Detecção de Anomalias em Aplicações de Monitoramento de Sistemas utilizando *Isolation Forest*

Anomaly Detection in Monitoring Systems Application using Isolation Forest

Anderson A. de Souza[1]
orcid.org/0000-0003-1629-8051

Agostinho A. F. Júnior[1]
orcid.org/0000-0002-6059-9014

Diego A. da Silva[1]
orcid.org/0000-0001-9078-2760

João V. R. de Andrade[1]
orcid.org/0000-0003-3554-0157

[1]Escola Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Brasil. E-mail: aas3@ecomp.poli.br

DOI: 10.25286/repa.v6i5.2152

Como citar este artigo pela NBR 6023/2018: SOUZA, A. A.; JÚNIOR, A. A. F.; SILVA, D. A.; ANDRADE, J. V. R. Detecção de Anomalias em Aplicações de Monitoramento de Sistemas utilizando Isolation Forest. Revista de Engenharia e Pesquisa Aplicada, Recife, v.6, n. 5, p. 100-109, Novembro, 2021.

RESUMO

A maior parte das aplicações reais são compostas, em sua maioria, por processos críticos para as operações das empresas. Dessa forma, quando se trata de monitoramento de sistemas, a análise preditiva de anomalias é vital para a manutenção do negócio. Atualmente a maior parte de soluções para esse contexto é baseada em análise temporal utilizando redes neurais recorrentes, porém este tipo de técnica demanda alto custo computacional, o que pode inviabilizar abordagens sujeitas à variabilidade (*Data* e *Concept Drift*). Portanto este trabalho foca no desenvolvimento de uma abordagem para detecção de anomalias para esses ambientes, sem a necessidade de consumir muitos recursos computacionais. A solução desenvolvida neste trabalho baseia-se na técnica *Isolation Forest*, os experimentos mostraram que é possível atingir um alto nível de generalização sem a necessidade de um alto poder de processamento, atingindo para classe de anomalia uma *Precision* de 0.99 e um *Recall* de 0.98, com um modelo treinado em aproximadamente 0.56 segundos.

PALAVRAS-CHAVE: Isolation Forest; Monitoramento de Sistemas; Detecção de Anomalias;

ABSTACT

Most real applications are composed of critical processes for business operations. Thus, when it comes to systems monitoring, predictive anomaly analysis is vital for business maintenance. Currently, most solutions for this context are based on temporal analysis using recurrent neural networks, but this technique demands a high computational cost, making approaches subject to variability (Data and Concept Drift) unfeasible. Therefore, this work focuses on developing an approach for detecting anomalies for these environments without consuming a lot of computational resources. The solution developed in this work is based on the Isolation Forest technique. The experiments showed that it is possible to achieve a high level of generalization without the need for high processing power, reaching a Precision of 0.99 and a Recall of 0.98 for the anomaly class, with a trained model in approximately 0.56 seconds.

KEY-WORDS: Isolation Forest; System Monitoring; Anomaly Detection;

1 INTRODUÇÃO

Com o avanço tecnológico, a capacidade de monitorar sistemas se tornou primordial nas companhias atuais. Afinal, uma forte tendência à digitalização das empresas vem acontecendo, criando sistemas cada vez mais complexos que, consequentemente, possuem um maior custo computacional, e de operação **[1]**. Além disso, estes são de grande importância para manutenção das empresas, pois auxiliam na execução de tarefas que são fundamentais para o funcionamento do negócio.

Para os atuais sistemas em funcionamento, a manutenção do correto fluxo de processamento e execução de funcionalidades é de vital importância. Para uma boa análise em longo prazo, os processos executados no sistema, precisam ser auditáveis. Assim, surge a necessidade de uma operação que monitore esses processos, de modo a detalhar seu funcionamento em dado momento e circunstância, ao monitorar aplicações é possível obter uma visão global sobre o comportamento do sistema **[2]**.

Dessa forma, um cenário crítico precisa ser considerado, manter o rastreio da saúde dos processos de diversos clientes é uma tarefa árdua e uma característica imprescindível de aplicações que desejam monitorar sistemas, pois impactam diretamente na qualidade do serviço prestado aos clientes. Além disso, para garantir uma melhor conformidade nas execuções dos ambientes monitorados, é necessário que a aplicação de monitoramento seja capaz de prever falhas de processos **[3]**.

Para a realização desses rastreios pode-se empregar diversas metodologias, muitas delas, baseadas em *Deep Learning*, porém o emprego dessas técnicas é de alto custo computacional, e impossibilitam o re-treino constante, isso, aliado ao volume de dados monitorados, gera um desvio de conceito, que ocorre quando os dados de treino já não são mais representativos em relação aos dados recebidos na aplicação, esse desvio é responsável pela queda de desempenho do modelo **[4]**.

Portanto, tentando resolver estes problemas, foram estabelecidos os seguintes objetivos específicos: estimar a probabilidade de ocorrência de uma anomalia no fluxo de uma aplicação, classificar quais são as anomalias mais impactantes ao funcionamento do sistema, utilizando um método que tenha um baixo custo computacional. Como sistemas de monitoramento vêm ganhando cada vez mais destaque nos dias atuais, as tecnologias de detecção de anomalias vêm se tornando partes vitais desses sistemas, deste modo evoluir tais tecnologias justifica-se, pois, seu desenvolvimento é desejável para garantir melhor funcionamento de aplicações.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados os tópicos necessários para o entendimento do problema e embasamento teórico, ou seja, tópicos relacionados ao monitoramento de sistemas e conceitos que estão presentes nessa área de negócio, além da predição de falhas. Por fim é feito um levantamento na literatura que utilizam da abordagem de mineração de dados para resolver problemas de predição de anomalias.

2.1 MONITORAMENTO DE SISTEMAS

Aplicações de monitoramento online vêm ganhando espaço no mercado B2B (*Business to Business*), soluções inteligentes que proporcionam análise de processos são indispensáveis para atingir a maturidade e qualidade do produto. O diferencial de um software de monitoramento de sistemas e de análise de dados está na qualidade e quantidade de informação que pode ser obtida a partir de um dado. Além disso, a capacidade do sistema de gerar respostas em tempo real também é importante, pois possibilita uma ação de correção de determinada falha rapidamente. Dessa forma, são reduzidos significativamente os períodos em que os serviços ficam indisponíveis, outro fator de impacto é o baixo custo de operação do software.

2.2 DETECÇÃO DE ANOMALIAS

A detecção de anomalia é também conhecida como detecção de *outilier*, detecção de novidade, detecção de desvios e mineração de exceções, sendo possível determinar quais instâncias se destacam como sendo diferentes para todos os outros dados do conjunto **[5]**. Existem várias maneiras de se definir o que vem a ser anomalia, uma definição amplamente aceita é a (Hawkins, 1980): "Uma anomalia é uma observação que desvia demasiadamente das outras, gerando suspeitas quanto ao mecanismo pela qual foi criada" **[6].** As anomalias podem ser

DOI: 10.25286/repa.v6i5.2152

categorizadas em três tipos: Anomalia Pontual, Anomalia Contextual e Anomalia Coletiva.

As anomalias são consideradas importantes, pois indicam eventos significativos, que podem estar relacionados a erros, solicitando que alguma ação pode ser tomada em diferentes tipos de aplicação, como por exemplo: um padrão de tráfego incomum em uma rede pode significar que um computador foi hackeado e os dados são transmitidos para destinos não autorizados; comportamento anômalo no cartão de crédito transações podem indicar atividades fraudulentas e uma anomalia em uma imagem de ressonância magnética pode indicar a presença de um tumor maligno, a detecção de anomalias tem sido utilizada em inúmeros domínios de aplicação, como detecção de fraude, detecção de intrusão, processamento de imagem, comportamento robótico e instabilidade em redes **[7]**.

2.2.1 Isolation Forest

Isolation Forest (iForest) **[12]** é um método de detecção de anomalias inspirado no algoritmo de *Random Forest* **[13]**. O IForest apresenta desempenho equiparável com o estado da arte em diversas aplicações quando se trata de detecção de anomalias **[14]**. O processo se trata basicamente de tentar separar as novas observações dos dados já existentes.

A premissa básica deste algoritmo é que as anomalias são poucas e destoam muito dos demais elementos e, por isso, podem ser separados pelo processo de isolamento, que ocorre a partir da geração de uma árvore. Nesta, cada folha é uma observação e cada nó interno é um recorte aleatório de dados. O iForest é uma abordagem baseada em um ensemble **[15]** (variadas técnicas trabalhando em conjunto), na qual o processo de isolação de elementos é repetido iterativamente, gerando diferentes configurações de árvores. Após isso, os outliers são identificados tomando como base o custo para que determinado elemento seja isolado dos demais, basicamente observando a distância entre a raiz da árvore para cada nova observação (folha).

2.3 TRABALHOS RELACIONADOS

Para o contexto de detecção de anomalias, alguns trabalhos propuseram o uso de Redes Neurais para a solução do problema. Por exemplo, Andy Brown *et al.* **[8]** utilizaram uma Rede Neural Recorrente para detecção de anomalias a partir de logs do sistema. Um dos objetivos deste trabalho é atingir um nível de interpretabilidade não presente nas redes tradicionais. Por conta das características da arquitetura utilizada, também foi proposto a modelagem de uma linguagem a partir da criação de tokens para cada item contido nas linhas do log. A proposta dos autores foi capaz de atingir os resultados do estado da arte possibilitando uma interpretação dos resultados, evidenciando as *features* mais importantes para a tomada de decisão.

Gian Antonio Susto, *et al.* **[9]** utiliza a técnica de detecção de anomalia *Isolation Forest* em um conjunto de dados industrial real relacionado a *Plasma Etching* (Gravura de Plasma), através de um monitoramento de sistemas, explorando dados de Espectroscopia Óptica. Os resultados obtidos pelo *Isolation Forest* tiveram os melhores resultados quando comparados com outros métodos de detecção de anomalias, obtendo um desempenho de 82.84% de *precision* e 92.50% de *recall* com a técnica *Isolation Forest*.

No trabalho desenvolvido por Minghua *et al.* **[4]** mostra que a detecção de anomalias é crítica para sistemas de software baseados na web, apresentando uma adaptação rápida e robusta para *Concept Drift* na detecção de anomalias do sistema de software, nele é levantado uma discussão sobre a precisão de um método de detecção de anomalias em um sistema que pode se degradar com o tempo, pois um método depende da distribuição de dados, que por sua vez pode ser mudado com a evolução do sistema em questão, que é o chamado desvio de conceito. Os autores apresentam uma estrutura *StepWise*, capaz de detectar o desvio de conceito sem ajuste de limite de detecção ou por KPI (Indicador Chave de Desempenho), ajudando o algoritmo de detecção de anomalias para lidar com isso rapidamente, o trabalho mostra que o *StepWise* melhora a pontuação F média em 206% para muitos detectores de anomalias amplamente utilizados em uma linha de base.

Por fim, Li, X. *et al.* **[10]** propuseram uma abordagem online com ênfase em técnicas de *data augmentation*, enquanto lidam com a predição de anomalia. Essa abordagem foi utilizada com o intuito de conseguir alta acurácia mesmo em *datasets* pequenos, ainda foi observado que estas técnicas funcionam melhor quando usadas combinadas.

3 MATERIAIS E MÉTODOS

Para este projeto foi utilizada a metodologia CRISP-DM de modo que o modelo proposto seja coerente com o problema **[11]**.

3.1 DESCRIÇÃO DA BASE DE DADOS

Para o desenvolvimento deste projeto foi utilizada uma base de dados disponibilizada pela Accenture, contendo informações acerca de dados coletados durante a execução de *jobs* nos ambientes dos clientes.

Os dados estão contidos em uma planilha que contém informações de datas, duração do *jobs* e status com relação à execução. A planilha tem um conteúdo de 491.459 elementos que estão compreendidos no período de 01/01/2020 até 24/05/2021, um total de 509 dias de execução, estes elementos são divididos em de três cadeias principais processos, distintas umas das outras, um nível abaixo, encontram-se as subcadeias, que possuem as instâncias mais atômicas da base de dados, os processos. Para cada um destes, um conjunto de *features* é associado. Todas essas particularidades estão listadas no Quadro 1.

Quadro 1- Descrição das *features* da base de dados.

VARIÁVEL	DESCRIÇÃO	TIPO
MAINCHAIN	Cadeia principal de processos	Texto
SUBCHAIN	Subcadeia criada a partir da cadeia principal	Texto
TYPE	Classificação do processo	Texto
STATE	Status da execução	Texto
ACTUAL_STATE	Último status da execução	Texto
RUNTIME	Tempo de execução do processo	Inteiro
LINESREAD	Dados lidos na execução	Inteiro
LINESTRANSFERRED	Dados transferidos na execução	Inteiro
START/END DATE	Intervalo de execução do processo	Datetime
FULL_DATE	Data do início do processo	Datetime
HOUR_INTERVAL	Caracterização dos intervalos de hora	Inteiro

Fonte: Os autores.

Para este *dataset*, a análise para detecção de anomalias deve ser feita para o segundo nível mais detalhado dos dados, as SUBCHAINs. Isso se dá pelo fato de um processo isolado não conseguir representar o comportamento de toda uma cadeia, da mesma forma que ele não carrega o nível de criticidade operacional do conjunto completo. Para as variáveis STATE e ACTUAL_STATE algumas codificações são adotadas, conforme Quadro 2.

Quadro 2 – Descrição dos códigos da tabela RSPC.

CÓDIGO	DESCRIÇÃO
R	Finalizou com erros
G	Finalizou com sucesso
F	Concluída
A	Ativa
X	Cancelada
P	Agendada
S	Ignorado na reinicialização
Q	Liberada
Y	Pronta
J	Finalizou com erro

Fonte: Os autores.

3.2 ANÁLISE DESCRITIVA DOS DADOS

Com o objetivo de obter um melhor entendimento do problema em questão e da distribuição dos dados, foi necessária a execução de uma análise descritiva. A partir desta, alguns fatores importantes foram notados por meio da visualização dos dados, principalmente no que tange ao comportamento de algumas cadeias.

Para o problema abordado neste trabalho, um desafio surge quando a análise das anomalias é voltada somente para os códigos de execução, que é o alto nível de desbalanceamento desses dados,

DOI: 10.25286/repa.v6i5.2152

como visto na distribuição de frequência do Quadro 3, aqueles considerados erros, ou anomalias (códigos J, X e R), somam, originalmente, 197 elementos, cerca de 0,04% da base de dados, o que pode gerar viés durante a modelagem do problema.

Quadro 3 – Distribuição de Frequência em relação as codificações do ACTUAL_STATE: frequência absoluta (FA), frequência relativa (FR), frequência absoluta acumulada (FAC), acumulado (AC).

VALOR	FA	FR (%)	FAC (%)	AC
A	0	0.0000	0.0000	0
Q	0	0.0000	0.0000	0
P	0	0.0000	0.0000	0
Y	0	0.0000	0.0000	0
J	10	0.0020	0.0020	10
X	40	0.0081	0.0102	50
R	147	0.0299	0.0400	197
S	289	0.0588	0.0989	486
F	52715	10.7262	10.8251	53201
G	438258	89.1749	100.0000	491459

Fonte: Os autores.

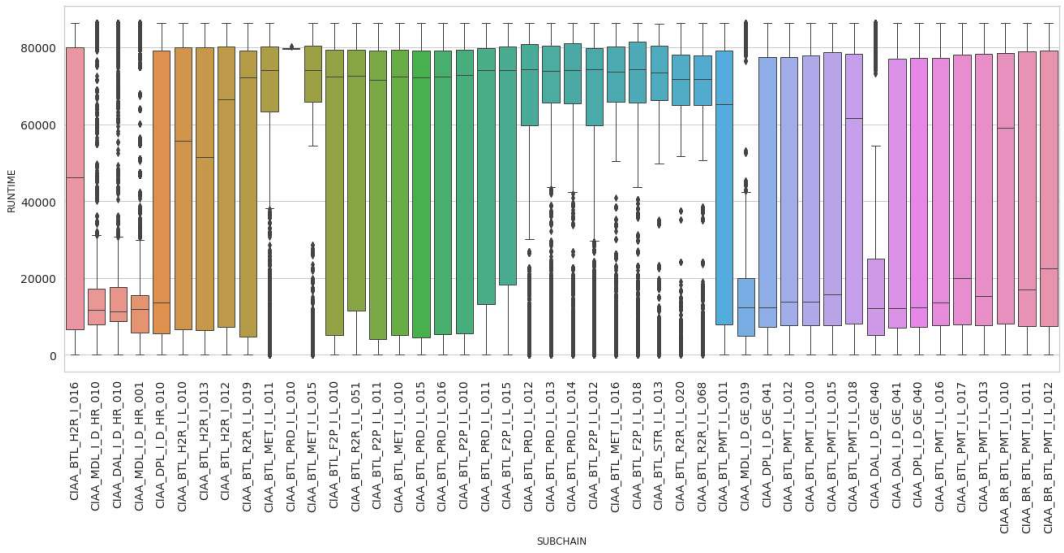
Além disso, alguns elementos do *dataset* demandam análise individual, que são as SUBCHAINs, por não apresentarem correlação entre si. Portanto, junto à necessidade de avaliar os grupos de processo individualmente, para a solução dessa problemática, a interpretação dos dados precisa considerar outras *features*, e não somente os resultados das execuções. Assim, estendemos nossa análise também para o RUNTIME.

Dessa forma, o primeiro passo a ser tomado é a identificação de cadeias problemáticas, tomando como base tempo de execução e códigos de execução. Para o primeiro item, foi gerado o box-plot para cada SUBCHAIN, que pode ser visto na Figura 1. Em um primeiro momento, é possível notar que cinco cadeias se destacam quando se trata de fuga de padrão de comportamento no tempo de execução, sendo elas:

- CIAA_MDL_I_D_HR_010
- CIAA_DAL_I_D_HR_010
- CIAA_MDL_I_D_HR_001
- CIAA_MDL_I_D_GE_019
- CIAA_DAL_I_D_GE_040.

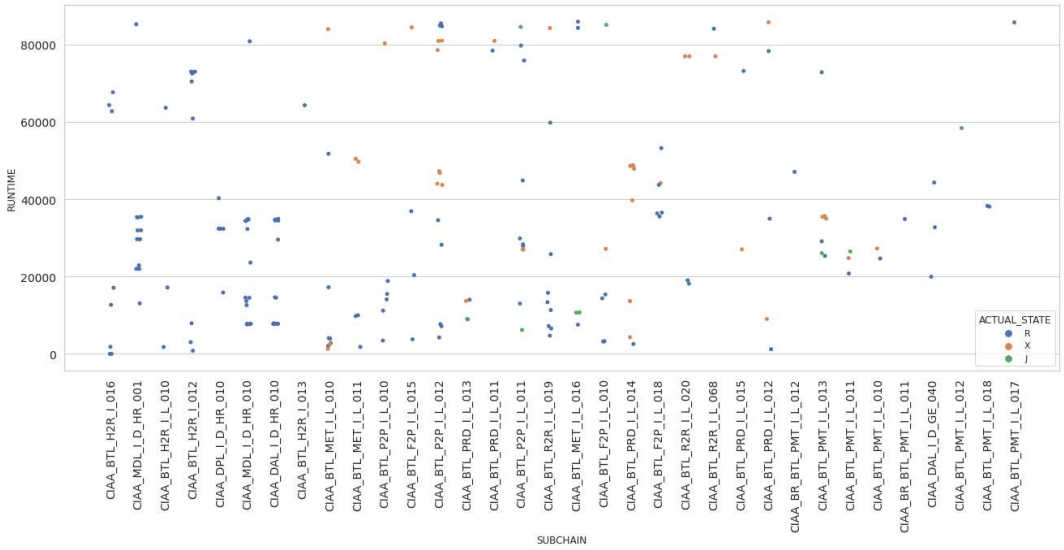
Por fim, corroborando com o que foi dito, na Figura 2 é possível observar que os códigos de execução relacionados aos erros não são capazes de representar a saúde de uma determinada cadeia de processos, principalmente pelo fato da maioria destes não estar inseridos nos conjuntos de anomalias, o que, consequentemente, não caracterizaria uma anomalia, criando um provável viés na interpretação dos dados.

Figura 1 – Boxplot da relação entre RUNTIME e SUBCHAIN.



Fonte: Os autores.

Figura 2 – Gráfico de Dispersão dos considerados erros com relação ao RUNTIME de cada SUBCHAIN.



Fonte: Os autores.

3.3 PRÉ-PROCESSAMENTO DOS DADOS

Nesta seção são expostas as etapas de pré-processamento aplicadas à base de dados utilizada. Com o objetivo de aumentar a interpretabilidade dos dados, algumas técnicas foram utilizadas, como redução vertical, padronização e categorização.

Em um primeiro momento foi utilizado a redução vertical para remover algumas *features* que continham valores corrompidos ou não aplicáveis ao problema, afinal, estes não teriam usabilidade na construção da solução deste trabalho.

Erros de formatação foram detectados, principalmente no que tange às datas de início e término dos processos, dessa forma utilizamos a padronização para manter a consistência de dados que se referem às datas.

Para aprimorar a capacidade de predição do modelo e superar o problema de desbalanceamento da base, foram feitas transformações baseadas na distribuição dos elementos da cadeia, categorizando-os entre erros e não erros.

Para uma melhor análise dos dados, foram criadas algumas *features* utilizando dados já existentes, listadas a seguir:

- **RUNTIME:** criada com o intuito de ter em uma *feature* com o tempo, em segundos, de execução da cadeia de processo.
- **FULL_DATE:** esta entrada é apenas a data inicial com uma formatação diferente, com o intuito de simplificar a leitura dos dados.
- **HAS_ERROR:** é uma *feature* concebida para abstrair códigos de erros do *dataset* original.

3.4 METODOLOGIA EXPERIMENTAL

Como apontado na análise descritiva, durante a visualização dos dados, foram encontradas cadeias de processos que apresentavam distorções nos tempos de execução (*outliers*), desta forma, essas SUBCHAINs foram selecionadas para a execução do conjunto de experimentos, uma vez que as demais apresentaram normalidade quanto ao comportamento dos processos.

Para cada cadeia de processo foi treinado um modelo de Isolation Forest, uma vez que esses conjuntos de processos são independentes, não é possível gerar ou atribuir uma correlação entre eles. Além disso, para cada cadeia de processo, foram feitas alterações nas *features* que alimentam o modelo, totalizando cinco combinações de input diferente, listadas a seguir:

- "rt": RUNTIME
- "lr": LINESREAD
- "lt": LINESTRANSFERRED
- "lrlt": LINESREAD, LINESTRANSFERRED
- "all": LINESREAD, LINESTRANSFERRED e RUNTIME

Ainda se tratando dos experimentos, cada modelo foi utilizado um *split* de 80% dos dados para treino e os 20% restantes para testes. Como

DOI: 10.25286/repa.v6i5.2152

já apontado anteriormente, a variável de maior valor para detecção de anomalias é a RUNTIME, portanto, esta será utilizada para como input para o modelo.

Além disso, com o objetivo de garantir a consistência dos resultados, todos os experimentos foram executados 30 vezes, aplicando variações nas amostras. Por fim, para avaliação e consolidação dos resultados, foram utilizadas as principais métricas para problemas de classificação: *precision, recall* e *f1-score*. Para efeitos de visualização, foram geradas matrizes de confusão para cada um dos modelos.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

4.1 RESULTADOS

De modo geral, para cada cadeia, um modelo apresentou boa capacidade na tarefa de detecção de anomalia, como dito anteriormente, para cada SUBCHAIN foram treinados cinco modelos diferentes ("rt", "lr", "lt", "lrlt", "all"), os resultados para cada cadeia de processos estão compilados a seguir, CIAA_MDL_I_D_HR_010 na Tabela 1, CIAA_DAL_I_D_HR_010 na Tabela 2, CIAA_MDL_I_D_HR_001 na Tabela 3, CIAA_MDL_I_D_GE_019 na Tabela 4 e CIAA_DAL_I_D_GE_040 na Tabela 5. Nestas foram registradas as métricas de avaliação e tempo necessário para treino de cada modelo, ambas contando com valor médio e intervalo de confiança.

Tabela 1 – Resultados dos experimentos para a SUBCHAIN "CIAA_MDL_I_D_HR_010".

Model	Classe	Precision	Recall	F1-score	Acurácia	Treinamento (s)
rt	**normal**	**0.9294 ± 0.0541**	**0.9959 ± 0.0083**	**0.9613 ± 0.0298**	**0.9597 ± 0.0324**	0.6558 ± 0.0089
	erro	**0.9955 ± 0.0090**	**0.9234 ± 0.0636**	**0.9578 ± 0.0354**		
lr	normal	0.5028 ± 0.0136	0.9285 ± 0.0332	0.6523 ± 0.0175	0.5052 ± 0.0252	0.5502 ± 0.0098
	erro	0.5327 ± 0.1728	0.0818 ± 0.0408	0.1413 ± 0.0651		
lt	normal	0.5083 ± 0.0148	0.9310 ± 0.0288	0.6575 ± 0.0173	0.5151 ± 0.0269	0.5504 ± 0.0094
	erro	0.5865 ± 0.1582	0.0991 ± 0.0450	0.1690 ± 0.0704		
lrlt	normal	0.5040 ± 0.0146	0.9301 ± 0.0316	0.6537 ± 0.0183	0.5074 ± 0.0268	0.5793 ± 0.0085
	erro	0.5462 ± 0.1786	0.0846 ± 0.0401	0.1461 ± 0.0642		
all	normal	0.6025 ± 0.0498	0.9497 ± 0.0353	0.7370 ± 0.0430	0.6605 ± 0.0681	**0.5492 ± 0.0114**
	erro	0.8780 ± 0.0966	0.3713 ± 0.1216	0.5197 ± 0.1312		

Fonte: Os autores.

Tabela 2 – Resultados dos experimentos para a SUBCHAIN "CIAA_DAL_I_D_HR_010".

Model	Classe	Precision	Recall	F1-score	Acurácia	Treinamento (s)
rt	**normal**	**0.9875 ± 0.0233**	**0.9991 ± 0.0067**	**0.9932 ± 0.0123**	**0.9932 ± 0.0125**	0.5587 ± 0.0113
	erro	**0.9991 ± 0.0067**	**0.9872 ± 0.0239**	**0.9931 ± 0.0127**		
lr	normal	0.4981 ± 0.0175	0.9621 ± 0.0489	0.6563 ± 0.0251	0.4963 ± 0.0334	**0.4766 ± 0.0124**
	erro	0.4742 ± 0.5326	0.0306 ± 0.0414	0.0566 ± 0.0740		
lt	normal	0.4982 ± 0.0166	0.9644 ± 0.0463	0.6570 ± 0.0238	0.4966 ± 0.0319	0.4773 ± 0.0102
	erro	0.4799 ± 0.5641	0.0288 ± 0.0394	0.0535 ± 0.0707		
lrlt	normal	0.4980 ± 0.0176	0.9621 ± 0.0463	0.6562 ± 0.0247	0.4961 ± 0.0337	0.5035 ± 0.0095
	erro	0.4670 ± 0.5424	0.0301 ± 0.0418	0.0558 ± 0.0750		
all	normal	0.5727 ± 0.0301	0.9694 ± 0.0458	0.7199 ± 0.0304	0.6226 ± 0.0459	0.4789 ± 0.0079
	erro	0.9038 ± 0.1267	0.2758 ± 0.0822	0.4207 ± 0.0979		

Fonte: Os autores.

Tabela 3 – Resultados dos experimentos para a SUBCHAIN "CIAA_MDL_I_D_HR_001".

Model	Classe	Precision	Recall	F1-score	Acurácia	Treinamento (s)
rt	**normal**	**0.7579 ± 0.0993**	**0.9507 ± 0.0287**	**0.8428 ± 0.0720**	**0.8212 ± 0.0922**	1.3606 ± 0.0210
	erro	**0.9313 ± 0.0509**	**0.6918 ± 0.1587**	**0.7922 ± 0.1233**		
lr	normal	0.5287 ± 0.0086	0.8761 ± 0.0217	0.6594 ± 0.0104	0.5475 ± 0.0138	**1.1256 ± 0.0114**
	erro	0.6387 ± 0.0400	0.2188 ± 0.0273	0.3258 ± 0.0321		
lt	normal	0.5434 ± 0.0096	0.8812 ± 0.0202	0.6722 ± 0.0107	0.5704 ± 0.0148	1.1354 ± 0.0160
	erro	0.6861 ± 0.0371	0.2596 ± 0.0285	0.3764 ± 0.0323		
lrlt	normal	0.5290 ± 0.0094	0.8765 ± 0.0201	0.6598 ± 0.0101	0.5480 ± 0.0150	1.2125 ± 0.0211
	erro	0.6399 ± 0.0397	0.2196 ± 0.0314	0.3267 ± 0.0372		
all	normal	0.6592 ± 0.0266	0.9252 ± 0.0209	0.7698 ± 0.0220	0.7232 ± 0.0308	1.1378 ± 0.0174
	erro	0.8744 ± 0.0346	0.5212 ± 0.0539	0.6528 ± 0.0482		

Fonte: Os autores.

Tabela 4 – Resultados dos experimentos para a SUBCHAIN "CIAA_MDL_I_D_GE_019".

Model	Classe	Precision	Recall	F1-score	Acurácia	Treinamento (s)
rt	**normal**	**0.7838 ± 0.0812**	**0.9435 ± 0.0683**	**0.8555 ± 0.0581**	**0.8401 ± 0.0690**	0.2034 ± 0.0031
	erro	**0.9301 ± 0.0799**	**0.7367 ± 0.1215**	**0.8206 ± 0.0853**		
lr	normal	0.5316 ± 0.0361	0.8367 ± 0.0918	0.6498 ± 0.0491	0.5497 ± 0.0563	**0.1874 ± 0.0036**
	erro	0.6198 ± 0.1357	0.2626 ± 0.0881	0.3668 ± 0.0999		
lt	normal	0.5466 ± 0.0404	0.8476 ± 0.0875	0.6643 ± 0.0507	0.5721 ± 0.0620	0.1880 ± 0.0034
	erro	0.6631 ± 0.1423	0.2966 ± 0.0939	0.4079 ± 0.1069		
lrlt	normal	0.5351 ± 0.0393	0.8422 ± 0.0888	0.6541 ± 0.0505	0.5551 ± 0.0612	0.1919 ± 0.0037
	erro	0.6307 ± 0.1424	0.2680 ± 0.0962	0.3742 ± 0.1125		
all	normal	0.6464 ± 0.0651	0.8850 ± 0.0836	0.7466 ± 0.0609	0.6993 ± 0.0757	0.1907 ± 0.0145
	erro	0.8192 ± 0.1155	0.5136 ± 0.1227	0.6292 ± 0.1070		

Fonte: Os autores.

Tabela 5 – Resultados dos experimentos para a SUBCHAIN "CIAA_DAL_I_D_GE_040".

Model	Classe	Precision	Recall	F1-score	Acurácia	Treinamento (s)
rt	**normal**	**0.7527 ± 0.0490**	**0.9206 ± 0.0273**	**0.8281 ± 0.0365**	**0.8085 ± 0.0453**	0.5854 ± 0.0095
	erro	**0.8974 ± 0.0376**	**0.6964 ± 0.0757**	**0.7839 ± 0.0582**		
lr	normal	0.5287 ± 0.0155	0.8082 ± 0.0437	0.6391 ± 0.0215	0.5438 ± 0.0237	**0.4884 ± 0.0089**
	erro	0.5935 ± 0.0526	0.2794 ± 0.0434	0.3794 ± 0.0445		
lt	normal	0.5441 ± 0.0155	0.8180 ± 0.0426	0.6535 ± 0.0217	0.5663 ± 0.0234	0.4912 ± 0.0138
	erro	0.6344 ± 0.0531	0.3146 ± 0.0380	0.4202 ± 0.0377		
lrlt	normal	0.5356 ± 0.0175	0.8141 ± 0.0420	0.6461 ± 0.0235	0.5541 ± 0.0267	0.5205 ± 0.0105
	erro	0.6134 ± 0.0587	0.2942 ± 0.0389	0.3973 ± 0.0422		
all	normal	0.6142 ± 0.0371	0.8554 ± 0.0427	0.7148 ± 0.0336	0.6585 ± 0.0453	0.4905 ± 0.0170
	erro	0.7614 ± 0.0617	0.4617 ± 0.0783	0.5741 ± 0.0717		

Fonte: Os autores.

DOI: 10.25286/repa.v6i5.2152

4.2 DISCUSSÃO

Como dito anteriormente, para todas as SUBCHAINs ao menos um modelo atingiu boa generalização, porém um comportamento específico se repetiu em todos os experimentos, que se dá ao tentar utilizar um conjunto de *features* como entrada do modelo, o que ocorre para os modelos "all" e "lrlt", mas, isto é justificável quando se analisa a correlação entre as variáveis contínuas do *dataset*, presente na Figura 3. A partir disto, percebe-se que LINESREAD e LINESTRANSFERED não apresentam correlação com o RUNTIME de um determinado processo, que é a variável central da solução deste trabalho, afinal, a partir dela é possível identificar quando um processo ou cadeia de processos apresenta um comportamento anômalo.

Figura 3 – Correlação entre as variáveis continuas.

Fonte: Os autores.

Dessa forma, os resultados obtidos são justificáveis a partir deste viés, como pode ser percebido nas tabelas de resultados, o modelo "rt", em todos experimentos, se destacou quanto a capacidade de classificação e detecção dos processos que apresentavam anomalias.

Além disso, em todos experimentos, algo importante para ambientes de produção com alta variabilidade de dados, basicamente os contextos de aplicação real. Afinal, nos casos de *data* e *concept drift*, modelos com a capacidade de treinar rapidamente, possibilitam a adaptação às mudanças das características dos dados.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como objetivo validar a criação de uma técnica de detecção de anomalias para ambiente de monitoramento, identificando potenciais problemas que podem surgir durante a execução de determinadas cadeias de processos. Para os experimentos, foi utilizada uma base de dados com informações de ambientes de produção real, disponibilizada pela empresa Accenture.

Após um conjunto de análises, foi notado que os dados que, originalmente, eram associados a erros no dataset não representavam o comportamento dos processos, foi preciso efetuar uma etapa de *feature engineering* nos dados, e, com isso, identificar as cadeias passíveis de análise e processos anômalos. Para modelar a solução, foram utilizadas *Isolation Forest* para identificação de outliers.

A limitação deste trabalho se deu por conta de uma característica própria do *dataset*, no qual os dados de interesse não apresentavam correlação entre si. Porém, durante os experimentos foi notada boa capacidade de generalização para a detecção de anomalias nos processos, algo benéfico para aplicações críticas, como ocorre no estudo de caso presente neste trabalho.

Como trabalhos futuros, têm-se a criação de uma métrica para quantificar o grau de confiança de anomalia para dada cadeia. Há ainda o possível emprego de um método de aprendizado online com o intuito de lidar com o *concept drift* que possa vir a ocorrer com a geração de novos dados.

REFERÊNCIAS

[1] SANTO LONGO, Claudio et al. **Big Data for advanced monitoring system: an approach to manage system complexity**. In: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). IEEE, 2018. p. 341-346.

[2] ETRO, Federico. **The economics of cloud computing**. In: Cloud Technology: Concepts, Methodologies, Tools, and Applications. IGI Global, 2015. p. 2135-2148.

[3] WORLD HEALTH ORGANIZATION et al. **Monitoring the building blocks of health systems: a handbook of indicators and**

their measurement strategies. World Health Organization, 2010.

[4] MA, Minghua et al. **Robust and rapid adaption for concept drift in software system anomaly detection**. In: 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2018. p. 13-24.

[5] CHALAPATHY, Raghavendra; CHAWLA, Sanjay. **Deep learning for anomaly detection: A survey**. arXiv preprint arXiv:1901.03407, 2019.

[6] AHMED, Mohiuddin; MAHMOOD, Abdun Naser; HU, Jiankun. **A survey of network anomaly detection techniques**. Journal of Network and Computer Applications, v. 60, p. 19-31, 2016.

[7] AHMED, Mohiuddin; MAHMOOD, Abdun Naser; ISLAM, Md Rafiqul. **A survey of anomaly detection techniques in financial domain**. Future Generation Computer Systems, v. 55, p. 278-288, 2016.

[8] BROWN, Andy et al. **Recurrent neural network attention mechanisms for interpretable system log anomaly detection**. In: Proceedings of the First Workshop on Machine Learning for Computing Systems. 2018. p. 1-8.

[9] SUSTO, Gian Antonio; BEGHI, Alessandro; MCLOONE, Seán. **Anomaly detection through on-line isolation forest: An application to plasma etching**. In: 2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC). IEEE, 2017. p. 89-94.

[10] LI, Xiang et al. **Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation**. Journal of Intelligent Manufacturing, v. 31, n. 2, p. 433-452, 2020.

[11] Christoph Schröer, Felix Kruse, Jorge Marx Gómez, **A Systematic Literature Review on Applying CRISP-DM Process Model**, Procedia Computer Science, Volume 181, 2021, Pages 526-534,

[12] LIU, Fei Tony; TING, Kai Ming; ZHOU, Zhi-Hua. **Isolation forest**. In: 2008 eighth ieee international conference on data mining. IEEE, 2008. p. 413-422.

[13] SVETNIK, Vladimir et al. **Random forest: a classification and regression tool for compound classification and QSAR modeling**. Journal of chemical information and computer sciences, v. 43, n. 6, p. 1947-1958, 2003.

[14] ZHANG, Daqing et al. **iBAT: detecting anomalous taxi trajectories from GPS traces**. In: Proceedings of the 13th international conference on Ubiquitous computing. 2011. p. 99-108.

[15] BREUNIG, Markus M. et al. **LOF: identifying density-based local outliers.** In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. 2000. p. 93-104.

DOI: 10.25286/repa.v6i5.2152