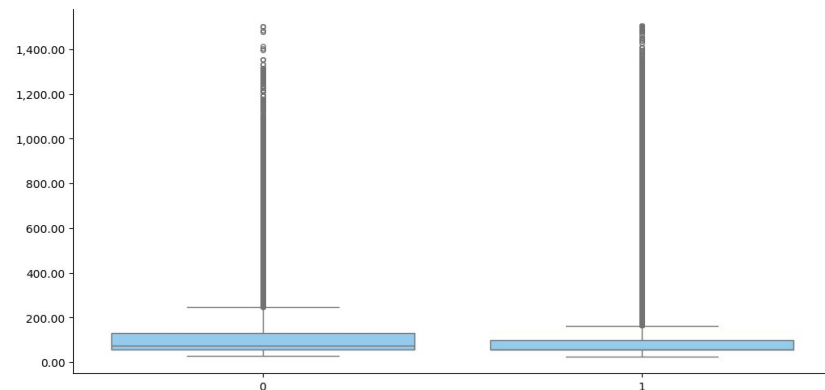
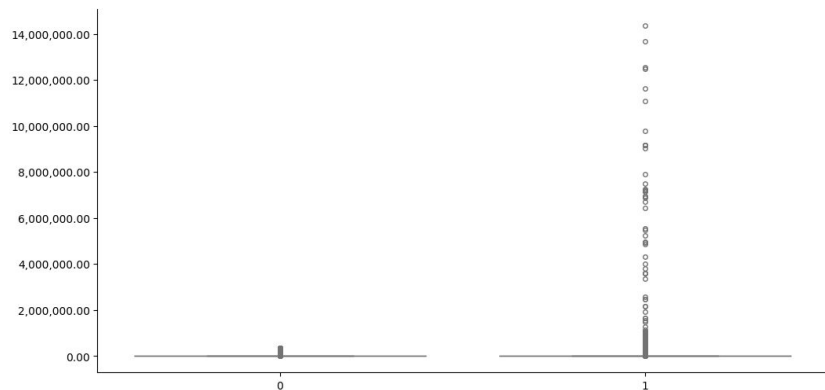
## 6. Visualização dos Resultados

Um dos grandes benefícios do LGBM é a capacidade de ranquear as features mais importantes para a classificação. Esta funcionalidade é crucial para a interpretabilidade do modelo, permitindo-nos compreender e explicar os fatores que mais influenciam suas previsões.



## 6. Visualização dos Resultados

A análise de importância de features revelou que sbytes (total de bytes de origem) e smean (tamanho médio dos pacotes de origem) são os atributos mais preditivos do nosso modelo. Isso é esperado, pois o volume e o tamanho dos pacotes são indicadores-chave do comportamento do tráfego. O boxplot de sbytes mostra uma dinâmica interessante: o tráfego de ataque, apesar de ter uma mediana de bytes menor que o tráfego benigno, exibe uma quantidade massiva de outliers com valores altíssimos. Isso captura a dualidade dos ataques: alguns são discretos, com poucos bytes, para roubo de informação, enquanto outros, como ataques de negação de serviço (DoS), inundam a rede com um volume enorme de dados. Já em smean, a mediana para ataques é visivelmente menor, o que sugere uma tática comum de sobrecarregar o sistema com um grande número de pacotes pequenos e maliciosos. As Figuras abaixo representam o boxplot do sbytes e smean, respectivamente.



## 7. Conclusão

### Resultados Alcançados:

- Implementamos com sucesso um pipeline de ponta a ponta, desde a análise exploratória até a avaliação de um modelo preditivo.
- O modelo final alcançou alta performance na detecção de ataques, validando a eficácia da abordagem de Machine Learning para a segurança de redes.

### A Importância da Detecção Inteligente:

- Projetos como este são fundamentais para criar sistemas de segurança proativos, capazes de analisar o comportamento do tráfego e identificar ameaças em tempo real, superando as limitações de abordagens baseadas apenas em assinaturas.

### Trabalhos Futuros:

- Aprimoramento do Modelo: Explorar técnicas de regularização mais robustas e métodos de reamostragem (como SMOTE) para reduzir o overfitting e melhorar a detecção da classe minoritária.
- Engenharia de Features: Investigar a criação de novas variáveis que possam capturar relações complexas no tráfego sem introduzir vieses.
- Classificação Multicasse: Expandir o modelo para identificar o tipo específico de ataque (DoS, Fuzzers, etc.), fornecendo insights mais acionáveis para a defesa da rede.

# Referências

“LightGBM 3.3.5 Documentation.” [Lightgbm.readthedocs.io, lightgbm.readthedocs.io/en/stable/](https://lightgbm.readthedocs.io/en/stable/). Acessado em: 09-09-2025.

Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." *Military Communications and Information Systems Conference (MilCIS)*, 2015. IEEE, 2015.

Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset." *Information Security Journal: A Global Perspective* (2016): 1-14.

Moustafa, Nour, et al. "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." *IEEE Transactions on Big Data* (2017).

Moustafa, Nour, et al. "Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models." *Data Analytics and Decision Support for Cybersecurity*. Springer, Cham, 2017. 127-156.