UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALEX FREDERICO MATHIAS FELIX DE MELO

AVALIAÇÃO DA REMOÇÃO DE ANOMALIAS EM DADOS COLETADOS POR SENSORES

RUSSAS

2019

ALEX FREDERICO MATHIAS FELIX DE MELO

AVALIAÇÃO DA REMOÇÃO DE ANOMALIAS EM DADOS COLETADOS POR SENSORES

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Dmontier Pinheiro Aragão Junior

RUSSAS

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M485a Melo, Alex Frederico Mathias Felix de.
Avaliação da remoção de anomalias em dados coletados por sensores / Alex Frederico Mathias Felix de Melo. – 2019.
59 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2019.
Orientação: Prof. Dr. Dmontier Pinheiro Aragão Junior.

1. Detecção de anomalias. 2. Aprendizado de máquina. 3. Indústria 4.0. 4. Avaliação e comparação. I. Título.

CDD 005

ALEX FREDERICO MATHIAS FELIX DE MELO

AVALIAÇÃO DA REMOÇÃO DE ANOMALIAS EM DADOS COLETADOS POR SENSORES

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Dmontier Pinheiro Aragão Junior (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Alexandre Matos Arruda
Universidade Federal do Ceará (UFC)

Prof. Ms. Alex Lima Silva
Universidade Federal do Ceará (UFC)

A Deus, meus pais, irmãos, minha esposa, professores, amigos e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa da minha vida.

AGRADECIMENTOS

A Deus, por ter me sustentado em todos os momentos difíceis, proporcionando vitórias e aprendizados. A minha família, que sempre me apoiou e incentivou a continuar. A minha esposa, Luana Keity de Lima da Silva, por ser minha inspiração e amor de minha vida.

A meu pai, Francisco José Félix de Melo, por ter proporcionado que eu chegasse tão longe e por ter me ensinado os valores e os princípios que regem minha vida. A minha mãe, Rosanna Andréa Mathias de Melo, por sempre mostrar que a educação é o único caminho a ser seguido. A minha sogra, Clécia Maria Moura de Lima, por sempre ter palavra de incentivos e ter me acolhido.

Agradeço aos professores Marcos Vinícius de Andrade Lima e Dmontier Pinheiro Aragão Júnior, que acompanharam a minha jornada acadêmica de perto e deram muito apoio em sala de aula. Obrigado pela incansável dedicação e confiança. Sou grato principalmente ao Doutor Dmontier Pinheiro Aragão Júnior, que foi o meu orientador, por ter confiado em mim e em meu trabalho, contribuindo muito com a realização dessa pesquisa.

Aos meus amigos, Marcos Alencar, Mateus Oliveira, Evilaine Paiva, Elyneker Silva, George Masullo, Elanne Mendes, Erik Almeida, Carlos Victor, Hugo Venâncio, Igor Mendes, Isaías Ferreira, Marcos Paulo, Thomas Dillan, Vinícius Almeida, Isaac Rahel, Tágila Lima e Afonso Matheus, por me ensinarem que juntos podemos ir mais longe, e agradeço a tantos outros que me apoiaram durante a graduação.

Obrigado Universidade Federal do Ceará pela oportunidade de realizar o curso de Ciência da Computação. Agradeço por me oferecer professores incríveis, um ambiente de estudo saudável e muitos estímulos para participar de atividades acadêmicas. Sou grato não só aos professores, mas também à direção, ao pessoal do administrativo, da limpeza e demais colaboradores da instituição.

Agradeço ao Laboratório Interdisciplinar de Computação e Engenharia de Software (LINCE), que me proporcionou a chance de expandir os meus horizontes. Obrigada pelo ambiente criativo e amigável nesses quatro anos de formação.

Agradeço à empresa Apodi, pela oportunidade de fazer estágio supervisionado. Foi com essa experiência que me tornei um profissional qualificado na minha área de formação. Obrigado Thiago Pasquini, Stefanos Anagnostou, Alexandre Arruda, Dmontier Pinheiro, Cíntia Lima, Hismael Costa, Rafael Costa, Rilmar Farias e Hevilla Souza por terem me acompanhado nessa jornada.

Ao Ednardo Moreira Rodrigues e Alan Batista de Oliveira, pela adequação do template utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Por fim, a todos que ajudaram diretamente ou indiretamente na realização desse trabalho. Sou imensamente grato pela paciência e incentivo.

RESUMO

Na industria 4.0, um grande número de sensores diferentes são instalados em sistemas de produção e conectados a inteligências artificiais com o intuito de fazer inferências sobre o ambiente e, através disso, auxiliar operadores humanos na supervisão, acompanhamento e otimização da produção. Entretanto as variáveis do ambiente em que o sensor está inserido podem induzir a ruídos e a perda de dados, tornando-se necessário empregar um mecanismo de triagem apropriado para a analise desses dados. Deste modo foi necessário analisar o desempenho de diferentes algoritmos estatísticos de detecção de anomalias com o propósito de identificar o algoritmo mais eficiente e rápido para a remoção de dados anômalos de um conjunto de dados coletados automaticamente por sensores. Foram realizados 216 experimentos, um experimento é definido como a combinação de um algoritmo de detecção de anomalias, um algoritmo de aprendizado de máquina e uma classe objetivo. Para os experimentos, cerca de 6 algoritmos de detecção de anomalias, 2 algoritmos de aprendizado de máquina e 3 classes diferentes foram utilizados. Constatou-se que os algoritmos de detecção de anomalias que possuem maior eficácia, em ordem decrescente, foram o KDE (*Gaussian*; 0,3), Box plot (0,5), KDE (*Exponential*; 0,3), KDE (*Tophat*; 0,3) e Box Plot (1,5). Portanto a remoção de dados considerados anômalos utilizando técnicas estatísticas, geralmente, produzem um impacto positivo para a previsão de novos valores, removendo as anomalias em um tempo desprezível.

Palavras-chave: Detecção de anomalias; Aprendizado de máquina; Indústria 4.0; Avaliação e comparação.

ABSTRACT

In industry 4.0, a large number of different sensors are installed in production systems and connected to artificial intelligence in order to make inferences about the environment and thereby assist human operators in the supervision, monitoring and optimization of production. However, the variables of the environment in which the sensor is inserted can induce noise and data loss, making it necessary to employ an appropriate screening mechanism for the analysis of these data. Thus, it was necessary to analyze the performance of different statistical algorithms for the detection of anomalies in order to identify the most efficient and fast algorithm for the removal of anomalous data from a set of data automatically collected by sensors. 216 experiments were performed, an experiment is defined as the combination of an anomaly detection algorithm, a machine learning algorithm and a target class. For the experiments, about 6 anomaly detection algorithms, 2 machine learning algorithms and 3 different classes were used. It was found that the most effective anomaly detection algorithms, in descending order, were KDE (Gaussian; 0.3), Box plot (0.5), KDE (textit Exponential; 0.3), KDE (Tophat), 0.3) and Box Plot (1.5). Therefore, the removal of data considered anomalous using statistical techniques generally produces a positive impact for the prediction of new values, removing the anomalies in a negligible time.

Keywords: Anomalies Detection; Machine learning; Industry 4.0; Evaluation and comparison.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

aLOCI	*Approximate Local Correlation Integral*
AM	Aprendizagem de Máquina
AUC	*Area Under the ROC Curve*
CBLOF	*Cluster-Based Local Outlier Factor*
CMGOS	*Clustering-based Multivariate Gaussian Outlier Score*
COF	*Connectivity-Based Outlier Factor*
HBOS	*Histogram Based Outlier Score*
INFLO	*Influenced Outlierness*
K-NN	*K-Nearest Neighbors*
KDE	*Kernel Density Estimate*
LDCOF	*Local Density Cluster-based Outlier Factor*
LOCI	*Local Correlation Integral*
LOF	*Local Outlier Factor*
MAE	*Mean Absolute Error*
MAPE	*Mean Absolute Percentage Error*
MASE	*Mean Absolute Scaled Error*
MLP	*Multilayer Perceptron*
NASA	*National Aeronautics and Space Administration*
RMSE	*Root Mean Square Deviation*
ROC	*Receiver Operating Characteristic*
rPCA	*Robust Principal Component Analysis*
SPC	*Statical Process Control*
SVM	*Support Vector Machine*
SVR	*Support Vector Regression*

SUMÁRIO

1 INTRODUÇÃO

A quarta revolução industrial, também chamada de Indústria 4.0, é definida por uma rede de computadores conectados a sensores e atuadores que são instalados como sistemas embarcados em materiais, equipamentos e peças de máquinas, sendo monitorados por uma inteligência artificial, tendo como intuito, auxiliar operadores humanos na supervisão e acompanhamento da produção (RAUCH *et al.*, 2019).

Possibilita o diagnóstico e a intervenção em tempo real no sistema de produção, as quais novas competências podem surgir para um aumento da produtividade e uma maior qualidade e confiabilidade do produto, tornando os operadores mais eficientes e eficazes (RAUCH *et al.*, 2019).

Os avanços tecnológicos proporcionaram que um grande número de sensores diferentes pudessem ser instalados em sistemas de produção para a captura do máximo de dados necessários em diferentes estágios do ciclo de vida de um produto. Isto colabora na avaliação e otimização da produção, o que pode levar à coleta de enormes quantidades de dados heterogêneos (BAEK; KIM, 2019; RAUCH *et al.*, 2019).

Diante da grande quantidade de dados heterogêneos coletados diretamente por diversos sensores, é difícil lidar com a alta complexidade e incerteza destes dados, tornando-se necessário empregar um mecanismo de triagem apropriado para a analise desses dados (BAEK; KIM, 2019).

Dados provenientes de sensores, além de possuírem diferentes tipos de informações, são concebidos em fluxo constante. As variáveis do ambiente em que o sensor está inserido podem induzir ruídos e perda de dados, saindo do padrão ideal. Estes padrões disformes nos dados podem ser denominados de anomalias, estas são comumente estudadas em detecção de fraudes, intrusões, falhas e erros (CHANDOLA *et al.*, 2009).

Várias são as pesquisas que avaliam algoritmos de detecção de anomalias a fim de constatar quais algoritmos removem mais dados anômalos, entretanto, estas pesquisas se restringem a comparar algoritmos pertencentes a diferentes técnicas aplicadas a diferentes conjuntos de dados, ou seja, não é estabelecida uma comparação entre variações de algoritmos pertencente a uma mesma técnica em um mesmo conjunto de dados. Com base nestas limitações observadas, esta pesquisa tem como objetivo confrontar diferentes variações de algoritmos estatísticos de detecção de anomalias na remoção dos dados anômalos de um conjunto de dados coletados automaticamente por sensores.

Este trabalho é relevante para profissionais que utilizam conjuntos de dados coletados por sensores para fazer inferências sobre o ambiente por meio da utilização de técnicas de otimização ou de aprendizado de máquina. Podem-se empregar algoritmos estatísticos de detecção de anomalias para obter melhores resultados. Este trabalho, pode ainda auxiliar profissionais que manuseiam algum algoritmo para remoção de anomalias; na substituição por um algoritmo de remoção de anomalias mais eficiente e rápido, útil para sistemas que estão em ambientes dinâmicos e que necessitam fazer inferências em um curto período de tempo.

1.1 Objetivos

- **Objetivo geral**

  Analisar o desempenho de algoritmos estatísticos de detecção de anomalias na remoção dos dados anômalos de 3 (três) conjuntos de dados coletados automaticamente por sensores, de maneira a aumentar a acurácia da previsão de algoritmos de aprendizado de máquina.
- **Objetivos específicos**
  - Extrair dados coletados automaticamente por sensores de uma empresa;
  - Selecionar algoritmos de aprendizado de máquina para a obtenção de previsões;
  - Selecionar diferentes algoritmos de detecção de anomalias estatísticos;
  - Experimentar o efeito de diferentes algoritmos de detecção de anomalias em diferentes algoritmos de aprendizado de máquina.

1.2 Trabalhos relacionados

Nesta seção são apresentadas pesquisas que utilizaram conceitos e técnicas semelhantes às propostas por esta pesquisa, destacando as diferenças de cada estudo. Por fim, é apresentado um resumo, exibido em uma tabela as principais características de cada trabalho (seção 1.2.4).

1.2.1 Técnicas de limpeza de dados para conjuntos de dados de engenharia de software

O estudo de Liebchen () abordou a utilização de três variações do algoritmo de árvores de decisão para detecção de anomalias em um conjuntos de dados de engenharia de software. Cada técnica representou uma abordagem diferente: filtragem robusta, onde os conjuntos de treinamento e teste são os mesmos; filtragem preditiva, em que os conjuntos de

treinamento e teste são diferentes; e filtragem e polimento, as quais instâncias ruidosas são corrigidas.

O algoritmo de árvores de decisão é um método de classificação supervisionada, ou seja, dependem da disponibilidade de rótulos precisos para todas as classes (CHANDOLA *et al.*, 2009). A presente pesquisa utilizou dados coletados automaticamente por sensores, portanto não houve disponibilidade de rótulos e, consequentemente, foi necessário utilizar algoritmos não supervisionados.

1.2.2 Um estudo comparativo de esquemas de detecção de anomalias na detecção de intrusão na rede

O trabalho de Lazarevic *et al.* () mostrou a utilização de técnicas de detecção de anomalias usadas para identificar ataques contra computadores e infra-estruturas de rede. O autor comparou várias técnicas de detecção de anomalias supervisionadas e não supervisionadas existentes, e suas variações foram avaliadas no conjunto de dados de conexões de rede.

Entretanto, para o presente trabalho, foi utilizado somente algoritmos não supervisionados para a detecção de dados anômalos, pois o conjunto de dados usado não dispõem de rótulos para a verificação da equivalência entre o valor previsto e o real.

1.2.3 Uma avaliação comparativa de algoritmos de detecção de anomalia não supervisionada para dados multivariados

A pesquisa de Goldstein e Uchida (2016) abordou a avaliação do desempenho de diferentes algoritmos de detecção de anomalia não supervisionada, avaliados em diversos conjuntos de dados em vários domínios de aplicação. Além disso, a pesquisa revelou os pontos fortes e fracos das diferentes abordagens e descreveu o esforço computacional, o impacto das configurações dos parâmetros, bem como o comportamento dos algoritmos de detecção de anomalias.

Referente ao trabalho citado, primeiramente foi realizado um tratamento no conjunto de dados, retirando-se o rotulo de cada instância e, posteriormente, utilizou-se algoritmos de detecção de anomalias para a classificação dos dados do conjunto, rotulando-os novamente. Ao final do processamento comparou-se os rótulos obtidos pela classificação feita por estes algoritmos e os rótulos reais do conjunto de dados, com o intuito de avaliar se os algoritmos de detecção de anomalias classificaram corretamente os dados.

O presente estudo diferencia-se na forma de avaliação do desempenho e na escolha dos algoritmos de detecção de anomalias, escolhendo somente aqueles que empregam técnicas estatísticas. Como os dados provenientes de sensores não são previamente rotulados, optou-se por mensurar a eficácia dos algoritmos de detecção de anomalias através da acurácia da previsão dos algoritmos de aprendizado de máquina, em relação a diferentes variáveis do conjunto.

1.2.4 Compêndio

A Tabela 1 apresenta o compêndio das principais características deste trabalho e daqueles relacionados, demonstrando uma visão geral das diferenças e semelhanças entre cada estudo, o qual foram abordados detalhadamente nas sub-seções anteriores.

Cada estudo foi classificado em cinco quesitos. A primeira coluna refere-se a citação dos estudos; a segunda coluna retrata quais algoritmos de detecção de anomalias foram utilizados; a terceira coluna classifica qual o paradigma de aprendizado foi usado pelos algoritmos de detecção de anomalias; a quarta coluna demostra quais conjuntos de dados foram selecionados para a pesquisa em questão; e a ultima coluna representa quais os métodos de avaliação foram empregados para mensurar o desempenho dos algoritmos de detecção de anomalias.

Tabela 1 – Compêndio dos trabalhos relacionados

Autores	**Algoritmos**	**Paradigmas**	**Conjunto de dados**	**Avaliação**
Liebchen ()	Árvores de decisão	Supervisionado	Projetos de engenharia de software	Acurácia da previsão
Lazarevic *et al.* ()	LOF, K-NN, SVM	Supervisionado e não supervisionado	Conexões de rede simulados e reais	ROC
Goldstein e Uchida (2016)	K-NN, LOF, COF, INFLO, , LOCI, aLOCI, CBLOF, LDCOF, CMGOS, HBOS, SVM e rPCA	Não supervisionado	Câncer de mama, texto escrito à mão de caneta, reconhecimento de letras, dados de sotaque na fala, ligação à terra, doença da tireoide, posições do radiador em um ônibus espacial da NASA, imagens de objetos e a detecção de invasão na rede	ROC, AUC e tempo de computação
Presente trabalho	*Box plot*, *6-Sigma*, Teste de Grubbs, HBOS, KDE e X^2 *Measure*	Não supervisionado	Processo de moagem de cimento em uma fábrica (dados coletados através de sensores acoplados ao moinho)	Acurácia da previsão, tempo de computação e quantidade de dados removidos

Fonte: Autoria própria.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é explicado os conceitos que serão relevantes para o presente trabalho. A seção 2.1 explica, detalhadamente, o que são anomalias e quais os métodos existentes para detectá-las; posteriormente, a seção 2.2 explica o que é aprendizado de máquina e como é aplicado, explanando a teoria de como é realizado a computação de alguns desses algoritmos.

Este capítulo é importante para compreender como o presente estudo irá analisar o desempenho de diferentes métodos para identificação de anomalias a partir da previsão de algoritmos de aprendizado de máquina, em conjuntos de dados coletados por sensores.

2.1 Detecção de anomalias

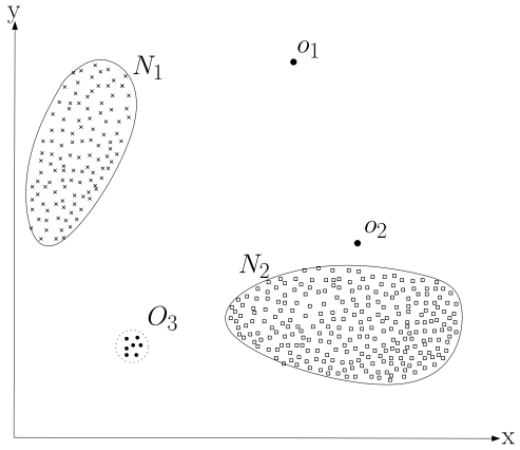
A detecção de anomalias concerne em identificar padrões em dados que diferem significativamente da média ou dos valores esperados, esses padrões disformes são denominados, na sua grande maioria, de anomalias ou *outliers* (CHANDOLA *et al.*, 2009). Uma dada instância em um conjunto de dados geralmente é considerada anomalia se possuir alguma anormalidade, ou seja, se não estiver semelhante a qualquer outra instância do conjunto (SURI; ATHITHAN, 2019). Técnicas de detecção de anomalias podem ser aplicadas para a limpeza de dados ou para a detecção de fraudes, intrusões, falhas, problemas, mudanças, diagnósticos, etc (CHANDOLA *et al.*, 2009; SURI; ATHITHAN, 2019).

A Figura 1 ilustra um exemplo simples de detecção de anomalias em um conjunto de dados bidimensional. Onde os pontos N1 e N2 são considerados duas regiões normais, e os pontos O1, O2 e O3 por se diferirem das regiões normais dos dados, são considerados anomalias.

Segundo Suri e Athithan (2019), instâncias de dados anômalas podem ser caracterizadas de várias maneiras: quanto ao número de instâncias envolvidas, ou seja, o tamanho da amostra que é considerada anomalia; aos diferentes tipos de natureza, ou seja, a diversidade de tipos e de variação de valores; e aos papéis que desempenham, ou seja, a importância de um *outlier* para a aplicação.

Em ambientes onde há uma vasta quantidade sensores inseridos, uma coleta frequente de dados e um canal de comunicação distribuída, ocasiona ruídos e a perda dos dados coletados. Por esse motivo, a presença de ruído nos dados coletados dos sensores torna a detecção de anomalias um processo dificultoso, pois é necessário distinguir quais instâncias de dados possuem

Figura 1 – Detecção de anomalias em um conjunto de dados



Fonte: Chandola *et al.* (2009).

anomalias significantes, valores ruidosos ou valores ausentes (CHANDOLA *et al.*, 2009).

De acordo com Suri e Athithan (2019), há uma distinção entre *outlier* e ruído aleatório, mesmo que os dois representem anomalias nos valores das instâncias, a diferença para um ruído ser considerado um *outlier* é normalmente um avaliação subjetiva baseado no interesse do analista.

O ruído pode ser definido como um fenômeno em dados que não interessam ao analista, mas atua como um obstáculo à análise de dados. A remoção de ruído é motivada pela necessidade de remover os objetos indesejados antes que qualquer análise seja executada nos dados (CHANDOLA *et al.*, 2009).

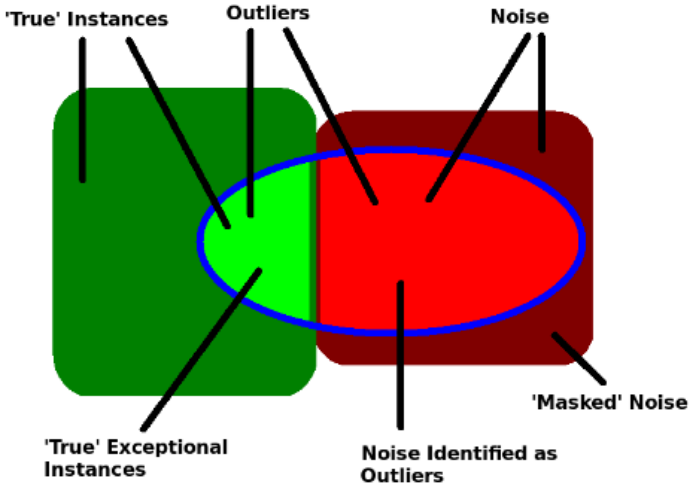
A Figura 2 demonstra um conjunto de dados que pode ser representado como a divisão de dois componentes: instâncias verdadeiras (*'True' instances*) e ruído (*noise*). A presença de *outliers* pode ser identificada nos dois componentes do conjunto.

Segundo Chandola *et al.* (2009), existem vários empecilhos no processo de detecção de anomalias, como:

- Dados que podem ser considerados anomalias, ou não, dependendo do domínio;
- Identificação do limite ideal de onde será considerado a região do comportamento normal e anormal dos dados;
- Dados normais podem se desenvolver bastante e sair dos limites de aceitação estabelecidos anteriormente;
- É difícil distinguir dados anômalos e com ruídos.

Além disso, de acordo com Goldstein e Uchida (2016), para a detecção de anomalias automática é possível utilizar, basicamente, três tipos de configuração:

Figura 2 – Conjunto dos dados normais e ruídos



Fonte: Liebchen ().

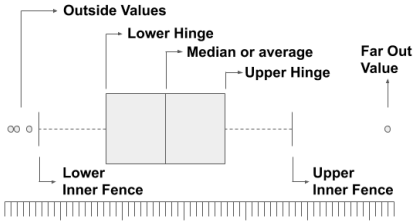
- Detecção de anomalia supervisionada: a qual descreve a configuração em que os dados incluem conjuntos de dados de treinamento e teste totalmente rotulados.
- Detecção de Anomalia Semi-supervisionada: que utiliza-se conjuntos de dados de treinamento e teste, pois os dados de treinamento consistem apenas em dados normais, sem quaisquer anomalias.
- Detecção de Anomalia não Supervisionada: que não é requerido rótulos; não há distinção clara entre um conjunto de dados de treinamento e um de teste.

Segundo Chandola *et al.* (2009), existem várias abordagens que permitem a identificação de anomalias nos dados, cada abordagem possui numerosos algoritmos que dependem das características dos dados, onde é possível classificar os diferentes tipos de abordagens de detecção de anomalias, que utilizam técnicas como, classificação, vizinho mais próximo, *cluster*, estatístico, teórico e espectral de informação. As técnicas estatísticas, por exemplo, tem como pressuposto que anomalias ocorram em regiões de baixa probabilidade, enquanto os dados normais ocorram em regiões de alta probabilidade do modelo estocástico, ou seja, caso um teste de inferência estatística determine que uma nova instância não pertence ao modelo, esta é declarada anômalo. (CHANDOLA *et al.*, 2009).

Este trabalho trata de técnicas detecção de anomalias estatísticas não supervisionada, ou seja, não será utilizado dados de treinamento com rótulos. Será considerado que os dados provenientes dos sensores ocasionalmente dispõem de anomalias. Cerca de seis algoritmos estatísticos serão apresentados, Box Plot, 6-Sigma, Teste de Grubbs, HBOS, KDE e X^2 *Measure*.

2.1.1 Box Plot

Figura 3 – Representação do *Box Plot*



Fonte: Autoria própria.

O *Box Plot*, demonstrado na Figura 3, possui um caixa que representa um intervalo de valores entre o *Upper Hinge* e o *Lower Hinge*, que representam do primeiro ao terceiro quartil, respectivamente. A linha vertical no interior da caixa representa o valor médio (*Average*) ou mediano (*Median*) da distribuição dos dados. A Equação 2.1 demonstra que valor do *Step* é obtido através de 1,5 vezes o valor do H-*Spread*.

$$Step = 1,5 \ * H - Spread \tag{2.1}$$

Subsequentemente o *H-Spread* é calculado a partir da diferença dos valores dos *Hinges*, como demonstrado na Equação 2.2.

$$H - Spread = UpperHinge - LowerHinge \tag{2.2}$$

Às linhas verticais na extremidade da caixa são as *inner fences*. O valor da *Lower Inner Fence* é calculada a partir da subtração do *Lower Hinge* pelo *Step* demonstardo da Equação 2.3.

$$LowerInnerFence = LowerHinge - Step \tag{2.3}$$

E o valor da *Upper Inner Fence* é calculado a partir da adição do *Upper Hinge* ao *Step*, demonstrado na Equações 2.4.

$$UpperInnerFence = UpperHinge + Step \tag{2.4}$$

Os valores que estão entre a *Lower Inner Fence* e o *Lower Hinge* ou entre a *Upper Inner Fence* e o *Upper Hinge*, são chamados de valores adjacentes, ou seja, que apresentam um comportamento normal em relação ao conjunto de dados. Há, também, duas linhas verticais ocultas, chamadas *Lower Outer Fence* e *Upper Outer Fence*, que separa os dados entre *outsiders* e *far out*, embora ambos sejam considerados anomalias (TUKEY, 1977).

Duas linhas horizontais, que não são apresentadas na imagem, são a *Lower Outer Fence* que é calculada a partir da subtração do *Lower Hinge* por duas vezes o valor do *Step*, e a *Upper Outer Fence* que é calculada a partir da soma do *Upper Hinge* por duas vezes o *Step* (TUKEY, 1977).

Os valores que estão entre o *Lower Inner Fence* e o *Lower Outer Fence* ou entre *Upper Inner Fence* e o *Upper Outer Fence*, são consideradas outsiders. Os valores que são menores que *Lower Outer Fence* ou maiores que o *Upper Outer Fence*, são considerados *far out*. (TUKEY, 1977).

2.1.2 6-Sigma

O processo *6-Sigma* é uma metodologias para a solução de problemas e se concentra na otimização e na mudança cultural dos envolvidos, tendo como objetivo evitar perdas e obter resultados significativos em um menor período de tempo, através da utilização de um conjunto de ferramentas rigorosas. Uma dessas ferramentas é o *Statical Process Control* (SPC), que busca enfocar desvios dos valores normais do processo evitando que a qualidade do produto seja comprometida. (RAISINGHANI *et al.*, 2005).

$$UpperControlLimit = \mu + 3\sigma \tag{2.5}$$

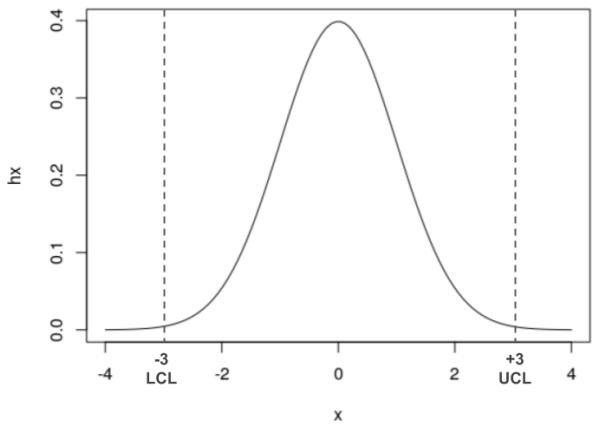
$$LowerControlLimit = \mu - 3\sigma \tag{2.6}$$

Shewhart (1931) desenvolveu um gráfico chamado de Carta de Controle, uma técnica simples para o controle de processos estatístico (SPC) utilizado, também, para a detecção de anomalias. A carta de controle define que todas as instâncias de dados que possuam valor maior que 3 vezes o desvio padrão de distância da distribuição média estão fora de controle ou são considerados anomalias, onde σ é o desvio padrão e α é o valor médio da distribuição, como mostrado nas Equações 2.5 e 2.6. A Figura 4 ilustra a região de controle onde contém 99,7% das instâncias de dados de um conjunto com distribuição normal.

2.1.3 X^2 Measure

Ye e Chen (2001) propuseram uma técnica de detecção de anomalias multivariadas baseada em uma estatística qui-quadrada. Essa técnica parte do teorema do limite central, ou seja, quando o número de atributos é grande o suficiente (isto é, maior que 30), a soma das

Figura 4 – Limites do controle de processo estatístico



Fonte: Autoria própria.

diferenças quadráticas entre o valor observado e o valor esperado (média) dos atributos possui, aproximadamente, uma distribuição normal.

$$X^2 = \sum_{i=1}^{N} \frac{(X_i - \overline{X_i})^2}{\overline{X_i}} \tag{2.7}$$

A Equação 2.7 proposta por Ye e Chen (2001) demonstra que é possível medir a distancia de um determinada instância para as demais, onde X_i é o valor real e $\overline{X_i}$ é o valor médio do i-ésimo atributo da instância que deseja avaliar e N é o número de atributos do conjunto de dados, na qual se uma determinada instância obtiver os maiores valores de X^2, isto indica que essa instância é uma anomalia.

$$\text{Upper Control Limit} = \overline{X^2} + 3S_X^2 \tag{2.8}$$

Os limites de controle para detecção de anomalias podem ser configurados para aplicar parâmetros de controle 6-Sigma (seção 2.1.2). Como o objetivo é detectar os maiores valores X^2, fez-se necessário definir apenas o limite de controle superior (*Upper Control Limit*). Isto é, um limite que se a saída X^2 calculada a partir de uma determinada instância for maior que este limite, definido pela soma da média mais 3 vezes o desvio padrão na Equação 2.8, então essa instância é considerada uma anomalia (YE; CHEN, 2001).

2.1.4 Teste de Grubbs

O Teste de Grubbs ou Teste de Valor Extremo, proposta por Grubbs (1969), é utilizado em dados univariados que possuem uma distribuição normal, onde seu procedimento consiste em detectar uma anomalia por vez, eliminando-a do conjunto de dados, repetindo esse

procedimento até não ser mais detectada anomalias no conjunto de dados.

$$G = \frac{\max_{i=1,\ldots,N} \left| X_i - \overline{X} \right|}{S} \tag{2.9}$$

O Teste de Grubbs (*double-sided*), ilustrado na Equação 2.9, o qual G é o maior desvio absoluto da média amostral, onde o $\overline{X}$ e S são, respectivamente, a média e o desvio padrão da amostra de tamanho N. Para a detecção de anomalias é verificado se G satisfaz determinado nível de significância, α (GRUBBS, 1969).

$$G > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/(2N),N-2}}{N-2+t^2_{\alpha/(2N),N-2}}} \tag{2.10}$$

A hipótese, para o Teste de Grubbs, de que não há anomalias no conjunto de dados é rejeitada com um nível de significância α, de acordo com a Equação 2.10, onde $t^2_{\alpha/(2N),N-2}$ é o valor critico da distribuição-t com (N/2) graus de liberdade e um nível de significância $\alpha/(2N)$ (GRUBBS, 1969).

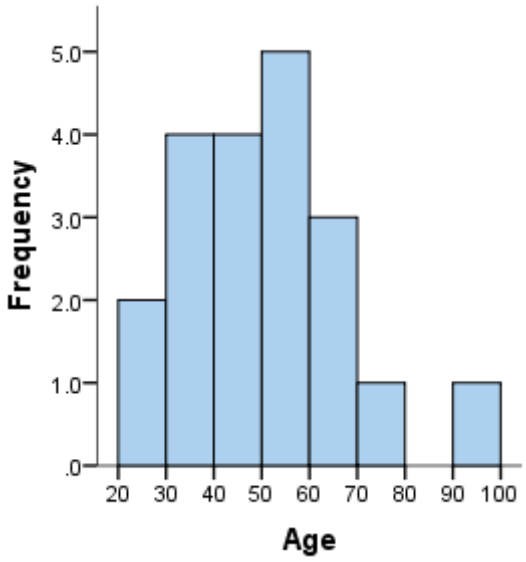
2.1.5 Histograma

O histograma é uma técnica básica que pode ser utilizada para detecção de anomalias em dados univariados, onde seu processo consiste em duas etapas. A primeira etapa envolve a construção de um histograma baseado nos diferentes valores obtidos nos dados de treinamento, dividindo cada instância entre o número de caixas previamente definidas e, na segunda etapa, a técnica verifica se uma instância de teste está em alguma caixa vazia, rara ou inexistente, caso esteja, essa instância será considerada uma anomalia (CHANDOLA *et al.*, 2009).

A Figura 5 exemplifica um histograma criado a partir da frequência dos dados contido no seguinte conjunto de idades: {36, 25, 38, 46, 55, 68, 72, 55, 36, 38, 67, 45, 22, 48, 91, 46, 52, 61, 58, 55}, é possível ver que as caixas criadas para idades acima de 70 anos possuem um frequência (*Frequency*) baixa ou nula, assim, de acordo com esse conjunto de dados, caso uma nova idade (*Age*) seja inserida no conjunto de dados e possuir um valor acima de 70 anos, pode ser considerada uma anomalias.

O *Histogram Based Outlier Score* (HBOS) é uma combinação de métodos univariados que não é capaz de modelar dependências entre atributos. No qual, para cada instância *p* e para cada atributo *d*, um histograma individual é calculado, em que a altura de cada caixa representa uma estimativa de densidade, onde a soma das alturas de cada caixa é igual a 1,0, garantindo um peso igual de cada característica à pontuação de anomalia (GOLDSTEIN; DENGEL,

Figura 5 – Exemplo de histograma de uma distribuição de idades



Fonte: Lund e Lund (2018)

2012).

$$HBOS(p) = \sum_{i=0}^{d} log(\frac{1}{hist_i(p)}) \tag{2.11}$$

A pontuação resultante é uma multiplicação do inverso das densidades estimadas. Entretanto, equivalentemente, em vez de multiplicação, utilizou-se a soma dos logaritmos, como demostrado na Equação 2.12 (GOLDSTEIN; DENGEL, 2012). Segundo Goldstein e Dengel (2012), a aplicação do inverso das densidades torna o algoritmo menos sensível a erros devido à precisão do ponto flutuante em distribuições muito desequilibradas.

$$log(a*b) = log(a) + log(b) \tag{2.12}$$

O limite, para um valor ser considerado uma anomalia, é definido através do q-ésimo percentil dos valores de HBOS calculados para cada instâncias do conjunto de dados, onde o valor de q é calculado através da Equação 2.13. Caso o valor do HBOS para instância p seja maior que o limite calculado, a instância p será considerada uma anomalia. *Contaminação* é um valor entre 0 e 1 fornecido pelo usuário, que pode ser representado como a porcentagem de dados anômalos no conjunto.

$$q = 100 * (1 - Contaminação) \tag{2.13}$$

2.1.6 Kernel Density Estimate

A estimativa de densidade do *kernel* (KDE) é uma forma de estimar a função de densidade de probabilidade de uma instância aleatória, em conjuntos de dados que possuem

distribuição desconhecida, fundamentalmente utilizado para suavização de dados, podendo ser aplicado para estimar riscos ou taxa condicional de falha (PARZEN, 1962).

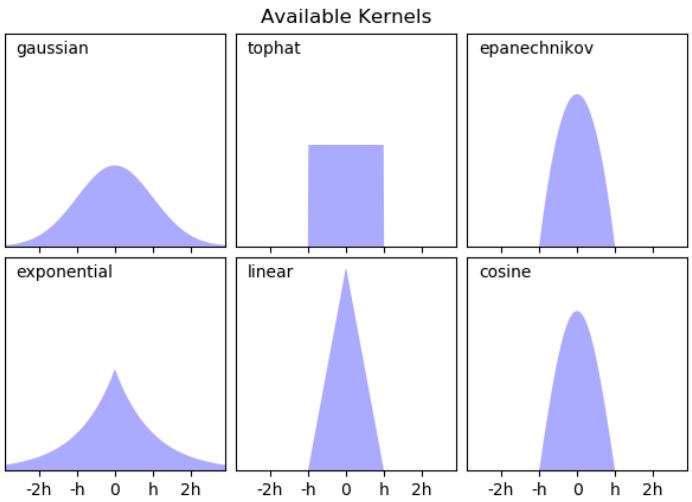
O KDE é semelhante ao histograma em relação a construção dos perfis de densidade, embora a principal diferença seja a construção de uma versão mais suave do perfil de densidade. O valor da densidade em um determinado ponto é estimado como a soma dos valores suavizados das funções do *kernel* $K_h(\cdot)$ associadas a cada ponto no conjunto de dados, podendo ser considerado como uma soma de 'colisões' de instâncias no conjunto. Cada função do *kernel* está associada a uma largura da janela h que determina o nível de suavização criado pela função, desempenhando um papel semelhante à largura da caixa no caso do histograma, e uma instância aleatória $\overline{X}$ do conjunto de tamanho n (AGGARWAL, 2015; DESFORGES *et al.*, 1998).

De acordo com Aggarwal (2015), a função de estimativa do kernel $K_h(\cdot)$ é definida da seguinte forma:

$$f(\overline{X}) = \frac{1}{n} \cdot \sum_{i=1}^{n} K_h(\overline{X} - \overline{X}_i) \tag{2.14}$$

Segundo Vanderplas (2007), a biblioteca *scikit-learn*, presente na linguagem de programação *Python*, implementa diversos tipos de *kernels* para lidar com diferentes situações. As expressões matemática utilizadas em cada um dos *kernels* disponíveis são as seguintes:

Figura 6 – Distribuição dos diferentes tipos de *kernels*



Fonte: Vanderplas (2007)

- Gaussiano ou *Gaussian*, sua distribuição é ilustrada na Figura 6, no primeiro quadro da parte superior, calculada de acordo com a Equação 2.15.

$$K_h(x) = exp(-\frac{x^2}{2h^2}) \tag{2.15}$$

- *Tophat*, o qual a Figura 6 ilustra a sua distribuição, no segundo quadro da parte superior, obtido a partir da Equação 2.16.

$$K_h(x) = \begin{Bmatrix} 1, & \text{se } x < h \\ 0, & \text{caso contrario} \end{Bmatrix} \tag{2.16}$$

- *Epanechnikov*, calculado a partir da equação 2.17 de forma que sua distribuição é representada pelo terceiro quadro da parte superior, na Figura 6.

$$K_h(x) = 1 - \frac{x^2}{h^2} \tag{2.17}$$

- Exponencial ou *Exponential*, é representado pela distribuição contida no primeiro quadro da parte inferior da Figura 6 e determinada de acordo com a Equação 2.18.

$$K_h(x) = exp(\frac{-x}{h}) \tag{2.18}$$

- Linear, possui sua distribuição calculada a partir da Equação 2.19 e traçada de acordo com o segundo quadro da Figura 6.

$$K_h(x) = \begin{Bmatrix} \frac{1-x}{h}, & \text{se } x < h \\ 0, & \text{caso contrario} \end{Bmatrix} \tag{2.19}$$

- Cosine, onde sua distribuição ilustrada no terceiro quadro da parte inferior da Figura 6 e é definida de acordo com a Equação 2.20.

$$K_h(x) = \begin{Bmatrix} cos(\frac{\pi x}{2h}), & \text{se } x < h \\ 0, & \text{caso contrario} \end{Bmatrix} \tag{2.20}$$

A pontuação de anomalias é representada pelo valor da densidade, ou seja, valores baixos de densidade têm maior probabilidade de serem anomalias. Essa técnica possui uma menor eficacia de acordo com o aumento do número de dimensões (AGGARWAL, 2015).

2.2 Aprendizado de máquina

Nas últimas décadas, com a crescente complexidade dos problemas e o grande volume de dados gerados, tornou se clara a necessidade de melhores técnicas de Inteligência Artificial que reduzisse a necessidade de intervenção humana e dependência de especialistas, que era como funcionava até o momento (CARVALHO *et al.*, 2011).

Os modelos estatísticos começaram a perder força com a advinda dos modelos de AM, que chamaram a atenção devido a utilização, somente, de valores de dados históricos para

aprender a dependência estocástica e prever os valores dos dados futuros (BONTEMPI *et al.*, 2012).

Esses modelos aplicavam de técnicas que descreviam o comportamento dos dados por meio de hipóteses e funções matemáticas. Técnicas, estas, denominadas indução, que a partir de uma base particular de exemplos podem-se extrair conclusões genéricas (CARVALHO *et al.*, 2011).

A AM é uma área interdisciplinar, que tem como foco extrair informações em bases de dados de maneira automática, envolvendo conceitos de estatística, inteligência artificial, filosofia, teoria da informação, biologia, ciências cognitivas, complexidade computacional, teoria de controle, etc. Os modelos de AM são capazes de se adaptar a ambientes dinâmicos de acordo com a base de dados de exemplo, podendo mudar o seu comportamento a fim de melhorar o resultado de suas previsões (FERRARI; SILVA, 2017).

Segundo Corrigan (),o crescimento do uso de modelos de AM deu-se pelo aprimoramento da capacidade de processamento do computador, avanços no desenvolvimento de algoritmos e da expansão do volume de dados coletados. Podendo ser aplicado a uma ampla gama de problemas, como a detecção de *spam* por email, o reconhecimento óptico de caracteres, a classificação de imagens, diagnósticos médicos, tradução automática, etc.

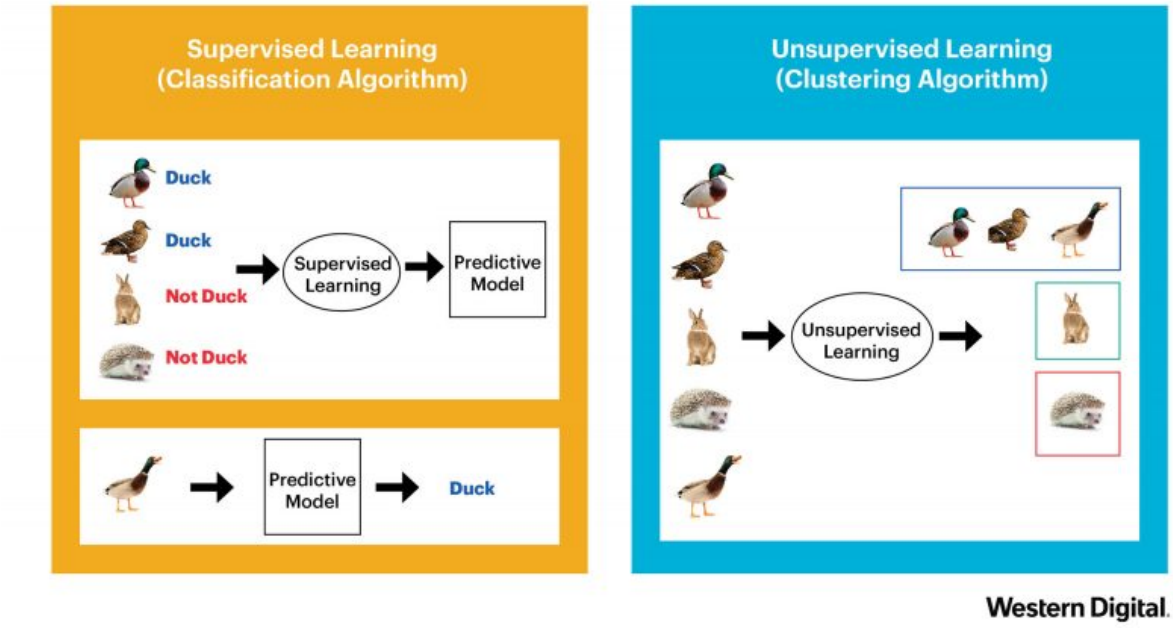
De acordo com Ferrari e Silva (2017), 'paradigma de aprendizagem' é a maneira pela qual a base de dados intervém no modelo a ser utilizada. Os dois mais comum são:

- Aprendizado supervisionado: baseado na aprendizagem de uma base de dados com valores de saída conhecidos.
- Aprendizado não supervisionado: baseado na aprendizagem de uma base de dados sem valores de saída pré-definidos, no qual serão categorizados.

É possível observar na Figura 7 as duas abordagens, o 'Aprendizado supervisionado' (*Supervised Learning*) que a partir de vários animais da base de dados com dois valores nominais de saída: pato (*Duck*) e o não pato (*Not Duck*), é possível classificar se um novo animal é '*Duck*' ou '*Not Duck*'; e o 'Aprendizado não supervisionado' (*Unsupervised Learning*) que a partir de vários animais da base de dados e sem valores de saída conseguem descrever quais animais são os mais similares e os caracterizam em três classes anônimas distintas.

Uma vez que um modelo do mapeamento, entre um conjunto de variáveis de entrada e uma ou mais variáveis de saída, esteja disponível, um algoritmo de AM poderá ser utilizado para aprendizagem supervisionada (BONTEMPI *et al.*, 2012). Segundo Fawcett e Provost (2018),

Figura 7 – Paradigmas de aprendizagem



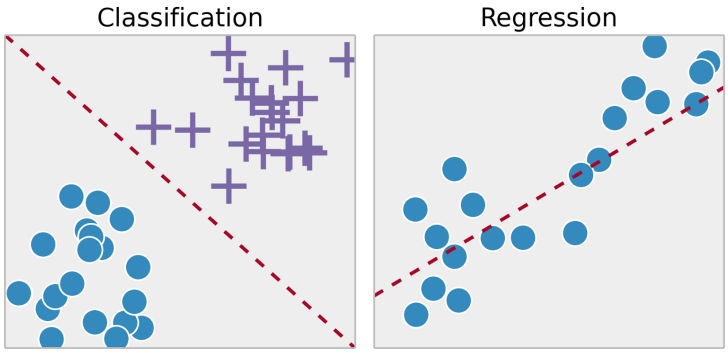
Fonte: Zhou (2018).

apesar da existência de números modelos de AM atualmente, há apenas um pequeno grupo de tarefas distintas em que esses algoritmos podem abordar, por exemplo:

- Regressão: tenta estimar ou prever um valor numérico de uma determinada instância.
- Classificação: tenta prever a qual conjunto de classes uma instância pertence.

É possível observar na Figura 8 as duas abordagens: a 'Classificação' (*Classification*) que divide os dados em dois conjuntos distintos, e a 'Regressão' (*Regression*) que produz uma função linear que mais se adéqua aos dados.

Figura 8 – Abordagens de previsão



Fonte: Soni (2018).

A *Support Vector Machine* (svm) e o *Multilayer Perceptron* (mlp) são alguns dos algoritmos de aprendizado de maquina aplicados para regressão em conjuntos de dados, esse

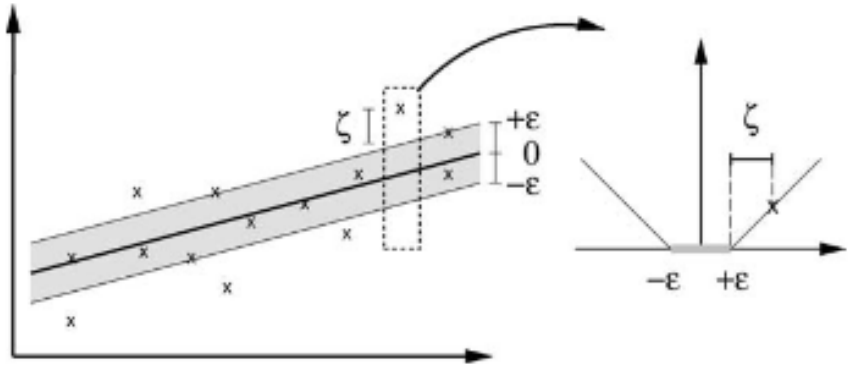
algoritmos empregam o aprendizado supervisionado para realizar suas previsões.

2.2.1 Support Vector Machine

A SVM mapeia, de forma não linear, os vetores de entrada para um espaço de recurso de grande dimensão, posteriormente, é construída uma superfície de decisão linear com alta capacidade de generalização. A SVM permite uma expansão do vetor de solução em vetores de suporte, estende as superfícies de solução linear para não linear e permite erros mínimos no conjunto de treinamento (CORTES; VAPNIK, 1995).

Será necessário o manuseio da *Support Vector Regression* SVR, ou seja, uma variação da SVM para regressão. A Figura 9 demonstra graficamente o funcionamento da SVR, que o x representa instâncias do conjunto de dados, ε é a distância máxima aceitável em relação a função linear ótima traçada, e a variável de folga ζ é a distância das instâncias fora da região aceitável até a região aceitável. Apenas os pontos fora da região sombreada contribuem para o custo da função, à medida em que os desvios são penalizados de forma linear (SMOLA; SCHÖLKOPF, 2004).

Figura 9 – Função linear SVR



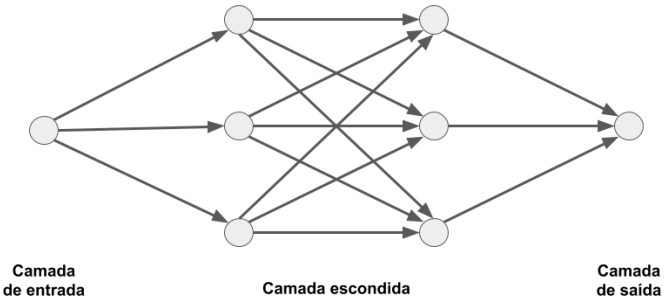
Fonte: Smola e Schölkopf (2004).

2.2.2 Multilayer Perceptron

O MLP consiste em uma rede de nós separados por três camadas, com cada nó conectado, por pesos e sinais de saída, a cada nó na camada seguinte e anterior, onde a saída de um nó alimenta as entrada da próxima camada da rede. Os pesos e sinais são a soma das entradas modificadas por uma função de transferência ou ativação, onde a função mais utilizada é a logística. A Figura 10 demonstra às três camadas que compõem a rede: a camada de entrada

(*Input layer*), que serve apenas para passar o vetor de entrada para o resto da rede; a camada escondida (*Hidden layer*), que pode possuir um ou mais nós; e a camada de saída (*Output layer*), que possui somente um nó (GARDNER; DORLING, 1998).

Figura 10 – Camadas do MLP



Fonte: Autoria própria.

Como o MLP é um algoritmo supervisionado, é requerido um treinamento, que consiste num conjunto de dados de vetores de entrada e de saída associados. Durante o treinamento, o *perceptron* multicamadas é repetidamente apresentado aos dados de treinamento para decidir os melhores pesos para a rede. Para cada treinamento um sinal de erro é produzido, definido como a diferença entre a saída prevista e a real, de modo que através do valor desse sinal de erro é possível determinar para que medida os pesos na rede devem ser ajustados, com o objetivo de minimizar o erro da previsão (GARDNER; DORLING, 1998).

O algoritmo de treinamento de retro-propagação usa a técnica de gradiente descendente para tentar localizar o mínimo absoluto (ou global) da superfície de erro. Os pesos na rede são inicialmente definidos para pequenos valores aleatórios e, assim, o algoritmo de retro-propagação calcula o gradiente local da superfície de erro e altera os pesos na direção do gradiente local mais acentuado. Com uma superfície de erro razoavelmente suave, espera-se que os pesos se dirijam pelo mínimo global da superfície de erro, minimizando-o (GARDNER; DORLING, 1998).

3 METODOLOGIA

Neste capítulo são apresentados quais os tipos de dados foram coletados para a previsão 3.1); a seção 3.2) demonstra quais os algoritmos de detecção de anomalias e previsão foram selecionados para a avaliação dos resultados; a seção 3.3 explica quais as métricas foram selecionadas para análise, examinação e verificação dos resultados obtidos pela mudança de cada parâmetros; Por fim, é explicado como foi feito o processo de avaliação do desempenho relativo dos algoritmos de aprendizado de máquina com o objetivo de classificar os algoritmos de detecção de anomalias (seção 3.4).

3.1 Extração dos dados

Os dados utilizados neste estudo foram extraídos de um banco de dados de uma fábrica que produz cimento, os dados persistidos no banco são oriundos de um conjunto de sensores que ficam acoplados a um moinho de cimento. Os dados são coletados a cada 30 segundos, existem 93 atributos numéricos diferentes, mas somente alguns atributos serão utilizados para prever as classes objetivo.

A partir da analise de correlação e entrevistas com especialistas, dos 93 atributos, foram selecionados apenas três atributos como classes objetivo, ou seja, atributos que serão utilizados como rótulos para a previsão dos algoritmos de AM. A seleção dessas três classes reduz a possibilidade de enviesar o experimento, possibilitando fazer os testes com quantidades e tipos diferentes de atributos causais.

Portanto, adiante e após a remoção dos dados considerados anômalos pelos algoritmos de detecção de anomalias, será feita uma regressão por meio dos algoritmos de aprendizagem de maquina para investigar se houve uma melhora na acurácia da previsão dessas classes objetivo.

A Tabela 2 apresenta o nome das classes cujos algoritmos de aprendizado de máquina deverão prever e o número de atributos que provocam alguma consequência àquela classe. Esses atributos possuem uma relação de causalidade, ou seja, acontecem de acordo com um dado grau de sincronicidade, em conformidade com a classe objetivo.

- A classe '*mill_motor_pwr_kw_pv*' mensura a potência do motor do moinho em Quilowatt e possui cerca de 38 atributos.
- A classe '*bucket_elv_mtr_pwr_kw_pv*' mensura a potência do motor do elevador que transporta rejeito em Quilowatt e possui cerca de 8 atributos.

- A classe '*mill_exit_temp_c_pv*' mensura a temperatura de saída do moinho em grau Celsius e possui cerca de 40 atributos.

Tabela 2 – Classes para previsão

Classes objetivo	**Número de atributos**
mill_motor_pwr_kw_pv	38
bucket_elv_mtr_pwr_kw_pv	8
mill_exit_temp_c_pv	40

Fonte: Autoria própria.

3.2 Seleção dos algoritmos

Durante o planejamento e definição das técnicas, tanto para a detecção de anomalias quanto para previsão dos dados, foi priorizado os algoritmos disponíveis nas bibliotecas da linguagem a fim de obter algoritmos mais otimizados e robustos. As subseções à seguir apresentam os algoritmos utilizados para a limpeza (seção 3.2.1) e para a previsão (seção 3.2.2) dos dados.

3.2.1 Detecção de anomalias

As características de cada técnica implicam em desvantagens específicas, tais como: aquelas baseadas na classificação e em vizinhos mais próximos, necessitam de dados rotulados para treinamento; baseado em *cluster* e espectral, geralmente requer alta complexidade computacional; aqueles baseados em informações teóricas frequentemente detectam a presença de anomalias somente quando há um grande número de anomalias presentes nos dados. Conjuntos de dados com distribuição normal e sem rótulos para treinamento são adequados para as técnicas estatísticas (CHANDOLA *et al.*, 2009).

A detecção de anomalias em redes de sensores representa um conjunto de desafios únicos. As técnicas de detecção de anomalias são necessárias para operar em uma abordagem online. Devido a restrições severas de recursos, as técnicas de detecção de anomalias precisam computar em um tempo desprezível (CHATZIGIANNAKIS *et al.*,).

Por serem de fácil implementação, possuírem um baixo tempo de execução, não exigirem rótulos nos dados e não requererem um grande número de anomalias, este trabalho tem se limitado a analisar técnicas estatísticas, ou seja, que detectam anomalias através de formulas matemáticas.

3.2.2 Aprendizado de máquina

Algoritmos de aprendizado de máquina foram necessários para a medição da eficácia dos algoritmos de detecção de anomalias, partindo do pressuposto: Quanto mais limpo, ou seja, mais consistente for o conjunto de dados de treinamento, maior será a proximidade entre o valor obtido experimentalmente e o valor coletado pelos sensores das classes objetivo. No qual o "valor obtido experimentalmente" refere-se ao valor previsto pelos algoritmos de aprendizado de máquina.

A partir de uma série de comparações feitas por Vanschoren *et al.* (2012), avaliando diversos algoritmos, de aprendizado supervisionado, foi identificado que a SVM e o MLP (abordados nas seções 2.2.1 e 2.2.2) estão entre os que possuem melhor desempenho, a partir de comparações tanto métricas individuais, como acurácia da previsão, medida F, precisão e recordação; quanto métricas coletivas, como *Root Mean Square Deviation* (RMSE) , média das métricas individuais e teste de Friedman.

3.3 Métodos de avaliação

Métricas de avaliação foram necessárias para a medição da eficiência dos algoritmos de detecção de anomalias. Para indicar qual a técnica é a mais adequada para o conjunto de dados em questão, foi medido o tempo de execução dos algoritmos, a precisão estatística calculada por meio do erro e a quantidade de dados que são considerado anomalias.

O objetivo das métricas de desempenho é comparar dois valores fornecendo uma pontuação quantitativa que descreva o grau de similaridade entre eles (WANG; BOVIK, 2009; KHAIR *et al.*, 2017). Ou seja, neste trabalho, é comparado o valor coletado pelos sensores e o valor previsto pelos algoritmos de AM.

Segundo Hyndman e Koehler (2006), muitas medidas de exatidão e previsão foram propostas no passado e geralmente são pouco aplicáveis, podendo produzir resultados enganosos, desse modo, mediante observações empíricas, recomendam o MASE para a comparação das precisões de previsão. Medidas existentes ainda podem ser preferidas nos seguintes casos:

- Caso todas as instâncias estiverem na mesma escala, o *Mean Absolute Error* (MAE) pode ser preferido porque é mais simples de utilizar, entretanto o conjunto de dados utilizado possui atributos com escalas bem divergente um dos outros.
- Caso todos os dados forem positivos e muito maiores que zero, o *Mean Absolute Percentage*

Error (MAPE) ainda pode ser preferido por motivos de simplicidade.

- Caso haja situações em que existem escalas muito diferentes, incluindo dados próximos de zero ou negativos, o MASE é a melhor medida disponível de precisão de previsão, demonstrado na Equação 3.1, onde Y_t é o valor real coletado, F_t é o valor previsto e e_t é o valor do erro calculado da seguinte forma, $e_t = Y_t \ \ F_t$.

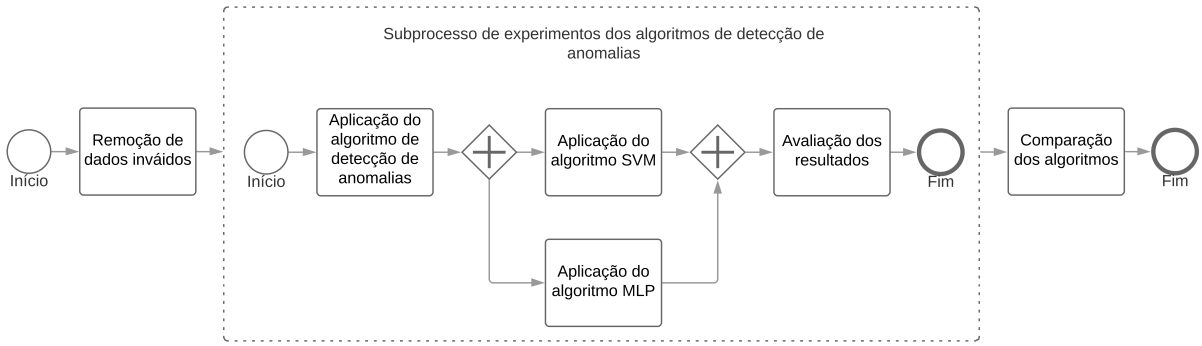
$$MASE = \frac{\sum_{t=1}^{T} |e_t|}{\frac{1}{T-1}\sum_{t=2}^{T} |Y_t - Y_{t-1}|} \tag{3.1}$$

3.4 Experimentos realizados

O objetivo da análise de desempenho é a avaliar as características de performance de um sistema de computador, tanto na parte de *software* quanto na parte de *hardware*. No qual é possível caracterizar como o desempenho de um sistema muda à medida que determinados técnicas ou parâmetros são variadas (LILJA, 2005).

Para determinar o impacto de um recurso, é necessário a comparação com cada alteração feita no sistema, o antes e o depois, e a troca por outros recursos. No qual o objetivo é identificar o fator que mais contribui com o desempenho geral do sistema. Neste caso, os recursos equivalem aos algoritmos de detecção de anomalias, e o desempenho é avaliado e comparado por meio da previsão dos algoritmos de aprendizado de máquina.

Figura 11 – Processo de avaliação de algoritmos de detecção de anomalias



Fonte: Autoria própria.

A Figura 11 demonstra como foi realizado o experimento, ou seja, o processo de avaliação dos algoritmos de detecção de anomalias propostos. O processo se inicia-se após a extração dos dados, seguindo para a remoção dos dados inválidos e, a partir desse momento, um subprocesso é repetido para cada um dos algoritmos de detecção de anomalias avaliados neste trabalho. Um experimento é definido como uma tupla contendo três elementos, ou seja, é a

combinação de um algoritmo de limpeza, um algoritmo de aprendizado de máquina e uma classe. Como pode ser observado no Quadro 1, neste trabalho foram experimentados 6 algoritmos de detecção de anomalias, envolvendo as previsões obtidas por 2 algoritmos de AM e utilizando 3 classes diferentes.

A tarefa de remoção dos dados inválidos é realizada através da remoção de instâncias que possuem dados com erro e instâncias em que o moinho não esteja em funcionamento, do conjunto de dados.

A aplicação do algoritmo de detecção de anomalias é realizada através da utilização de um dos algoritmos propostos na seção 3.2.1, no qual dados que se diferenciam drasticamente dos outros são removidos do conjunto de dados, para não prejudicar a acurácia das previsões.

As tarefas de aplicação dos algoritmos SVM e MLP são realizadas paralelamente. Os algoritmos de aprendizado de máquina são aplicados em subdivisões do conjunto de dados contendo cada uma das classe com seus atributos, com o objetivo de prever as classes propostas na Tabela 2.

A avaliação dos resultados é a tarefa que realiza as medições propostas na seção 3.3 para cada um dos algoritmos de aprendizado de máquina, reservando os resultados para posterior analise. Esse subprocesso, apresentado da Figura 11, é reexecutado até todos os algoritmos de detecção de anomalias sejam avaliados.

Após a realização de todos os experimentos, demonstrado no Quadro 1, é feito a comparação dos resultados das medições, onde cada uma das métricas de avaliação são ponderadas com um grau de importância e, posteriormente, são classificadas de acordo com o desempenho de cada algoritmo de detecção de anomalias.

3.5 Código fonte

O código utilizado para fazer os experimentos está disponível no GitHub, o qual é possível acessar através do link github.com/alexfrederico/anomaly-detection-benchmarking.

4 RESULTADOS E DISCUSSÕES

A máquina utilizada para os experimentos possui o sistema operacional *Debian* 9 de 64 bits, *kernel Linux*, e um processador *Intel*® *Core*™ i5-4570 de 4ª geração, com frequência de 3.20GHz, 4 núcleos, 6MB de memória cache e 16GB de memória RAM do tipo DDR3.

4.1 Remoção de dados inválidos

Para os experimentos foram utilizados dados coletados pelos sensores durante um período de 10 dias, um total de 23.767 instâncias foram extraídos do banco de dados dos sensores. Registros que possuíam erros em algum dos atributos ou que foram coletados sem o moinho estar funcionando foram removidos do conjunto de dados, restando, assim, cerca de 14.987 instâncias sem erros.

Como os dados são coletados a cada 30 segundos, cerca de 10 instâncias foram removidas do conjunto total, sobrando cerca de 14.977 instâncias, com o objetivo que os algoritmos de aprendizado de máquina prevejam a tendência dos dados para 5 minutos a frente sem a necessidade de utilização dos atributos causais. Os 10 dados removidos serão reaproveitados para comparar com a tendência prevista.

Das 14.977 instâncias sem erro, foram selecionadas cerca de 80% das instâncias mais antigas para o conjunto de treinamento dos algoritmos de AM, de acordo com a data e a hora da coleta. Ou seja, um total de 11.981 instâncias foram utilizados para treinamento.

Foi utilizado somente os dados mais antigos para o treinamento com o intuito de simular os dados coletados pelos sensores, pois em uma aplicação real só há possibilidade de dados anteriores preverem dados posteriores.

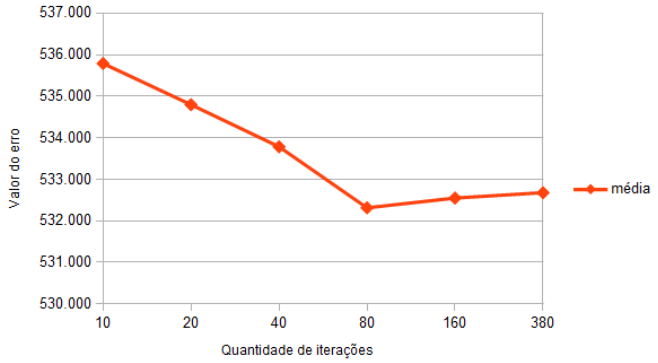
4.2 Resultados dos experimentos

Após a remoção das anomalias nos dados de treinamento, os algoritmos de AM são necessários para prever os dados futuros. Como o SVM é um algoritmo não estocástico, foi preciso realizar somente uma execução para obter o valor de MASE dos dados previstos. Entretanto, o MLP é um algoritmo estocástico, por conta disso, foi necessário utilizar a média dos valores de MASE obtidos a partir de 160 iterações realizadas.

O número de iterações ideal para obter um valor médio de MASE, usando o MLP, foi determinado a partir de uma série de testes produzidos com mudanças no número de iterações,

com o intuito de encontrar a quantidade de iterações necessárias para que o MASE tenha um valor médio com poucas variações, como demonstrado na Figura 12.

Figura 12 – Valor médio do MASE utilizando o MLP



Fonte: Autoria própria.

Os valores obtidos pelos resultados dos experimentos foram definidos como uma tupla contendo três elementos, ou seja, o valor obtido pelo cálculo do MASE, o tempo de execução dos algoritmos de detecção de anomalias e a quantidade de dados de treino que restaram após a remoção das anomalias. O Quadro 1 é a associação das tuplas definidas na seção 3.4 e das tuplas definidas nesta seção, contendo os valores resultante dos experimentos.

Para cada um dos algoritmo de detecção de anomalias experimentado foram feito analises alterando alguns parâmetros para comparar qual o impacto desses parâmetros na detecção de anomalias no conjunto de dados, demonstrado no Quadro 1. Na coluna 'Algoritmo de detecção de anomalias' foram experimentados os seguintes métodos:

- **Nenhum:** Significa que nenhum algoritmo de detecção de anomalias foi utilizado para remover as anomalias contidas no conjunto de dados de treinamento, somente foi realizada a remoção dos dados inválidos, como explicado na seção 4.1
- **Box Plot:** O número presente dentro dos parenteses representa o valor que será multiplicado ao H-Spread, como mostrado na Equação 2.1
- X**-Sigma:** A incógnita X representa o dobro do valor que será multiplicado ao desvio padrão (σ), como mostrado nas Equações 2.5 e 2.6
- X^2 **Measure:** O número presente dentro dos parenteses representa o valor que será multiplicado ao desvio padrão, como mostrado nas Equação 2.8
- **Teste de Grubbs:** O número presente dentro dos parenteses representa o nível de significância, como mostrado nas Equação 2.10
- **HBOS :** O número presente dentro dos parenteses representa o valor da contaminação do

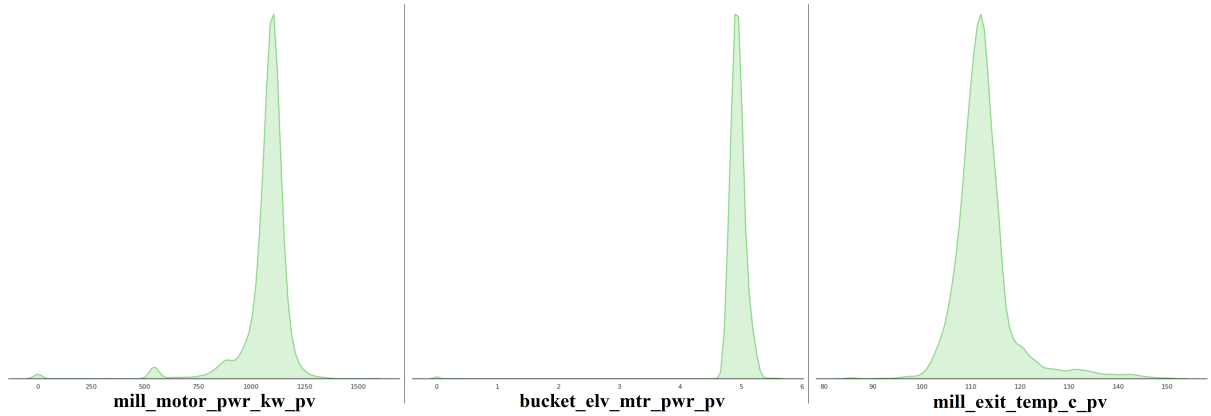
conjunto de dados, como mostrado nas Equação 2.13

- **KDE:** A tupla presente dentro dos parenteses, contendo dois elementos, representa o *kenel* e a quantidade de dados removidos do conjunto em porcentagem, respectivamente, como mostrado nas Equação 2.14

4.3 Comparação dos algoritmos de detecção de anomalias

Para analise dos resultados, foi gerado um histograma da distribuição dos valores da três classes objetivo, mill_motor_pwr_kw_pv, bucket_elv_mtr_pwr_kw_pv e mill_exit_temp_c_pv como é ilustrado na Figura 13, respectivamente. É possível observar que os valores das três classes objetivo podem ser representados por uma curva normal. Observa-se, também, que os valores contidos no histograma da classe mill_motor_pwr_kw_pv e mill_exit_temp_c_pv possuem seus valores mais distribuídos ao longo do eixo X.

Figura 13 – Histograma da distribuição dos valores de cada classe



Fonte: Autoria própria.

Para uma melhor compreensão da distribuição dos dados, o Quadro 2 ilustra tando medidas de tendências central como medidas de dispersão do conjunto antes da remoção de anomalias. No qual, foram analisadas as seguintes medidas: valor mínimo, valor máximo, amplitude, média, Mediana, Desvio Padrão, coeficiente de variação, variância e variância relativa de cada classe, onde é possível perceber nuances estatísticas de cada distribuição.

Examinando o Quadro 2, é razoável afirmar que tanto a classe *bucket_elv_mtr_pwr_kw_pv* quanto a classe *mill_exit_temp_c_pv* possuem valores com uma menor dispersão, onde a classe *bucket_elv_mtr_pwr_kw_pv* também possui valores com uma menor tendência. Consequentemente, identificar quais dados são anomalias na classe *mill_motor_pwr_kw_pv*, que possui grandes valores de tendência e dispersão, torna-se uma tarefa difícil.

Quadro 1 – Experimentos

Cód.	**Algoritmo de detecção de anomalias**	**AM**	**Classe**	**MASE**	**Tempo**	**Dados para treino**
1	Nenhum	SVM	mill_motor_pwr_kw_pv	485428,31	-	11981
2	Nenhum	SVM	bucket_elv_mtr_pwr_kw_pv	814524,19	-	11981
3	Nenhum	SVM	mill_exit_temp_c_pv	6740955,92	-	11981
4	Nenhum	MLP	mill_motor_pwr_kw_pv	514107,65	-	11981
5	Nenhum	MLP	bucket_elv_mtr_pwr_kw_pv	2457933,19	-	11981
6	Nenhum	MLP	mill_exit_temp_c_pv	4844259,45	-	11981
7	Box Plot (1,5)	SVM	mill_motor_pwr_kw_pv	499446,70	5,94	6714
8	Box Plot (1,5)	SVM	bucket_elv_mtr_pwr_kw_pv	996504,85	1,66	8984
9	Box Plot (1,5)	SVM	mill_exit_temp_c_pv	5170602,13	5,18	8647
10	Box Plot (1,5)	MLP	mill_motor_pwr_kw_pv	500210,02	5,91	6714
11	Box Plot (1,5)	MLP	bucket_elv_mtr_pwr_kw_pv	2766548,94	1,68	8984
12	Box Plot (1,5)	MLP	mill_exit_temp_c_pv	4674643,16	5,25	8647
13	Box Plot (1,0)	SVM	mill_motor_pwr_kw_pv	487837,59	5,92	5917
14	Box Plot (1,0)	SVM	bucket_elv_mtr_pwr_kw_pv	1027321,59	1,65	7979
15	Box Plot (1,0)	SVM	mill_exit_temp_c_pv	5168734,90	5,21	8299
16	Box Plot (1,0)	MLP	mill_motor_pwr_kw_pv	516645,98	5,91	5917
17	Box Plot (1,0)	MLP	bucket_elv_mtr_pwr_kw_pv	2821080,54	1,67	7979
18	Box Plot (1,0)	MLP	mill_exit_temp_c_pv	4774625,19	5,24	8299
19	Box Plot (0,5)	SVM	mill_motor_pwr_kw_pv	487881,27	5,88	3480
20	Box Plot (0,5)	SVM	bucket_elv_mtr_pwr_kw_pv	1157145,15	1,64	5666
21	Box Plot (0,5)	SVM	mill_exit_temp_c_pv	4835360,29	5,19	6343
22	Box Plot (0,5)	MLP	mill_motor_pwr_kw_pv	535749,24	5,82	3480
23	Box Plot (0,5)	MLP	bucket_elv_mtr_pwr_kw_pv	2047766,53	1,65	5666
24	Box Plot (0,5)	MLP	mill_exit_temp_c_pv	4706096,37	5,27	6343
25	6-Sigma	SVM	mill_motor_pwr_kw_pv	483536,97	5,19	11311
26	6-Sigma	SVM	bucket_elv_mtr_pwr_kw_pv	842765,24	1,41	11495
27	6-Sigma	SVM	mill_exit_temp_c_pv	7257819,08	4,62	11520
28	6-Sigma	MLP	mill_motor_pwr_kw_pv	543863,66	5,13	11311
29	6-Sigma	MLP	bucket_elv_mtr_pwr_kw_pv	2644877,17	1,41	11495
30	6-Sigma	MLP	mill_exit_temp_c_pv	4944074,86	4,69	11520
31	5-Sigma	SVM	mill_motor_pwr_kw_pv	480308,31	5,14	10337
32	5-Sigma	SVM	bucket_elv_mtr_pwr_kw_pv	864687,64	1,40	11394
33	5-Sigma	SVM	mill_exit_temp_c_pv	6992284,95	4,63	10477
34	5-Sigma	MLP	mill_motor_pwr_kw_pv	538186,56	5,10	10337
35	5-Sigma	MLP	bucket_elv_mtr_pwr_kw_pv	2815452,13	1,41	11394
36	5-Sigma	MLP	mill_exit_temp_c_pv	5246013,22	4,65	10477
37	4-Sigma	SVM	mill_motor_pwr_kw_pv	469546,62	5,06	7986
38	4-Sigma	SVM	bucket_elv_mtr_pwr_kw_pv	940744,06	1,39	9977
39	4-Sigma	SVM	mill_exit_temp_c_pv	5289577,80	4,58	9058
40	4-Sigma	MLP	mill_motor_pwr_kw_pv	521631,15	5,08	7986
41	4-Sigma	MLP	bucket_elv_mtr_pwr_kw_pv	2674827,44	1,40	9977
42	4-Sigma	MLP	mill_exit_temp_c_pv	5017631,60	4,60	9058
43	X^2 Measure (3,0)	SVM	mill_motor_pwr_kw_pv	485335,86	2,90	11885
44	X^2 Measure (3,0)	SVM	bucket_elv_mtr_pwr_kw_pv	824368,29	0,87	11825
45	X^2 Measure (3,0)	SVM	mill_exit_temp_c_pv	6740231,52	2,65	11975
46	X^2 Measure (3,0)	MLP	mill_motor_pwr_kw_pv	512049,66	2,91	11885
47	X^2 Measure (3,0)	MLP	bucket_elv_mtr_pwr_kw_pv	2761198,90	0,86	11825
48	X^2 Measure (3,0)	MLP	mill_exit_temp_c_pv	4994085,20	2,66	11975
49	X^2 Measure (2,0)	SVM	mill_motor_pwr_kw_pv	484337,44	2,88	10783
50	X^2 Measure (2,0)	SVM	bucket_elv_mtr_pwr_kw_pv	823189,69	0,88	11457

Fonte: Autoria própria.

Quadro 1 – Continuação dos Experimentos

Cód.	Algoritmo de detecção de anomalias	AM	Classe	MASE	Tempo	Dados para treino
51	X^2 Measure (2,0)	SVM	mill_exit_temp_c_pv	6845638,71	2,64	10592
52	X^2 Measure (2,0)	MLP	mill_motor_pwr_kw_pv	515728,89	2,89	10783
53	X^2 Measure (2,0)	MLP	bucket_elv_mtr_pwr_kw_pv	2955177,94	0,87	11457
54	X^2 Measure (2,0)	MLP	mill_exit_temp_c_pv	4841115,09	2,64	10592
55	X^2 Measure (1,0)	SVM	mill_motor_pwr_kw_pv	484805,93	2,88	10049
56	X^2 Measure (1,0)	SVM	bucket_elv_mtr_pwr_kw_pv	808829,89	0,88	9862
57	X^2 Measure (1,0)	SVM	mill_exit_temp_c_pv	7205758,09	2,62	10125
58	X^2 Measure (1,0)	MLP	mill_motor_pwr_kw_pv	523086,22	2,88	10049
59	X^2 Measure (1,0)	MLP	bucket_elv_mtr_pwr_kw_pv	3215967,98	0,87	9862
60	X^2 Measure (1,0)	MLP	mill_exit_temp_c_pv	4867446,16	2,61	10125
61	X^2 Measure (0,5)	SVM	mill_motor_pwr_kw_pv	484690,55	2,90	9970
62	X^2 Measure (0,5)	SVM	bucket_elv_mtr_pwr_kw_pv	812967,22	0,88	9418
63	X^2 Measure (0,5)	SVM	mill_exit_temp_c_pv	7201103,11	2,61	10037
64	X^2 Measure (0,5)	MLP	mill_motor_pwr_kw_pv	539562,46	2,92	9970
65	X^2 Measure (0,5)	MLP	bucket_elv_mtr_pwr_kw_pv	2260583,79	0,88	9418
66	X^2 Measure (0,5)	MLP	mill_exit_temp_c_pv	4967919,55	2,62	10037
67	Teste de Grubbs (0,05)	SVM	mill_motor_pwr_kw_pv	482499,12	6,80	11433
68	Teste de Grubbs (0,05)	SVM	bucket_elv_mtr_pwr_kw_pv	950631,47	2,28	11484
69	Teste de Grubbs (0,05)	SVM	mill_exit_temp_c_pv	7892360,30	5,23	11694
70	Teste de Grubbs (0,05)	MLP	mill_motor_pwr_kw_pv	544540,77	6,93	11433
71	Teste de Grubbs (0,05)	MLP	bucket_elv_mtr_pwr_kw_pv	3157853,53	2,30	11484
72	Teste de Grubbs (0,05)	MLP	mill_exit_temp_c_pv	4871077,62	5,34	11694
73	Teste de Grubbs (0,1)	SVM	mill_motor_pwr_kw_pv	482698,26	6,82	11401
74	Teste de Grubbs (0,1)	SVM	bucket_elv_mtr_pwr_kw_pv	959479,01	2,32	11458
75	Teste de Grubbs (0,1)	SVM	mill_exit_temp_c_pv	7210048,00	5,26	11663
76	Teste de Grubbs (0,1)	MLP	mill_motor_pwr_kw_pv	556760,22	6,89	11401
77	Teste de Grubbs (0,1)	MLP	bucket_elv_mtr_pwr_kw_pv	2833549,17	2,34	11458
78	Teste de Grubbs (0,1)	MLP	mill_exit_temp_c_pv	4920797,14	5,28	11663
79	Teste de Grubbs (0,2)	SVM	mill_motor_pwr_kw_pv	483301,72	6,94	11328
80	Teste de Grubbs (0,2)	SVM	bucket_elv_mtr_pwr_kw_pv	957991,78	2,34	11451
81	Teste de Grubbs (0,2)	SVM	mill_exit_temp_c_pv	6745110,71	5,37	11591
82	Teste de Grubbs (0,2)	MLP	mill_motor_pwr_kw_pv	549053,46	6,96	11328
83	Teste de Grubbs (0,2)	MLP	bucket_elv_mtr_pwr_kw_pv	2708755,83	2,35	11451
84	Teste de Grubbs (0,2)	MLP	mill_exit_temp_c_pv	4977482,99	5,43	11591
85	Teste de Grubbs (0,3)	SVM	mill_motor_pwr_kw_pv	482605,26	7,03	11288
86	Teste de Grubbs (0,3)	SVM	bucket_elv_mtr_pwr_kw_pv	944252,96	2,36	11441
87	Teste de Grubbs (0,3)	SVM	mill_exit_temp_c_pv	6239531,19	5,47	11541
88	Teste de Grubbs (0,3)	MLP	mill_motor_pwr_kw_pv	542974,77	7,04	11288
89	Teste de Grubbs (0,3)	MLP	bucket_elv_mtr_pwr_kw_pv	3191780,43	2,38	11441
90	Teste de Grubbs (0,3)	MLP	mill_exit_temp_c_pv	5034876,92	5,52	11541
91	HBOS (0,1)	SVM	mill_motor_pwr_kw_pv	488304,62	0,17	10787
92	HBOS (0,1)	SVM	bucket_elv_mtr_pwr_kw_pv	874168,72	0,13	10786
93	HBOS (0,1)	SVM	mill_exit_temp_c_pv	6956188,16	0,16	10784
94	HBOS (0,1)	MLP	mill_motor_pwr_kw_pv	512367,57	0,18	10787
95	HBOS (0,1)	MLP	bucket_elv_mtr_pwr_kw_pv	2438830,35	0,13	10786
96	HBOS (0,1)	MLP	mill_exit_temp_c_pv	4928238,22	0,17	10784
97	HBOS (0,2)	SVM	mill_motor_pwr_kw_pv	494680,62	0,16	9592
98	HBOS (0,2)	SVM	bucket_elv_mtr_pwr_kw_pv	857054,49	0,13	9625
99	HBOS (0,2)	SVM	mill_exit_temp_c_pv	7375874,69	0,15	9705
100	HBOS (0,2)	MLP	mill_motor_pwr_kw_pv	519869,80	0,16	9592

Fonte: Autoria própria.

Quadro 1 – Continuação dos Experimentos

Cód.	Algoritmo de detecção de anomalias	AM	Classe	MASE	Tempo	Dados para treino
101	HBOS (0,2)	MLP	bucket_elv_mtr_pwr_kw_pv	2583624,00	0,14	9625
102	HBOS (0,2)	MLP	mill_exit_temp_c_pv	4922743,61	0,15	9705
103	HBOS (0,3)	SVM	mill_motor_pwr_kw_pv	485759,51	0,16	8400
104	HBOS (0,3)	SVM	bucket_elv_mtr_pwr_kw_pv	2005414,15	0,13	8442
105	HBOS (0,3)	SVM	mill_exit_temp_c_pv	7248356,77	0,17	8412
106	HBOS (0,3)	MLP	mill_motor_pwr_kw_pv	510742,08	0,17	8400
107	HBOS (0,3)	MLP	bucket_elv_mtr_pwr_kw_pv	1859488,53	0,13	8442
108	HBOS (0,3)	MLP	mill_exit_temp_c_pv	4945664,32	0,16	8412
109	KDE (Gaussian;0,1)	SVM	mill_motor_pwr_kw_pv	488593,96	7,35	10783
110	KDE (Gaussian;0,1)	SVM	bucket_elv_mtr_pwr_kw_pv	982379,05	6,38	10783
111	KDE (Gaussian;0,1)	SVM	mill_exit_temp_c_pv	5855641,14	7,82	10783
112	KDE (Gaussian;0,1)	MLP	mill_motor_pwr_kw_pv	525760,47	8,00	10783
113	KDE (Gaussian;0,1)	MLP	bucket_elv_mtr_pwr_kw_pv	3298638,67	6,17	10783
114	KDE (Gaussian;0,1)	MLP	mill_exit_temp_c_pv	4867337,28	7,76	10783
115	KDE (Gaussian;0,2)	SVM	mill_motor_pwr_kw_pv	487101,28	7,46	9585
116	KDE (Gaussian;0,2)	SVM	bucket_elv_mtr_pwr_kw_pv	1025845,31	6,27	9585
117	KDE (Gaussian;0,2)	SVM	mill_exit_temp_c_pv	5405136,70	7,85	9585
118	KDE (Gaussian;0,2)	MLP	mill_motor_pwr_kw_pv	529424,82	7,00	9585
119	KDE (Gaussian;0,2)	MLP	bucket_elv_mtr_pwr_kw_pv	2361175,38	6,20	9585
120	KDE (Gaussian;0,2)	MLP	mill_exit_temp_c_pv	5272687,59	7,77	9585
121	KDE (Gaussian;0,3)	SVM	mill_motor_pwr_kw_pv	488828,31	7,06	8387
122	KDE (Gaussian;0,3)	SVM	bucket_elv_mtr_pwr_kw_pv	937880,09	6,42	8387
123	KDE (Gaussian;0,3)	SVM	mill_exit_temp_c_pv	4797602,48	7,88	8387
124	KDE (Gaussian;0,3)	MLP	mill_motor_pwr_kw_pv	534789,14	7,04	8387
125	KDE (Gaussian;0,3)	MLP	bucket_elv_mtr_pwr_kw_pv	2106051,69	6,19	8387
126	KDE (Gaussian;0,3)	MLP	mill_exit_temp_c_pv	4640325,79	7,80	8387
127	KDE (Tophat;0,1)	SVM	mill_motor_pwr_kw_pv	479266,63	0,39	10783
128	KDE (Tophat;0,1)	SVM	bucket_elv_mtr_pwr_kw_pv	925259,07	0,62	10783
129	KDE (Tophat;0,1)	SVM	mill_exit_temp_c_pv	6982254,56	1,61	10783
130	KDE (Tophat;0,1)	MLP	mill_motor_pwr_kw_pv	520163,96	0,38	10783
131	KDE (Tophat;0,1)	MLP	bucket_elv_mtr_pwr_kw_pv	2407813,98	0,60	10783
132	KDE (Tophat;0,1)	MLP	mill_exit_temp_c_pv	4951656,99	1,60	10783
133	KDE (Tophat;0,2)	SVM	mill_motor_pwr_kw_pv	475173,63	0,39	9585
134	KDE (Tophat;0,2)	SVM	bucket_elv_mtr_pwr_kw_pv	934797,24	0,64	9585
135	KDE (Tophat;0,2)	SVM	mill_exit_temp_c_pv	5111182,18	1,61	9585
136	KDE (Tophat;0,2)	MLP	mill_motor_pwr_kw_pv	499563,05	0,38	9585
137	KDE (Tophat;0,2)	MLP	bucket_elv_mtr_pwr_kw_pv	2980805,08	0,60	9585
138	KDE (Tophat;0,2)	MLP	mill_exit_temp_c_pv	5214629,38	1,59	9585
139	KDE (Tophat;0,3)	SVM	mill_motor_pwr_kw_pv	475688,33	0,38	8387
140	KDE (Tophat;0,3)	SVM	bucket_elv_mtr_pwr_kw_pv	903764,75	0,62	8387
141	KDE (Tophat;0,3)	SVM	mill_exit_temp_c_pv	5279209,90	1,62	8387
142	KDE (Tophat;0,3)	MLP	mill_motor_pwr_kw_pv	498682,76	0,38	8387
143	KDE (Tophat;0,3)	MLP	bucket_elv_mtr_pwr_kw_pv	2004604,68	0,60	8387
144	KDE (Tophat;0,3)	MLP	mill_exit_temp_c_pv	5250172,93	1,59	8387
145	KDE (Epanechnikov;0,1)	SVM	mill_motor_pwr_kw_pv	479399,70	0,39	10783
146	KDE (Epanechnikov;0,1)	SVM	bucket_elv_mtr_pwr_kw_pv	1008120,80	0,70	10783
147	KDE (Epanechnikov;0,1)	SVM	mill_exit_temp_c_pv	6812713,93	1,66	10783
148	KDE (Epanechnikov;0,1)	MLP	mill_motor_pwr_kw_pv	512533,68	0,38	10783
149	KDE (Epanechnikov;0,1)	MLP	bucket_elv_mtr_pwr_kw_pv	2367644,01	0,70	10783
150	KDE (Epanechnikov;0,1)	MLP	mill_exit_temp_c_pv	4922670,59	1,65	10783

Fonte: Autoria própria.

Quadro 1 – Continuação dos Experimentos

Cód.	Algoritmo de detecção de anomalias	AM	Classe	MASE	Tempo	Dados para treino
151	KDE (Epanechnikov;0,2)	SVM	mill_motor_pwr_kw_pv	479505,59	0,39	9585
152	KDE (Epanechnikov;0,2)	SVM	bucket_elv_mtr_pwr_kw_pv	930974,83	0,70	9585
153	KDE (Epanechnikov;0,2)	SVM	mill_exit_temp_c_pv	5236129,85	1,66	9585
154	KDE (Epanechnikov;0,2)	MLP	mill_motor_pwr_kw_pv	509502,03	0,38	9585
155	KDE (Epanechnikov;0,2)	MLP	bucket_elv_mtr_pwr_kw_pv	2240303,51	0,69	9585
156	KDE (Epanechnikov;0,2)	MLP	mill_exit_temp_c_pv	5212632,63	1,65	9585
157	KDE (Epanechnikov;0,3)	SVM	mill_motor_pwr_kw_pv	480211,99	0,39	8387
158	KDE (Epanechnikov;0,3)	SVM	bucket_elv_mtr_pwr_kw_pv	909822,79	0,70	8387
159	KDE (Epanechnikov;0,3)	SVM	mill_exit_temp_c_pv	5953096,78	1,65	8387
160	KDE (Epanechnikov;0,3)	MLP	mill_motor_pwr_kw_pv	503425,27	0,38	8387
161	KDE (Epanechnikov;0,3)	MLP	bucket_elv_mtr_pwr_kw_pv	2263004,68	0,70	8387
162	KDE (Epanechnikov;0,3)	MLP	mill_exit_temp_c_pv	5436507,01	1,64	8387
163	KDE (Exponential;0,1)	SVM	mill_motor_pwr_kw_pv	488152,57	7,79	10783
164	KDE (Exponential;0,1)	SVM	bucket_elv_mtr_pwr_kw_pv	858973,29	7,29	10783
165	KDE (Exponential;0,1)	SVM	mill_exit_temp_c_pv	5963724,26	8,98	10783
166	KDE (Exponential;0,1)	MLP	mill_motor_pwr_kw_pv	529553,61	7,74	10783
167	KDE (Exponential;0,1)	MLP	bucket_elv_mtr_pwr_kw_pv	2874101,25	7,26	10783
168	KDE (Exponential;0,1)	MLP	mill_exit_temp_c_pv	4949688,12	8,90	10783
169	KDE (Exponential;0,2)	SVM	mill_motor_pwr_kw_pv	484657,19	7,95	9585
170	KDE (Exponential;0,2)	SVM	bucket_elv_mtr_pwr_kw_pv	1030001,48	7,62	9585
171	KDE (Exponential;0,2)	SVM	mill_exit_temp_c_pv	5831834,25	9,13	9585
172	KDE (Exponential;0,2)	MLP	mill_motor_pwr_kw_pv	540612,49	7,76	9585
173	KDE (Exponential;0,2)	MLP	bucket_elv_mtr_pwr_kw_pv	2700029,88	7,27	9585
174	KDE (Exponential;0,2)	MLP	mill_exit_temp_c_pv	5381859,11	8,90	9585
175	KDE (Exponential;0,3)	SVM	mill_motor_pwr_kw_pv	500534,44	9,08	8387
176	KDE (Exponential;0,3)	SVM	bucket_elv_mtr_pwr_kw_pv	871189,59	7,29	8387
177	KDE (Exponential;0,3)	SVM	mill_exit_temp_c_pv	4728023,19	9,03	8387
178	KDE (Exponential;0,3)	MLP	mill_motor_pwr_kw_pv	536687,84	7,74	8387
179	KDE (Exponential;0,3)	MLP	bucket_elv_mtr_pwr_kw_pv	2727055,12	7,27	8387
180	KDE (Exponential;0,3)	MLP	mill_exit_temp_c_pv	4705016,84	9,15	8387
181	KDE (Linear;0,1)	SVM	mill_motor_pwr_kw_pv	479538,27	0,40	10783
182	KDE (Linear;0,1)	SVM	bucket_elv_mtr_pwr_kw_pv	1025209,30	0,71	10783
183	KDE (Linear;0,1)	SVM	mill_exit_temp_c_pv	6932189,39	1,65	10783
184	KDE (Linear;0,1)	MLP	mill_motor_pwr_kw_pv	511167,73	0,38	10783
185	KDE (Linear;0,1)	MLP	bucket_elv_mtr_pwr_kw_pv	2513850,81	0,69	10783
186	KDE (Linear;0,1)	MLP	mill_exit_temp_c_pv	4887075,05	1,65	10783
187	KDE (Linear;0,2)	SVM	mill_motor_pwr_kw_pv	478847,86	0,39	9585
188	KDE (Linear;0,2)	SVM	bucket_elv_mtr_pwr_kw_pv	990100,03	0,70	9585
189	KDE (Linear;0,2)	SVM	mill_exit_temp_c_pv	5227114,10	1,65	9585
190	KDE (Linear;0,2)	MLP	mill_motor_pwr_kw_pv	505858,92	0,38	9585
191	KDE (Linear;0,2)	MLP	bucket_elv_mtr_pwr_kw_pv	2625963,66	0,70	9585
192	KDE (Linear;0,2)	MLP	mill_exit_temp_c_pv	5258731,07	1,64	9585
193	KDE (Linear;0,3)	SVM	mill_motor_pwr_kw_pv	481681,45	0,39	8387
194	KDE (Linear;0,3)	SVM	bucket_elv_mtr_pwr_kw_pv	912049,20	0,69	8387
195	KDE (Linear;0,3)	SVM	mill_exit_temp_c_pv	5930707,71	1,67	8387
196	KDE (Linear;0,3)	MLP	mill_motor_pwr_kw_pv	506961,88	0,38	8387
197	KDE (Linear;0,3)	MLP	bucket_elv_mtr_pwr_kw_pv	2500398,62	0,70	8387
198	KDE (Linear;0,3)	MLP	mill_exit_temp_c_pv	5397986,69	1,64	8387
199	KDE (Cosine;0,1)	SVM	mill_motor_pwr_kw_pv	482699,65	0,39	10783
200	KDE (Cosine;0,1)	SVM	bucket_elv_mtr_pwr_kw_pv	905487,85	0,74	10783

Fonte: Autoria própria.

Quadro 1 – Continuação dos Experimentos

Cód.	Algoritmo de detecção de anomalias	AM	Classe	MASE	Tempo	Dados para treino
201	KDE (Cosine;0,1)	SVM	mill_exit_temp_c_pv	7419880,72	10,63	10783
202	KDE (Cosine;0,1)	MLP	mill_motor_pwr_kw_pv	514696,10	0,39	10783
203	KDE (Cosine;0,1)	MLP	bucket_elv_mtr_pwr_kw_pv	3060096,22	0,74	10783
204	KDE (Cosine;0,1)	MLP	mill_exit_temp_c_pv	4866384,74	10,55	10783
205	KDE (Cosine;0,2)	SVM	mill_motor_pwr_kw_pv	482609,24	0,39	9585
206	KDE (Cosine;0,2)	SVM	bucket_elv_mtr_pwr_kw_pv	933896,82	0,74	9585
207	KDE (Cosine;0,2)	SVM	mill_exit_temp_c_pv	6080100,19	10,55	9585
208	KDE (Cosine;0,2)	MLP	mill_motor_pwr_kw_pv	509834,91	0,38	9585
209	KDE (Cosine;0,2)	MLP	bucket_elv_mtr_pwr_kw_pv	2392045,82	0,74	9585
210	KDE (Cosine;0,2)	MLP	mill_exit_temp_c_pv	4961964,66	10,55	9585
211	KDE (Cosine;0,3)	SVM	mill_motor_pwr_kw_pv	482365,89	0,39	8387
212	KDE (Cosine;0,3)	SVM	bucket_elv_mtr_pwr_kw_pv	909826,35	0,74	8387
213	KDE (Cosine;0,3)	SVM	mill_exit_temp_c_pv	5833147,81	10,54	8387
214	KDE (Cosine;0,3)	MLP	mill_motor_pwr_kw_pv	508941,50	0,39	8387
215	KDE (Cosine;0,3)	MLP	bucket_elv_mtr_pwr_kw_pv	2185639,82	0,75	8387
216	KDE (Cosine;0,3)	MLP	mill_exit_temp_c_pv	4738818,14	10,60	8387

Fonte: Autoria própria.

Quadro 2 – Analise estatística de cada classe

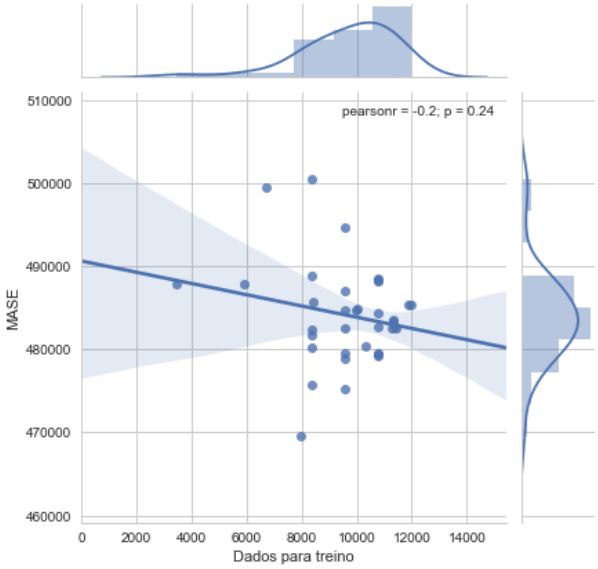
	mill_motor_pwr_kw_pv	bucket_elv_mtr_pwr_kw_pv	mill_exit_temp_c_pv
Mínimo	0.00	0.00	84.67
Máximo	1554.97	5.62	151.74
Amplitude	1554.97	5.62	67.07
Média	1067.62	4.94	112.61
Mediana	1091.91	4.94	111.89
Desvio Padrão	123.84	0.25	6.10
Coeficiente de Variação	11.60	5.01	5.42
Variância	15336.60	0.06	37.27
Variância Relativa	$1346 * 10^{-5}$	$251 * 10^{-5}$	$294 * 10^{-5}$

Fonte: Autoria própria.

As Figuras 14, 15, 16, 17, 18 e 19 ilustram gráficos gerados a partir do Quadro 1, onde o eixo y é o valor do MASE e o eixo x é a quantidade de dados utilizados para treino, isto é, dados usados para treinamento após a limpeza. Cada ponto contido no gráfico representa um experimento realizado, no qual é enfatizado os cinco pontos que mais se destacam.

Analisando a Figura 14, percebe-se que não há uma tendência linear clara, pois a margem de erro possui uma discrepância elevada que consente com o nível de significância igual a 24% e a correlação absoluta nesse subconjunto está próxima a zero, onde ponto mais a esquerda representa o experimento referente ao código número 19, ou seja, o experimento que aplica o Box Plot, com parâmetro igual a 0,5 e 3.480 dados para treino; este ponto demonstra que, mesmo removendo cerca de 8.501 dados o valor de MASE foi maior do que se nenhuma

Figura 14 – Gráfico da classe mill_motor_pwr_kw_pv para classificação de algoritmos de detecção de anomalias utilizando o SVM

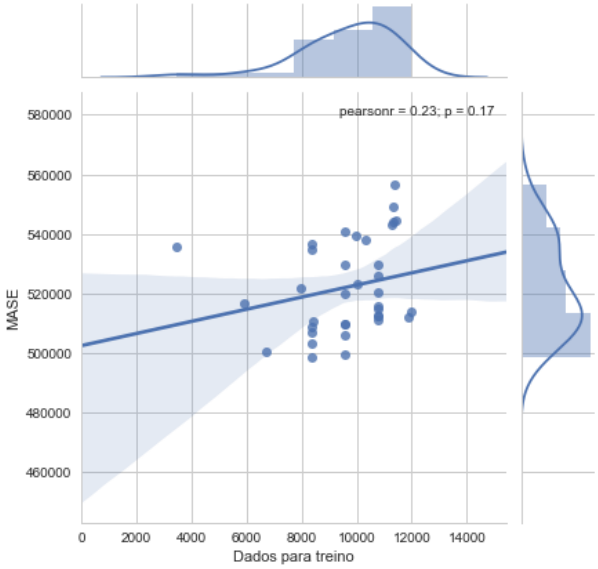


Fonte: Autoria própria.

remoção de anomalias fosse realizada.

O ponto mais acima na Figura 14, demonstra o pior algoritmo de detecção de anomalias que, neste caso, foi o KDE utilizando *kernel* exponencial com remoção de 30% dos dados. Os dois últimos pontos destacados mais abaixo no gráfico, representam os melhores algoritmo de detecção de anomalias que, neste caso, foram o *4-Sigma* e o KDE utilizando *kernel tophat* com remoção de 20% dos dados.

Figura 15 – Gráfico da classe mill_motor_pwr_kw_pv para classificação de algoritmos de detecção de anomalias utilizando o MLP

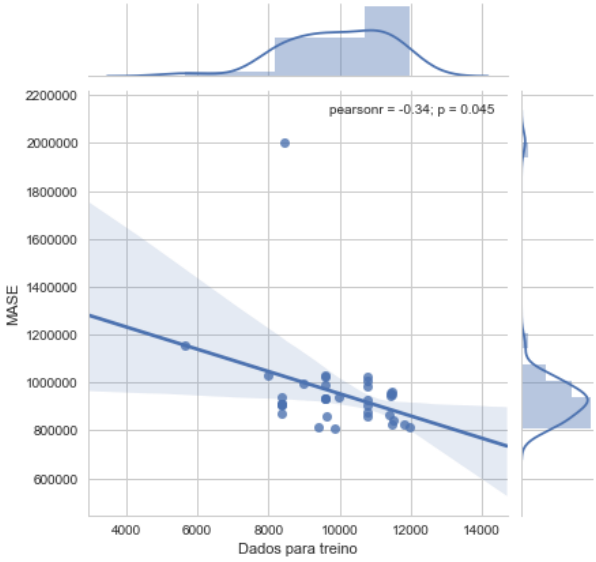


Fonte: Autoria própria.

Ao examinar a Figura 15, nota-se que há uma tendência linear crescente, mesmo que ínfima, de diminuição no valor de MASE quando há uma aumento na quantidade de dados removidos, melhorando pouco em relação a correlação dos dados e nível de significância do SVM. No qual os dois pontos destacados mais acima no gráfico representam os algoritmos de detecção de anomalias que obtiveram os piores resultado de MASE em relação a quantidade de dados removidos que, neste caso, foram o Box Plot com parâmetro igual a 0,5 e o Teste de Grubbs com nível de significância igual a 0,1.

Os dois últimos pontos destacados na Figura 15, mais abaixo no gráfico, representam variações do algoritmo KDE, utilizando *kernel tophat*, onde os experimentos que consideraram que 20% e 30% dos dados são anomalias foram os que obtiveram os melhores resultados de MASE.

Figura 16 – Gráfico da classe bucket_elv_mtr_pwr_kw_pv para classificação de algoritmos de detecção de anomalias utilizando o SVM

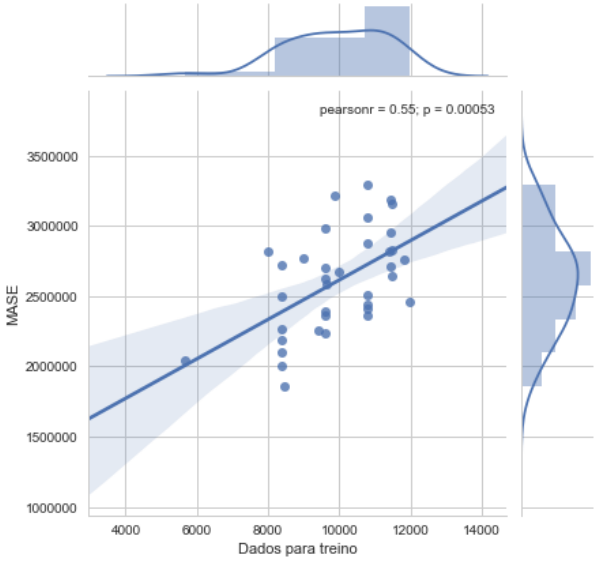


Fonte: Autoria própria.

Na Figura 16 há uma melhora significante na relação entre esses dois dados, pois a uma maior correlação absoluta dos dados e um nível de significância abaixo de 5%. Entretanto há uma tendência linear decrescente, ou seja, quanto menos dados removidos melhor será o MASE. Na Figura 16 é possível ver o ponto mais acima é referente ao experimento com o algoritmo HBOS com 30% de contaminação (código número 104), pois o mesmo possui um valor de MASE maior do que o experimento com o segundo maior MASE calculado. O Box Plot com parâmetro multiplicador igual a 0,5, foi identificado como o ponto que está na parte superior mais a esquerda no gráfico (código número 20).

Os experimentos representados pelos dois pontos presentes mais abaixo (códigos número 56 e 62) no gráfico, presente na Figura 16, não alcançaram um valor significativo de MASE comparado ao experimento que não utilizou nenhum método de detecção de anomalias (código número 2).

Figura 17 – Gráfico da classe bucket_elv_mtr_pwr_kw_pv para classificação de algoritmos de detecção de anomalias utilizando o MLP



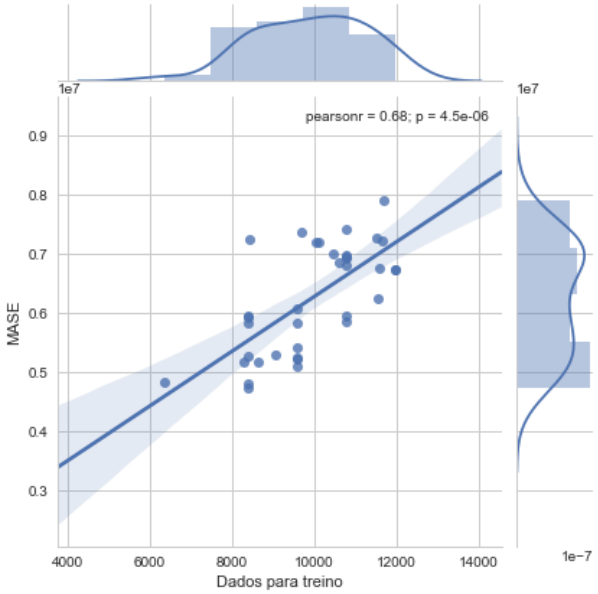
Fonte: Autoria própria.

Ao explorar a Figurar 17 percebe-se que há uma tendência linear crescente, nota-se, também, que o resultado do algoritmo MLP foi melhor que o algoritmo SVM em relação a classe *bucket_elv_mtr_pwr_kw_pv*. Essa tendência significa que quanto menor a quantidade de dados para treino, menor será o MASE. É possível notar que o ponto mais acima no gráfico (código número 113) representa o algoritmo KDE utilizando *kernel* Gaussiano com remoção de 10% dos dados anômalos, no qual possui o pior valor de MASE, neste caso.

Os pontos com um menor valor de MASE representam os algoritmos HBOS com 30% de contaminação (código número 107) na Figura 17, KDE utilizando *kernel* tophat com remoção de 30% dos dados anômalos (código número 143) e Box Plot com parâmetro multiplicador igual a 0,5 (código número 23).

Na Figura 18 é possível identificar que a remoção de dados anômalos para o sub-conjunto da classe *mill_exit_temp_c_pv* e para o algoritmo SVM foi a mais eficaz e possui uma melhor correlação e um menor nível de significância, no qual é nítido que medida que menos dados são utilizados para treino melhor será o valor de MASE. No qual dois experimentos, a direita no gráfico, que estão praticamente no mesmo ponto (códigos número 3 e 45), isso significa

Figura 18 – Gráfico da classe mill_exit_temp_c_pv para classificação de algoritmos de detecção de anomalias utilizando o SVM



Fonte: Autoria própria.

que não utilizar algoritmos de detecção de anomalias e utilizar o X^2 *Measure* usando 3 vezes o desvio padrão surte, praticamente, o mesmo efeito sobre os dados, neste caso.

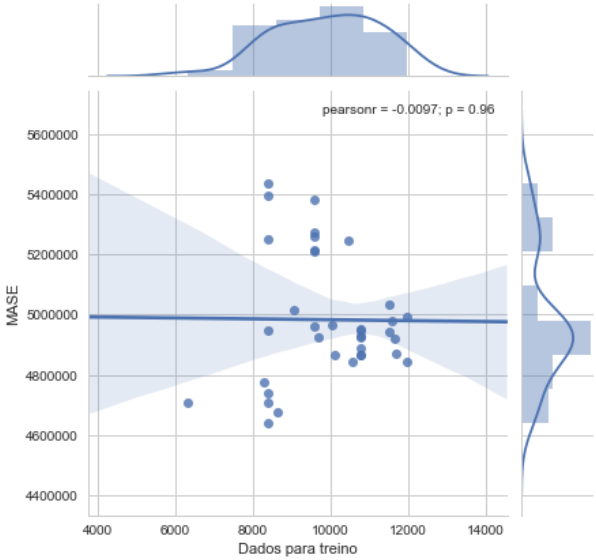
O experimento que obteve o pior valor de MASE é ilustrado, na Figura 18, como o ponto mais acima no gráfico (código número 69), indicando que o Teste de Grubbs com nível de significância igual a 0,05 piora as previsões dos algoritmos de AM.

Os três experimentos que obtiveram os melhores resultados foram os pontos mais abaixo no gráfico (códigos número 21, 123 e 177), presente na Figura 18, foram os algoritmos Box Plot com parametro igual a 0,5, KDE com *kernel* gaussiano com remoção de 30% dos dados e KDE com *kernel* exponencial com remoção de 30% dos dados, sendo que o primeiro algoritmo citado necessita remover uma maior quantidade de dados.

Nota-se, na Figura 19, que não há uma tendência linear clara nos dados e seus valores de correlação e nível de significância são os piores entre todos os blocos, com seus valores chegando a praticamente zero e próximo a 100%, respectivamente. Portanto não é possível obter conclusões significantes sobre este gráfico. É possível ver que o ponto mais acima dos outros no gráfico (código número 162), é representado pelo experimento com o algoritmo KDE utilizando *kernel epanechnikov* com remoção de 30% dos dados, que possui o pior valor de MASE, até mesmo se nenhum algoritmo de detecção de anomalias fosse aplicado, como demonstrado pelo ponto mais a direita (código número 6)

Percebe-se, também, na Figura 19, que três pontos destacam-se no gráfico (código

Figura 19 – Gráfico da classe mill_exit_temp_c_pv para classificação de algoritmos de detecção de anomalias utilizando o MLP



Fonte: Autoria própria.

número 12, 24 e 126), representando os algoritmos Box Plot com parâmetro igual a 1,5, Box Plot com parâmetro igual a 0,5 e KDE utilizando *kernel* com remoção de 30% dos dados, como os que possuem melhor MASE. Entretanto o segundo algoritmo citado necessita remover uma maior quantidade de dados.

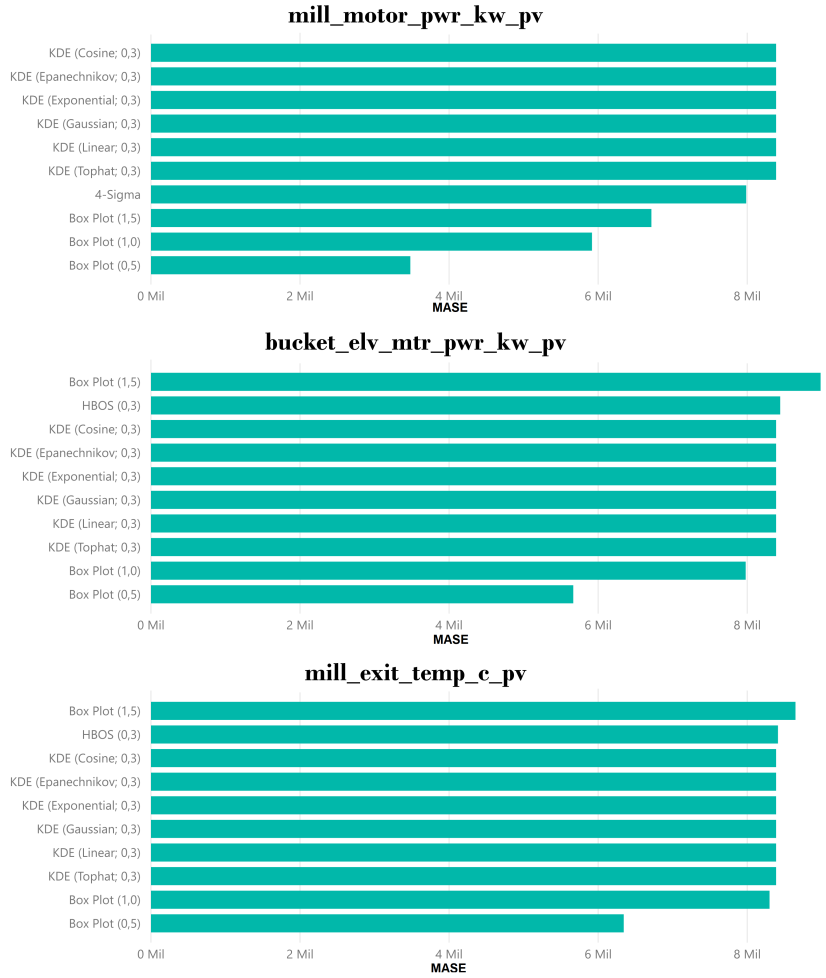
A tendência linear crescente encontrada na maioria dos gráficos reforça a ideia de que a remoção de anomalias é eficaz e, geralmente, leva a uma melhor acurácia na previsão pelos algoritmos de AM. No qual os algoritmos de detecção de anomalias mais eficientes são aqueles que possuem seus pontos mais a esquerda e abaixo no gráfico.

Na Figura 20, ilustra um gráfico que analisa quais os 10 algoritmos de detecção de anomalias removeram mais instâncias consideradas anômalas do conjunto de treinamento, ou seja, o eixo *x* ilustra a quantidade de dados restantes do conjunto de treinamento após a remoção das anomalias de cada uma das classes (seção 4.1).

Os algoritmos que mais removeram dados do conjunto, de acordo com a analise feita na Figura 20, em ordem decrescente, foram: Box Plot (0,5), Box Plot (1,0), Box Plot (1,5), KDE ([Gaussian, Tophat, Epanechnikov, Exponential, Linear, Cousine],0,3]) e HBOS (0,3). Onde, se considerar que o primeiro experimento contem 100% dos dados de treino, cada algoritmo citado anteriormente manteve cerca de 43,09%, 61,75%, 67,73%, 70% e 70,26% dos dados do conjunto, respectivamente.

A Figura 21 demonstra que o algoritmo de AM SVM possui uma baixa melhora no MASE em comparação com o MLP. Isso pode ser devido a alta dispersão dos dados para a classe

Figura 20 – Total de dados de treinamento em relação aos algoritmos de detecção de anomalias para cada classe



Fonte: Autoria própria.

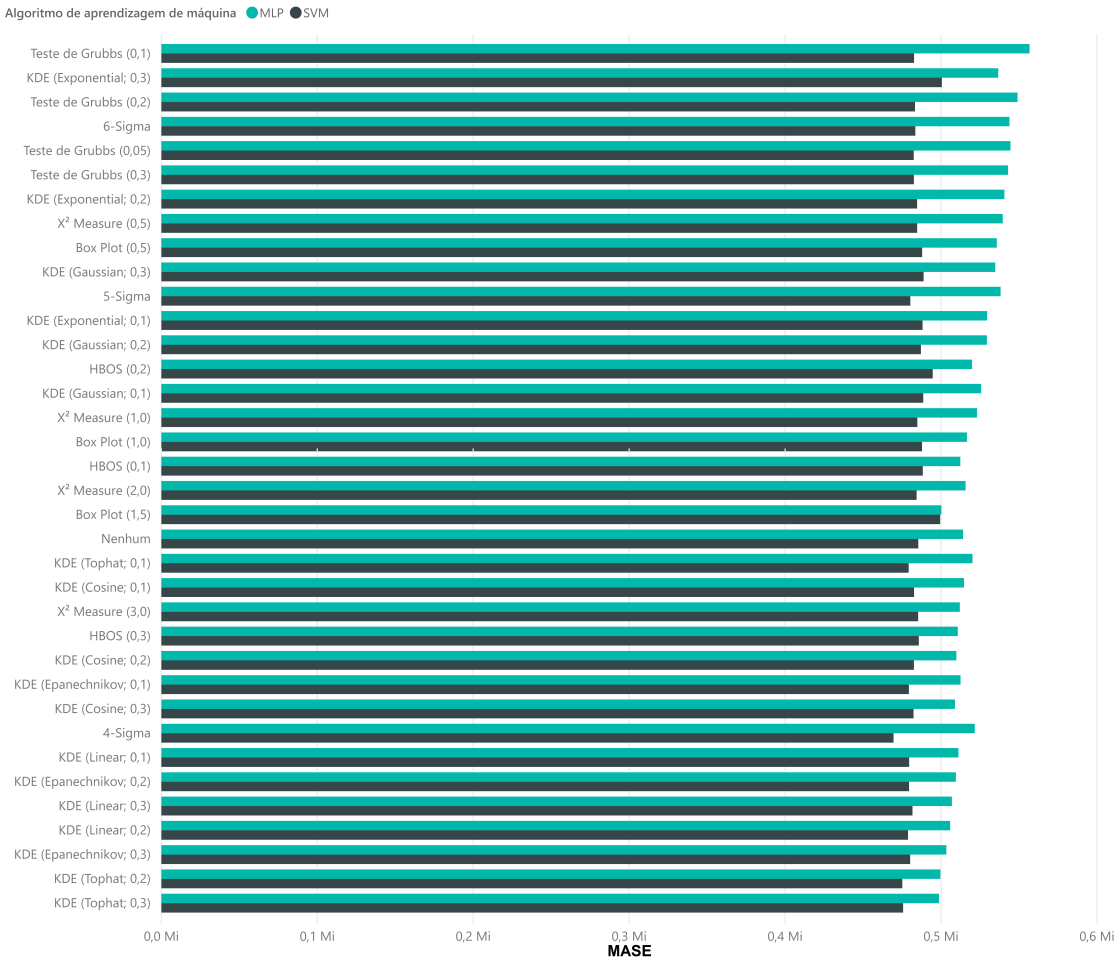
mill_motor_pwr_kw_pv, como calculado no Quadro 2.

A Figura 22 demonstra que o algoritmo de AM SVM possui um valor de MASE, para as previsões feitas na classe bucket_elv_mtr_pwr_kw_pv, menor que o MLP, para maioria das limpezas feitas pelos algoritmos de detecção de anomalias.

O algoritmo de AM MLP, para previsões feitas para a classe mill_exit_temp_c_pv, possui um valor de MASE menor que as previsões feitas pelo SVM, em relação a maioria das limpezas feitas pelos algoritmos de detecção de anomalias. Nas Figuras 21, 22 e 23 é possível identificar que o SVM obteve melhores resultados de MASE em relação ao MLP em duas das três classes objetivo, havendo somente uma diferença ns resultados de MASE para a classe mill_exit_temp_c_pv.

Essa diferença em relação as outras classes pode ser devido ao baixo desempenho que o SVM dispõem para dados que não possuem uma correlação linear, outro fator que pode contribuir para essa diferença é que as medidas de dispersões não são altas para a classe

Figura 21 – Diferença de MASE entre os algoritmos de AM utilizando a classe mill_motor_pwr_kw_pv



Fonte: Autoria própria.

mill_exit_temp_c_pv, ilustradas no Quadro 2, isso significa que o MLP possui desempenho melhor para correlações não lineares e com baixo grau de dispersão.

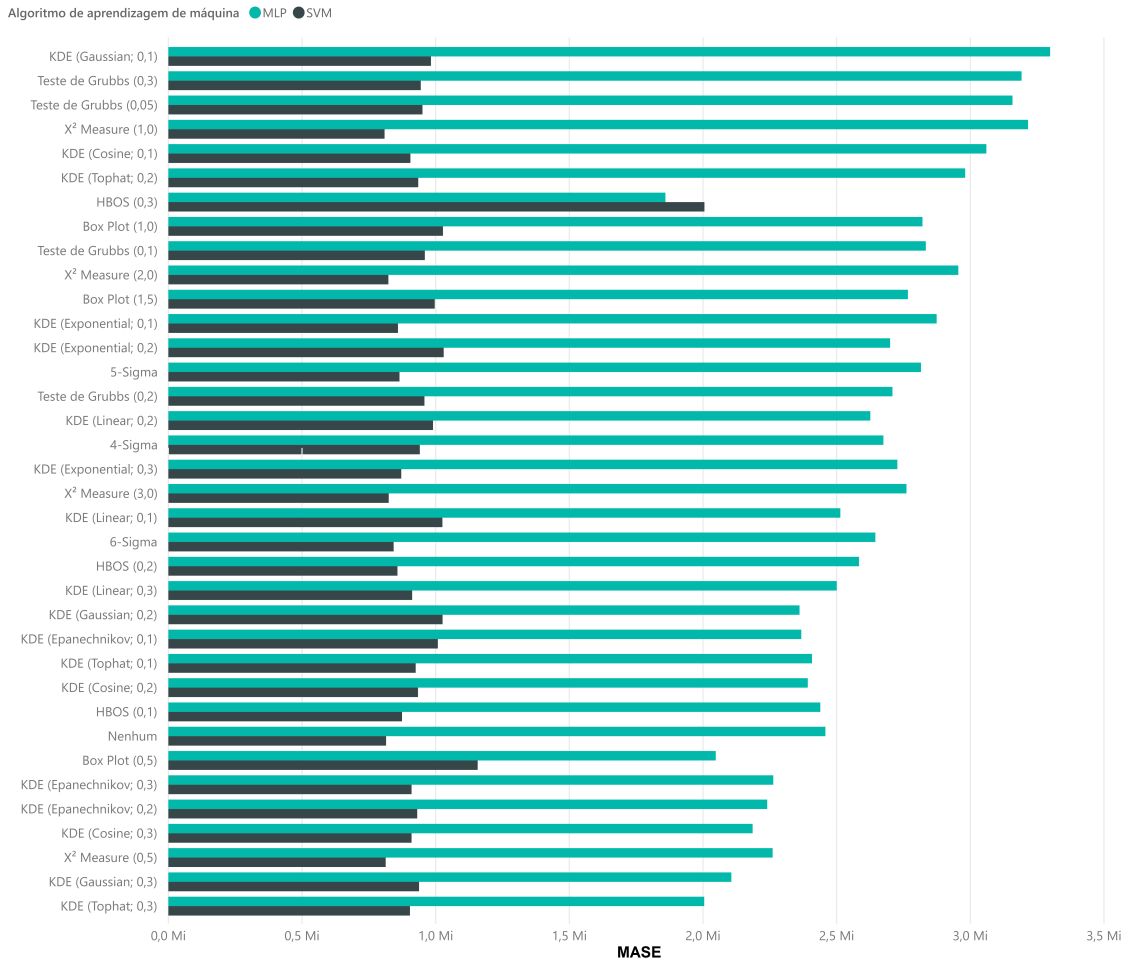
Abaixo estão listados os cinco algoritmos de detecção de anomalias que possuíram o menor valor de MASE, em ordem decrescentes, em relação a classe:

- mill_motor_pwr_kw_pv: KDE (Tophat; 0,3), KDE (Tophat; 0,2), KDE (Epanechnikov; 0,3), KDE (Linear; 0,2) e KDE (Linear; 0,3);
- bucket_elv_mtr_pwr_kw_pv: KDE (Tophat; 0,3), KDE (Gaussian; 0,3), X^2 Measure (0,5), KDE (Cosine; 0,3) e KDE (Epanechnikov; 0,2);
- mill_exit_temp_c_pv: KDE (Exponential; 0,3), KDE (Gaussian; 0,3), Box Plot (0,5), Box Plot (1,5) e Box Plot (1,0).

Em relação aos algoritmos de AM, os cinco algoritmos de detecção de anomalias que possuíram o menor valor de MASE estão listado abaixo, em ordem decrescentes, para o:

- SVM: KDE (Exponential; 0,3), KDE (Gaussian; 0,3), Box Plot (0,5), KDE (Tophat; 0,2) e

Figura 22 – Diferença de MASE entre os algoritmos de AM utilizando a classe bucket_elv_mtr_pwr_kw_pv



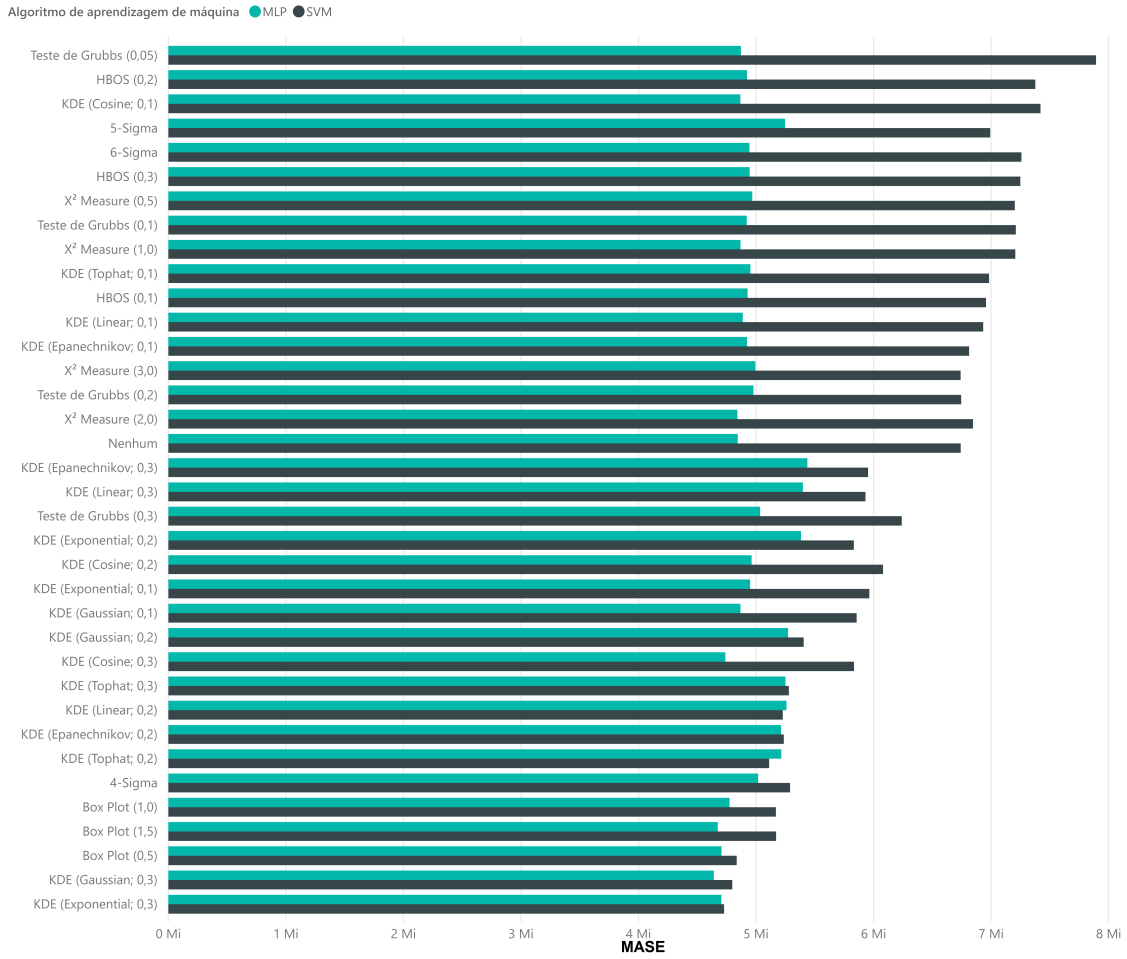
Fonte: Autoria própria.

KDE (Epanechnikov; 0,2);

- MLP: KDE (Gaussian; 0,3), Box Plot (0,5), HBOS (0,3), KDE (Cosine; 0,3) e KDE (Tophat; 0,3).

Após a analise de cerca de 216 experimentos e através das comparações dos resultados de MASE, da quantidade de dados removidos e dos algoritmos de AM, é possível constatar que os algoritmos de detecção de anomalias que mais se destaram, provendo um menor valor de MASE para os dois algoritmos de AM, em ordem decrescente, foram o KDE (Gaussian; 0,3), Box plot (0,5), KDE (Exponential; 0,3), KDE (Tophat; 0,3) e Box Plot (1,5). No qual esses cinco algoritmos estão entre os 10 que removeram mais dados do conjunto.

Figura 23 – Diferença de MASE entre os algoritmos de AM utilizando a classe mill_exit_temp_c_pv



Fonte: Autoria própria.

5 CONCLUSÕES E TRABALHOS FUTUROS

Durante a execução deste trabalho, foram feitas analises que avaliaram quais algoritmos estatísticos de detecção de anomalias foram mais eficazes na remoção de dados considerados anômalos do conjunto, ou seja, quais algoritmos aumentaram a acurácia da previsão dos algoritmos de aprendizado de máquina.

Após uma série de analises, percebeu-se que, independentemente da quantidade de dados removidos, os algoritmos de aprendizagem de máquina ainda podem prover boas previsões a partir dos dados permanecentes .

Apesar dos algoritmos de detecção de anomalias poderem aumentar da acurácia das previsões, estes variam de acordo com as características dos dados analisados e dos algoritmos de aprendizado de máquina utilizados. O mesmo depende diretamente de dois outros fatores, que são o conjunto de dados e o algoritmo de aprendizado de máquina utilizado. Constatou-se que dados identificados como anômalos por um algoritmo de detecção de anomalias pode levar a uma maior acurácia para um determinado algoritmo de aprendizagem de máquina e uma menor acurácia para um outro.

Portanto a remoção de dados considerados anômalos utilizando técnicas estatísticas, geralmente, produzem um impacto positivo para a previsão de novos valores, removendo rapidamente as anomalias, útil para sistemas que estão em ambientes dinâmicos e que necessitam fazer diversas inferências em um curto período de tempo.

Como trabalhos futuros, é possível avaliar o impacto na previsão da remoção de anomalias utilizando algoritmos baseados em outras técnicas, como, baseado em classificação, baseado em vizinho mais próximos, baseado em *cluster*, baseados em informações teóricas ou espectral, aplicados a um conjunto dados coletados por sensores. Pra tanto, pode-se refazer os experimentos para outros conjuntos de dados disponíveis ou para outros algoritmos de aprendizagem de máquina existentes. Pode-se ainda testar variações de parâmetros que não foram testados tanto nos algoritmos de detecção de anomalias quanto nos algoritmos de aprendizagem de maquina.

Além de variações em diversos pontos dos experimentos realizados, pode-se realizar analises em relação ao comportamento dos dados de cada classe para identificar o porquê que alguns algoritmos de aprendizagem de máquina são melhores para o conjunto de dados ou para uma remoção de anomalias específica.

REFERÊNCIAS

AGGARWAL, C. C. **Data mining:** the textbook. New York: Springer, 2015.

BAEK, S.; KIM, D.-Y. Abrupt variance and discernibility analyses of multi-sensor signals for fault pattern extraction. **Computers & Industrial Engineering**, Elsevier, v. 128, p. 999–1007, 2019.

BONTEMPI, G.; TAIEB, S. B.; BORGNE, Y.-A. L. Machine learning strategies for time series forecasting. In: AUFAURE, Marie-Aude; ZIMÁNYI, Esteban. **Business intelligence**. Berlin, Heidelberg: Springer, 2012. p. 62–77.

CARVALHO, A.; FACELI, K.; LORENA, A.; GAMA, J. **Inteligência Artificial–uma abordagem de aprendizado de máquina**. Rio de Janeiro: LTC, 2011.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: a survey. **ACM computing surveys (CSUR)**, ACM, v. 41, n. 3, p. 15, 2009.

CHATZIGIANNAKIS, V.; PAPAVASSILIOU, S.; GRAMMATIKOU, M.; MAGLARIS, B. Hierarchical anomaly detection in distributed large-scale sensor networks. In: SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, 11., 2006, Cagliari. **Proceedings** [...]. [S.l.]: IEEE. 2006. p. 761–767.

CORRIGAN, O. **An investigation into machine learning solutions involving time series across different problem domains.** 2018. Tese (Doutorado em Filosofia) — Dublin City University, Dublin.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

DESFORGES, M.; JACOB, P.; COOPER, J. Applications of probability density estimation to the detection of abnormal conditions in engineering. **Journal of Mechanical Engineering Science**, London, England, v. 212, n. 8, p. 687–703, 1998.

FAWCETT, T.; PROVOST, F. **Data science para negócios:** o que você precisa saber sobre mineração de dados e pensamento analítico de dados. Rio de Janeiro: Alta Books, 2018.

FERRARI, D. G.; SILVA, L. N. de C. **Introdução a mineração de dados**. São Paulo: Saraiva, 2017.

GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric environment**, Elsevier, v. 32, n. 14-15, p. 2627–2636, 1998.

GOLDSTEIN, M.; DENGEL, A. Histogram-based outlier score (hbos): a fast unsupervised anomaly detection algorithm. **KI-2012:** Poster and Demo Track, Citeseer, p. 59–63, 2012.

GOLDSTEIN, M.; UCHIDA, S. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. **PloS one**, Public Library of Science, v. 11, n. 4, p. e0152173, 2016.

GRUBBS, F. E. Procedures for detecting outlying observations in samples. **Technometrics**, Taylor & Francis, v. 11, n. 1, p. 1–21, 1969.

HYNDMAN, R. J.; KOEHLER, A. B. Another look at measures of forecast accuracy. **International journal of forecasting**, Elsevier, v. 22, n. 4, p. 679–688, 2006.

KHAIR, U.; FAHMI, H.; HAKIM, S. A.; RAHIM, R. Forecasting error calculation with mean absolute deviation and mean absolute percentage error. **Journal of Physics**, v. 930, p. 012002, 2017.

LAZAREVIC, A.; ERTOZ, L.; KUMAR, V.; OZGUR, A.; SRIVASTAVA, J. A comparative study of anomaly detection schemes in network intrusion detection. In: INTERNATIONAL CONFERENCE ON DATA MINING, 2003, San Francisco, California. **Proceedings**[...]. [S.l.]: SIAM. 2003. p. 25–36.

LIEBCHEN, G. A. **Data cleaning techniques for software engineering data sets.** 2010. Tese (Doutorado em Filosofia) — Brunel University, School of Information Systems, Computing and Mathematics, London.

LILJA, D. J. **Measuring computer performance:** a practitioner's guide. New York: Cambridge university press, 2005.

LUND, A.; LUND, M. **Histograms**. 2018. Disponível em: <https://statistics.laerd.com/statistical-guides/understanding-histograms.php>. Acesso em: 09 jul. 2019.

PARZEN, E. On estimation of a probability density function and mode. **The annals of mathematical statistics**, JSTOR, v. 33, n. 3, p. 1065–1076, 1962.

RAISINGHANI, M. S.; ETTE, H.; PIERCE, R.; CANNON, G.; DARIPALY, P. Six sigma: concepts, tools, and applications. **Industrial management & Data systems**, Emerald Group Publishing Limited, v. 105, n. 4, p. 491–505, 2005.

RAUCH, E.; LINDER, C.; DALLASEGA, P. Anthropocentric perspective of production before and within industry 4.0. **Computers & Industrial Engineering**, Elsevier, 2019. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360835219300233>. Acesso em: 03 mai. 2019.

SHEWHART, W. A. **Economic control of quality of manufactured product**. New York: ASQ Quality Press, 1931.

SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. **Statistics and computing**, Springer, v. 14, n. 3, p. 199–222, 2004.

SONI, D. **Supervised vs. unsupervised learning**. 2018. Disponível em: <https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d>. Acesso em: 17 out. 2018.

SURI, N. R.; ATHITHAN, G. **Outlier detection:** techniques and applications. Cham, Switzerland: Springer, 2019.

TUKEY, J. W. Exploratory data analysis. **[S.l.]: Addison-Wesley**, 1977.

VANDERPLAS, J. **Density Estimation**. 2007. Disponível em: <https://scikit-learn.org/stable/modules/density.html#kernel-density>. Acesso em: 09 jul. 2019.

VANSCHOREN, J.; BLOCKEEL, H.; PFAHRINGER, B.; HOLMES, G. Experiment databases. **Machine Learning**, Springer, v. 87, n. 2, p. 127–158, 2012.

WANG, Z.; BOVIK, A. C. Mean squared error: love it or leave it? a new look at signal fidelity measures. **IEEE signal processing magazine**, IEEE, v. 26, n. 1, p. 98–117, 2009.

YE, N.; CHEN, Q. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. **Quality and Reliability Engineering International**, Wiley Online Library, v. 17, n. 2, p. 105–112, 2001.

ZHOU, L. **Simplify machine learning pipeline analysis with object storage**. 2018. Disponível em: <https://blog.westerndigital.com/machine-learning-pipeline-object-storage//>. Acesso em: 17 out. 2018.