Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

MATHEUS GUILHERME ARRAES VELOSO

MONOGRAFIA DE PROJETO ORIENTADO EM COMPUTAÇÃO

ANÁLISE DE DESEMPENHO DE ALGORITMOS DE DETECÇÃO DE ANOMALIAS NO CONTEXTO DE INTRUSÃO DE REDES

Belo Horizonte

2019 / 2º semestre
Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

ANÁLISE DE DESEMPENHO DE ALGORITMOS DE DETECÇÃO DE ANOMALIAS NO CONTEXTO DE INTRUSÃO DE REDES

por

MATHEUS GUILHERME ARRAES VELOSO

Monografia de Projeto Orientado em Computação I

Apresentado como requisito da disciplina de Projeto Orientado em Computação I do Curso de Bacharelado em Ciência da Computação da UFMG

Prof. Dr. Leonardo Barbosa
Orientador

Belo Horizonte
2019 / 2º semestre

À Deus,
aos professores,
aos colegas de curso e
aos meus familiares,
dedico este trabalho.

AGRADECIMENTOS

Inicialmente quero agradeço a Deus, pelos dons recebidos.

Agradeço aos meus pais, pelo amor incondicional.

Aos meus professores, pelos conhecimentos adquiridos.

E finalmente aos colegas de curso pela convivência e trocas.

"Viva como se fosse morrer amanhã.
Aprenda como se fosse viver para sempre.”

Mahatma Gandhi

RESUMO

O principal objetivo do presente trabalho é fazer uma análise dos algoritmos de detecção de anomalias, com o objetivo principal de identificar quais são os mais indicados para a tarefa de detecção de intrusão de redes. O trabalho é inspirado na terceira competição internacional de descoberta de conhecimento e ferramentas de mineração de dados de 1999, também conhecida como KDD-99 Cup (Knowledge Discovery and Data Mining Competition), em que os participantes deveriam criar modelos capazes de distinguir entre conexões maliciosas, chamadas de ataques, e conexões normais. Assim, este trabalho também tem o objetivo de comparar métodos utilizados há 20 anos, com algoritmos recentes, utilizando principalmente os resultados divulgados e as métricas de Precisão e Revocação. Os métodos analisados são algoritmos de mineração de dados popularmente utilizados na detecção de anomalias, isto é, a identificação de itens, observações ou eventos raros que levantam suspeitas por serem significativamente diferentes da maioria dos dados. Foram analisados neste trabalho 9 métodos: PCA (Principal Component Analysis), MCD (Minimum Covariance Determinant), OCSVM (One-Class Support Vector Machines), LOF (Local Outlier Factor), CBLOF (Clustering-Based Local Outlier Factor), kNN (k Nearest Neighbors), HBOS (Histogram-based Outlier Score), IForest (Isolation Forest) e o XGBOD (Extreme Boosting Based Outlier Detection), todos presentes na biblioteca python PyOD.

Palavras-chave:. Detecção de Intrusão de Redes, Detecção de Anomalia, KDD-99, PyOD, Comparação, Análise.

ABSTRACT

The main objective of the present work is to analyze the anomaly detection algorithms, with the main objective of identifying which are the most suitable for a network intrusion detection task. The work is inspired by the third international Knowledge Discovery and Data Mining Competition of 1999, also known as the KDD-99 Cup, in which participants had to build models capable of distinguishing between bad connections, called attacks and normal connections. Thus, this paper also aims to compare methods used 20 years ago with recent algorithms, using mainly the published results and the Precision and Recall metrics. The methods analyzed are data mining algorithms often used to detect anomalies, that is, an identification of rare items, notifications, or events that may cause suspicions of use other than common data. Nine methods were analyzed: PCA (Principal Component Analysis), MCD (Minimum Covariance Determinant), OCSVM (One-Class Support Vector Machines), LOF (Local Outlier Factor), CBLOF (Local Outlier Based Factor). in Cluster), kNN (k Nearest Neighbors), HBOS (External Hysterogram Score), IForest (Isolation Forest), and XGBOD (Extreme Stimulus Out of Range Detection), all present in the PyOD Python library.

Keywords*:* Intrusion Detection, Outiler Detection, Anomaly Detection, KDD-99, PyOD, Analysis.

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE SIGLAS

API	Application Programming Interface
CBLOF	Clustering-Based Local Outlier Factor
DOS	Denial-of-Service
HBOS	Histogram-based Outlier Score
IForest	Isolation Forest
IP	Internet Protocol
KDD Cup	Knowledge Discovery and Data Mining Tools Competition
kNN	k Nearest Neighbors
LAN	Local Area Network
LOF	Local Outlier Factor
MacOS	Mac Operating System
MCD	Minimum Covariance Determinant
MIT	Massachusetts Institute of Technology
NIDS	Network Intrusion Detection Systems
OCSVM	One-Class Support Vector Machines
PCA	Principal Component Analysis
PyOD	Python Outlier Detection (toolkit)
R2L	Remote-to-Local
ROC	Receiver Operating Characteristic
TCP	Transmission Control Protocol
U2R	User-to-Root
XGBOD	Extreme Boosting Based Outlier Detection

SUMÁRIO

1 INTRODUÇÃO

A cada dia aumenta mais, o número de dispositivos conectados à internet, e muitos deles acabam se tornando vítimas de ataques e atividades maliciosas que ameaçam a integridade, a confidencialidade ou a disponibilidade de um recurso [**Erro! Fonte de referência não encontrada.**].

Com o avanço da tecnologia e com a complexidade dos protocolos de redes, novas estratégias de ataque vêm surgindo e nem mesmo os firewalls mais potentes tornam um computador 100% seguro [0]. Por isso, cada vez mais se tem investido em Sistemas de Detecção de Intrusão em Redes (Network Intrusion Detection Systems, NIDS), para que seja possível detectar e identificar as novas formas ataque e então criar medidas de remediação ou prevenção para estes.

Os desenvolvedores de NIDS sempre tiveram o propósito de solucionar este problema de maneira mais eficiente e com o menor custo [0]. Atualmente a estratégia mais popular utilizada são técnicas de aprendizado de máquina e mineração de dados, que são métodos capazes de tratar grande quantidade de dados resultando na aquisição de conhecimento. Métodos de aprendizado são mais autônomos, o que diminui a necessidade de intervenção humana, o que facilita o desenvolvimento de sistemas especialistas ou sistemas baseados em conhecimento sem a necessidade constante de um especialista [0].

Dentro do campo de mineração de dados, há a chamada mineração das exceções ou detecção de outliers/anomalias, que é uma tarefa importante de análise de dados que detecta dados anômalos ou anormais de um determinado conjunto de dados. É considerada uma área importante da pesquisa em mineração de dados, por envolver a descoberta de padrões fascinantes e raros nos dados. As anomalias são consideradas importantes porque indicam eventos significativos, mas raros, e podem levar a ações críticas a serem tomadas em uma ampla gama de domínios de aplicativos; por exemplo, um padrão de tráfego incomum em uma rede pode significar que um computador foi invadido e que os dados são transmitidos para destinos não autorizados.

Em 1999, foi realizada a quinta conferência da Descoberta de Conhecimento e Mineração de Dados (KDD), onde uma competição foi proposta para que fossem desenvolvidos modelos capazes de distinguir transações de redes normais das transações de ataque. As estratégias utilizadas foram divulgadas junto com os resultados.

O presente estudo pretende realizar testes de desempenhos sobre algoritmos de detecção de anomalia comparar os resultados não só entre os métodos aqui citados, mas também com resultados de 1999 com o objetivo de responder as seguintes questões:

- Abordar o problema de detecção de intrusão de redes como um problema de detecção de anomalia é uma estratégia viável?
- Dentre os algoritmos implementados, qual obteve melhores resultados?
- Houve alguma evolução nos métodos de detecção de intrusão desde a proposta do problema, há 20 anos?

2 CONTEXTUALIZAÇÃO E TRABALHOS RELACIONADOS

Existem poucos datasets rotulados, com dados referentes à intrusão de rede, de domínio público que podem ser usados para pesquisa. Um dos mais populares é o fornecido na conferência da Descoberta de Conhecimento e Mineração de Dados de 1999 (KDD 99), que foi o único do tipo até o ano de 2005 [5] e é ainda ativamente utilizado por pesquisadores da área [6].

Este dataset foi criado em 1998 no MIT Lincoln Laboratory, onde foram simuladas transações de uma rede local da Força Aérea dos Estados Unidos (U. S. Air Force LAN). Esta rede foi monitorada e toda forma de transação foi armazenada durante 9 semanas. Além de replicar as transações realizadas pela força aérea, os pesquisadores também atacaram propositalmente a rede, para gerar as anomalias nos dados.

O Dataset é composto de conexões rotuladas como normal ou como ataque, com exatamente um tipo de ataque específico. Uma conexão é uma sequência de pacotes TCP iniciando e terminando em momentos bem definidos, entre os quais os dados fluem de um endereço IP de origem para um endereço IP de destino sob algum protocolo bem definido. Cada registro de conexão consiste em cerca de 100 bytes.

Os ataques se enquadram em quatro categorias principais:

- DOS (denial-of-service): negação de serviço*;*
- R2L (remote-to-local): acesso não autorizado de uma máquina remota;
- U2R (user-to-root): acesso não autorizado a privilégios locais de superusuário (root);
- Probing: vigilância e outras sondagens.

Dentro das quatro categorias listadas acima, estão distribuídos 23 tipos de ataques existentes no dataset. A tabela abaixo mostra quais rótulos pertencem a cada classe:

Tabela 1 - Rótulos do dataset agrupados em classes

Classe de Ataque	Nome dos Ataques
Normal	Normal
Probe	Ipsweep, nmap, portsweep, satan
DOS (Denial of Service)	Back, land, Neptune, pod, smurf, teardrop
U2R (Uset-To-Root)	buffer_overflow, loadmodule, perl, rootkit
R2L(Remote-To-Local)	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster

Fonte: Elaborada pelo autor.

Foram armazenadas 41 características de cada conexão armazenada no dataset, como, duração, tipo de protocolo, serviço, IP de origem, IP de destino, contador para o número de conexões de uma mesma porta, número de uso de terminais, número de operações de acesso à arquivos, dentre outras. Algumas informações são dinâmicas, como o número da porta, por

isso o contador considera apenas conexões em um intervalo de 2 segundos para computar estas características.

O conjunto de dados é fornecido em várias formas. Neste trabalho foram utilizadas tanto a versão completa quanto uma outra versão contendo apenas 10% dos dados. Na tabela 2 estão listadas as distribuições de cada um dos rótulos em cada um dos datasets.

Tabela 2 - Distribuição da classe de ataques no dataset

Classe	**10% Dataset**	**Dataset Completo**
Normal	97.278	972.781
Probe	4.107	41.102
DOS	391.458	3.883.370
U2R	52	52
R2L	1.126	1.126
TOTAL:	**494.021**	**4.898.431**

Fonte: Elaborada pelo autor.

Muitas soluções para este problema foram implementadas e seus resultados foram divulgados [6]. Entretanto nenhum dos estudos realizados sobre este dataset utiliza a estratégia de detecção de anomalia, isto é, considerar que as conexões de ataque são na verdade exceções do dataset, e não uma classe a mais para o classificador identificar. E portanto, alguns métodos nunca foram explorados. O algoritmo XGBOD (Extreme Gradient Boosting Outlier Detection) é um exemplo, que usa múltiplos algoritmos não supervisionados para identificar quais dimensões dos dados são mais expressivas na tomada de decisão, e então usa esta informação em algoritmos supervisionados que realizam agrupamentos sobre o espaço reduzido [7]. Trata-se de um algoritmo recente (2018), ainda não tão utilizado em testes ou outros estudos, mas que aparenta ser uma ferramenta poderosa para aplicações como NIDS.

3 DESENVOLVIMENTO DO TRABALHO

Nesta seção serão discutidas as metodologias empregadas para realizar a análise dos algoritmos de detecção de anomalias no contexto de intrusão de redes.

3.1 PyOD

O PyOD é um kit de ferramentas Python abrangente e escalável para detectar objetos periféricos em dados multivariados. Yue Zhao, um de seus criadores, afirma que comparado à outras bibliotecas, PyOD tem seis vantagens: Primeiro, contém mais de 20 algoritmos que cobrem desde técnicas clássicas até técnicas recentes. Segundo, a PyOD implementa a combinação de métodos para combinar os resultados de múltiplos detectores. Terceiro, inclui uma API unificada, documentação detalhada e exemplos interativos de todos os algoritmos para maior clareza e facilidade de uso. Quarto, todos os modelos são cobertos por testes de unidade, com integração contínua entre plataformas, cobertura e verificação e manutenção de código. Quinto, instrumentos de otimização são empregados sempre que possível. Por fim, o PyOD é compatível com Python 2 e 3, disponíveis para os principais sistemas operacionais

(Windows, Linux e MacOS). [0] Por esses motivos, para este trabalho, optou-se pelo uso desta biblioteca.

A página da web contendo a documentação desta ferramenta [0] possui também algumas informações extras como um benchmark dos principais modelos, capaz de fornecer uma visão geral das implementações. A partir destes resultados foram selecionados 9 modelos para serem analisados neste trabalho, sendo eles PCA (Principal Component Analysis), MCD (Minimum Covariance Determinant), OCSVM (One-Class Support Vector Machines), LOF (Local Outlier Factor), CBLOF (Clustering-Based Local Outlier Factor), kNN (k Nearest Neighbors), HBOS (Histogram-based Outlier Score), IForest (Isolation Forest) e o XGBOD (Extreme Boosting Based Outlier Detection).

A instalação da biblioteca foi feita de acordo com as instruções contidas na documentação. Os exemplos disponibilizados também foram executados para confirmar o bom funcionamento das ferramentas, além de possibilitar um primeiro contato. Embora os exemplos tenham executado perfeitamente, alguns modelos enfrentaram problemas ao serem testados sobre o dataset, como o XGBOD. Enquanto os outros algoritmos convergiam em menos de um dia para o resultado, o XGBOD nunca convergiu, mesmo após 10 dias em execução. Uma solução foi proposta para que o algoritmo não fosse retirado da análise proposta por este trabalho, visto que ele tem muito a contribuir para as perguntas propostas na introdução, por se tratar de um algoritmo recente (2018) e focado na detecção de anomalias [**Erro! Fonte de referência não encontrada.**].

O XGBOD usa métodos não supervisionados da biblioteca PyOD para extrair uma representação mais rica dos dados e então concatena estas novas características obtidas com as originais construindo um espaço de características maior. E então aplica o algoritmo XGBoost neste espaço aumentado. No artigo que acompanha o lançamento da ferramenta, Yue Zhao compara o seu método com o XGBoost puro, cujos dados estão replicados aqui na tabela 3.

Tabela 3 - Comparação de valor da curva ROC entre os modelos XGBOD e XGBoost em diferentes datasets

Datasets	XGBOD	XGBoost
Arrhythmia	0.8110	0.8816
Letter	0.9593	0.9729
Cardio	0.9868	0.9976
Speech	0.7819	0.8591
Satellite	0.9254	0.9666
Mnist	0.9980	0.9999
Mammography	0.9105	0.9431

Fonte: Elaborada pelo autor.

A partir dos resultados acima é possível ver que nem sempre o XGBOD demonstra uma melhoria nos resultados, como é o caso do teste sobre o dataset Mammography. E mesmo naqueles em que houve melhoria, o ganho é bem baixo, não indo além de 2%. O XGBoost ainda, foi capaz de convergir sobre o dataset KDD-99 em menos de um hora. Por estas razões citadas, foi decidido que o algoritmo XGBoost seria utilizado nesta análise no lugar do XGBOD existente na biblioteca PyOD.

3.2 Balanceando o Banco de Dados

Anomalias são, por definição, itens, eventos ou observações raras, que chamem a atenção por diferirem significativamente da maioria dos dados [9]. Analisando a distribuição de cada classe do dataset (Tabela 2), é possível perceber que quase 80% dos dados recebe o rótulo de ataque DOS e isto configura um grande problema para o trabalho, pois este ataque não pode ser considerado uma anomalia.

Os algoritmos implementados na PyOD possuem um parâmetro chamado *contamination* (contaminação) onde é passado o percentual de outliers no dataset, e este parâmetro é limitado para valores entre 0 (não incluso) e 0.5. E isto mais uma vez se torna um empecilho em analisar os dados de intrusão da classe DOS. Para solucionar este problema, optou-se por remover um grande número das conexões do tipo DOS.

Foi descrito um programa em python que fazia a leitura do dataset contando quantas conexões eram de cada classe. Então uma segunda leitura era feita sobre os dados, de forma aleatória e apenas algumas transações eram efetivamente utilizadas nos métodos, as demais eram descartadas. Este processo foi testado várias vezes, alterando o número de transações mantidas ou descartadas até se chegar na melhor proporção de contaminação. Os resultados destes testes podem ser vistos nos gráficos 1 e 2.

3.3 Hiperparâmetro

Para que os valores dos hiperparâmetros sejam bem selecionados, o especialista em aprendizado de máquina deve ser capaz de explorar as diversas configurações para escolher uma combinação que resulte em um bom desempenho do modelo. Entretanto, esse processo de exploração requer que o treinamento do modelo seja realizado para cada uma das combinações, e, além disso, que as combinações de parâmetros utilizadas sejam registradas [**Erro! Fonte de referência não encontrada.**].
O parâmetro de contaminação do PyOD é utilizado para fazer os ajustes durante a fase de treino.[0] Este hiperparâmetro é comum a todos os modelos, e portanto o mais indicado para exploração em testes. Intuitivamente espera-se que o valor deste parâmetro corresponda a realidade, isto é, que o valor de contaminação declarado seja exatamente a razão de outliers existente no dataset. Na prática, porém, este valor pode não ter esta correspondência, principalmente quando o modelo não é capaz de predizer corretamente a totalidade dos dados.

O especialista deve então definir suas prioridades, isto é, se ele considera melhor a presença de falsos positivos ou de falsos negativos, e ajustar este hiperparâmetro para alcançar tal resultado.

Segundo Željko Ivezić, este compromisso entre a contaminação e completude pode ser demonstrado pela curva ROC (Receiver Operating Characteristic), que tipicamente é um gráfico da fração entre verdadeiros positivos e verdadeiros negativos. Na astronomia, curvas ROC são comumente plotadas como completude esperada x contaminação [0].
Para descobrir o valor ótimo para o parâmetro de contaminação foram realizadas diversas execuções do modelo que pudesse cobrir de maneira linear alguns dos valores dentro do intervalo permitido (0, 0.5]. Os resultados podem ser vistos dos gráficos 3 ao 10.

3.4 Variância dos dados

Um dos problemas clássicos da mineração de dados é encontrar o equilíbrio viés-variância [**Erro! Fonte de referência não encontrada.**]. Quando um modelo é complexo o suficiente para atingir bons resultados no modelo de treino, mas não é generalizável o suficiente para manter este padrão nos dados de teste, é dito que este é um caso de *overfitting,* e sua causa é especialmente a alta variância dos dados.
O conjunto de dados fornecidos pelo KDD-99 possui 4 classes de ataques bem diferentes. Ataques de DOS, por exemplo, tem a característica muito comum de envolver inúmeras conexões sequenciais no mesmo dispositivo, em um curto período de tempo, enquanto ataques R2L ou U2R são ataques incorporados em conjuntos de dados de um pacote, e geralmente envolvem uma conexão simples [**Erro! Fonte de referência não encontrada.**]. E esta distinção entre as classes de ataque caracteriza uma alta variância nos dados, que é prejudicial à tentativa dos modelos de aprender a distinção entre uma transação normal ou um ataque.

Para corrigir este problema, o dataset foi quebrado em 4. Em todos eles, as transações consideradas normais foram mantidas, mas cada um continha apenas uma das classes de transações de ataque. Desta forma, a variância entre o que é considerado como ataque se torna bem menor.

3.5 Seleção de Características

Gareth James define a seleção de características, também conhecida como seleção de atributos ou seleção de subconjunto de variáveis, como o processo de selecionar um subconjunto de recursos relevantes para uso na construção do modelo, com o objetivo de simplificar o modelo e torna-lo simples de interpretar[**Erro! Fonte de referência não encontrada.**]. Além disto, esta estratégia também encurta o tempo de treino, evita problemas como a maldição da dimensionalidade e reduz a variância dos dados e consequentemente as chances de *overfitting*[**Erro! Fonte de referência não encontrada.**].

O dataset KDD-99 oferece 41 características das transações (Anexo 1), em que 9 delas (duration, protocol+type, service, src_bytes, dst_bytes, flag, land, wrong_fragment, urgente) são referentes à informações do cabeçalho do protocolo TCP. Outras 13 (hot, num_failed_logins, logged_in, num_compromised, root_shell, su_attempted, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, is_hot_login, is_guest_login) são características do conteúdo da conexão, derivadas por conhecimento do domínio. O restante (count, serror_rate, rerror_rate, same_srv_rate, diff_srv_rate, srv_count, srv_serror_rate, srv_rerror_rate, srv_diff_host_rate) são características de tráfego considerando uma janela temporal de 2 segundos [**Erro! Fonte de referência não encontrada.**]. E nem todas elas são importantes para caracterizar uma transação como ataque ou não.

Em 2005, os pesquisadores Gunes Kayacik, A. Nur Zincir-Heywood e Malcolm I. Heywood publicaram um artigo onde descreveram a importância de cada característica de uma conexão na tarefa de identificar ataques[**Erro! Fonte de referência não encontrada.**]. Eles identificaram qual a característica mais importante na detecção de cada ataque e um

percentual desta importância. Por exemplo, ataques de smurf são identificados pela característica source_bytes, que representa um ganho de informação de 98.59%. Além disto, o artigo também disponibiliza uma lista de características mais relevantes para as quais a classe de ataque é corretamente identificada. Fazendo uma combinação destas informações foi possível realizar uma seleção manual de características, reduzindo o dataset de 41 para até 12 características. A tabela 4 mostra quais características foram selecionadas para cada tipo de ataque. O Apêndice A.1. nomeia e descreve cada uma das características.

Tabela 4 - Características selecionadas como importantes para a detecção de cada tipo de ataque.

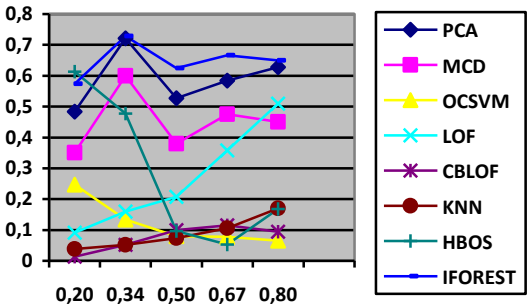
Classe de Ataque	Características Utilizadas	Quantidade
Probe	2,3,4,5,23,24,27,29,30,35,36,37	12
DOS	2,3,4,5,6,7,8,10,13,23,24,25,26,27,28,29,30,33,34,35,36,38,39,40,41	21
U2R	2,3,4,5,6,14,16,23,24,29,30,35,36	13
R2L	2,3,4,5,6,9,11,23,24,29,30,35,36,39	14

Fonte: Elaborada pelo autor.

Definidas as melhores características para seleção, os melhores valores para parâmetros, e a melhor proporção de contaminação, os algoritmos foram então testados e os resultados finais estão contidos nas tabelas 5 e 6.

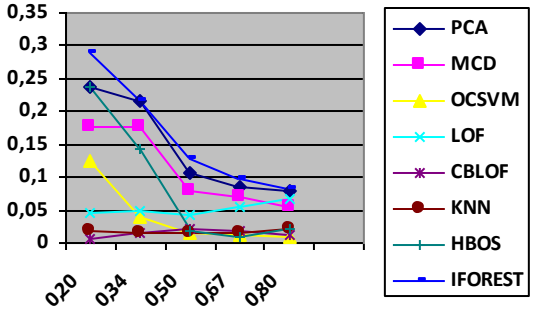
4 RESULTADOS E DISCUSSÃO

Gráficos 1 - Precisão dos métodos em função da contaminação do dataset



Fonte: Elaborado pelo autor.

Gráficos 2 - Revocação dos métodos em função da contaminação do dataset

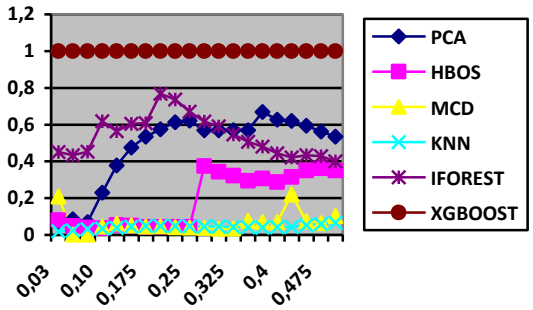


Fonte: Elaborado pelo autor.

A partir dos resultados acima, é possível notar que as melhores revocações estão associadas a uma inferior taxa de contaminação, enquanto a precisão obtém seu valor ótimo quando a proporção entre transações normais e ataques é de 2/3 para 1/3. Isto é, quando a contaminação é próxima de 33,33% é quando se extrai o melhor dos algoritmos. Além disso, alguns algoritmos se mostraram sempre com péssimos resultados, como o OCSVM, CBLOF e o LOF, e portanto a análise destes foi descontinuada.

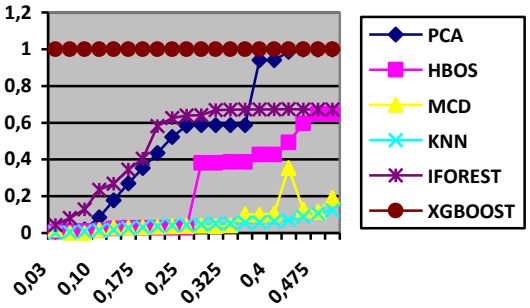
Os datasets foram então separados em 4, cada um contendo apenas um tipo de ataque na proporção de contaminação ótima. Então vieram os testes do hiperparâmetro de contaminação. O XGBoost não possui esse parâmetro (presente porém no XGBOD, sua versão implementada na PyOD). Portanto seu valor foi replicado por toda a tabela de dados na hora de plotar os gráficos de 3 a 10.

Gráficos 3 - Análise de Precisão em função do hiperparâmetro contaminação no dataset contendo apenas conexões do tipo normal ou com ataques do tipo DOS, em uma contaminação real de 33%.



Fonte: Elaborado pelo autor.

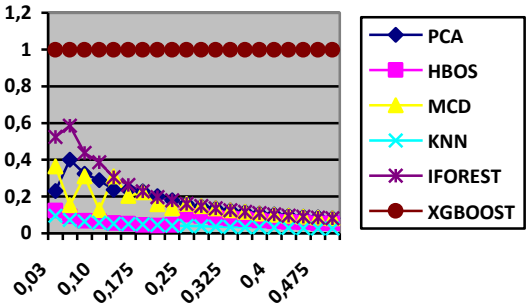
Gráficos 4 - Análise de Revocação em função do hiperparâmetro contaminação no dataset contendo apenas conexões do tipo normal ou com ataques do tipo DOS, em uma contaminação real de 33%.



Fonte: Elaborado pelo autor.

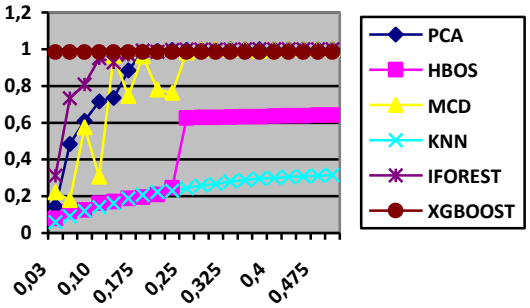
A partir dos resultados acima, é visto que mesmo tendo uma contaminação real de apenas 33%, o valor do parâmetro contamination em 50% gerou resultados mais expressivos quando se analisa a revocação dos modelos. No contexto de identificação de intrusão de redes, uma revocação baixa indicaria a existência de muitos falsos negativos, isto é, ataques seriam identificados como transações normais pelo modelo. Por isso, é desejável que a revocação seja ligeiramente priorizada sobre a precisão, uma vez que a ideia principal de um NIDS é identificar esses ataques. Então, mesmo que a precisão dos modelos, tendo o parâmetro contamination em 0.5, não seja a ótima, ainda seria este o valor ótimo para o parâmetro.

Gráficos 5 - Análise de Precisão em função do hiperparâmetro contaminação no dataset contendo apenas conexões do tipo normal ou com ataques do tipo PROBE, em uma contaminação real de 4.08%.



Fonte: Elaborado pelo autor.

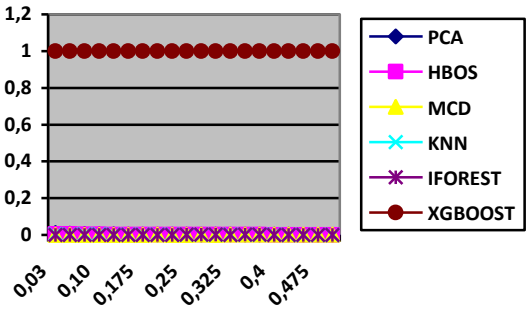
Gráficos 6 - Análise de Revocação em função do hiperparâmetro contaminação no dataset contendo apenas conexões do tipo normal ou com ataques do tipo PROBE, em uma contaminação real de 4.08%.



Fonte: Elaborado pelo autor.

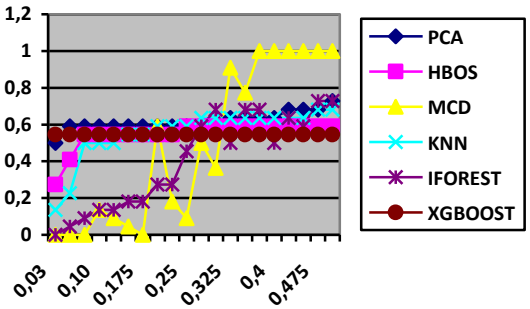
A partir dos testes acima podemos notar uma maior resistência dos modelos em distinguir entre ataques do tipo PROBE, em relação aos ataques de DOS. De fato, mesmo em 1999 após a competição do KDD, o DOS sempre foi o ataque mais fácil de se identificar visto que se trata de transações sequenciais em um curto período de tempo. Mesmo que conexões de ataque de PROBE também sejam sequenciais, estas são mais esparças, acontecendo apenas uma transação por minuto, ou uma a cada dois minutos. E esta esparcialidade, somado com o fato de que há menos dados deste tipo de ataque para os modelos treinarem, resulta em um resultado ligeiramente inferior. Mesmo assim, o padrão dos resultados anteriores se mantém, com o XGBoost, PCA e IForest se destacando em ambos os casos.

Gráficos 7 - Análise de Precisão em função do hiperparâmetro contaminação no dataset contendo apenas conexões do tipo normal ou com ataques do tipo U2R, em uma contaminação real de 0.06%.



Fonte: Elaborado pelo autor.

Gráficos 8 - Análise de Revocação em função do hiperparâmetro contaminação no dataset contendo apenas conexões do tipo normal ou com ataques do tipo U2R, em uma contaminação real de 0.06%.

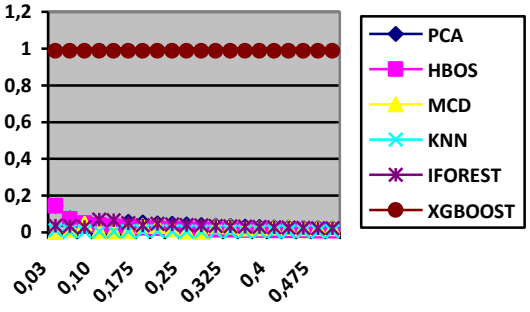


Fonte: Elaborado pelo autor.

Os resultados do tipo de ataque U2R são os piores dentre os quatro tipos de ataque, e a principal razão para isto é a falta de dados disponíveis para os modelos treinarem. Dentre as 4 milhões de transações existentes no dataset, apenas 52 são rotuladas como ataque U2R. Um fato interessante sobre estes resultados é que uma revocação ruim indica uma alta taxa de falsos negativos, isto é transações de ataque sendo preditas como transações normais. Enquanto uma precisão baixa indica um alto número de falsos positivos, isto é, transações normais sendo consideradas como ataque.

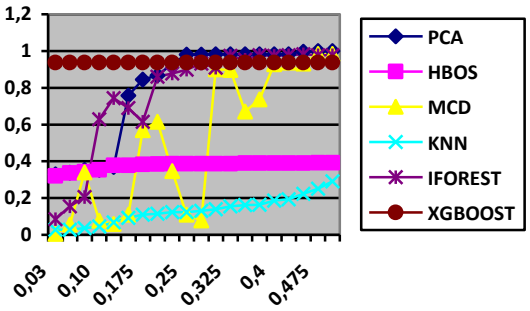
Quando um modelo consegue um resultado inferior em ambas as métricas, significa que ele errou praticamente toda a sua previsão. E isto pode ser consertado com um simples inversor que pega a saída do modelo e inverte. Portanto o algoritmo IForest, com o parâmetro de contaminação em 0.125 ter obtido o valor de 0.0001 em precisão e revocação é equivalente ao resultado de 0.9999 nessas métricas.

Gráficos 9 - Análise de Precisão em função do hiperparâmetro contaminação no dataset contendo apenas conexões do tipo normal ou com ataques do tipo R2L, em uma contaminação real de 1.13%.



Fonte: Elaborado pelo autor.

Gráficos 10 - Análise de Revocação em função do hiperparâmetro contaminação no dataset contendo apenas conexões do tipo normal ou com ataques do tipo R2L, em uma contaminação real de 1.13%.



Fonte: Elaborado pelo autor.

Como resultado dos gráficos acima foi possível encontrar os valores ótimos do parâmetro contaminação para cada tipo de ataque. Reunidos na tabela 5.

Tabela 5 - C = Valor do hiperparâmetro contamination, P = Precisão alcançada pelo modelo, R = Revocação alcançada pelo modelo.

Modelo	PROBE			DOS			U2R			R2L		
	C	P	R	C	P	R	C	P	R	C	P	R
PCA	0.2	20 %	99 %	0.25	62 %	58 %	0.025	99 %	50 %	0.025	88 %	68 %
HBOS	0.25	10 %	63 %	0.5	35 %	65 %	0.025	99 %	73 %	0.475	95 %	66 %
MCD	0.175	22 %	96 %	0.425	95 %	97 %	0.1	99 %	87 %	0.025	99 %	99 %
KNN	0.5	97 %	69 %	0.5	94 %	88 %	0.025	99 %	87 %	0.125	99 %	93 %
IFOREST	0.1	39 %	95 %	0.3	59 %	67 %	0.025	99 %	99 %	0.5	97 %	91 %

Fonte: Elaborada pelo autor.

Repetindo a parametrização acima nos testes com seleção de features foram obtidos os resultados das tabelas 6 e 7.

Tabela 6 - Precisão dos modelos após a seleção de características

Modelo	PROBE	DOS	U2R	R2L
PCA	23%	53%	99%	90%
HBOS	18%	95%	99%	99%
MCD	23%	94%	99%	99%
KNN	98%	94%	99%	99%
IFOREST	40%	53%	99%	99%
XGBOOST	99%	99%	100%	97%

Fonte: Elaborada pelo autor.

Tabela 7 - Revocação dos modelos após a seleção de características

Modelo	PROBE	DOS	U2R	R2L
PCA	99%	99%	60%	70%
HBOS	99%	95%	60%	94%
MCD	96%	96%	99%	94%
KNN	71%	89%	88%	81%
IFOREST	95%	99%	99%	93%
XGBOOST	99%	99%	54%	93%

Fonte: Elaborada pelo autor.

Todos esses resultados podem ainda serem comparados com os resultados de 1999 disponibilizados pelo KDD [**Erro! Fonte de referência não encontrada.**].

Tabela 8 - Resultados dos testes de 1999, utilizando o algoritmo KNN com K = 1.

Classes	Precisão	Revocação
Probe	83.3%	64.8%
DOS	97.1%	99.9%
U2R	13.2%	71.4%
R2L	8.4%	98.8%

Fonte: Elaborada pelo autor.

5 CONCLUSÕES (E TRABALHO FUTUROS)

De maneira geral, o algoritmo XGBoost é o que obteve melhores resultados, e caso apenas um dos algoritmos fosse escolhido com o propósito geral de identificar os quatro tipos de ataque, então este seria o indicado.

Mesmo sem muito destaque, todos os outros modelos conseguiram superar os resultados de 1999. Isso mostra que a estratégia de tratar um problema de intrusão de redes como um problema de detecção de anomalias é realmente eficiente.

As maiores melhorias são em relação aos tipos de ataque U2R e R2L. Disso se intui que com o passar do tempo, os algoritmos foram desenvolvidos capacitados para aprender mais com menos informação e assim obter precisão e revocação tão elevados mesmo no tipo de ataque que possui apenas 52 amostras em um dataset com 4 milhões de instâncias.

Para trabalhos futuros é proposto a avaliação dos outros métodos existentes na biblioteca PyOD, visto que menos da metade deles foram cobertos neste trabalho. Até mesmo métodos de outras bibliotecas podem ser testados e comparados. Em especial destaca-se o XGBOD, que foi apenas parcialmente analisado, deixando o entendimento de sua completude para uma possível continuação deste trabalho.

Realizar um teste similar em outros datasets para confirmar a eficácia desta metodologia quanto à detecção de intrusão de redes, é também de suma importância e considerado para um projeto futuro.

REFERÊNCIAS

1. SUBHY, M.; IBRAHIM, L. M.; BASHEER, D. A comparison study for intrusion database (KDD99, NSL-KDD) based on self organization map (SOM) artificial neural network. **Journal of Engineering Science and Technology**, 8, 2013. Disponível em: <https://www.researchgate.net/publication/329450947_A_comparison_study_for_intrusion_database_KDD99_NSL-KDD_based_on_self_organization_map_SOM_artificial_neural_network>. p. 107-119.

2. KEMMERER, R. A.; VIGNA, G. Intrusion detection a brief history and overview. **Computer**, v. 35, n. 4, abr. 2002. p. 27-30. DOI: 10.1109/MC.2002.1012428. Disponível em: <https://ieeexplore.ieee.org/document/1012428>. Acesso em: 8 set. 2019.

3. LORENZO-FONSECA, I. *et al*. Intrusion detection method using neural networks based on the reduction of characteristics. **Lecture Notes in Computer Science**, v. 5517, 2009, p. 1296-1303.

4. ZAMONER, F. W. **Técnica de aprendizado semissupervisionado para detecção de outliers**. 2014. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2014. Disponível em: <https://teses.usp.br/teses/disponiveis/55/55134/tde-07042014-100038/pt-br.php>. Acesso em: 8 set. 2019.

5. KAYACIK, G., ZINCIR-HEYWOOD, A.; HEYWOOD, M. **Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99**. 2005. Disponivel em: <https://www.researchgate.net/publication/220919984_Selecting_Features_for_Intrusion_Detection_A_Feature_Relevance_Analysis_on_KDD_99>. Acesso em: 8 set. 2019.

6. ÖZGÜR, A.; ERDEM, H. (2016). **A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015**. 2016. Disponível em: <https://doi.org/10.7287/peerj.preprints.1954v1>. Acesso em: 8 set. 2019.

7. ZHAO, Y.; HRYNIEWICKI, M. K. (2018). **XGBOD**: Improving Supervised Outlier Detection with Unsupervised Representation Learning. doi: 10.1109/IJCNN.2018.8489605. Disponível em: <https://arxiv.org/ftp/arxiv/papers/1912/1912.00290.pdf>. Acesso em: 8 set. 2019.

8. ZHAO, Y.; NASRULLAH, Z.; LI, Z. PyOD: A Python Toolbox for Scalable Outlier Detection. **Journal of Machine Learning Research**. 20. p. 1-7. Disponível em: <https://arxiv.org/pdf/1901.01588.pdf>. Acesso em: 5 dez. 2019.

9. ZIMEK, A.; SCHUBERT, E. Outlier Detection. In: LIU, L.; ÖZSU, M. (Orgs.). **Encyclopedia of Database Systems**. Springer, New York: 2015. Disponível em: <https://doi.org/10.1007/978-1-4899-7993-3>. Acesso em: 5 dez. 2019.

10. WELCOME to PyOD Documentation!. Disponível em <https://pyod.readthedocs.io/en/latest/>. Acesso em: 5 dez. 2019.

11. PINA, D. B. *et al*. Análise de Hiperparâmetros em Aplicações de Aprendizado Profundo por meio de Dados de Proveniência. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS (SBBD), 34, 2019, Fortaleza. Anais do XXXIV Simpósio Brasileiro de Banco de Dados. Porto Alegre: Sociedade Brasileira de Computação, nov. 2019. p. 223-228.

12. IVEZIC, Z.; CONNOLLY, A. J.; VANDERPLAS, J. T. **Statistics, Data Mining, and Machine Learning in Astronomy**: A Practical Python Guide for the Analysis of Survey Data, Updated Edition. Princeton, NJ: Princeton University Press, 2014. ISBN: 978-0691151687.

13. GEMAN, S.; BIENENSTOCK, E.; DOURSAT, R. Neural Networks and the Bias/Variance Dilemma. **Neural Computation**, Massachusetts, v. 4, n.1, p. 1-58, jan. 1992. Disponível em: <http://www.mitpressjournals.org/doi/10.1162/neco.1992.4.1.1>. Acesso em: 05 dez. 2019.

14. Gareth JAMES, G. *et al*. An Introduction to Statistical Learning. Los Angeles: Springer, 2013. p. 204.

15. Bermingham, M. L. *et al*. Application of high-dimensional feature selection: evaluation for genomic prediction in man. **Sci. Rep.**, 5. DOI:10.1038/srep10312. Disponível em: <https://www.nature.com/articles/srep10312#citeas>. Acesso em: 5 dez. 2019.

16. KAYACIK, G.; ZINCIR-HEYWOOD, A.; HEYWOOD, M. Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99. 2005. Disponível em: <https://www.researchgate.net/publication/220919984_Selecting_Features_for_Intrusion_Detection_A_Feature_Relevance_Analysis_on_KDD_99>. Acesso em: 5 dez. 2019:

APÊNDICE 1. Descrição das características do dataset KKD 99

Tabela A.1. Lista de características com suas descrições e tipo de dado

Feature	Description	Type	Feature	Description	Type
1. duration	Duration of the connection.	Cont.	**22. is guest login**	1 if the login is a "guest" login; 0 otherwise	Disc.
2. protocol type	Connection protocol (e.g. tcp, udp)	Disc.	**23. Count**	number of connections to the same host as the current connection in the past two seconds	Cont.
3. service	Destination service (e.g. telnet, ftp)	Disc.	**24. srv count**	number of connections to the same service as the current connection in the past two seconds	Cont.
4. flag	Status flag of the connection	Disc.	**25. serror rate**	% of connections that have "SYN" errors	Cont.
5. source bytes	Bytes sent from source to destination	Cont.	**26. srv serror rate**	% of connections that have "SYN" errors	Cont.
6. destination bytes	Bytes sent from destination to source	Cont.	**27. rerror rate**	% of connections that have "REJ" errors	Cont.
7. land	1 if connection is from/to the same host/port; 0 otherwise	Disc.	**28. srv rerror rate**	% of connections that have "REJ" errors	Cont.
8. wrong fragment	number of wrong fragments	Cont.	**29. same srv rate**	% of connections to the same service	Cont.
9. urgent	number of urgent packets	Cont.	**30. diff srv rate**	% of connections to different services	Cont.
10. hot	number of "hot" indicators	Cont.	**31. srv diff host rate**	% of connections to different hosts	Cont.
11. failed logins	number of failed logins	Cont.	**32. dst host count**	count of connections having the same destination host	Cont.
12. logged in	1 if successfully logged in; 0 otherwise	Disc.	**33. dst host srv count**	count of connections having the same destination host and using the same service	Cont.
13. # compromised	number of "compromised" conditions	Cont.	**34. dst host same srv rate**	% of connections having the same destination host and using the same service	Cont.
14. root shell	1 if root shell is obtained; 0 otherwise	Cont.	**35. dst host diff srv rate**	% of different services on the current host	Cont.
15. su attempted	1 if "su root" command attempted; 0 otherwise	Cont.	**36. dst host same src port rate**	% of connections to the current host having the same src port	Cont.
16. # root	number of "root" accesses	Cont.	**37. dst host srv diff host rate**	% of connections to the same service coming from different hosts	Cont.
17. # file creations	number of file creation operations	Cont.	**38. dst host serror rate**	% of connections to the current host that have an S0 error	Cont.
18. # shells	number of shell prompts	Cont.	**39. dst host srv serror rate**	% of connections to the current host and specified service that have an S0 error	Cont.
19. # access files	number of operations on access control files	Cont.	**40. dst host rerror rate**	% of connections to the current host that have an RST error	Cont.
20. # outbound cmds	number of outbound commands in an ftp session	Cont.	**41. dst host srv rerror rate**	% of connections to the current host and specified service that have an RST error	Cont.
21. is hot login	1 if the login belongs to the "hot" list; 0 otherwise	Disc.			

Fonte: Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 [17]