

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DAINF - DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
ESPECIALIZAÇÃO EM CIÊNCIAS DE DADOS**

JONAS OLIVEIRA DE SOUZA

DETECÇÃO DE OUTLIERS EM PIPELINES DE DADOS

CURITIBA / PR

2021

JONAS OLIVEIRA DE SOUZA

DETECÇÃO DE OUTLIERS EM PIPELINES DE DADOS

OUTLIERS DETECTION IN DATA PIPELINES

Monografia apresentada como requisito parcial à obtenção do título de Especialista em Ciência de Dados e suas Aplicações, do DAINF - Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador(a): Prof. Dr. Marco A. Wehrmeister.

CURITIBA

2021



[4.0 Internacional](https://creativecommons.org/licenses/by-nc-sa/4.0/)

Insira aqui a nota explicativa da licença *Creative Commons* regulamentada pelo curso/programa. Folha de Rosto: <http://portal.utfpr.edu.br/biblioteca/trabalhos-academicos>. Antes de baixar o modelo, certifique-se da licença adotada pelo Curso de Graduação ou Programa de Pós-Graduação *Stricto Sensu* no qual o trabalho foi defendido. Você pode consultar esta informação na página do Curso/Programa. Atualizar o logo e o *link* (ao lado) para o acesso à página da licença, caso necessário.



Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
UTFPR - CAMPUS CURITIBA
DIRETORIA-GERAL - CAMPUS CURITIBA
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO - CAMPUS CURITIBA
DEPARTAMENTO DE APOIO DAS ESPECIALIZAÇÕES LATO-SENSU DOS
CURSOS DE INFORMÁTICA - CAMPUS CURITIBA
CURSO DE ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS E SUAS APLICAÇÕES



TERMO DE APROVAÇÃO

DETECÇÃO DE OUTLIERS EM PIPELINES DE DADOS.

por

Jonas Oliveira De Souza

Este Trabalho de Conclusão de Curso foi apresentado às 20h30min do dia 11 de agosto de 2021 por videoconferência como requisito parcial à obtenção do grau de Especialista em Ciência de Dados e suas Aplicações na Universidade Tecnológica Federal do Paraná - UTFPR - Campus Curitiba. O aluno foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

Prof. Dr. Marco Aurelio Wehrmeister (Presidente/Orientador – DAINF-CT/ UTFPR-CT)

Prof. Marcelo de Oliveira Rosa (Avaliador 1– DAELT-CT/UTFPR-CT)

Profa. Dra. Rita Cris na Galarraga Berardi (Avaliadora 2 – DAINF-CT/ UTFPR-CT)

O Termo de Aprovação assinado encontra-se no sistema SEI- Processo nº 23064.031963/2021-49

AGRADECIMENTOS

A gratidão alimenta nossa humildade, e faz com que pessoas se tornem inesquecíveis em nossas vidas. Nesta etapa da minha vida que se conclui, os agradecimentos são inúmeros.

Inicialmente agradeço minha família que me incentivou, e me ajudou a concluir o curso. A minha mãe que infelizmente não está mais entre nós, mas que com toda certeza estaria toda orgulhosa.

Agradeço aos professores do Curso, que compartilharam seus conhecimentos, e que durante a pandemia se adaptaram rapidamente, mantendo a mesma qualidade das aulas presenciais.

Agradeço ao meu orientador Prof. Dr. Marco A. Wehrmeister, que com paciência, com críticas construtivas e um curto espaço de tempo, fez com que pudesse concluir a pesquisa.

Agradeço a todos os colegas de sala que contribuíram para o enriquecimento das aulas, compartilhando suas dúvidas, experiências e opiniões.

Agradeço a coordenação do Curso, pela transparência, atenção e acompanhando que foi prestado durante todo o Curso.

RESUMO

Com o avanço tecnológico das últimas décadas, a velocidade e a quantidade de dados gerados aumentaram consideravelmente. A gestão do fluxo dos dados tornou-se relevante para obtenção de informações. *Pipelines* de dados podem ser utilizados para transportar os dados, bem como transforma-los de acordo com cada necessidade. Os *pipelines* de dados são sensíveis a mudanças que podem ocorrer tanto em suas extrações quanto em suas regras de negócio, ocasionando inconsistência de dados. Este estudo busca uma solução para identificar alterações de comportamentos em *pipelines* de dados (*outliers*) através do emprego de *Machine Learning*. Os algoritmos de *Machine Learning* serão aplicados durante processo de carga dos dados, utilizando as métricas dos *pipelines* para identificar as anomalias. O trabalho proposto foi avaliado através de alguns casos de estudos que utilizou um *dataset* público e outro gerado para o contexto do estudo, nomeados, *US Accidents Updated until Dec 2020* e métricas obtidas da máquina local. Para viabilizar o estudo foi criado um ambiente, através do Docker, composto por três containers, um StreamSets para criação de *pipelines* de dados, um banco de dados Mysql para armazenar as métricas dos *pipelines*, outro container com o Jupyter para realizar as análises exploratórias dos dados e realização de testes dos algoritmos. Os algoritmos utilizados para este estudo foram o PCA e o Dbscan. Os resultados obtidos, ainda que em um ambiente simulado, foram satisfatórios apontando os *pipelines* que apresentaram mudanças de comportando.

Palavras-chave: Outlier. Anomalia. Pipeline de Dados. Machine Learning.

ABSTRACT

The technological advances of the last decades allowed a huge increase in the rate and amount of data generated by computing systems. Data flow management has become relevant for obtaining information. Data pipelines are used to transport the data from its source to different sorts of targets, transforming the data itself whether necessary. Data pipelines are sensitive to changes that occur both in the extraction process and in their business rules, leading to data inconsistency. This study proposes a solution to identify behavior changes in data pipelines (outliers), through the use of machine learning. Some machine learning algorithms have been applied during the data loading process by using the pipeline metrics to identify anomalies. The proposed work has been evaluated throughout some case studies that use a publicly available dataset and another created in the context of the study, namely, US Accidents Updated until Dec 2020 and Local Machine Metrics. An environment was created using Docker. It is comprised of three containers: one using StreamSets for creating data pipelines; one using MySQL database to store pipeline metrics, and one using Jupyter to perform exploratory analysis of the data and testing of the algorithms. The algorithms used in this study were PCA and DbSCAN. The obtained results are considered satisfactory since the pipelines that had divergent comportment were correctly identified and reported.

Keywords: Outlier. anomalies. Data Pipeline. Machine Learning.

LISTA DE ILUSTRAÇÕES

Figura 1 - Ilustração fluxo de dados	20
Figura 2 - Streamsets pipeline de dados	21
Figura 3 – Streamsets pipelines de dados criados.....	24
Figura 4 – Arquitetura Processo.....	24
Figura 5 - Unsupervised Anomaly Detection Algorithms.....	25
Figura 6 - Algoritmos de Outliers Globais vs Locais.....	26
Figura 7 - DBSCAN clustering	27
Figura 8 - PCA.....	29
Figura 9 - Processos específicos no pipeline	30
Figura 10 - usacciden7 registros de entrada	39
Figura 11 - Comparação usacciden2 com usacciden7	39
Figura 12 - readmachi métricas	40
Gráfico 1 – Análise de dados Boxplot – Input Records.....	35
Gráfico 2 – Análise de dados Boxplot – Error Records.....	36
Gráfico 3 - Erros encontrados no pipeline usacciden1.....	36
Gráfico 4 - Execuções do Pipeline readmachi4.....	37
Quadro 1 - <i>Métricas da API do Streamsets</i>	31
Tabela 1 - Resultado Anomalias dos Pipelines	38

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

LISTA DE SIGLAS

AED	Análise Exploratória de Dados
API	Application Programming Interface
DBSCAN	Density-based spatial clustering of applications with noise
DIKW	Data-Information-Knowledge-Wisdom
JDBC	Java™ EE Database Connectivity
JVM	Java Virtual Machine
KDD	Knowledge Discovery and Data Mining
PB	Petabytes
PCA	Principal Component Analysis

SUMÁRIO

1 INTRODUÇÃO	13
1.1 OBJETIVOS	16
1.1.1 Objetivo Geral	16
1.1.2 Objetivos Específicos	16
1.2 LIMITAÇÕES DO ESCOPO DO TRABALHO	16
1.3 ORGANIZAÇÃO DO TRABALHO	17
2 MOTIVAÇÃO E JUSTIFICATIVA	18
3 MATERIAIS E MÉTODOS	20
3.1 PIPELINES DE DADOS (STREAMSETS)	20
3.2 CRIAÇÃO DO AMBIENTE	22
3.3 DADOS UTILIZADOS NOS <i>PIPELINES</i>	23
3.4 DESENVOLVIMENTO DOS COMPONENTES	23
3.5 ALGORITMOS PARA DETECÇÃO DE OUTLIERS	25
3.5.1 DBSCAN	26
3.5.2 PCA	28
4 ANÁLISE DOS DADOS	30
4.1 <i>API STREAMSETS</i>	30
4.2 ANÁLISE EXPLORATÓRIA DE DADOS (AED)	33
5 RESULTADOS	38
6 CONSIDERAÇÕES FINAIS	42
REFERÊNCIAS	44
APÊNDICE A - Questionário de Pesquisa	46
APÊNDICE B - Roteiro da Entrevista	48
ANEXO A - Direitos autorais - Lei nº 9.610, de 19 de fevereiro de 1998. Disposições preliminares	50
ANEXO B - Capa do livro: Normas para Elaboração de Trabalhos	52

1 INTRODUÇÃO

A ascensão tecnológica dos últimos anos contribuiu para diversas áreas da sociedade, não só econômica como a social também. Mudou-se a maneira de gerar dados, a velocidade com que são gerados e conseqüentemente, a quantidade com a qual são gerados também cresce. Este cenário ilustra o termo do *Big Data*, que aproximadamente desde os anos 2000 tem ganhado relevância na comunidade científica, em especial na Ciência de Dados, que por sua vez tem contribuído com diversas respostas, seja para definir uma rota de um ponto a outro, ou para indicar que tipo de tratamento pode ser mais eficaz contra uma determinada doença, através dos mais variados métodos desenvolvidos até hoje. Porém para que todos os dados possam trafegar desde sua origem até um ou mais destino, são utilizados os *pipelines* de dados, que além de transportar os dados, também podem transformar estes dados de acordo com a finalidade de cada *pipeline* (STREAMSETS, [s.d.]).

O termo *pipeline* de dados, comum entre os profissionais da Engenharia de Dados, responsáveis normalmente por construir e mantê-los, é composto normalmente por uma origem, um ou mais passos de processamento dos dados da origem, e um ou mais destinos. Dentro de um ambiente corporativo, os *pipelines* podem ser comparados ao transporte rodoviário, aonde as estradas seriam os *pipelines*, interligando diversos lugares, que durante a viagem é possível adquirir mais produtos, ir descarregando até que chegue ao destino final. O desenvolvimento dos *pipelines de dados*, variam conforme a necessidade: por exemplo um *pipeline* de E.T.L (*Extract Transform and Load*) é diferente de um de *streaming*, ambos podendo ser executados dentro de uma mesma ferramenta.

Um ambiente corporativo onde a movimentação de dados forma um complexo ecossistema, várias necessidades surgem, a fim de garantir que toda orquestração dos processos de movimentações de dados, ocorram da melhor forma possível. Como todo software problemas podem ocorrer, gargalos de memória, derrubando o ambiente, publicação de novas funcionalidades mal testadas, problemas nas origens de onde dados são extraídos, entre outros. Processos são definidos de acordo com a realidade corporativa de cada um, afim de evitar tais transtornos. Soluções oferecidas de forma nativa por grandes desenvolvedores de softwares que proporcionam o desenvolvimento dos *pipelines* de dados, oferecem

possibilidades de monitoramento facilitando a vida dos Engenheiros de Dados. Porém conforme o tamanho do ambiente aumenta o desafio de monitorar e garantir que tudo está ocorrendo corretamente, pois as vezes um problema não é capturado por relatórios de acompanhamento dos processos, muitas vezes configurações pontuais precisam ser feitas diretamente em cada *pipeline* de dados, no caso do Streamsets preencher complexos formulários dentro da solução, para que seja gerado alertas de possíveis problemas.

A detecção de anomalias em *pipeline* de dados pode contribuir com um ecossistema de dados corporativo mais saudável, pois ao identificar algo incomum em um pipeline de dados, é possível atuar antes que o problema se agrave. Empresas de diferentes ramos investem grandes montantes na gestão de seus dados, desde uma geração consistente até que estes dados transformem-se em informação, para posteriormente obter sabedoria, conforme ilustra a Hierarquia DIKW (Data-Information-Knowledge-Wisdom) (BELLINGER; CASTRO; MILLS, 2004) para a empresa. No ambiente corporativo é comum escutar de muitos gestores que “os dados são o novo petróleo” (JAVORNIK; NADOH; LANGE, 2019), e assim como construir e manter o processo extração de petróleo é caro, o mesmo aplica-se para os dados.

Através da Ciência de Dados, diversos problemas/atividades das mais variadas áreas de negócio estão sendo resolvidos e otimizadas através de soluções baseadas especialmente em dados, como detecção de fraudes, qualidade de produção, oportunidades de negócios entre outras (FOX, 2018). Buscando efetuar um monitoramento ativo de *pipelines* de dados, o presente estudo através de métodos de identificação de anomalias ou *outliers*, propõe uma análise do comportamento individual de cada *pipeline* ao longo do tempo, sem que seja necessário interromper sua execução ou sobrecarregar o ambiente para o realizar seu monitoramento. O principal objetivo é de mitigar possíveis problemas conforme já citados, durante o tráfego de dados, a partir das métricas fornecidas normalmente por ferramentas do mercado. O emprego de algoritmos de *Machine Learning* pode colaborar com a redução de custos com dados de pouca qualidade, que geram informação incorreta. Uma pesquisa da Gartner (MOORE; GARTNER, 2018), destaca que aproximadamente US\$15 milhões são perdidos devido a dados ruins.

Compreender os padrões de comportamento de um pipeline de dados pode evitar que possíveis problemas sejam levados adiante, como por exemplo registros

duplicados, a não entrega dos dados corretos ao destino correto, um novo valor em uma das colunas do registro, seja por alguma mudança de negócio não prevista para no pipeline de dados, aumentando o volume de registros, um consumo maior de recursos computacionais com processando dados incorretos, faltantes, acarretando em muitas vezes um reprocessamento de todos os dados.

Detecção de anomalias é a identificação de comportamentos que não se enquadram nos padrões identificados (VON EYE; SCHUSTER, 1998). Segundo (HAWKINS, 1980) *outlier* é “uma observação que difere tanto de outras observações que suscita suspeita que tenha sido criada por um mecanismo diferente das outras observações”. Aplicando conceito de Hawkins no cenário de *pipeline* de dados, uma nova observação que está distante das demais observadas, pode ser caracterizada como uma anomalia, porém cabe a ressalva que pode ser um novo padrão do *pipeline* observado. Assim o objetivo do trabalho é facilitar a identificação de possíveis anomalias, para que um time de Engenharia de Dados possa construir mecanismos baseados em métricas dos *pipelines* de dados, que auxiliem no monitoramento e reduza o tempo em que um processo execute de forma indevida, ao identificar que um pipeline apresente divergências em relação ao seu padrão comportamental observado anteriormente.

O estudo de realizado por Guilherme Campos (CAMPOS, 2015) destaca que métodos para identificação de anomalias vem ganhando espaço entre pesquisadores. Com isso surgem novas técnicas para detecção de *outliers*. As técnicas diferenciam-se de acordo com contexto que se propõe a identificar *outliers*. Inicialmente é crucial o entendimento do que pode ser de fato um *outlier*, pois em alguns casos pode ser que ainda não tenha ocorrido um *outlier*. Compreendido o que pode ser um outlier, próximo passo é analisar os dados pois pode ser que o *outlier* já esteja rotulado nos registros, facilitando assim o treinamento do modelo. Finalmente como será a resposta do modelo, por exemplo: um *ranking* com as observações mais ou menos propensas a serem um *outlier*, uma classificação binária indicando se uma observação é ou não um *outlier*. Todos os pontos mencionados influenciam na escolha do algoritmo a ser escolhido para identificar anomalias.

Seguindo os pontos apresentados anteriormente, o trabalho em questão sugere uma solução computacional para identificar outliers com um baixo custo computacional, ainda nos processos de ingestões dos dados, através de

interpretação das métricas dos pipelines, contribuindo para obtenção de informações melhores, e redução de custo com reprocessamentos de dados.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Este trabalho tem como objetivo aplicar métodos e técnica da Ciências de Dados na identificação de anomalias em *pipelines* de dados, de maneira simples e com um baixo custo computacional.

1.1.2 Objetivos Específicos

- Obter métricas de pipelines de dados;
- Compreender qual abordagem para identificação de outliers melhor se aplicam ao problema proposto;
- Treinar um ou mais modelos para tentar identificar anomalias;
- Reportar os resultados obtidos para discussão dos resultados.

1.2 LIMITAÇÕES DO ESCOPO DO TRABALHO

A fim de delimitar a abrangência do trabalho, será utilizada apenas uma ferramenta de *pipeline* de dados. O Streamsets *Data Collector* é um software de código aberto leve, que disponibiliza um conjunto um robusto de APIs, das quais podemos obter as métricas de cada *pipeline*, como por exemplo, volume, período, quantidade de erros, quantidade de processamentos realizados, entre outras (STREAMSETS, [s.d.]).

Os algoritmos para detecção de anomalias foram escolhidos baseados em suas características: simples implementação, não supervisionados e multivariáveis. Não serão discutidas questões matemáticas e metodologias de implementação nos pacotes utilizados do *Python*. Também não é objetivo do estudo comparar diversos algoritmos de detecção de *outliers* entre si.

Os *pipelines* usados neste trabalho não são oriundos de um ambiente real, foram desenvolvidos apenas para gerar métricas de execução, que serão avaliadas posteriormente por algoritmos de *Machine Learning*, com o objetivo de identificar possíveis variações em seus comportamentos. Não foram levadas em conta questões de segurança como autenticação da API do *Streamsets*, configurações de otimização do ambiente, quantidade reduzida de pipelines monitorados, pontos que poderiam alterar a arquitetura proposta, como a adoção um processo paralelo para a captura das métricas da API.

1.3 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está organizado da seguinte maneira:

Capítulo 2 - motivação e justificativa;

Capítulo 3 - explicação sobre as tecnologias e métodos utilizados na análise e desenvolvimento do estudo;

Capítulo 4 – análise das métricas obtidas dos *pipelines*;

Capítulo 5 – apresentação dos resultados do estudo;

Capítulo 6 – considerações finais do estudo.

2 MOTIVAÇÃO E JUSTIFICATIVA

O presente estudo busca colaborar com a Engenharia de Dados, propondo uma abordagem simples de monitoramento de *pipelines* de dados, identificando variações em seus comportamentos por meio do uso algoritmos para detecção de anomalias. O estudo propõe como entrada para os modelos as métricas de cada processo, tornando desnecessária uma análise detalhada de cada metadado utilizado em cada pipeline.

Como mencionado na Introdução deste trabalho, *pipeline* de dados estão cada vez mais presentes nas arquiteturas corporativas. Um estudo conduzido pela *Dimensional Research* em parceria com a *Fivetran* (SURVEY; MARCH, 2021), retrata a dimensão em que os *pipelines* impactam no dia a dia de uma empresa. A pesquisa aponta que 51% dos pipelines falham pelo menos mensalmente e que normalmente se leva um dia de trabalho para repará. Ainda de acordo com a pesquisa, 66% dos participantes relatam que falhas nos *pipelines* de dados afetam a eficiência operacional do negócio. Outro fato relevante demonstrado pela pesquisa de que a satisfação dos clientes é afetada, devido a falhas em pipelines, de acordo 39% dos entrevistados. Outros 25% da pesquisa destacaram perda de receita (vendas perdidas), apenas 3% afirmaram não terem impactos no negócio devido uma falha no *pipeline*. Os principais motivos pelos quais as falhas nos *pipelines* ocorrem, de acordo com 52% dos entrevistados, é devido a mudanças dos metadados, seguido da disponibilidade e da corrupção da origem dos dados, com 51% e 47% respectivamente. Pelos resultados, é nítido o quão crítico os processos de *pipeline* de dados tornaram-se para o negócio, apesar de sua sensibilidade a falhas. O estudo busca auxiliar e fortalecer o processo apontando quais processos estão comportando-se fora do contexto até então observado.

Um estudo conciliando Inteligência Operacional com modelos de *machine learning* (FEDUSHKO; USTYIANOVYCH; GREGUS, 2020) relata que a maior dificuldade foi tratar os dados e realizar ingestão de grandes volumes de dados em tempo real, com o objetivo de identificar problemas antes que ocorram. Com objetivos similares o estudo presente difere-se pelo fato de não precisar tratar dados faltantes, a qualidade dos dados influencia bastante nos modelos de *machine learning*, com o *Streamsets* todas as métricas serão extraídas via API, garantindo

que os dados sejam uniformes, facilitando a implementação para a detecção de outliers. Uma característica interessante do trabalho de Solomia Fedushko é a realização de uma análise preditiva de possíveis falhas, não realizada no estudo atual.

Alguns estudos de detecção de anomalias normalmente consistem em analisar dados transacionais, como por exemplo o serviço de detecção de anomalias na Microsoft (REN et al., 2019): parte da solução para detectar anomalias consiste inicialmente em carregar os dados da origem, informado pelo usuário do serviço. A proposta apresentada aqui não faz uso de dados transacionais das origens e não concorre com o sistema de origem por recursos computacionais.

Conforme já mencionado o estudo de Guilherme Campos (CAMPOS, 2015) destaca o esforço no tratamento das bases de dados, tais como ajustes de atributos faltantes, remoção de dados duplicados, pontos que podem induzir algoritmos a uma interpretação equivocada de *outliers*. No presente estudo não é esperado encontrar dificuldades com os dados analisados para detecção de *outliers*, pois os dados, comuns a todos os pipelines, estão estruturados.

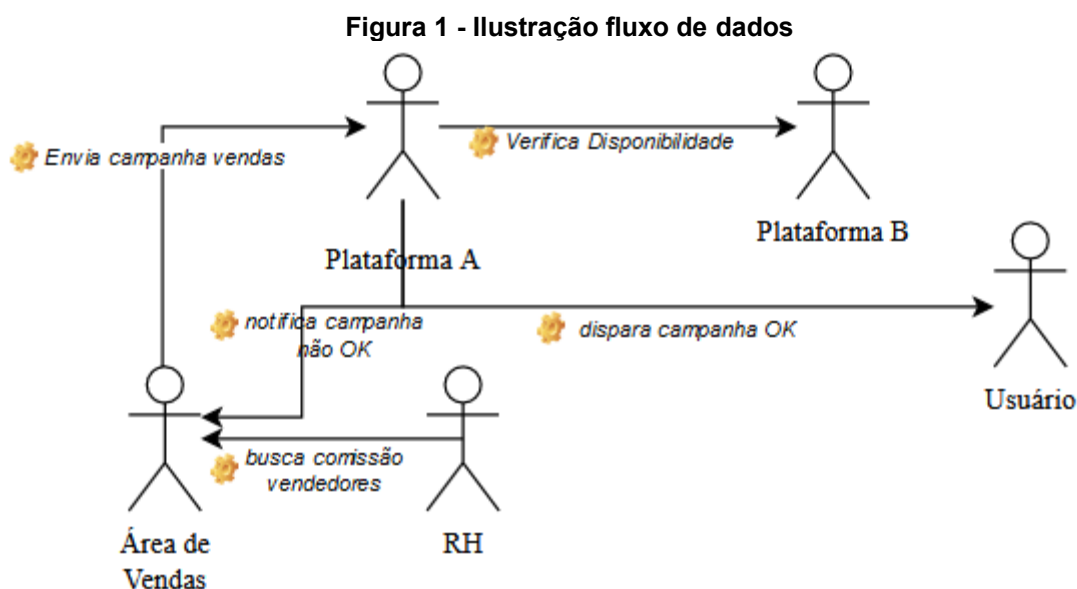
Soluções de *Machine Learning* podem contar com combinações de algoritmos. No presente estudo foram utilizados dois algoritmos, similar a um trabalho desenvolvido para detectar alterações em imagens (MURSI; SALAMA; HABEB, 2017), servindo de inspiração para o presente estudo que conciliou os algoritmos DBSCAN e PCA, por serem algoritmos já implementados em diversas linguagens de programação, em especial no Python, permitindo uma simples implementação de código.

As próprias ferramentas para desenvolvimento de *pipelines* oferecem recursos para monitorar e gerar alertas, como no caso do Streamsets. Porém é necessário configurar manualmente cada *pipeline* do seu ambiente. Uma característica do trabalho apresentado é a abordagem genérica para monitorar os *pipelines*, informando simplesmente o nome do processo que será monitorado.

3 MATERIAIS E MÉTODOS

3.1 PIPELINES DE DADOS (STREAMSETS)

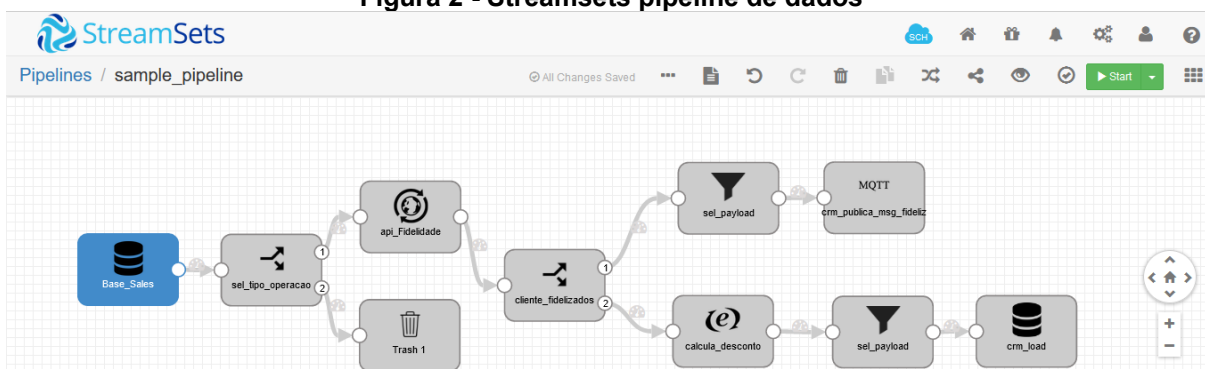
Pipeline de dados, conforme já mencionado, em resumo movimentam dados de diferentes formas, com diferentes objetivos, para algum destino. Seja através de mensagerias, ou através de um pipeline de E.T.L (*Extract Transform and Load*), ou de *Streaming*. Grandes cadeias de conexões com fluxos de dados, tornam desafiador o trabalho da Engenharia de Dados, especialmente o de manter saudável todo o processo. Conforme ilustra a Figura 1, um processo resumido, onde cada engrenagem representa um *pipeline* de dados, que tem como objetivo final disparar campanhas de vendas para diferentes usuários de sua plataforma.



Fonte: Autoria própria

Cada engrenagem poderia ser um processo semelhante ao apresentado na Figura 2, em que um pipeline, desenvolvido no *Streamsets*, executa uma consulta via JDBC, algumas regras de validação são aplicadas, publicando resultados em seus respectivos destinos.

Figura 2 - Streamsets pipeline de dados



Fonte: Autoria própria

O trabalho se propõe a monitorar e apontar anomalias em qualquer *pipeline* na ferramenta, através de métricas obtidas a partir de qualquer *pipeline*. Métricas comuns disponibilizadas por *softwares* como o Streamsets, ou similares são:

- quantidade de *batches* executadas;
 - indica quantas vezes o pipeline processou uma quantidade específica de registros, desde o seu início da execução;
- quantidade de erros no *pipeline*;
 - indica a quantidade de erros que ocorreram no *pipeline*. Alguns *pipelines* são tolerantes a falhas e continuam em execução mesmo apresentando falhas, enquanto outros podem encerrar a execução a qualquer falha;
- quantidade de registro extraídos da origem;
 - indica quantos registros no total foram extraídos da origem desde o início da execução do *pipeline*;
- quantidade de registros publicados nos destinos.
 - Indica quantos registros foram publicados em todos os destinos do *pipeline*. Pipelines com mais de um destino, por exemplo, podem apresentar uma quantidade muito maior de registros publicados: um *pipeline* que extraí 100 registros , por *batch*, e escreve em três tabelas de diferentes sistemas, teria uma quantidade de registros publicados de 300, caso nenhum registro fosse perdido.

O conjunto de métricas mencionadas são comuns para qualquer *pipeline*, no Streamsets. Qualquer *pipeline*, indiferente de seu *design*, pode ser monitorado

através dessas métricas, que por sua vez podem apontar variações no comportamento de cada *pipeline*.

No Streamsets existe a possibilidade de configurar alertas, baseados nas métricas do pipeline, porém são baseados em regras como “Se Quantidade de Erros > 100 então envie um e-mail ou interrompa a *pipeline*”, são úteis, porém precisam ser configuradas uma a uma, sendo necessário conhecer o comportamento do pipeline para definir os alertas. O presente estudo tem como uma das propostas, evitar a configuração individual de cada *pipeline*, para o monitoramento dos pipelines.

3.2 CRIAÇÃO DO AMBIENTE

O ambiente construído no Windows, que por meio do WSL (*Windows Subsystem for Linux*) (MICROSOFT, 2021) permite executar binários e scripts em Linux, necessário para utilizar o Docker. As seguintes imagens foram utilizadas com as seguintes finalidades:

- **jupyter/datascience-notebook:latest** – Executar análise exploratória dos dados e testar os algoritmos de *outliers*.
- **streamsets/datacollector:latest** – Ferramenta de ingestão/*streaming* de dados
- **mysql:latest** – Banco de dados transacional, para utilização nos pipelines e repositório das métricas obtidas.

Customizações mínimas foram realizadas para tornar possível a condução do trabalho:

- integração entre os containers criados no Docker
 - para tornar possível a comunicação dos containers criados a partir das imagens citadas acima. O container do MYSQL foi o primeiro a ser criado, na sequência os demais os containers foram “linkados” entre si:
 - `docker run -p 8888:8888 -d --name jupyter --link mysql jupyter/datascience-notebook`
 - `docker run --restart on-failure -p 18630:18630 -d --name sdc --link mysql --net python_default streamsets/datacollector`

- modo de autenticação do *Streamsets*
 - `http.authentication = "basic"` – Evitar a necessidade de geração de token para utilizar a API da ferramenta, já que é um ambiente sem dados sensíveis, criado apenas para realização do trabalho;
- instalação de pacote python
 - PyOD – um conjunto de métodos para detecção de outliers.

3.3 DADOS UTILIZADOS NOS *PIPELINES*

Os dados utilizados nos *pipelines* são referentes a duas origens apenas:

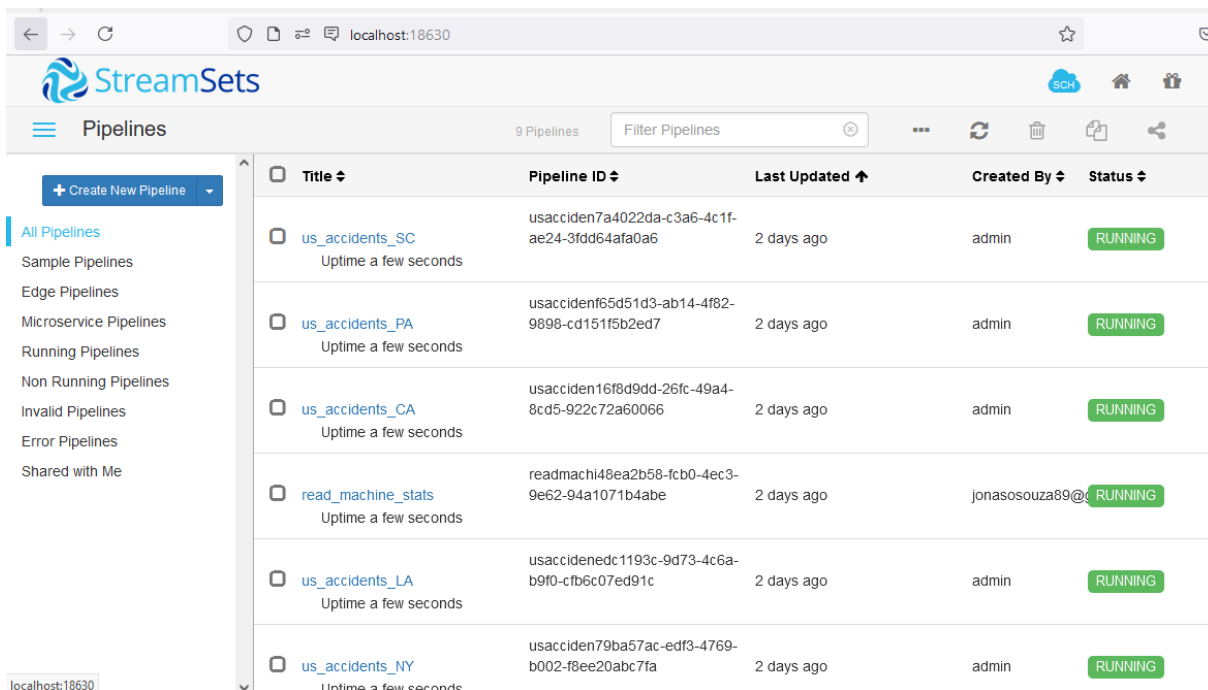
- um csv contendo os acidentes ocorridos no Estados Unidos, entre os anos de 2016 a 2020 (SOBHAN; KAGGLE, 2021). O *dataset* foi selecionado por ser tabular, possuir mais de três milhões de registros, além de possibilitar a criação de alguns *pipelines*, com o intuito aplicar uma regra para que os dados sejam distribuídos entre as unidades federativas, assim obtendo métricas específicas para cada *pipeline*;
- uma tabela criada no Mysql Server, que contém os dados referente ao consumo de memória RAM, Disco e Hardware do computador em que foi simulado o ambiente. O objetivo é ter uma origem de dados contínua, na forma de um *streaming*.

3.4 DESENVOLVIMENTO DOS COMPONENTES

Para processar os dados selecionados para o estudo, detalhados na seção 4.1, foram construídos oito *pipelines* de dados, de baixa complexidade, com o objetivo de gerar métricas heterogêneas, mesmo em um ambiente simulado. Sete *pipelines* executam uma extração de um arquivo referente a acidentes nos ocorridos no EUA, obtidos no Kaggle (SOBHAN; KAGGLE, 2021) e cada *pipeline* separa os acidentes por estados os acidentes e grava em uma tabela do *Mysql*. O *pipeline* restante consome informações de uma base *MYSQL*, contendo as métricas

capturadas durante a execução do servidor que mantem todos os recursos. Em comum todos tem como destino a base no MYSQL.

Figura 3 – Streamsets pipelines de dados criados

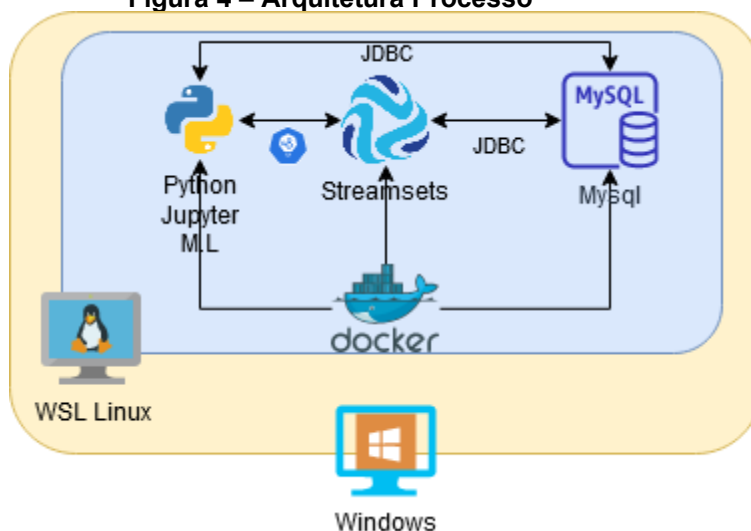


Title	Pipeline ID	Last Updated	Created By	Status
us_accidents_SC Uptime a few seconds	usacciden7a4022da-c3a6-4c1f-ae24-3fdd64afa0a6	2 days ago	admin	RUNNING
us_accidents_PA Uptime a few seconds	usaccidenf65d51d3-ab14-4f82-9898-cd151f5b2ed7	2 days ago	admin	RUNNING
us_accidents_CA Uptime a few seconds	usacciden16f8d9dd-26fc-49a4-8cd5-922c72a60066	2 days ago	admin	RUNNING
read_machine_stats Uptime a few seconds	readmachi48ea2b58-fcb0-4ec3-9e62-94a1071b4abe	2 days ago	jonasosouza89@	RUNNING
us_accidents_LA Uptime a few seconds	usaccidenedc1193c-9d73-4c6a-b9f0-cfb6c07ed91c	2 days ago	admin	RUNNING
us_accidents_NY Uptime a few seconds	usacciden79ba57ac-edf3-4769-b002-f8ee20abc7fa	2 days ago	admin	RUNNING

Fonte: Autoria própria

Adicionalmente, um script Python foi desenvolvido para extrair as métricas de todos os *pipelines*, através da API do próprio *Streamsets*, a cada 30 segundos. Também foram desenvolvidos notebooks, no *Jupyter*, para execução dos algoritmos de detecção de *outliers*. A arquitetura resultante é mostrada na figura 4.

Figura 4 – Arquitetura Processo



Fonte: Autoria própria

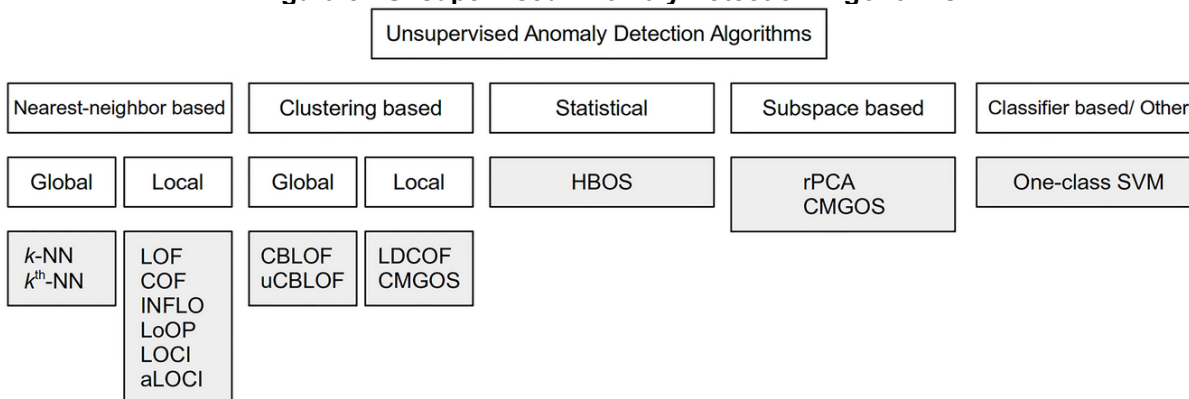
3.5 ALGORITMOS PARA DETECÇÃO DE OUTLIERS

Detectar anomalias é muitas vezes desafiador, pois uma observação apontada como outlier por um algoritmo, pode ser questionável em um contexto de aprendizagem não supervisionada, onde não existe uma pré-classificação, informando o que é ou não é um *outlier*.

Algoritmos de *Machine Learning* podem ser classificados como algoritmos Supervisionado e Não Supervisionado. Resumidamente a diferença está nos dados de treinamento, quando é informado ao algoritmo o que são registros corretos e incorretos, trata-se de um algoritmo Supervisionado. Quando não existe esta sinalização, trata-se de um algoritmo Não Supervisionado.

No presente trabalho foram utilizados os algoritmos PCA e DBSCAN, ambos não supervisionados, pois não é de conhecimento prévio, quais *pipelines* estão divergentes em relação ao seus comportamentos prévios, quais são outliers. Os algoritmos não supervisionados para detecção de anomalia podem ser divididos em 5 grupos, conforme ilustra a figura 6 (MEEHAN; ZDONIK, 2017).

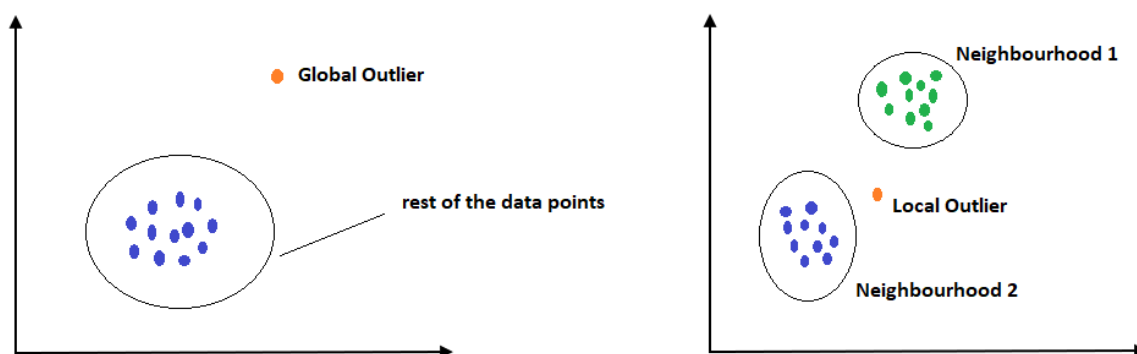
Figura 5 - Unsupervised Anomaly Detection Algorithms



Fonte: Goldstein M, Uchida S (2017)

Outra propriedade dos algoritmos para detecção de outliers é com relação ao contexto em que se atribui um outlier. Algoritmos podem ser GLOBAL ou LOCAL, conforme ilustrado na figura 7, em que a diferença entre eles é que os algoritmos Globais definem como *outliers* observações que divergem em relação a maior ao único ponto observado. Já algoritmos Locais consideram que observações alheias aos grupos observados são considerados *outliers* (CAMPOS, 2015).

Figura 6 - Algoritmos de Outliers Globais vs Locais



Fonte: Blog Nandhini N (2021)

O trabalho aplica uma abordagem Local na detecção dos *outliers*, pois o comportamento de um *pipeline* ao longo do tempo pode variar drasticamente de acordo com o contexto em que é executado. Um *batch pipeline*, tende a executar durante um certo período de tempo e finalizar, processam normalmente grandes volumes de dados. Já um *streaming pipeline* que tende a executar continuamente, processo sob demanda os dados que chegam. Por exemplo um *pipeline* que processa solicitações de clientes para um determinado serviço, pode apresentar picos em determinados horários do dia, todos os dias, mas que é um comportamento esperado do negócio. Já um monitoramento com algoritmo Global pode ser melhor em processos uniformes, aonde é esperado pouca variância.

Os algoritmos selecionados para o trabalho enquadram-se como não supervisionado e Global. Como o trabalho se propõe a análise de *pipelines* de dados, o comportamento pode variar de um instante a outro ao longo de um dia, um mês. Uma abordagem com algoritmo Global poderia ser aplicada a uma análise de observações de sensores industriais, por exemplo, em que espera uma uniformidade nos dados observados.

3.5.1 DBSCAN

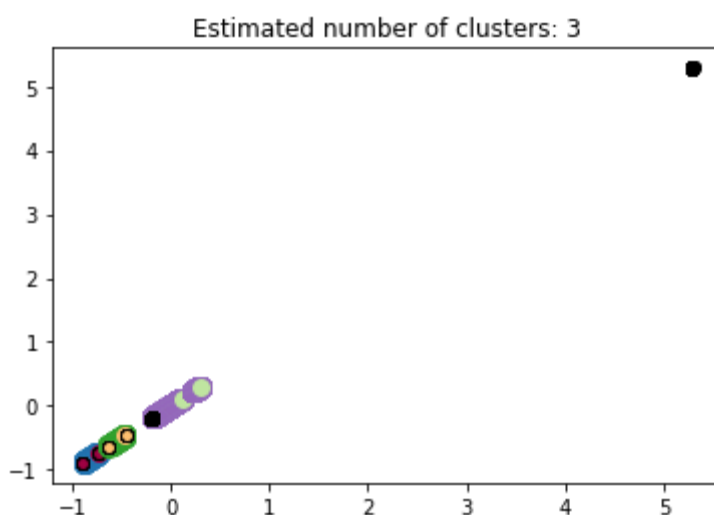
O DBSCAN é um algoritmo baseado em *clustering*, com detecção local, com fácil implementação no Python, via a *sklearn*. O método requer dois parâmetros principais:

- *Eps* - indica a densidade necessária entre os vizinhos ao redor dos pontos mínimos inicialmente fornecidos através do segundo parâmetro;
- *min_samples* - define a tolerância do modelo aos ruídos ((SKLEARN), [s.d.]). O retorno do algoritmo é um vetor com resultados de 1 e -1, em que as observações com o valor de -1 são *outliers*.

A figura 8 mostra o resultado gráfico de um teste executado para o trabalho, aonde é apresentado três *clusters* de um *pipeline* observado. Cada *cluster* é representado pelas cores azul, verde e roxo, as demais cores são os ruídos, anomalias, identificadas pelo DBSCAN, execução do *pipeline*.

Figura 7 - DBSCAN clustering

```
DBSCAN(eps=0.3, min_samples=60)
Estimated number of clusters: 3
Estimated number of noise points: 83
```



Fonte: Autoria própria

Na detecção de outliers em pipeline de dados, o DBSCAN foi utilizado para analisar o comportamento de cada pipeline individualmente. Resultando na formação de *clusters* de dados, com alguns ruídos observações que não se enquadraram a nenhum *cluster* formado. O ruído dos dados obtido através do DBSCAN é interpretado como anomalias.

A solução proposta no trabalho consiste em capturar um período de execução do *pipeline*, processar os dados através do algoritmo DBSCAN. Em seguida as novas observações capturadas serão adicionadas ao modelo. Os resultados devem manter o mesmo valor de ruído para indicar que a observação

atual do *pipeline*, encontra-se dentro dos parâmetros previamente observados, mantendo uma estabilidade.

3.5.2 PCA

O PCA é um algoritmo não supervisionado, baseado em um subespaço ideal para agrupar múltiplas variáveis, facilitando a interpretação dos dados em um plano cartesiano. É importante destacar a performance com a qual o processo é executado, pois com redução das dimensões do *dataset*: o processamento ocorre apenas nos atributos que realmente são mais relevantes para identificar os ruídos nos dados. (AYYADEVARA, 2018).

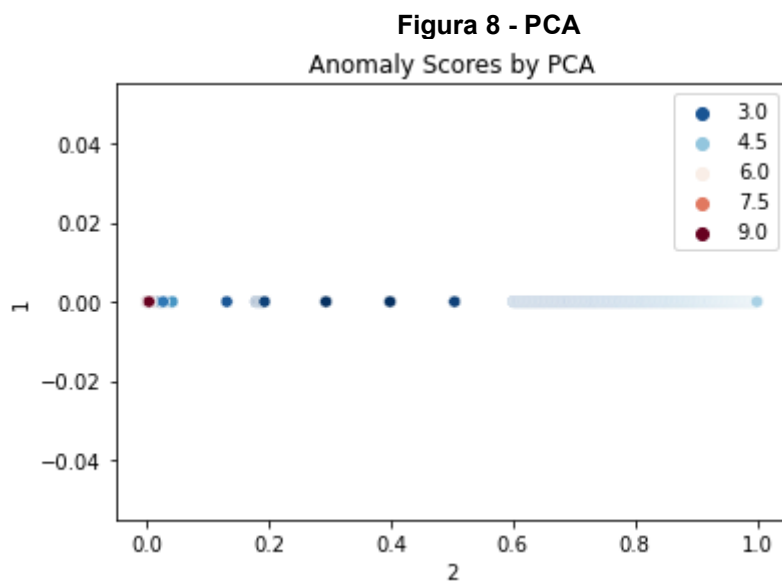
O algoritmo utilizado para o estudo faz parte do pacote *pyod*, disponível para o Python, que além do PCA, contém outros métodos implementados para detecção de outliers. No presente estudo o método do PCA foi implementado da seguinte maneira:

- *svd_solver* – atribuído o valor “*full*”, que de acordo com a documentação do PYOD, realiza uma fatoração da matriz dos dados, reduzindo para duas matrizes unitárias. O valor “*full*” foi escolhido para combinar com o parâmetro seguinte, com o intuito de fazer com o que o modelo defina a quantidade de componentes a serem utilizadas para cada *pipeline* avaliado;
- *n_components* – atribuído o valor “*mle*”, pois combinado com o parâmetro *svd_solver* = “*full*”, o próprio algoritmo vai decidir quantos componentes (atributos disponíveis para identificar *outliers*) serão utilizados.

O algoritmo retorna após o treinamento:

- *n_components_* - a quantidade de componentes utilizadas pelo algoritmo para identificar os *outliers*;
- *decision_scores_* - o valor de registro observado durante o treinamento do modelo. Segundo a documentação, quanto maior o valor, maior a possibilidade de ser um *outlier*;
- *labels_* - contém 0 ou 1 para cada observação realizada, em que 1 é compreendido como *outlier*.

A figura 9 retrata um resultado da redução de oito dimensões a um plano cartesiano, obtido após execução do modelo. Os *decision_scores_* maiores que 6.0, são interpretados como anomalias.



Fonte: Autoria própria

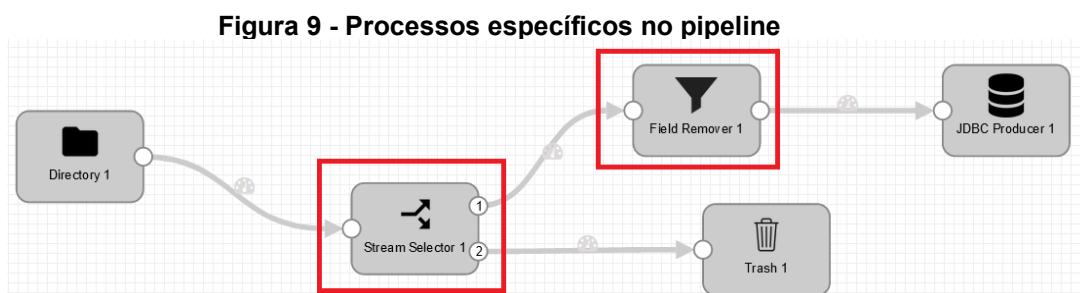
4 ANÁLISE DOS DADOS

Nesta seção será apresentado as métricas dos *pipelines*, análise das métricas capturadas de cada *pipeline* desenvolvido.

4.1 API STREAMSETS

As métricas extraídas da API do Streamsets estão disponíveis para qualquer *pipeline* em execução na ferramenta, contudo foram selecionadas métricas que são comuns a grande maioria dos *pipelines*, indiferente do *software*. Dados como quantidade de entrada de registros, quantidade de saída de dados, quantidade de erros o que tornam o estudo replicável a outras ferramentas, similares ao Streamsets, sem grandes esforços.

Como escopo de dados para identificar outliers nos *pipelines*, foram extraídas as métricas fornecidas pela API do Streamsets dos 8 pipelines desenvolvidos. Métricas de processos específicos foram descartados: por exemplo, quantidade de registros que entraram e saíram de um determinado processo de filtro de algum *pipeline*. A figura 9 destaca em vermelho, exemplos de processos em que as métricas não foram utilizados.



Fonte: Autoria própria

Outro ponto a ser destacado é a utilização das métricas da JVM, seu monitoramento leva em conta todo o ambiente do Streamsets, ou seja, não é possível mensurar quanto cada *pipeline* consome de recursos da JVM.

O Quadro 1 apresenta as métricas extraídas da API do Streamsets, e suas respectivas importâncias para o estudo. Os itens do N° 1 ao 11 são as métricas

comuns a qualquer *pipeline* na ferramenta. Os itens a seguir são referentes a utilização da JVM.

Quadro 1 - Métricas da API do Streamsets

Nº	Atributo	Descrição
1	timeDataObserverRunnable	Data e hora da verificação. Dado temporal referente ao momento em que a métrica foi capturada. Este dado é utilizado para realizar os cortes de acordo com o período em que se deseja avaliar o pipeline.
2	timeOfLastReceivedRecord	Data e hora do último registro extraído na origem do pipeline. Dado importante para ser avaliado de acordo com o período do recorte em que será avaliado o <i>pipeline</i> . Por exemplo um <i>pipeline</i> que fica um período maior do que o habitual sem receber novos registros, o intervalo de tempo será maior, dando indícios que algo não está normal com o processo.
3	pipeline.batchErrorMessage.meter	Quantidade de mensagens de erro da <i>batch</i> (Da Pipeline). Alguns <i>pipelines</i> podem estar configurados para abortar em qualquer caso de erro. Porém para os casos em que os pipelines são mais tolerantes a erros, aumentos ou reduções de erros capturados ao longo da execução do <i>pipeline</i> podem indicar comportamentos anormais aos até então avaliados.
4	pipeline.batchErrorRecords.meter	Quantidade de registros de erro da <i>batch</i> (Da Pipeline), referente a um período de tempo. Alguns <i>pipelines</i> podem estar configurados para abortar em qualquer caso de erro. Similar ao item nº 3.
5	pipeline.batchInputRecords.meter	Quantidade de registros extraídos da origem da <i>pipeline</i> , referente a um período de tempo. A cada execução do fluxo da <i>pipeline</i> , é adicionado a quantidade de registros extraídos a esta métrica.
6	pipeline.batchOutputRecords.meter	Quantidade registros da <i>pipeline</i> que foram processados com sucesso até os destinos, referente a um período de tempo. <i>Pipelines</i> com mais de um destino,

		ou com uma transposição de coluna em linhas, podem apresentar um número muito maior de registros sendo publicados.
7	pipeline.batchCount.counter	Contagem de Batch - Indica a quantidade de vezes em que a origem foi processada. A cada reinício da <i>pipeline</i> , retorna ao zero. Um pipeline estável, sem interrupções em sua execução, ao apresentar uma quantidade de batch menor as até então observadas, podem indicar que algum problema está ocorrendo na pipeline.
8	pipeline.batchErrorMessages.counter	Quantidade de mensagens de erro <i>da batch</i> . Similar ao item nº 3, porém o valor não faz referência a um determinado período.
9	pipeline.batchErrorRecords.counter	Quantidade de registros de erro <i>da batch</i> . Similar ao item nº 4, porém o valor não faz referência a um determinado período.
10	pipeline.batchInputRecords.counter	Quantidade de entrada de registros <i>da batch</i> . Similar ao item nº 5, porém o valor não faz referência a um determinado período.
11	pipeline.batchOutputRecords.counter	Quantidade de saídas de registros <i>da batch</i> . Similar ao item nº 6, porém o valor não faz referência a um determinado período.
12	jvm.memory.heap.usage	Utilização da memória Heap da JVM. Local onde todas instâncias e <i>arrays</i> são alocadas. Auto gerencial, pois ao atingir próximo de 100%, se disponível mais memória será alocada.
13	jvm.memory.non-heap.usage	Utilização da memória não Heap. Similar a Heap, porém é a memória aonde é instanciado os métodos, campos, construtores, etc.
14	jvm.memory.pools.CMS-Old-Gen.usage	Geração antiga do <i>garbage collector</i> .
15	jvm.memory.pools.Code-Cache.usage	Local onde o compilador armazena os códigos compilados.
16	jvm.memory.pools.Metaspace.usage	Uma espécie de memória extra, para evitar o <i>"java.lang.OutOfMemoryError:PermGen"</i> . Disponível nas versões 8 do java.

17	jvm.memory.pools.Par-Eden-Space.usage	Espaço alocado para os novos objetos na memória Eden
18	jvm.memory.pools.Par-Survivor-Space.usage	Objetos da memória Eden são movidos para os Survivors, quando lota.
19	jvm.threads.blocked.count	Quantidade de bloqueios nas <i>threads</i> na JVM.
20	jvm.threads.count	Quantidade de <i>threads</i> na JVM.
21	jvm.threads.daemon.count	Quantidade de <i>threads</i> com baixa prioridade na JVM.
22	jvm.threads.deadlock.count	Quantidade de <i>deadlocks</i> na JVM.
23	jvm.threads.new.count	Quantidade de novas <i>threads</i> na JVM.
24	jvm.threads.runnable.count	Quantidade de <i>threads</i> em execução na JVM.
25	jvm.threads.terminated.count	Quantidade de <i>threads</i> finalizadas na JVM.
26	jvm.threads.timed_waiting.count	Quantidade de <i>threads</i> em <i>sleep</i> na JVM.
27	jvm.threads.waiting.count	Quantidade de <i>threads</i> em espera na JVM.

Fonte: Autoria própria, JVM Memory (GESSO; KULSUM, 2020)

Com estes dados disponíveis é possível separá-los em dois grupos a análise: um no monitoramento geral do ambiente do Streamsets que são os dados da JVM, outro que são os dados de cada *pipeline* baseado nos padrões de erros e quantidade de entrada e saída de registros. Porém foi abordado no estudo apenas as observações dos *pipelines*: dentre os motivos o principal é disponibilidade de ferramentas no mercado, que já realizam esse monitoramento de ambiente, como Dynatrace, Grafana entre outros.

4.2 ANÁLISE EXPLORATÓRIA DE DADOS (AED)

No mercado existe uma diversa gama de ferramentas que podem ser utilizadas na análise de dados. Python tem ganho cada vez mais espaço por questões de custos e flexibilidade, que é obtida através de algumas bibliotecas abertas, como no caso deste trabalho o Pandas. No Pandas é possível trabalhar com um objeto chamado *data frame*, que é similar a uma tabela de banco de dados, uma estrutura bidimensional (EMBARAK, 2018), possui diversos métodos já implementados que auxiliam na AED. Os métodos mais comuns na AED utilizando Pandas são:

1. Análises Estatísticas

- a. Em colunas numéricas de um *data frame*, é possível obter a média dos valores encontrados em cada coluna, valores máximos e mínimos, correlação entre colunas, desvios padrão, entre outros dados que mostram um perfil dos valores numéricos de cada coluna no *data frame* analisado.

2. Agrupamento de Dados

- a. Transformar o *data frame* em subgrupos de dados, como agrupar os *Pipelines Ids* por quantidade de erros, e realizar uma análise mais específica neste subgrupo extraído.

3. Iteração dos Agrupamentos

- a. Quando necessário é possível realizar iterações em agrupamentos realizados, como listar o período observado de cada *pipeline* que foi agrupado.

4. Transformação

- a. Operações que resultam a modificação ou criação de alguma nova coluna, baseadas em alguma condição, como aplicar uma transformação para que todas as colunas que representem peso estejam em gramas e não quilogramas. Transformações não impactam a quantidade de registros no *data frame*.

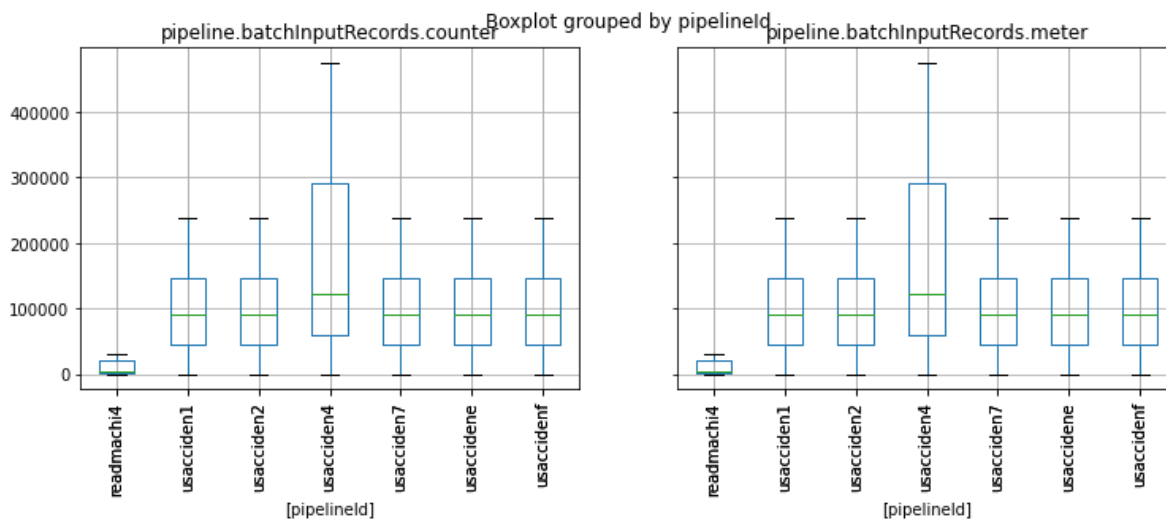
5. Filtragem

- a. Diferentemente da transformação a quantidade de registros é afetada no processo de filtragem. Por exemplo filtrar todos os registros não vazios de uma determinada coluna.

Todos estes processos tem como principal objetivo criar uma prévia compreensão e uma sumarização das principais características de um *dataset*. (EMBARAK, 2018).

Algumas métricas como quantidade de entrada e saída de registros dos *pipelines* observados apresentaram o mesmo comportamento, tanto o “*meter*” quanto o “*counter*”, conforme o Gráfico 1. Ambos atributos mostraram-se redundantes no estudo, porém o mesmo pode não ocorrer, caso o ambiente fosse real, logo ambos continuaram sendo utilizados no modelo.

Gráfico 1 – Análise de dados Boxplot – Input Records



Fonte: Autoria própria

Apenas 3 *pipelines* apresentaram erros durante suas execuções, conforme o gráfico 2. Todos os *pipelines* são tolerantes a falhas, o que torna a métrica *batchErrorRecords.meter* relevante para o processo de detecção de outliers. Qualquer *pipeline* que venha apresentar uma quantidade de erro superior ao comportamento observado pode indicar que algo anormal está ocorrendo em sua execução. Analisado no gráfico 3, o *pipeline* *usacciden1* apresentou mais erros que os demais, o comportamento observado talvez possa ser tratado como uma anomalia pelo modelo, já que na execução do dia 25/06 a quantidade de erros, aumentou consideravelmente em relação ao período anterior.

Gráfico 2 – Análise de dados Boxplot – Error Records

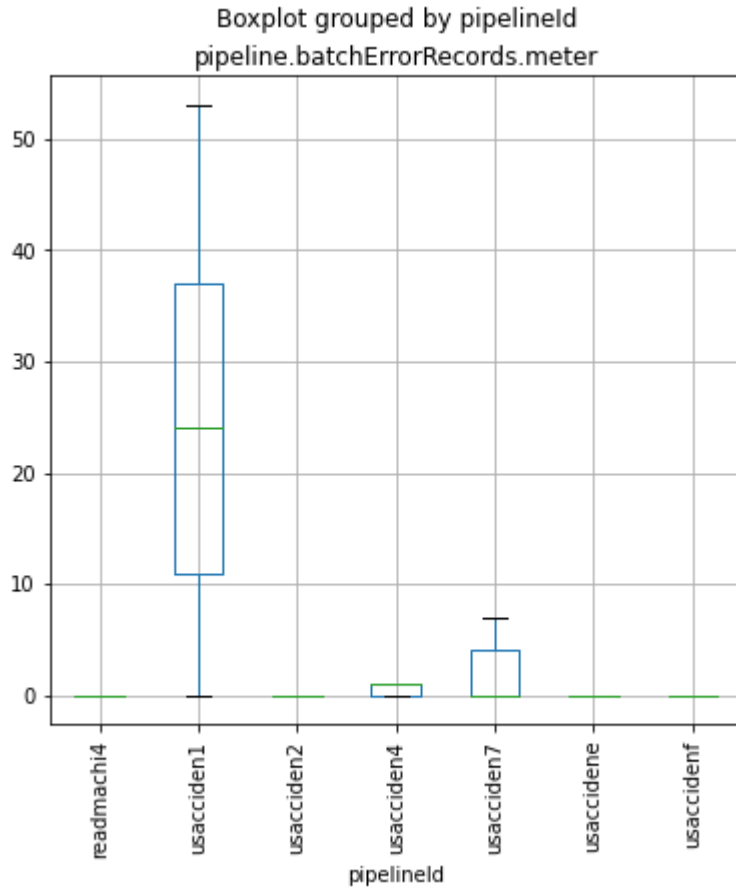
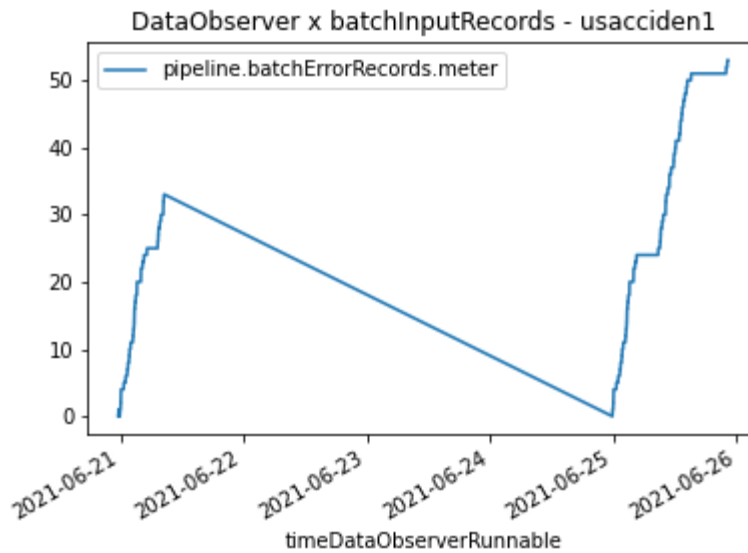


Gráfico 3 - Erros encontrados no pipeline usacciden1



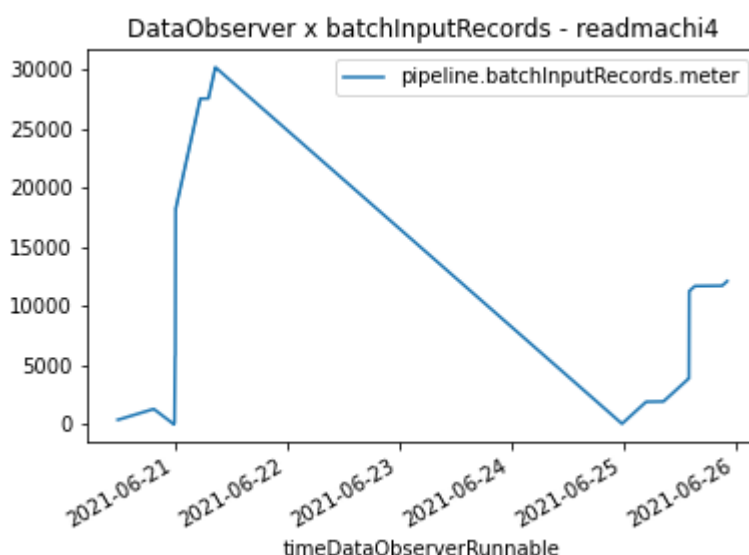
Fonte: Autoria própria

Avaliando os dados obtidos a partir dos atributos dos pipelines, alguns atributos destacam-se como possíveis *features* para os modelos. *Feature* é descrito com uma contextualização de um atributo (KOHAVI; PROVOST, 1998). Dados como

quantidade de erros, quantidade de registros extraídos/publicados pelo *pipeline*, são as principais *features* dos modelos.

O período observado de cada *pipeline*, conforme o gráfico 4, destacam três momentos em que o processo reiniciou, pois, conforme o gráfico a quantidade de registros extraídas retorna a zero, voltando a subir, com a execução incremental de novas batches. Processos que são reiniciados com frequência, indiferente do motivo, podem não serem apontados como anomalias dependendo do período observado, pois um longo período observado com reinicializações poderá ser compreendido pelo modelo como um comportamento normal. Os motivos podem ser dos mais variados como uma intervenção manual do processo, paradas do ambiente ou falha no *pipeline* que reiniciou na sequência com sucesso. O comportamento do gráfico 4 também foi verificado nos demais *pipelines*.

Gráfico 4 - Execuções do Pipeline readmachi4



Fonte: Autoria própria

5 RESULTADOS

Foram utilizados os algoritmos PCA e o DBSCAN, na tentativa de capturar comportamentos anormais dos *pipelines*. O resultado de ambos algoritmos foram combinados, multiplicando o ruído do resultado PCA pela quantidade de ruídos obtidos pelo DBSCAN, para criação de um *ranking*, indicando a ordem dos *pipelines* que demandam atenção e apresentaram um comportamento mais anormal dentro todos. Os resultados obtidos são apresentados na tabela 1, a coluna “*result*” é o indicador de anomalia final.

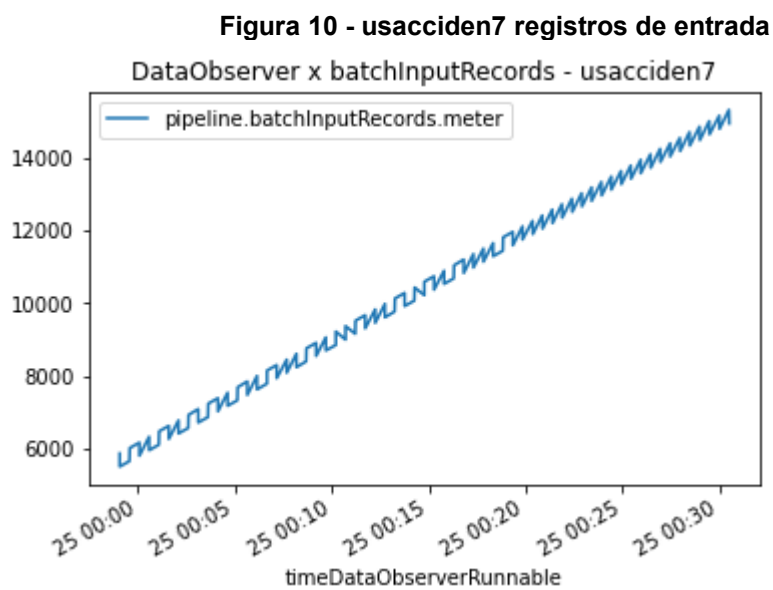
Tabela 1 - Resultado Anomalias dos Pipelines

pipeline	PCA_Ruido	DBA_Clusters	DBA_Ruido	result
usacciden79ba57ac-edf3-4769-b002-f8ee20abc7fa	4.474573e-32	7	49	2.192541e-30
readmachi48ea2b58-fcb0-4ec3-9e62-94a1071b4abe	4.597908e-33	4	4	1.839163e-32
usacciden16f8d9dd-26fc-49a4-8cd5-922c72a60066	1.013265e-32	3	0	0.000000e+00
usaccidenf65d51d3-ab14-4f82-9898-cd151f5b2ed7	1.085344e-35	3	0	0.000000e+00
usacciden7a4022da-c3a6-4c1f-ae24-3fdd64afa0a6	1.171444e-34	3	0	0.000000e+00
usacciden260f4ed9-7b75-4f01-9698-9a5d0facdb60	2.506479e-34	3	0	0.000000e+00
usaccidenedc1193c-9d73-4c6a-b9f0-cfb6c07ed91c	1.804165e-33	3	0	0.000000e+00
usacciden41d98042-7159-448b-b7ba-f3e6a066852d	9.625169e-32	4	0	0.000000e+00

Fonte: Autoria própria

Apenas dois *pipelines* apresentaram um ruído maior que zero, ao comparar a execução do dia 25 as suas execuções anteriores, os *pipelines* readmachi e usacciden7.

A figura 10 apresenta uma oscilação da quantidade dos registros de entrada do *pipeline* usacciden7, o comportamento esperado para o processo, era uma evolução contínua, porém com o aumento do *batch size* dos *pipelines* para processar 1000 registros por vez, a escassez de *hardware* do servidor devido a alta demanda, fez com que este *pipeline* processasse por vez a quantidade máxima possível no servidor que não foi 1000 registros, ocasionando este ruído.



Fonte: Autoria própria

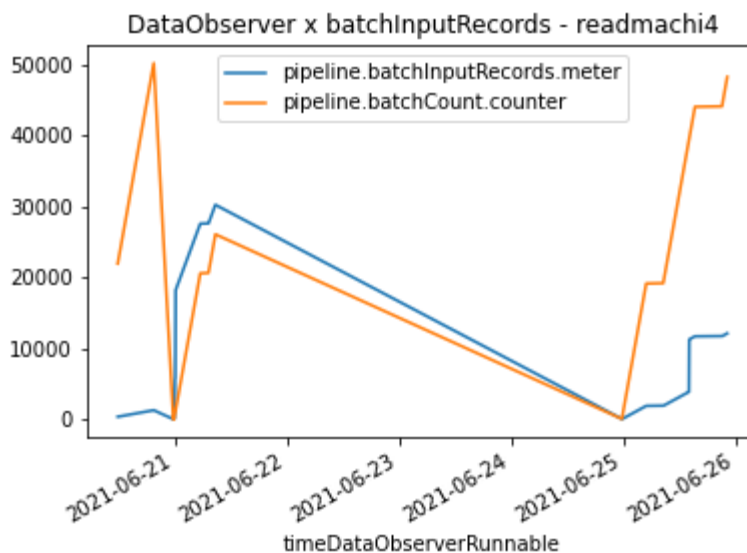
A figura 11 realiza uma comparação do usacciden2 com o usacciden7, aonde é possível notar a oscilação do *batchInputRecords*.

Figura 11 - Comparação usacciden2 com usacciden7

Fonte: Autoria própria

O *pipeline* readmachi, segundo colocado no *ranking*, apresentou uma diminuição na quantidade de registros extraídos pela origem, em relação aos períodos anteriores, conforme a mostra a figura 12.

Figura 12 - readmachi métricas



Fonte: Autoria própria

O *pipeline* usacciden1, apresentou na AED verificou-se os maiores índices de erros durante as suas execuções, manteve-se no dia 25/06 como uma execução normal, considerando que o aumento na quantidade de erros não foi tratado como um *outlier* pelo processo. Porém não foi classificado como um *outlier* pelo processo.

Os resultados obtidos através da solução inicialmente apresentaram resultados que poderiam auxiliar no monitoramento dos pipelines, apenas com a utilização das métricas dos pipelines como base para os modelos de detecção de *outlier*. O fato do ambiente ser simulado limitou o comportamento dos *pipelines*, pois em um ambiente produtivo, a ocorrência de fatos gerada por cada pipeline, seria muito maior. Seria possível avaliar o processo em tempo real, comparando períodos diferentes, para avaliar o processo.

O fator obtido através dos dois modelos utilizados facilita a interpretação dos resultados gerados pela solução. Tornou-se prático, com a tabela, para qualquer usuário compreender quais processos podem estar com um comportamento

anormal, os resultados também poderiam ser reproduzidas em ferramentas gráficas como Tableau, PowerBI, Grafana, entre outras.

O modelo do DBSCAN apresentou uma importância maior na definição do ranking, da figura 10. Porém caso o ruído do modelo PCA tivesse um peso maior no resultado, pouca coisa mudaria no ranking. A liderança seria do pipeline usacciden4, seguido do usacciden7, usacciden1 e readmachi4. Um *ranking* que também varia sentido pois o usacciden4 é o pipeline que mais apresentou erros durante seu monitoramento. Pode ser que alguma calibração no treinamento dos modelos seja necessária para resultados ainda mais consistentes.

6 CONSIDERAÇÕES FINAIS

Os resultados obtidos foram satisfatórios considerando que foram obtidos a partir ser um ambiente simulado. Contudo para objetivos propostos para o trabalho, algumas ressalvas devem ser postas:

- A combinação dos resultados de ambos modelos ainda pode ser otimizada;
- Outros algoritmos podem ser testados na tentativa de gerar melhores resultados na detecção de anomalias;
- Apenas atributos comuns a todos os *pipelines* da ferramenta foram utilizados para detectar *outliers*. A verificação de métricas específicas a cada processo do *pipeline*, pode entregar um resultado muito mais detalhado dos pipelines, porém pode impactar de forma negativa na performance do processo para identificar *outliers*;
- A avaliação dos *pipelines* em tempo real pode não ser satisfatória, pois a solução atual foi testada com base em um dia de dados, de cada *pipeline*;
- Criação de um processo de avaliação dos resultados obtidos pela solução seria importante para avaliar em médio/longo prazo, como o processo impactou na saúde do ambiente;
- A extensão do escopo do estudo, pode ser ampliada ao sistema de gestão dos *pipelines*, com a utilização das métricas da JVM;
- A performance do processo poderá onerar com um histórico de dados maior.

Além das ressalvas pode-se destacar os seguintes pontos positivos:

- Modelos ficaram leves, exigindo um baixo custo computacional;
- Indicação de pontos de ruídos relevantes;
- A criação do ambiente foi realizada com sucesso;
- Facilidade para monitorar novos processos, sendo necessário apenas indicar o *pipelineId* no processo que obtém as métricas da API.

A implementação de uma solução como a proposta pelo estudo não geraria um custo adicional para aquisição de um novo *software*, algoritmos de detecção de *outliers*, disponíveis no Python sem custo, podem gerar ganhos relevantes, com um

ambiente monitorado, indicando processos que podem estar apresentando inconsistências, que poderiam levar tempo até serem percebidos por um cliente, ou alguma área de negócio da empresa.

Vários caminhos podem ser explorados a partir do ponto final deste estudo, conforme citado nos itens acima. O principal seria a implementação em um ambiente produtivo, avaliando em tempo real os pipelines. Na sequência avaliar os *pipelines* de forma mais individualizada, olhando todos os processos que compõem o *pipeline*, a qualidade dos monitoramentos seria extremamente especializada.

A Ciência de Dados é complexa em sua essência, porém toda complexidade pode ser abstraída, com o auxílio de pacotes implementados em diversas linguagens de programação. Exigindo uma compreensão do contexto, para em seguida decidir com quais ferramentas(algoritmos) utilizar para obter respostas.

REFERÊNCIAS

(SKLEARN). **2.3. Clustering**. Disponível em: <https://scikit-learn.org/stable/modules/clustering.html#dbscan>.

AYYADEVARA, V. K. **Pro Machine Learning Algorithms**. Berkeley, CA: Apress, 2018.

BELLINGER, G.; CASTRO, D.; MILLS, A. **Data, Information, Knowledge, and Wisdom**. Disponível em: <http://www.systems-thinking.org/dikw/dikw.htm>.

CAMPOS, G. O. **Estudo, avaliação e comparação de técnicas de detecção não supervisionada de outliers**. p. 67, 2015.

EMBARAK, D. O. **Data Analysis and Visualization Using Python**. Berkeley, CA: Apress, 2018. v. 7

FEDUSHKO, S.; USTYIANOVYCH, T.; GREGUS, M. **Real-time high-load infrastructure transaction status output prediction using operational intelligence and big data technologies**. Electronics (Switzerland), v. 9, n. 4, 2020.

FOX, C. **“Data Science” and “Big Data”**. In: [s.l: s.n.]. p. 1–14.

GISSO, J. (BETSOL); KULSUM, U. **Java Memory Management for Java Virtual Machine (JVM)**. Disponível em: https://www.betsol.com/blog/java-memory-management-for-java-virtual-machine-jvm/#Java_Virtual_Machine_JVM.

HAWKINS, D. M. **Identification of Outliers**. 1. ed. Dordrecht: Springer Netherlands, 1980.

JAVORNIK, M.; NADOH, N.; LANGE, D. **Data Is the New Oil**. Disponível em: https://ana.blogs.com/maestros/2006/11/data_is_the_new.html.

KOHAVI, R.; PROVOST, F. **Special Issue on Applications of Machine Learning and the Knowledge Discovery Process**. Disponível em: <http://robotics.stanford.edu/~ronnyk/glossary.html>.

MEEHAN, J.; ZDONIK, S. **Data Ingestion for the Connected World**. Cidr, 2017.

MICROSOFT. **Guia de instalação do Subsistema Windows para Linux para Windows 10**. Disponível em: <https://docs.microsoft.com/pt-br/windows/wsl/install-win10>.

MOORE, S.; GARTNER. **How to Stop Data Quality Undermining Your Business**. Disponível em: <https://www.gartner.com/smarterwithgartner/how-to-stop-data-quality-undermining-your-business/>.

MURSI, M. F. M.; SALAMA, M. M.; HABEB, M. H. **An Improved SIFT-PCA-Based Copy-Move Image Forgery Detection Method**. International Journal of Advanced Research in Computer Science and Electronics Engineering, v. 6, n. 3, p. 23–28, 2017.

REN, H. et al. Time-series anomaly detection service at Microsoft. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, v. 3330680, n. c, p. 3009–3017, 2019.

SOBHAN, M.; KAGGLE. **US Accidents (3 million records -- updated)**. Disponível em: <https://www.kaggle.com/sobhanmoosavi/us-accidents>.

STREAMSETS. **What Is a Data Pipeline?** Disponível em: <https://streamsets.com/getting-started/building-data-pipelines/>.

STREAMSETS. **Streamsets Data Collector Documentation**. Disponível em: https://docs.streamsets.com/portal/#datacollector/latest/help/datacollector/UserGuide/Getting_Started/GettingStarted_Title.html#concept_htw_ghg_jq.

SURVEY, G.; MARCH, D. P. **DATA ENGINEERS : READY TO DELIVER MORE BUSINESS VALUE** , YET MIRED IN MANUAL TASKS A Global Survey of Data Professionals. n. March, 2021.

VON EYE, A.; SCHUSTER, C. Outlier Analysis. In: **Regression Analysis for Social Sciences**. [s.l.] Elsevier, 1998. p. 81–98.

APÊNDICE A -Questionário de Pesquisa

Ministério da Educação
Universidade Tecnológica Federal do Paraná
 Diretoria de Graduação e Educação Profissional
Secretaria de Gestão Acadêmica
 Departamento de Biblioteca

APLICAÇÃO DO QUESTIONÁRIO PARA TRABALHOS ACADÊMICOS

1. Você tem conhecimento do trabalho que está sendo realizado na UTFPR que criará o padrão da instituição para elaboração de trabalhos acadêmicos?

	EM	G	PG	P	TA	TOTAL
Sim						
Não						

2. Se a resposta da pergunta anterior foi afirmativa, de que maneira tomou conhecimento?

	EM	G	PG	P	TA	TOTAL
Pela Internet, na página da instituição						
Pelo jornal da instituição						
Por outra maneira						

3. Na realização de trabalhos acadêmicos (relatório, TCC, dissertação, tese, etc.) você costuma consultar normas que norteiam a elaboração dos mesmos?

	EM	G	PG	P	TA	TOTAL
Sempre						
Nunca						
Às vezes						

4. Se utiliza normas para elaboração de trabalhos acadêmicos, quais costuma consultar?

	EM	G	PG	P	TA	TOTAL
ABNT						
UTFPR						
A que seu orientador passou						
A elaborada pela biblioteca e professores de nosso Campus						
De outra instituição						

APÊNDICE B - Roteiro da Entrevista

ROTEIRO DE ENTREVISTA

1- Identificação Pessoal:

Nome: _____
D/N: _____
Nacionalidade: _____
Sexo: _____
Idade: _____

Outras pessoas que moram na casa:

Informante: _____

2- Encaminhado por: _____

Motivo da solicitação: _____

3 - Antecedentes Pessoais:

3.1- Concepção

Quanto tempo após o casamento? _____

Foi desejada? _____

Sexo esperado? _____

Abortos anteriores (espontâneos ou provocados e época) _____

Observações: _____

ANEXO A - Direitos autorais - Lei nº 9.610, de 19 de fevereiro de 1998. Disposições preliminares



**Presidência da República
Casa Civil
Subchefia para Assuntos Jurídicos**

LEI Nº 9.610, DE 19 DE FEVEREIRO DE 1998¹.

Mensagem de veto

Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências.

O PRESIDENTE DA REPÚBLICA Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Título I - Disposições Preliminares

Art. 1º Esta Lei regula os direitos autorais, entendendo-se sob esta denominação os direitos de autor e os que lhes são conexos.

Art. 2º Os estrangeiros domiciliados no exterior gozarão da proteção assegurada nos acordos, convenções e tratados em vigor no Brasil.

Parágrafo único. Aplica-se o disposto nesta Lei aos nacionais ou pessoas domiciliadas em país que assegure aos brasileiros ou pessoas domiciliadas no Brasil a reciprocidade na proteção aos direitos autorais ou equivalentes.

Art. 3º Os direitos autorais reputam-se, para os efeitos legais, bens móveis.

Art. 4º Interpretam-se restritivamente os negócios jurídicos sobre os direitos autorais.

Art. 5º Para os efeitos desta Lei, considera-se:

I - publicação - o oferecimento de obra literária, artística ou científica ao conhecimento do público, com o consentimento do autor, ou de qualquer outro titular de direito de autor, por qualquer forma ou processo;

II - transmissão ou emissão - a difusão de sons ou de sons e imagens, por meio de ondas radioelétricas; sinais de satélite; fio, cabo ou outro condutor; meios óticos ou qualquer outro processo eletromagnético;

III - retransmissão - a emissão simultânea da transmissão de uma empresa por outra;

IV - distribuição - a colocação à disposição do público do original ou cópia de obras literárias, artísticas ou científicas, interpretações ou execuções fixadas e fonogramas, mediante a venda, locação ou qualquer outra forma de transferência de propriedade ou posse;

V - comunicação ao público - ato mediante o qual a obra é colocada ao alcance do público, por qualquer meio ou procedimento e que não consista na distribuição de exemplares;

VI - reprodução - a cópia de um ou vários exemplares de uma obra literária, artística ou científica ou de um fonograma, de qualquer forma tangível, incluindo qualquer armazenamento permanente ou temporário por meios eletrônicos ou qualquer outro meio de fixação que venha a ser desenvolvido;

VII - contrafação - a reprodução não autorizada;

VIII - obra:

a) em co-autoria - quando é criada em comum, por dois ou mais autores;

b) anônima - quando não se indica o nome do autor, por sua vontade ou por ser desconhecido;

c) pseudônima - quando o autor se oculta sob nome suposto;

d) inédita - a que não haja sido objeto de publicação;

e) póstuma - a que se publique após a morte do autor;

f) originária - a criação primígena;

g) derivada - a que, constituindo criação intelectual nova, resulta da transformação de obra originária;

h) coletiva - a criada por iniciativa, organização e responsabilidade de uma pessoa física ou jurídica, que a publica sob seu nome ou marca e que é constituída pela participação de diferentes autores, cujas contribuições se fundem numa criação autônoma;

i) audiovisual - a que resulta da fixação de imagens com ou sem som, que tenha a finalidade de criar, por meio de sua reprodução, a impressão de movimento, independentemente dos processos de sua captação, do suporte usado inicial ou posteriormente para fixá-lo, bem como dos meios utilizados para sua veiculação;

IX - fonograma - toda fixação de sons de uma execução ou interpretação ou de outros sons, ou de uma representação de sons que não seja uma fixação incluída em uma obra audiovisual;

X - editor - a pessoa física ou jurídica à qual se atribui o direito exclusivo de reprodução da obra e o dever de divulgá-la, nos limites previstos no contrato de edição;

XI - produtor - a pessoa física ou jurídica que toma a iniciativa e tem a responsabilidade econômica da primeira fixação do fonograma ou da obra audiovisual, qualquer que seja a natureza do suporte utilizado;

XII - radiodifusão - a transmissão sem fio, inclusive por satélites, de sons ou imagens e sons ou das representações desses, para recepção ao público e a transmissão de sinais codificados, quando os meios de decodificação sejam oferecidos ao público pelo organismo de radiodifusão ou com seu consentimento;

XIII - artistas intérpretes ou executantes - todos os atores, cantores, músicos, bailarinos ou outras pessoas que representem um papel, cantem, recitem, declamem, interpretem ou executem em qualquer forma obras literárias ou artísticas ou expressões do folclore.

Art. 6º Não serão de domínio da União, dos Estados, do Distrito Federal ou dos Municípios as obras por eles simplesmente subvencionadas.

¹ Disponível em: http://www.planalto.gov.br/ccivil_03/leis/19610.htm.

ANEXO B - Capa do livro: Normas para Elaboração de Trabalhos