Estudo da Influência das Features e do Desempenho dos Classificadores na Detecção de Anomalias em Redes

João Lucas Oliveira Mota[1], **João Lucas Rodrigues da Silva**[2], **José Alberto Rodrigues Neto**[2]

[1] Departamento de Engenharia de Teleinformática – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brasil

[2]Departamento de Computação – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brasil

{jlucasoliveira2002}@gmail.com, {joao.lsilva,albertoo11}@alu.ufc.br

Resumo. *Este trabalho investiga o impacto da seleção e remoção manual de atributos no desempenho de classificadores de Aprendizado de Máquina para detecção de anomalias em redes de computadores. Utilizando o conjunto de dados Network Intrusion Detection, disponibilizado publicamente por Sampada Bhosale [Bhosale 2018], foram avaliados os modelos Random Forest (RF), K-Nearest Neighbors (KNN), Multilayer Perceptron (MLP) e Isolation Forest (IF). Os experimentos simularam cenários de indisponibilidade de dados através da exclusão controlada de features críticas, como o parâmetro src_bytes. Os resultados demonstraram que o Random Forest obteve o melhor desempenho inicial, porém apresentou maior sensibilidade à perda de informações, com queda no desempenho ao remover atributos influentes. Em contraste, o MLP revelou maior robustez e capacidade de generalização, mantendo F1-Scores superiores a 0,99 mesmo com conjuntos reduzidos. O estudo conclui que a escolha do classificador deve considerar não apenas a precisão absoluta, mas a resiliência do modelo em ambientes reais onde a integridade dos dados pode ser comprometida.*

1. Introdução

O crescimento contínuo da demanda por serviços digitais, tanto no cotidiano quanto em ambientes corporativos, tem impulsionado a ampla utilização de redes de computadores, incluindo data centers e servidores privados virtuais (VPS). Como consequência, observou-se um aumento significativo no número e na complexidade das ameaças, de acordo com o Relatório de Índice de Cibersegurança da ENISA (2024)[ENISA 2025], embora haja um esforço para fortalecer a resiliência cibernética, a capacidade do setor privado em detectar e analisar incidentes ainda enfrenta desafios de maturidade técnica. Essa evolução torna a detecção de intrusões um dos principais desafios da área de segurança da informação. Ataques cada vez mais sofisticados exigem mecanismos capazes de identificar comportamentos anômalos de forma precisa e eficiente, mesmo em cenários caracterizados por grandes volumes de dados e alta dimensionalidade.

Nesse contexto, técnicas de aprendizado de máquina têm sido amplamente empregadas em sistemas de detecção de intrusões[Buczak and Guven 2015][Zamani and Movahedi 2013][He et al. 2023], devido à sua capacidade de aprender padrões complexos a partir do tráfego de rede e de se adaptar

a diferentes tipos de ataques. Historicamente, a eficácia desses sistemas depende da qualidade das bases de dados utilizadas para o treinamento. Conforme analisado previamente no artigo "A detailed analysis of the KDD CUP 99 data set" [Tavallaee et al. 2009], conjunto de dados podem apresentar deficiências como a redundância de registros que foi avaliado no conjunto do KDD CUP 99, dataset criado para uma competição da área de mineração de dados, baseando-se em tráfego de rede simulado para distinguir entre conexões normais e invasões. O que pode causar um viés nos classificadores em direção a padrões frequentes, resultando em taxas de acerto inflacionadas que não refletem o desempenho real em redes modernas.

Diversos estudos na literatura investigam estratégias automáticas de redução de dimensionalidade e seleção de atributos, como a Análise de Componentes Principais (PCA), com o objetivo de melhorar a eficiência computacional e a capacidade de generalização dos modelos[Jia et al. 2022][Mladenić 2005][Zaheer et al. 2025]. Apesar disso, ainda há uma lacuna na compreensão do impacto individual das features mais relevantes no desempenho de diferentes classificadores, especialmente quando tais atributos deixam de estar disponíveis. Dessa forma, este trabalho tem como objetivo analisar a influência da seleção e da remoção de atributos no desempenho de diferentes modelos de aprendizado de máquina aplicados à detecção de anomalias em redes de computadores. Avaliando modelos supervisionados e não supervisionados, incluindo Random Forest, K-Nearest Neighbors (KNN), Multilayer Perceptron (MLP) e Isolation Forest. Os experimentos buscam, assim, fornecer uma análise comparativa que contribua para a compreensão da robustez e das limitações de cada abordagem em cenários realistas de detecção de intrusões.

2. Dataset

O conjunto de dados utilizado foi obtido a partir da plataforma Kaggle, no repositório *Network Intrusion Detection*, disponibilizado publicamente por Sampada Bhosale [Bhosale 2018].

O dataset foi construído a partir da simulação de um ambiente de rede militar típico da Força Aérea dos Estados Unidos, desenvolvido para representar diferentes tipos de tráfego de rede, incluindo tanto comunicações legítimas quanto diversos tipos de ataques. Cada instância do conjunto de dados representa uma conexão de rede, contendo atributos como a quantidade de bytes transmitidos pela origem, a duração da conexão e outros parâmetros, conforme descrito na tabela apresentada no Apêndice.

O conjunto de dados contém aproximadamente 25 mil instâncias, descritas por 42 atributos. Após a simulação supervisionada do ambiente, as conexões foram rotuladas como tráfego normal ou intrusão, caracterizando um problema de classificação binária, no qual a variável alvo assume os valores *Normal* ou *Anomalia*. A descrição detalhada dos atributos utilizados é apresentada no Apêndice deste trabalho.

3. Pré-processamento

3.1. Tratamento de Dados

Devido às diferenças entre os modelos avaliados, as etapas de pré-processamento foram adaptadas aos requisitos específicos de cada classificador, mantendo a comparabilidade

dos resultados. Inicialmente, o conjunto de dados foi separado em atributos preditores e rótulos, seguido da divisão em conjuntos de treinamento e teste por meio de amostragem estratificada.

As variáveis categóricas foram codificadas por meio de *One-Hot Encoding* ou *Label Encoding*, conforme o modelo utilizado, enquanto as variáveis numéricas foram padronizadas quando necessário. Além disso, foram conduzidos experimentos com conjuntos de dados reduzidos, nos quais atributos relevantes foram removidos de forma controlada, com o objetivo de avaliar a robustez dos modelos frente à ausência parcial de informações. A codificação das variáveis categóricas por meio de *One-Hot Encoding* e *Label Encoding* foi necessária para permitir que os algoritmos de aprendizagem de máquina processem atributos originalmente representados em formato textual. O *One-Hot Encoding* transforma categorias nominais em vetores binários, evitando a introdução de relações ordinais artificiais entre valores distintos, sendo especialmente indicado para modelos sensíveis à representação dos dados [Hastie et al. 2009]. Por outro lado, o *Label Encoding* atribui valores inteiros às categorias, apresentando menor custo computacional e sendo apropriado para algoritmos baseados em árvores de decisão, nos quais a relação ordinal não compromete o processo de aprendizagem [Géron 2022].

Adicionalmente, a padronização das variáveis numéricas foi realizada por meio da normalização baseada na média e no desvio padrão, garantindo que atributos com escalas distintas contribuíssem de forma equilibrada para o processo de treinamento dos modelos [Bishop 2006]. A divisão do conjunto de dados em subconjuntos de treinamento e teste foi conduzida utilizando amostragem estratificada, assegurando a preservação da proporção original entre tráfego normal e anomalias. Para essa etapa, adotou-se uma proporção de 70% dos dados para treinamento e 30% para teste, o que é especialmente relevante em problemas de classificação com classes desbalanceadas, pois contribui para uma avaliação mais representativa do desempenho dos modelos [Kohavi 1995].

3.2. Seleção e remoção de Atributos

A aplicação da Análise de Componentes Principais (PCA) como etapa de pré-processamento tem se mostrado eficaz para a redução da dimensionalidade dos dados sem perda significativa de informação. Estudos indicam que o PCA é capaz de preservar cerca de 99% da variância original mesmo com reduções superiores a 50% no número de atributos, conforme observado por Santos e Miani [Santos and Miani 2025]. Nesse sentido, técnicas de redução e seleção de atributos são amplamente empregadas na literatura como forma de mitigar redundâncias e melhorar a generalização dos modelos.

É importante destacar a diferença conceitual entre técnicas de redução de dimensionalidade, como o PCA, e a remoção manual de atributos adotada neste estudo. Enquanto o PCA realiza uma transformação linear dos dados, projetando-os em um novo espaço de menor dimensão por meio da combinação dos atributos originais, a abordagem empregada neste trabalho consistiu na exclusão direta de variáveis específicas do conjunto de dados. Essa escolha permitiu preservar a interpretabilidade das *features* remanescentes, além de possibilitar a análise individual do impacto da ausência de cada atributo no desempenho dos classificadores. Ademais, essa estratégia representa de forma mais fiel cenários práticos, nos quais nem sempre todas as informações ou sensores estão disponíveis, contribuindo para a avaliação da robustez dos modelos frente à perda par-

cial de dados. Os scripts utilizados para o pré-processamento e para o treinamento dos modelos estão disponíveis no repositório do projeto [1].

4. Metodologia

4.1. Random Forest

O Random Forest (RF) é um algoritmo de aprendizado de máquina supervisionado baseado em conjuntos de árvores de decisão, no qual múltiplas árvores são treinadas a partir de subconjuntos aleatórios dos dados e dos atributos, combinando suas previsões por meio de votação majoritária. Sua utilização neste trabalho se deve à robustez frente a dados de alta dimensionalidade, à capacidade de modelar relações não lineares e à possibilidade de analisar a importância das features, características relevantes para problemas de detecção de intrusão em redes [Breiman 2001].

4.2. K-Nearest Neighbors

Outro modelo utilizado foi o K-Nearest Neighbors (KNN). Esse modelo se diferencia dos demais por não realizar uma etapa explícita de aprendizado a partir dos dados de treinamento. Em vez disso, ele armazena as instâncias conhecidas e classifica uma nova entrada com base na similaridade em relação a um número arbitrário (K) de exemplos mais próximos previamente registrados [Cover and Hart 1967]

4.3. Isolation Forest

O Isolation Forest (IF) é um algoritmo de aprendizado de máquina não supervisionado, projetado especificamente para a detecção de anomalias (ou outliers) em conjuntos de dados. A escolha do Isolation Forest foi motivada pelo fato de que, em alguns cenários, não é possível obter todos os dados rotulados necessários para a detecção de intrusões em redes. Dessa forma, buscou-se avaliar se o IF seria capaz de identificar anomalias mesmo na ausência parcial dos rótulos utilizados no treinamento [Liu et al. 2008].

4.4. Multilayer Perceptron (MLP)

O Multilayer Perceptron (MLP) é um modelo de rede neural artificial supervisionado, composto por camadas de neurônios interconectados que aplicam transformações não lineares sobre os dados de entrada. A escolha do MLP foi motivada por sua capacidade de aprender padrões complexos no tráfego de rede, sendo adequado para a detecção de intrusões que apresentam comportamentos sutis.

Neste trabalho, o modelo foi avaliado em dois cenários: um utilizando todas as features disponíveis e outro com a remoção da feature mais influente identificada no modelo de melhor resultado, com o objetivo de analisar o impacto da redução de atributos em seu desempenho.

Ressalta-se que o desempenho do MLP depende fortemente da escolha adequada de hiperparâmetros e do pré-processamento dos dados, conforme discutido na literatura [Haykin 2009].

[1] `https://github.com/Jolusca/net-intrusion/tree/main/Ataque-Intrusao-machine`

5. Resultados e Discussão

5.1. Métricas de Avaliação

Para analisar o desempenho dos classificadores, foram utilizadas métricas fundamentais de aprendizado de máquina: *Precision*, *Recall*, *F1-Score* e *Support*.

A *Precision* avalia a porcentagem de instancias dadas como positivas que eram realmente positivas, ajudando a identificar a quantidade adivinhações falsas positivas. Em contrapartida, o *Recall* mede a capacidade do modelo de identificar corretamente todas as instancias positivas presentes no dataset, servindo para medir a proporção de adivinhações falsas negativas.

O *F1-Score* consiste na média harmônica entre a precisão e o recall, sendo particularmente útil quando se busca um equilíbrio entre essas duas métricas. Por fim, o *Support* refere-se ao número de ocorrências de cada instancia no dataset tanto negativas(Normal) quanto negativas(Anomala)

A métrica de *acurácia* não é ideal para essa analise tendo em vista que a maioria dos acessos de rede no mundo real são normais e raramente anômalos, então é muito mais interessante analisar falsos negativos e positivos do que o acerto geral.

5.2. Resultados com K-Nearest Neighbors

Para o modelo KNN, foram realizados alguns testes, com os resultados mais precisos foram encontrados com $K = 1$, resultando em um F1-Score de 0.9946 e uma quantidade quase nula de falsos negativos/positivos. Provavelmente, esse comportamento ocorre devido à grande quantidade de dados disponíveis para cada entrada de treinamento, o que torna a maioria dos ataques mais evidentes. Valores superiores de K levaram ao sobreajuste e ao decréscimo progressivo do desempenho do modelo.

Após uma análise baseada na remoção individual de cada atributo, observou-se que o parâmetro *hot* foi o mais decisivo para o modelo, reduzindo o F1-Score para 0.9931 quando removido. Outra informação relevante é que oito atributos, quando removidos, aumentaram a precisão do modelo. Após essa remoção, o F1-Score atingiu 0.9960, sendo o atributo *diff_srv_rate* o mais prejudicial, cuja exclusão resultou em um aumento de 0.0005 pontos no F1-Score.

Table 1. Métricas de desempenho do KNN

Métrica	Classe Normal	Classe Anomalia
Precision	0.99	0.99
Recall	0.99	0.99
F1-Score	0.99	0.99
Support	4034	3524

5.3. Resultados com Isolation Forest

O Isolation Forest se mostrou ineficiente quando comparado aos outros modelos utilizados durante a pesquisa, apresentando a menor acurácia (0.748). Entretanto, como falado anteriormente, a acurácia não é a métrica mais confiável em conjuntos de dados desbalanceados, o que é comum em problemas de detecção de anomalias, pois pode ser

inflacionada pela capacidade do modelo em classificar corretamente a classe majoritária (tráfego normal). Dessa forma, uma análise mais aprofundada torna-se necessária.

Table 2. Métricas de desempenho do Isolation Forest

Métrica	Classe Normal	Classe Anomalia
Precision	1.00	0.65
Recall	0.53	1.00
F1-Score	0.69	0.79
Support	2690	2349

Como pode ser observado na Tabela 2, todos os eventos classificados como normais eram, de fato, normais. A principal dificuldade do modelo esteve na identificação correta das anomalias, apresentando precisão de 65% (0.65) ao rotular eventos como intrusão. Além disso, considerando todos os eventos analisados, o modelo foi capaz de identificar corretamente apenas 53% (0.53) dos eventos normais, o que indica a geração de um elevado número de falsos positivos, isto é, eventos normais classificados como ataques.

Por outro lado, o Isolation Forest não apresentou o problema de gerar falsos negativos, uma vez que ataques não foram classificados como eventos normais. Em síntese, o modelo apresenta dificuldades em verificar se um evento é realmente normal, gerando falsos positivos que podem ocasionar retrabalho para equipes de segurança, as quais precisam analisar manualmente eventos normais identificados como anômalos.

5.4. Resultados com Random Forest

O Random Forest apresentou o melhor desempenho entre todos os modelos avaliados quando utilizado com o conjunto completo de atributos, alcançando uma acurácia de 0.9975. O modelo demonstrou excelente capacidade de generalização, classificando corretamente tanto o tráfego normal quanto as intrusões, o que evidencia seu elevado poder discriminativo em um cenário de detecção de anomalias em redes.

Conforme apresentado na Tabela 3, os valores de Precision, Recall e F1-Score foram iguais a 1.00 para ambas as classes, indicando a ausência praticamente total de falsos positivos e falsos negativos. Esses resultados reforçam a robustez do Random Forest e justificam sua utilização como modelo de referência para a análise de importância dos atributos. Nesse contexto, a feature *src_bytes* foi identificada como a mais influente no modelo, motivando sua remoção em uma segunda etapa experimental.

Com o objetivo de avaliar a robustez do Random Forest frente a conjuntos de dados incompletos, o modelo foi reavaliado após a exclusão da feature *src_bytes*. Nesse cenário, observou-se uma queda significativa no desempenho, com a acurácia reduzida para 0.9630, conforme apresentado na Tabela 4. Embora o modelo ainda apresente bons resultados globais, nota-se uma redução no Recall da classe normal (0.93), indicando um aumento na taxa de falsos positivos, ou seja, eventos normais classificados incorretamente como anomalias.

Esse comportamento evidencia a forte dependência do Random Forest em relação a atributos altamente informativos e reforça a importância da qualidade e da completude dos dados em aplicações de detecção de intrusões em redes. Além disso, os resultados

obtidos corroboram a motivação deste trabalho em investigar como diferentes modelos de aprendizado de máquina reagem à ausência de atributos relevantes no conjunto de dados.

Table 3. Métricas de desempenho do Random Forest (dataset completo)

Métrica	Classe Normal	Classe Anomalia
Precision	1.00	1.00
Recall	1.00	1.00
F1-Score	1.00	1.00
Support	3523	4035

Table 4. Métricas de desempenho do Random Forest (dataset reduzido, sem *src_bytes*)

Métrica	Classe Normal	Classe Anomalia
Precision	0.99	0.94
Recall	0.93	0.99
F1-Score	0.96	0.97
Support	3523	4035

5.5. Resultados com MLP

O Multilayer Perceptron foi avaliado em dois cenários distintos: um utilizando todas as features disponíveis no conjunto de dados e outro com a remoção da feature *src_bytes*, identificada como a mais influente no modelo Random Forest, que apresentou o melhor desempenho geral. O objetivo dessa análise foi avaliar o impacto da ausência de uma feature relevante no desempenho do MLP, simulando cenários em que o conjunto de dados pode estar incompleto.

No cenário completo, o MLP obteve uma acurácia de 0.9960, apresentando desempenho equilibrado entre as classes. A classe Anomalia apresentou precisão de 0.9969 e recall de 0.9946, enquanto a classe Normal obteve precisão de 0.9953 e recall de 0.9973, resultando em F1-Scores superiores a 0.995 para ambas as classes, conforme apresentado na Tabela 5.

Table 5. Métricas de desempenho do MLP com todas as features

Métrica	Classe Anomalia	Classe Normal
Precision	0.9969	0.9953
Recall	0.9946	0.9973
F1-Score	0.9957	0.9963
Support	3523	4035

Após a remoção da feature *src_bytes*, o MLP apresentou uma leve redução de desempenho, com acurácia de 0.9952. Observa-se que, embora o recall da classe Anomalia tenha aumentado para 0.9969, houve redução na precisão dessa classe, passando para 0.9929. Para a classe Normal, verificou-se comportamento inverso, com aumento da precisão e redução do recall. Ainda assim, os valores de F1-Score permaneceram elevados para ambas as classes, indicando que o modelo manteve desempenho consistente

mesmo com a remoção de uma feature altamente relevante em outro classificador, conforme mostrado na Tabela 6.

Table 6. Métricas de desempenho do MLP sem a feature *src_bytes*

Métrica	Classe Anomalia	Classe Normal
Precision	0.9929	0.9973
Recall	0.9969	0.9938
F1-Score	0.9949	0.9955
Support	3523	4035

De forma geral, os resultados indicam que o MLP é relativamente robusto à remoção de uma feature altamente influente em outro modelo, apresentando apenas pequenas variações nas métricas de desempenho. Esse comportamento reforça a capacidade do MLP de aprender representações distribuídas dos dados, reduzindo sua dependência de atributos individuais.

6. Conclusão

Este trabalho analisou a influência da seleção de atributos e do desempenho de diferentes classificadores na detecção de anomalias em redes de computadores. Através de uma abordagem de remoção controlada de atributos, foi possível observar como a integridade dos dados impacta diretamente a eficácia dos sistemas de detecção de intrusões.

Os resultados demonstraram que o modelo Random Forest apresentou o desempenho mais elevado com o conjunto de dados completo, alcançando resultados globais mais proximos de 1.0. Contudo, este modelo revelou-se extremamente sensível à ausência de informações críticas, como o parâmetro *src_bytes*, sofrendo uma degradação significativa na sua capacidade de identificar corretamente o tráfego normal.

Em contrapartida, o Multilayer Perceptron (MLP) destacou-se pela sua robustez e capacidade de generalização. Mesmo após a remoção de atributos altamente influentes, o MLP manteve um desempenho consistente com F1-Scores superiores a 0,99. Este comportamento sugere que redes neurais conseguem aprender representações distribuídas dos dados, tornando-as menos dependentes de variáveis individuais em comparação com modelos baseados em árvores de decisão.

Adicionalmente, o estudo revelou as limitações do Isolation Forest, que apresentou um elevado índice de falsos positivos, e a sensibilidade do KNN ao parâmetro K e a atributos específicos como o *hot* e o *diff_srv_rate*.

Conclui-se que a seleção de um classificador para segurança de redes não deve basear-se exclusivamente na precisão absoluta em ambientes controlados. Em cenários reais, onde a captura de pacotes pode ser imperfeita ou incompleta, a resiliência do modelo torna-se um fator determinante.

References

Bhosale, S. (2018). Network intrusion detection. Kaggle. Acesso em 2025.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Buczak, A. L. and Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

ENISA (2025). The EU-cybersecurity index 2024: EU-level insights and next steps. Technical report, European Union Agency for Cybersecurity. Acesso em: 6 jan. 2026.

Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.

Haykin, S. (2009). *Neural Networks and Learning Machines*. Prentice Hall, 3 edition.

He, K., Kim, D. D., and Asghar, M. R. (2023). Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 25(1):538–566.

Jia, W., Sun, M., Lian, J., and Hou, S. (2022). Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3):2663–2693.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation. *IJCAI*.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 413–422. IEEE.

Mladenić, D. (2005). Feature selection for dimensionality reduction. In *International Statistical and Optimization Perspectives Workshop" Subspace, Latent Structure and Feature Selection"*, pages 84–102. Springer.

Santos, K. C. and Miani, R. S. (2025). Impacto da redução de dimensão e seleção de atributos na generalização de modelos de detecção de intrusão. In *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, Uberlândia, MG, Brazil. SBC.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6.

Zaheer, R., Hanif, M. K., Sarwar, M. U., and Talib, R. (2025). Evaluating the effectiveness of dimensionality reduction on machine learning algorithms in time series forecasting. *IEEE Access*.

Zamani, M. and Movahedi, M. (2013). Machine learning techniques for intrusion detection. *arXiv preprint arXiv:1312.2177*.

Apêndice A — Descrição Completa do Dataset

Table 7. Descrição completa dos atributos do dataset de intrusão em redes

Atributo	Descrição
duration	Duração da conexão de rede
protocol_type	Protocolo utilizado na conexão (TCP, UDP ou ICMP)
service	Serviço de rede acessado
flag	Estado da conexão conforme o protocolo
src_bytes	Quantidade de bytes enviados pela origem
dst_bytes	Quantidade de bytes enviados pelo destino
land	Indica se origem e destino possuem mesmo IP e porta
wrong_fragment	Número de fragmentos incorretos
urgent	Número de pacotes marcados como urgentes
hot	Indicadores de comportamentos suspeitos
num_failed_logins	Número de tentativas de login malsucedidas
logged_in	Indica se o login foi realizado com sucesso
num_compromised	Número de condições comprometidas
root_shell	Indica obtenção de shell com privilégio root
su_attempted	Tentativas de uso do comando *su*
num_root	Número de acessos root
num_file_creations	Número de arquivos criados
num_shells	Número de shells abertos
num_access_files	Número de acessos a arquivos sensíveis
num_outbound_cmds	Número de comandos externos enviados
is_host_login	Indica login como host
is_guest_login	Indica login como convidado
count	Conexões com o mesmo host em janela de tempo
srv_count	Conexões com o mesmo serviço em janela de tempo
serror_rate	Taxa de erros SYN
srv_serror_rate	Taxa de erros SYN para o serviço
rerror_rate	Taxa de erros de resposta
srv_rerror_rate	Taxa de erros de resposta para o serviço
same_srv_rate	Taxa de conexões para o mesmo serviço
diff_srv_rate	Taxa de conexões para serviços diferentes
srv_diff_host_rate	Taxa de serviços acessando hosts distintos
dst_host_count	Número de conexões para o mesmo host destino
dst_host_srv_count	Número de conexões para o mesmo serviço no host
dst_host_same_srv_rate	Taxa de serviços iguais para o host destino
dst_host_diff_srv_rate	Taxa de serviços diferentes para o host destino
dst_host_same_src_port_rate	Taxa de conexões com mesma porta de origem
dst_host_srv_diff_host_rate	Taxa de serviços com hosts distintos
dst_host_serror_rate	Taxa de erros SYN no host destino
dst_host_srv_serror_rate	Taxa de erros SYN por serviço no host destino
dst_host_rerror_rate	Taxa de erros de resposta no host destino
dst_host_srv_rerror_rate	Taxa de erros de resposta por serviço no host destino