Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Instituto de Matemática e Estatística

Bruno Andrade Martins

Detecção de anomalias em séries temporais variáveis com LSTM-RNNS

Rio de Janeiro
2020

Bruno Andrade Martins

Detecção de anomalias em séries temporais variáveis com LSTM-RNNS

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Orientadora: Profª. Dra Nayat Sánchez Pi
Orientadora: Profª. Dra Rosa Maria Esteves Moreira da Costa

Rio de Janeiro
2020

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC-A

M386 Martins, Bruno Andrade.
Detecção de anomalias em séries temporais variáveis com LSTM-RNNS / Bruno Andrade Martins. – 2020.
58 f. : il.

Orientadoras: Nayat Sánchez-Pi, Rosa Maria Esteves Moreira da Costa
Dissertação (Mestrado em Ciências Computacionais) - Universidade do Estado do Rio de Janeiro, Instituto de Matemática e Estatística.

1. Aprendizado do computador - Teses. 2. Séries temporais - Teses. 3. Segurança de dados - Teses. 4. Séries temporais – Teses. I. Sánchez-Pi, Nayat. II. Costa, Rosa Maria Esteves Moreira da. III. Universidade do Estado do Rio de Janeiro. Instituto de Matemática e Estatística. IV. Título.

CDU 004

Patricia Bello Meijinhos CRB7/5217 -Bibliotecária responsável pela elaboração da ficha catalográfica

_____ _____

Assinatura Data

Bruno Andrade Martines

Detecção de anomalias em séries temporais variáveis com LSTM-RNNS

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Aprovada em 17 de abril de 2020.

Banca Examinadora:

Profª. Dra Nayat Sánchez Pi (Orientadora)
Instituto de Matemática e Estatística - UERJ

Profª. Dra Rosa Maria Esteves Moreira da Costa (Orientadora)
Instituto de Matemática e Estatística - UERJ

Prof. Drº. Luis Martí
Universidade Federal Fluminense - UFF

Prof. Drº. Zochil González Arenas
Instituto de Matemática e Estatística - UERJ

Rio de Janeiro
2020

DEDICATÓRIA

Dedico esta, bem como todas as minhas demais conquistas, aos meus mentores que acenderam a chamada da ciência em mim. (Dr. Luis Alfredo Vidal de Carvalho e Dr. Manuel Martins Filho), não posso deixar de mencionar os notáveis amigos, pesquisadores (Dr. Saul Eliahú Mizrahi e Dr.José Otávio Motta Pompeu e Silva), minha esposa (Michelle Pollyana S. S. de O. Martins) e meus filhos (Melissa e Leonardo) que tiveram que entender a minha ausência neste período e minha mãe (Ivone Andrade Martins), - Que falta você me faz, sei que estaria orgulhosa.

AGRADECIMENTOS

Quero agradecer, em primeiro lugar, a Deus, pela força e coragem durante toda esta longa caminhada.

Agradeço também a todos os professores que me acompanharam durante o mestrado, em especial a Prof. Dra. Rosa Maria Esteves Moreira da Costa.

‘‘Não tentes ser bem sucedido, tenta antes ser um homem de valor.’’ Sempre acreditei nisto e espero que este trabalho ajude a comunidade cientifica. —

Albert Einstein

RESUMO

MARTINS, Bruno Andrade. *Detecção de anomalias em séries temporais variáveis com LSTM-RNNS*. 2020. 57 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2020.

A quantidade e disponibilidade de dados e informações cresce continuamente, e paralelamente, cresce o número de anomalias, ameaças, intrusões ou ataques cibernéticos. Neste trabalho é proposto um método, que utiliza modelos combinados, híbridos de apren- dizagem de máquina, para analisar amostras de dados explorando previsão de cenários para detectar ameaças e anomalias. Nem sempre encontramos grandes conjuntos de dados reais - datasets - com amostras de dados equilibradas, ou seja, um número de sinalizações de anomalias e estados normais de um sistema, ou ambiente, em proporções iguais. O método foi desenvolvido explorando *Long Short-Term Memory* para trabalhar satisfato- riamente com diferentes configurações e tipos de conjunto de dados. Ele usa técnicas de agrupamento, classificação e previsão de dados, analisando diferentes estados ao longo do tempo, definindo o comportamento normal e detectando anomalias. Essa abordagem também usa técnicas para tornar menos necessário ter dados rotulados e sinalizados, ou qualquer outra indicação de seu uso ou informações confidenciais que não podem ser re- veladas. O modelo permitiu analisar as anomalias, observando suas vantagens e suas limitações, com modelos satisfatórios para detectá-las.

Palavras-chave: Aprendizagem de máquina. Séries temporais. Intrusão. Anomalias.

ABSTRACT

MARTINS, Bruno Andrade. *Anomaly detection in variable time series with LSTM-RNNS.* 2020. 57f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2020.

The amount and availability of data and information grow continuously, and in parallel, the number of anomalies, corrections, intrusions, or cybernetics increases. This work proposes a combined model, which explores hybrid machine learning, to analyze data samples and detect anomalies and threats through scenario prediction. We do not always find large sets of real data - datasets - with balanced data samples, that is, many anomaly signals and normal states of a system, or environment, in equal proportions. The method was developed by exploring Long Short-Term Memory to work satisfactorily with different configurations and types of data set. It uses data grouping, classification, and fo- recasting techniques, analyzing states over time, defining normal behavior, and detecting anomalies. This approach also uses techniques to make it less necessary to have labeled and flagged data or any other indication of its use or sensitive information that cannot be revealed. The model allowed analyzing the anomalies, observing their advantages and limitations, with satisfactory models to detect them.

Keywords: Machine learning. Time series. Intrusion. Anomalies.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da quantidade de dados e da necessidade de manipulação desses dados para a extração de conhecimentos, principalmente por parte das empresas, surgem uma série de problemas relacionados a transações, segurança, armazenagem entre outros. A detecção de anomalias é uma técnica usada para identificar padrões incomuns, que não estão em conformidade com o comportamento padrão esperado(CHANDOLA; BANERJEE; KUMAR, 2009). Outro termo usado para abordar esse problema é *outlier* (uma mudança no padrão estabelecido ao longo de uma série). A detecção de anomalias em sua forma mais genérica poderia ser empregada em vários campos da atuação e problemas, como sistemas de rede de computadores, ou detecção de intrusão (BONTEMPS et al., 2016). Esta atividade tem um papel muito importante em organizações privadas, ou governamentais. Tradicionalmente, as estratégias para detecção de anomalias, intrusão ou falhas em sistemas cibernéticos não são capazes de detectar eventos ainda desconhecidos (CHAUHAN; VIG, 2015). A detecção de anomalias, no geral, está relacionada a problemas de classificação, que constroem modelos comportamentais em seu estado normal e faz uso desses modelos para detectar novos padrões, que se desviam do mesmo (ANDO; GOMI; TANAKA, 2016; BONTEMPS et al., 2016). A detecção de anomalias em sua forma genérica, que é o objetivo deste trabalho, visa distinguir entre eventos ilegais, ou mal-intencionados, e comportamento normal nos ambientes analisados.

Em sistemas na área de segurança de redes, frequentemente as séries temporais multivariadas (MTS) são produzidas quando vários fluxos correlacionados dos dados são gravados ao longo do tempo (RANJAN et al., 2018). Frequentemente, as séries temporais podem conter dados aparentemente caóticos e aleatórios, principalmente, para os olhos humanos.Mesmo usando técnicas avançadas de mineração de dados sem uma abordagem que possa correlacionar dados com relação ao tempo, ou seja séries temporais, não seria possível um eficiente modelo de detecção de padrões e anomalias.

Atualmente, com o avanço da ciência de dados e com o uso de técnicas de inteligência artificial com aprendizagem de máquina, como as Redes Neurais Recorrentes (RNN, pela sigla em inglês) e a Memória Longa de Curto Prazo (LSTM, pela sigla em inglês) (BONTEMPS et al., 2016), é possível identificar as menores variações em dados de séries temporais, que podem conter padrões. Dessa maneira são detectadas as anomalias. Em geral, as técnicas de detecção de anomalias, ou intrusões, são baseadas no aprendizado do comportamento normal e esperado e nas ações anômalas já conhecidas e não inclui uma abordagem, como o uso de uma memória para trabalhar com eventos anteriores, ou previsões de estados nunca vistos, ou ainda, desconhecidos. Neste caso, temos vários trabalhos que utilizam Redes Neurais Recorrentes e LSTM como ferramenta de detecção de anomalias, pelo fato de serem ferramentas de maior precisão, de aplicação em

diferentes sistemas e que exploram múltiplas técnicas para obter resultados (DONAHUE et al., 2015; GREFF et al., 2016; VASSALI, 2018).

1.1 **Motivação**

Com o aumento da produção, armazenagem e uso de dados em geral, a partir da expansáo do uso de novas tecnologias com manipulação de grandes bases de dados, temos o fenômeno recorrente do aumento de problemas de intrusão e do número de anomalias. As vantagens de usar redes de comunicação para interconectar controladores e plantas físicas, por exemplo, conduz ao aumento de sistemas de controle em rede. No entanto, essa integração entre sistemas computacionais e ambientes físicos, também expõe esses sistemas a novas vulnerabilidades, ameaças e anomalias típicas do domínio cibernético, por meio de ataque por degradação de serviço, ataque de identificação do sistema e outros. Estes ataques afetam de maneira oculta e precisa, o comportamento do sistema (SÁ; CARMO; MACHADO, 2017), podendo gerar anomalias físicas e virtuais.

Segundo Pacheco et al. (2018), a qualidade dos dados é um aspecto fundamental para a criação de modelos para algoritmos de aprendizagem de máquina. *datasets* desequilibrados, ou relacionados à detecção de casos raros prejudicam a qualidade dos dados. Nesse caso, os algoritmos dependem de *datasets* corretamente construídos para avaliar, evoluir e gerar melhores modelos. Logo, desenvolver algoritmos de aprendizagem de máquina, que sejam menos dependentes de balanceamento ou rotulações pode ser estratégico.

Em geral, redes recorrentes podem ser usadas para identificações de vários tipos de ameças e anomalias e que têm sido objeto de estudos baseados em aprendizagem de máquina. Este assunto é de importância fundamental para a segurança cibernética (FERREIRA et al., 2019).

A detecção de anomalias tem sido bastante estudada e utilizada em uma ampla variedade de aplicações, como detecção de fraude e intrusão (ESKIN et al., 2002), detecção de falhas em sistemas críticos (KING et al., 2002), finanças (BORRAJO et al., 2011) ou indústrias (WOZNIAK; GRANA; CORCHADO, 2014), dentre outras. Entretanto, com o avanço da área de Aprendizado de Máquina, novas combinações tecnológicas podem ampliar as possibilidades de pesquisa nessa área.

1.2 **Problema de Pesquisa**

O problema que está estabelecido na pesquisa está representado pela seguinte indagação: É possível utilizar abordagem baseada em redes neurais recorrentes (RNN) e

modelos híbridos de machine learning para detectar falhas, anomalias, em *datasets* utilizando uma arquitetura genérica?

1.3 **Objetivos**

1.3.1 Objetivo Geral

Desenvolver um sistema que utiliza aprendizagem de máquina usando RNN e LSTM, que seja capaz de detectar anomalias em *datasets*.

1.3.2 Objetivos Secundários

(a) Analisar formas de detectar anomalias em *datasets* não equilibrados, contendo eventos raros em séries temporais;

(b) Verificar as diferenças e eficiência dos métodos utilizados;

(c) Utilizar o algoritmo em *datasets* com dados multivariados e que a abordagem possa ser adaptada para outros trabalhos, sem que haja necessidade de grande mudanças nos Hiper-parâmetros, ou com apenas leves calibrações.

1.4 **Justificativa**

Tradicionalmente, as estratégias para detecção de anomalias, intrusão ou falhas em sistemas cibernéticos e ciber-físicos não são capazes de detectar eventos desconhecidos de maneira eficiente. Assim, surge a necessidade de repensar novos modelos que superem os sistemas tradicionais para atender estas novas demandas. Com o uso de técnicas de inteligência artificial com aprendizagem de máquina acredita-se ser possível detectar as menores variações nos dados de séries temporais, que podem conter padrões. Identificando mudanças nos padrões, detecta-se as anomalias.

Em geral, os dados observados são desequilibrados e isso afeta a precisão e treinamento de um modelo de classificação (RANJAN et al., 2018). Em um problema de evento raro, temos um conjunto de dados desequilibrados. Ou seja, temos menos amostras marcadas positivamente do que negativas. Em um problema típico de evento raro, os dados marcados positivamente são de 5 a 10% do total. Em um problema de evento raro extremo, temos menos de 1% de dados rotulados positivamente. Por exemplo, no principal conjunto de dados usados nesta pesquisa e experimentos, esse número é de cerca

de 2,36%.

1.5 **Organização do Trabalho**

Este trabalho é composto por 6 (seis) capítulos. O primeiro apresenta a introdução geral, sua contextualização com dados históricos sobre a problemática. Em seguida são apresentadas as justificativas e os objetivos. O segundo capítulo trata do referencial teórico, apresentando os conceitos preliminares, que forneceram embasamento ao trabalho. O terceiro capítulo explora a compreensão do que são os métodos de busca, apresentando técnicas utilizadas no desenvolvimento do sistema. O quarto capítulo apresenta procedimentos e métricas adotadas. O quinto capítulo descreve os resultados das simulações e testes, e as discussões associadas. Por fim, são apresentadas as considerações finais, bem com as propostas de implementações futuras, seguida da lista de referências.

2 REFERENCIAL TEÓRICO

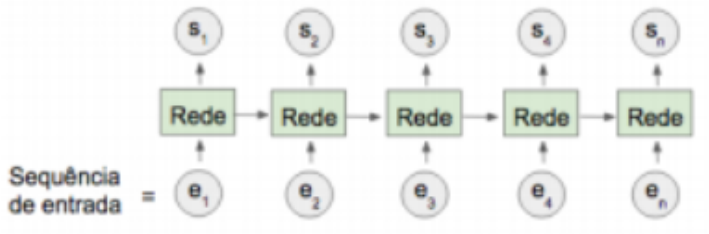
Neste capítulo, a revisão bibliográfica aborda temas relacionados a anomalias e estabelece alguns parâmetros e categorias quanto a aplicação de RNN LSTM para detectar falhas, padrões e seus desvios. Apresenta também, alguns trabalhos correlatos, para ilustrar aspectos apresentados.

2.1 Redes Neurais Recorrentes e Memória de Longo Prazo

As LSTMs (do inglês *Long Short-Term Memories*) são consideradas um avanço das redes neurais recorrentes tradicionais, as quais podem ser utilizadas em vários tipos de problemas, por serem capazes de perceber e recordar interações entre informações temporais, permitindo melhores respostas e ajustes, compensando as limitações inerentes às tradicionais RNN (do inglês *Recurrent Neural Network*). Na figura 1 é apresentada a estrutura de uma RNN.

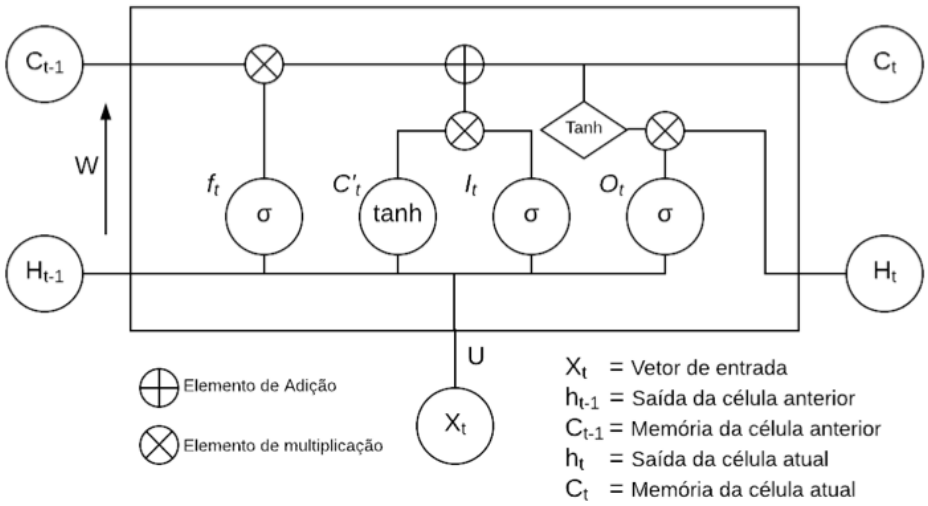
Percebe-se que cada célula executa uma mesma operação em cada etapa da sequência sendo n o número máximo de sequências de entradas, onde cada rede gera um resultado que influencia a saída. O resultado desta interação é a dependência da cadeia para o entendimento das informações atuais. O desenvolvimento das LSTM foi realizado a partir de 1997, por Sepp Hochreiter e Jürgen Schmidhuber que propuseram os primeiros modelos, atendendo vários tipos de problemas (DONAHUE et al., 2015). Uma das grandes vertentes das LSTM em relação às RNN é a possibilidade de uma rede de interação diferente dos módulos de repetição das RNN, com sua configuração permitindo maior autonomia à longo prazo. Segundo Greff et al. (2016), as redes LSTM possuem bons resultados para previsões temporais. Na figura 2 é apresentada uma estrutura de LSTM, composta de redes neurais e diferentes blocos de memória.

Figura 1 - Arquitetura RNN - O módulo de repetição em um modelo de RNN padrão contém uma única camada.



Fonte: Melo, 2018.

Figura 2 - Arquitetura LSTM - O módulo de repetição em uma LSTM contém quatro camadas de interação.



Fonte: Melo, 2018.

A estrutura LSTM é uma variação da RNN onde na representação na figura 2 temos uma linha na parte superior do diagrama, que é o estado da célula c e remete à memória interna da unidade. Na parte inferior temos o estado oculto e os *gates* f, i, o que consistem na engrenagem em torno do funcionamento do LSTM (ANDRYCHOWICZ et al., 2016).

Vassali (2018) estabelece alguns componentes para o LSTM:

(a) estado oculto: usado para determinar o que esquecer, entrar e sair no passo seguinte;

(b) estado de entrada: combinação do estado oculto e a entrada atual;

(c) estado interno: valores com função de memória;

(d) portão de entrada: decide se a entrada alcança o estado interno. Sendo este definido através da equação: (1) $i^{(t)} = \sigma(W^{ih}x^{(t)} + W^{ih}h^{(t-1)} + bi)$;

$i(t)$ representa o portão de entrada W^i os pesos, $h^{(t-1)}$ a saída do período anterior, $x^{(t)}$ entrada do período atual e bi bias.

(e) portão de esquecimento: decide se o estado interno desconsidera o anterior. Sendo definido pela equação: (2) $f^{(t)} = \sigma(W^{fx}x^{(t)} + W^{fh}h^{(t-1)} + bf)$;

$f(t)$ representa o portão de esquecimento W^f os pesos, $h^{(t-1)}$ a saída do período anterior, $x^{(t)}$ entrada do período atual e bf bias. Onde f é uma função sigmoide que é estabelecido qual valor deve ser "esquecido".

(f) portão de saída: decide se o estado interno é passado à saída e ao estado oculto no passo seguinte: (3) $o^{(t)} = \sigma(W^{ox}x^{(t)} + W^{oh}h^{(t-1)} + bo)$;

Finalmente $o(t)$ representa a saida, que determina qual parte da entrada $o(t)$ através de uma função sigmoid deve ir para saida, $h^{(t)}$.

O LSTM irá desenvolver um processo similar à RNN, mas as unidades LSTM incluem uma 'célula de memória', que pode manter as informações na memória por períodos maiores de tempo, tendo necessidade de parametrizar alguns parâmetros: (VASSALI, 2018):

(a) modelo computacional – a necessidade de escolha de um software para estruturação dos dados, construção dos gráficos, simulação, normalização dentre outros aspectos;

(b) descrição de condicionantes – são parâmetros pré estabelecidos no sentido de adicionar funcionalidades ao melhorar o modelo de forma a garantir desempenho e controle;

(c) parâmetros de rede – refere-se as amostras testes, números de entradas, as unidades de LSTM, função de perda, função de ativação.

2.2 As anomalias: características e definições

As anomalias podem ser classificadas segundo três tipo: as pontuais, contextuais (ou condicionais) e coletivas (CHANDOLA; BANERJEE; KUMAR, 2009).

As anomalias pontuais são consideradas os tipos mais simples e se configuram por estarem distantes do conjunto restante de dados. Este tipo de anomalia é a mais comumente encontrada. Um exemplo deste tipo de erro ocorre em transações bancárias, em processos de compra e venda, dentre outros.

As anomalias contextuais (ou condicionais) estão presentes sobretudo em séries temporais onde esta pode ser detectada a partir de dois atributos: atributos contextuais e comportamentais. O primeiro é utilizado para determinar o contexto para esta instancia, enquanto o segundo define as características não contextuais de uma instancia. Este tipo de anomalia é detectado a partir de um determinado dado fora de um contexto. Desta forma um exemplo deste tipo de erro seria um pico de dados, em um período regular sem demanda aparente.

Por fim, as anomalias coletivas consistem em uma ocorrência conjunto de pontos anômalo, necessitando de um conjunto de dados, ou seja, a partir da percepção de demanda normal sendo esta afetado por algo não aparente acarreta pico(s) repentino(s).

Algumas características são importantes no sentido de detectar a anomalia. Uma única instância de dados pode ser considerada anômala se estiver muito distante das

demais e é denominada de *Point Anomalie*. Podemos observar este tipo de caso em detecção de fraudes com cartão de crédito, se baseando pelo valor gasto conforme descrito por Oliveira (2015), que investigou a ocorrência combinando mais de 141 características, a fim de identificar padrões fraudulentos. Estes tipos de anomalias são comuns em dados de séries temporais (ST). Por sua vez, há necessidade de decompor esta estrutura em três partes: sazonalidade, tendência e ruído. As sazonalidades são padrões que se repetem com determinada frequência na ST. Segundo Brocklebank e Dickey (2002), esta repetição pode ser decorrente de causas naturais, econômicas e sociais e pode ser dividida em aditiva e multiplicativa. Na aditiva os padrões são estáveis sem que ocorra mudanças a nível global. Já a multiplicativa ocorre em função das mudanças macro. A tendência é, portanto, um padrão que pode ocorrer, normalmente, associado a um crescimento ou aumento de popularidade, ou seja, um movimento regular. Conforme destaca Ehlers (2009), esta pode ser do tipo:

(a) Linear;

(b) Exponencial;

(c) Amortecido.

Por fim, o ruído, ou resíduo, é o resultado do tratamento efetuado no sentido de excluir a sazonalidade e a tendência. Quando verificamos vários dados diferentes, estes ajudam coletivamente na detecção da anomalia, sendo especificados como *Collective Anomalies*. A detecção de anomalias se assemelha à remoção de ruído e detecção mudanças (*noise removal and novelty detection*). A detecção de novidade ou desvios está relacionada à identificação de um padrão não observado em outras novas observações. Logo, a remoção de ruído (NR) é o processo de imunizar a análise da ocorrência de observações indesejadas, removendo o ruído de um sinal de outra forma significativa. Em geral, anomalias podem ter uma natureza positiva ou não, e esta classificação dependerá do seu contexto.

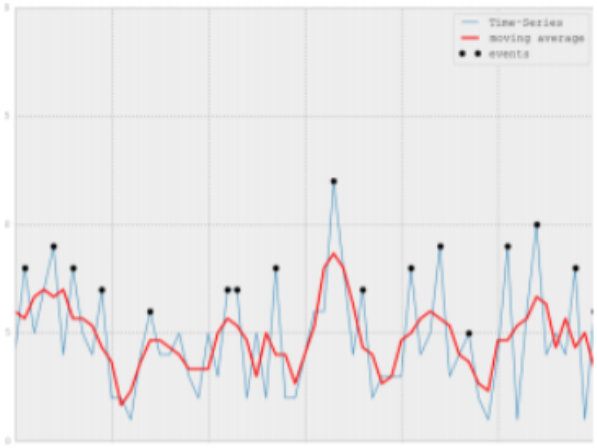
2.3 **Técnicas de Detecção de Anomalias**

Existem diferentes técnicas que podem ser utilizadas para detecção de anomalias além das já mencionadas, RNNs e LSTM, as quais são apresentadas a seguir:

2.3.1 Métodos Estatísticos ou Métodos Baseados em Modelos

Os métodos estatísticos, ou baseados em modelos, apoiam a identificação de irregularidades em dados e sinalizam os pontos de dados que se desviam das proprieda-

Figura 3 - Exemplo de um Modelo Estatístico



Fonte: Mata, 2017.

des estatísticas comuns de uma distribuição, incluindo média, mediana, modo e quantis. Esta técnica pode ser dividida em dois tipos: paramétricas e não paramétricas (HAN; PEI; KAMBER, 2011). Os paramétricos testam hipóteses sobre parâmetros específcicos, como, média, desvio-padrão, etc. Os não-paramétricos testam hipóteses sobre parâmetros, distribuição, ou relações entre classes de amostras.

Matematicamente, uma média móvel simples de período n também pode ser definida como um "filtro de baixa passagem". A figura 3 mostra uma linha vermelha que caracteriza uma suavização. As médias móveis suavizam os valores em séries temporais, filtrando ruído provocado por flutuações de curto prazo e destacando tendências, ou ciclos de longo prazo.

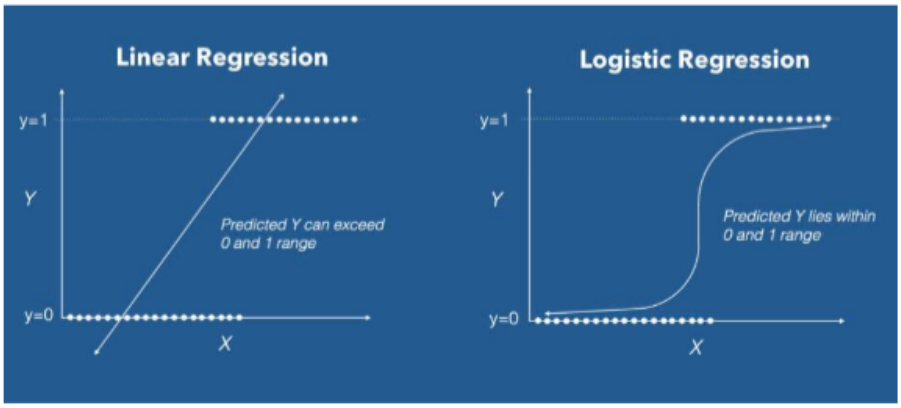
O filtro de baixa passagem permite identificar anomalias em casos de uso simples, mas há certas situações em que essa técnica não funciona. Os dados contém ruído que pode ser semelhante ao comportamento anormal, porque o limite entre o comportamento normal e anormal muitas vezes não é preciso. A definição de anormal, ou normal, pode frequentemente mudar. Portanto, o limite com base na média móvel nem sempre se aplica.

Uma vantagem deste tipo de análise é a confiabilidade tendo em vista o uso de modelos matemáticos, que justificam o conjunto de dados na detecção com base em hipóteses assumidas, conforme destaca Tan, Steinbach e Kumar (2009), que em seus estudos evidenciaram a eficiência na execução com o método.

Porém, com base em um padrão de sazonalidade, há necessidade de envolver outros métodos mais sofisticados, como decompor os dados em várias tendências, a fim de identificar a mudança na sazonalidade.

Algumas técnicas populares de aprendizado de máquina para detecção de anomalias são destacadas a seguir.

Figura 4 - Comparação modelo linear x logístico



Fonte: Estatsite, 2018

2.3.2 Regressão Logística

A regressão logística é uma técnica estatística que tem como objetivo produzir um modelo que realiza a predição de valores a partir de uma variável frequentemente binária, considerando uma série de variáveis explicativas contínuas e/ou binárias. Segundo Hosmer (2000), por meio do modelo de regressão logística, é possível calcular a probabilidade de um evento usando uma função de ligação.

Esta técnica pode ser dividida em três tipos: regressão logística binominal, regressão logística ordinária e regressão logística multinomial. Em especial, o modelo de regressão logística é o mais usual e apresenta uma característica que refere-se a variável resposta Y, que passa a ser binária ou dicotômica, assumindo dois valores. Assim, pode ser definido binominalmente através da equação a seguir:

$log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x$, onde n é conhecido como ensaios de Bernoulli e p é a probabilidade.

Como resultado da transformação obtemos a seguinte representação de acordo com a figura **??**.

Entre os modelos, é o mais usado em decorrência da facilidade de manuseio dos dados, não necessidade de sistemas computacionais robustos nem dos parâmetros do modelo serem usuais nas estimativas. Assim, o modelo busca estimativas probabilísticas da variável dependente a partir do conhecimento de outras variáveis sendo estes contidos em um intervalo entre 0 e 1. Na figura 4 há um comparativo do modelo de regressão linear e o logístico.

Em relação ao modelo linear a técnica apresenta algumas vantagens no que diz respeito a normalidade e linearidade. Dentre as vantagens do modelo de regressão logístico podemos destacar a facilidade no manuseio das variáveis, resultados probabilísticos, facilidade de interpretação, poucas condicionantes além do alto grau de confiabilidade.

Segundo Jr., Lemeshow e Sturdivant (2013), os modelos apresentam algumas características importantes:

- As variáveis explicativas podem possuir tanto características quantitativas, quanto qualitativas;
- A variável resposta será categórica;
- Os erros provenientes desta regressão não seguirão uma distribuição normal.

2.3.3 Detecção de Anomalia Baseada em Densidade

A Detecção de Anomalia Baseada em Densidade (do inglês *Density-Based Anomaly Detection*) baseia-se no algoritmo de vizinhos k mais próximos (ZHANG; LIN; KARIM, 2018). Suposição: Os pontos de dados normais ocorrem em torno de um bairro denso e as anormalidades estão distantes.

O conjunto de pontos de dados mais próximo é avaliado usando uma pontuação, que pode ser a distância Euclidiana ou uma medida semelhante, dependendo do tipo de dados (categórico ou numérico). Eles podem ser classificados em dois algoritmos:

K-vizinho mais próximo: k-NN é uma técnica de aprendizagem não paramétrica usada para classificar os dados com base nas semelhanças nas métricas de distância, como a distância Euclidiana, Manhattan, Minkowski ou Hamming.

Densidade relativa dos dados (*Relative density of data*): conhecida como fator *outlier* local (LOF). Este conceito é baseado em uma métrica de distância chamada distância de alcançabilidade.

2.3.4 Detecção de Anomalia Baseada em *Cluster*

O *clustering* é um dos conceitos mais populares no domínio da aprendizagem não supervisionada. Suposição: Os pontos de dados que são semelhantes tendem a pertencer a grupos ou *clusters* semelhantes, conforme determinado pela sua distância dos centroides locais. Então, temos:

K-means é um algoritmo de *clustering* amplamente usado. Cria 'k' grupos semelhantes de pontos de dados. As instâncias de dados que estão fora desses grupos podem ser marcadas como anomalias.

2.3.5 Detecção de anomalia baseada em máquina de vetores de suporte

Uma máquina de vetores de suporte (do inglês *Support Vector Machine* (SVM) é outra técnica eficaz para detectar anomalias, também baseada em sistema não supervisionado. A figura 5 apresenta uma análise simplificada deste método.

Um SVM é normalmente associado ao aprendizado supervisionado, mas existem extensões como OneClassCVM, que podem ser usadas para identificar anomalias como problemas não supervisionados (nos quais os dados de treinamento não são rotulados). O algoritmo aprende um limite flexível para agrupar as instâncias de dados normais usando o conjunto de treinamento e, em seguida, usando a instância de teste, ajusta-se para identificar as anormalidades que estão fora da região aprendida. Dependendo do caso de uso, a saída de um detector de anomalia pode ser valores escalares numéricos para filtragem em limites específicos de domínio ou rótulos textuais (como binário/vários rótulos).

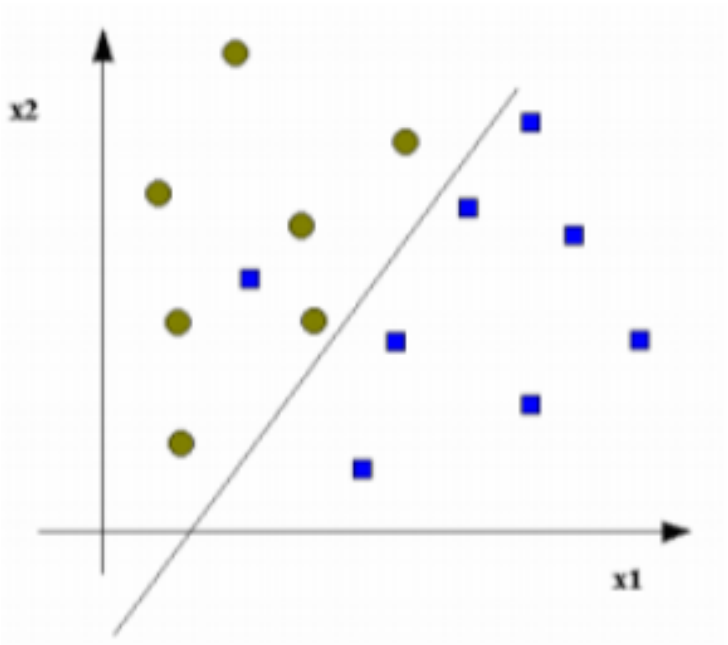
O filtro de baixa passagem permite identificar anomalias em casos de uso simples, mas há certas situações em que essa técnica não funciona. Alguns exemplos podem ser destacados:

- Os dados contêm ruído que pode ser semelhante ao comportamento anormal, porque o limite entre o comportamento normal e anormal muitas vezes não é preciso;
- A definição de anormal ou normal pode frequentemente mudar, pois adversários maliciosos se adaptam constantemente. Portanto, o limite com base na média móvel nem sempre se aplica;
- O padrão é baseado na sazonalidade. Isso envolve métodos mais sofisticados, como decompor os dados em várias tendências, a fim de identificar a mudança na sazonalidade.

2.3.6 Detecção de anomalia baseada em Floresta de isolamento (IForest)

Floresta de isolamento (IForest): Uma maneira eficiente de realizar formar detecção de outliers em conjuntos de dados de alta dimensão é usar florestas aleatórias. Ele isola as observações selecionando aleatoriamente um recurso e em seguida, escolhendo aleatoriamente um valor de divisão entre os valores máximo e mínimo do recurso selecionado. (HUANG et al., 2017)

Figura 5 - Modelo de Vetores de Suporte



Fonte: Fonte: Kinto, 2011.

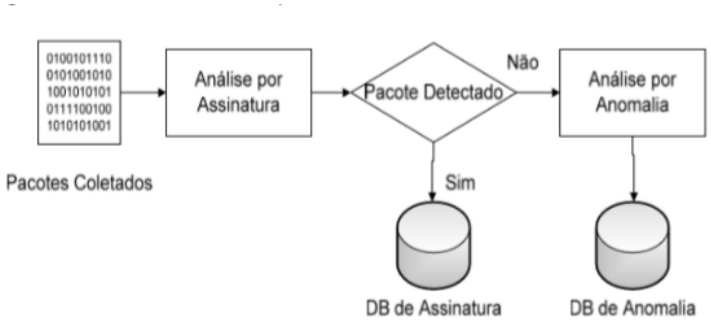
2.3.7 Detecção com algoritmo de agrupamento baseado em densidade (DBSCAN/ARENA)

O algoritmo DBSCAN agrupa pontos que estão intimamente agrupados. Pontos de dados isolados em baixa densidade regiões são marcadas como outliers. O treinamento do modelo de predição também é baseado no algoritmo de agrupamento espacial baseado em densidade. O sistema Arena usa os dados históricos de monitoramento como seu conjunto de dados de treinamento. Todos os pontos de monitoramento históricos são classificados em três categorias: pontos centrais, pontos de borda e outliers. O treinamento do modelo de predição requer apenas dois parâmetros: minPts e eps. Um ponto p é um ponto central se pelo menos os pontos minPts estiverem dentro da distância de eps. Um ponto p é um ponto de borda se pelo menos um ponto estiver dentro da distância de eps. Os pontos de descanso, que não são alcançáveis de nenhum outro ponto, são outliers. No processo de treinamento, os pontos centrais e os pontos de borda são mantidos como o modelo do sistema Arena. (HUANG et al., 2017)

2.3.8 Sistema Híbrido

Os Sistemas de Detecção de Intrusão Híbridos (IDS híbridos ou IDS-H) são instrumento capazes de detectar anomalias a partir da centralização de outros sistemas com

Figura 6 - Sistema Híbrido



Fonte: Neto, 2011.

diferentes aplicações, assim o método combina conhecimento (uso indevido) e comportamento (anomalia). Damaceno (2008) destaca que estes sistemas reúnem as características de NIDs (*network intrusion detection systems*) e HIDs (*host intrusion detection systems*).

A figura 6 apresenta uma representação esquemática do sistema híbrido.

O sistema é baseado na análise por meio da captura de pacotes sendo estes analisados e comparados com padrões, ou assinaturas já conhecidas, desta forma tem como funções a coleta, análise, armazenagem e respostas a estas informações.

Por sua vez, estes sistemas utilizam fontes de informações de determinado equipamento (*host*), ou de fontes oriundas da rede o qual está interligado, assumindo o controle e a segurança computacional do ambiente, tornando a detecção mais eficaz. Dentre as vantagens de um sistema híbrido, temos uma maior segurança de rede, utiliza de padrões já pré-estabelecidos e fácil manuseio do sistema, bem como auxilia as partes interessadas em alertas de prevenção.

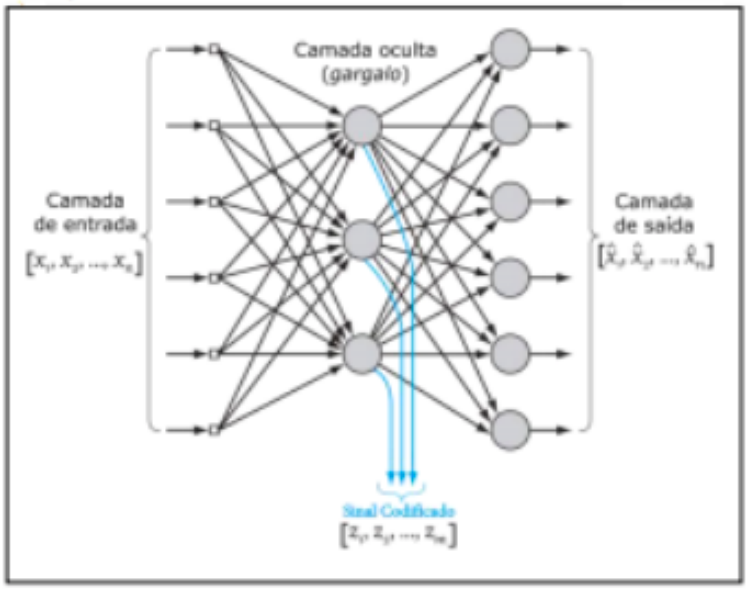
Entretanto, são poucos os sistemas com este ferramental e de forma integrada, logo, usualmente muitos destes meios estão limitados em tráfego ou anomalias, porém são os mais aconselháveis a serem utilizados em sistema de detecção de anomalias.

2.3.9 Redes *Feedforward* com *autoencoder*

Um *autoencoder*, representado na figura 7, é um tipo de rede neural artificial utilizado para aprender codificações de dados eficientes sem supervisão humana. O objetivo de um *autoencoder* é aprender uma representação (codificação) para um conjunto de dados, tipicamente para redução de dimensionalidade. Junto com o lado de redução, um lado de reconstrução é apreendido, onde o *autoencoder* tenta gerar a partir da codificação reduzida uma representação o mais próximo possível de sua entrada original:

$L(x, g(f(x)))$, onde L pode ser uma função de perda, sendo x e $g(f(x))$ as diferenças entre entrada e saída.

Figura 7 - Modelo *autoencoder*



Fonte: Hakin, 2009.

O *autoencoder* tenta aprender a aproximar a função de identidade. O princípio utilizado é o de que, quando ocorrer uma anomalia, isso deve afetar a interação entre as variáveis existentes no contexto a ser analisado. Quando isso acontece, aumenta o erro nas redes de reconstrução das variáveis de entrada. Ao monitorar o erro de reconstrução, pode-se obter uma indicação da possível anomalia, pois esse erro aumentará, se manterá, ou diminuirá, mas possivelmente irá gerar um padrão detectável. Na distância de Mahalanobis é usada a distribuição de probabilidade do erro de reconstrução para identificar se um ponto de dados é normal ou anômalo, conforme a figura 8.

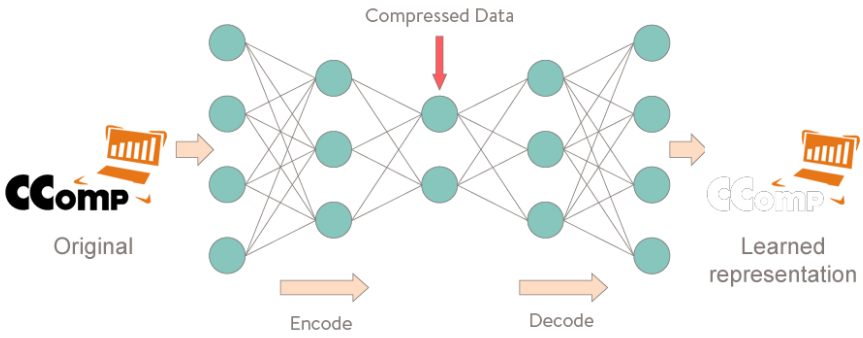
O objetivo de um *autoencoder* é aprender uma representação (codificação) para um conjunto de dados, tipicamente para redução de dimensionalidade. Junto com o lado de redução, um lado de reconstrução é apreendido, onde o *autoencoder* tenta gerar a partir da codificação reduzida uma representação o mais próximo possível de sua entrada original. O *autoencoder* tenta aprender a aproximar a seguinte equação de identidade:

$f_{w,b}(x) \approx x$

O erro de reconstrução passa a ser otimizado segundo o modelo de *autoencoder* padrão, para que o erro de reconstrução seja minimizado. O erro quadrático tradicional é frequentemente usado, como explicitado na equação a seguir:

$L(x,x') = \|x - x'\|^2$

Figura 8 - Representação da Rede Neuronal *autoencoder*



Fonte: Elaborado pelo autor, 2019.

2.4 Modelo LSTM *autoencoder*

Na sessão anterior, foi descrito o método *autoencoder*. No entanto, os *autoencoders* lidam alguns problemas em relação aos dados de séries temporais, como o tamanho diferente de entradas e capacidade de análise de uma ordem temporal, por exemplo.

As LSTM são redes neuronais que foram desenhadas para receberem dados sequenciais como entrada. Estas redes conseguem aprender sobre uma linha temporal e guardar na memória as informações importantes. Logo, uma LSTM *autoencoder* é uma rede neural capaz também, de trabalhar e perceber nuances em dados sequenciais, com a combinação de duas poderosas arquiteturas: uma arquitetura de LSTM na parte de *encoding* e de *decoding* do *autoencoder* (BROWNLEE, 2018). As primeiras camadas deste tipo de arquitetura, *encoding*, fazem uma compressão dos dados de entrada. Depois, é usada uma camada de *repeat vector* de forma a distribuir os dados comprimidos pelas varias etapas temporais do *decoder* e por último, vem a camada de *decoder* que tenta reconstruir a entrada (LARZALERE, 2019). Este tipo de rede foi usada em 2015 em suas primeiras versões, com um modelo sem supervisão (SRIVASTAVA; MANSIMOV; SALAKHUDINOV, 2015). A escolha deste tipo de redes neural nesta dissertação deveu-se ao fato da abordagem ter a capacidade de detectar anomalias escondidas na linha temporal que o *autoencoder*, por si só, poderá não conseguir extrair; e a capacidade do *autoencoder* de camada densa poder detectar anomalias de forma mais genérica, a fim de obter um modelo genérico, abrangente e eficiente.

2.5 **Trabalhos Correlatos**

Das diferentes problemáticas existentes na área de detecção de anomalias, a aplicação em Internet das Coisas (IoT - Internet of Things), por abranger novas tecnologias e regras de comunicação, merece atenção como destaca Bueno (2018), em seu trabalho de aplicação de Sistema de Detecção de Intrusão (IDS) a partir de anomalias de tráfego. Neste mesmo sentido, Sanz et al. (2018), usando grafos, obteve uma melhor análise das anomalias e uma redução de erros em mais de 1430 vezes. Já Souza et al. (2016), propuseram a utilização de algoritmo de descoberta de anomalias Isolation Forest, para proteger sensores fisicamente, de maneira a não gerarem dados incorretos. Outro aspecto que inspira estudos é quanto há anomalias em decorrência de intrusão, assim como destaca Silva (2009), por meio de algoritmo das Células Dendríticas (DCA) que permitiu uma análise temporal, conseguiu melhorias na qualidade da detecção de intrusão em redes de computadores e melhor medição do desempenho. Já Zarpelão (2009), propôs um sistema de detecção de anomalias em três níveis e o resultado mostrou altas taxas de detecção, a partir de correlacionamentos, o que permitiu uma melhor análise e tomada de decisão. Outra vertente geradora de grande problemática são as fraudes nas instituições financeiras, que sujeitam o cidadão a erros e perdas materiais. Neste domínio, Oliveira (2015), destaca em seu trabalho um comparativo da detecção de fraudes financeiras utilizando vários modelos como, sistemas baseados em regras, algorítimos genéticos, árvores de decisão e regressão logística. Ele observou algumas dificuldades em suas avaliações, mas também, a alta precisão e rapidez para classificar problemas complexos usando modelos baseados em redes neurais.

2.5.1 Trabalhos Correlatos com Aprendizagem de Máquina

A aprendizagem de máquina convencional comumente faz uso de análise estatística para resolução de problemas. Sua eficiência é correlacionada com o pré-processamento das informações de entrada, realizado por um processo de engenharia de dados. O aprendizado profundo é uma subclasse do aprendizado de máquina, que por sua vez, tem como característica usar grande quantidade de dados, precisando de menos ou nenhum pré-processamento, além de possuir menor variação a mudanças irrelevantes para o aprendizado, sendo sensível a anomalias importantes (LECUN; BENGIO; HINTON, 2015).

Moustafa et al. (2019), a partir de um levantamento bibliográfico, mostram vários métodos utilizando *Machile Leaning* e *Deep Learning*, que podem ser usados na detecção de intrusão e anomalias, bem como critérios para validação dos métodos. Ramakrishnan e Soni (2018) abordam a previsão de tráfego de rede como séries temporais, usando RNNs. Os autores concluem que as técnicas são promissoras, obtendo melhores resultados do que

métodos estatísticos tradicionais de previsão.

Em (VINAYAKUMAR et al., 2017) é feito um levantamento de técnicas de *Deep Learning* utilizadas para realizar previsões em tráfego de rede; os procedimentos avaliados são RNNs e suas subcategorias. O LSTM, apresentou melhor performance com relação às técnicas *Feedforward* e RNN comuns.

Vários autores (QIN; CHEN; LIN, 2018; ALTHUBITI; JONES; ROY, 2018; MIRZA; COSAN, 2018) usam LSTM para detecção de anomalias e em todos eles os autores argumentam sobre a necessidade de manter a dependência dos dados durante a análise, bem como a resolução do problema de dependências a longo prazo, que é resolvido com essa técnica.

Por fim, Lopez-Martin et al. (2017) combinam dois métodos de *Deep Learning*, o LSTM e CNN (*Convolutional Neural Network*), para a criação de um classificador de tráfego. Para o uso da LSTM, os dados são organizados na forma de uma matriz contendo uma dimensão temporal e um vetor de atributos. Já no segundo modelo, os dados utilizados para a classificação do tráfego são tratados como se fossem uma imagem. O trabalho concluiu que a combinação de ambos os métodos possui a melhor performance de classificação, porém segundo eles,o LSTM é o que mais contribui para um bom resultado.

3 PROCEDIMENTOS METODOLÓGICOS

A seguir, são apresentados os procedimentos metodológicos utilizados na pesquisa. De acordo com Souza (2004), esta metodologia atua como instrumento que auxilia a formulação das etapas da pesquisa. Já para Teixeira (2006), "é através da metodologia que se estuda, descreve e explica os métodos aplicados ao longo do trabalho, de forma a sistematizar os procedimentos adotados durante as várias etapas, procurando garantir a validade e a fidelidade dos resultados. A metodologia, segundo o autor, tem como objetivo analisar as características dos vários métodos disponíveis, observando as suas vantagens e desvantagens." Desta forma, o trabalho parte de uma fundamentação teórica e de estudos de trabalhos correlatos e descreve os procedimentos adotados ao longo deste capítulo.

3.1 Tipo de Pesquisa

A pesquisa é de caráter exploratório a partir de pesquisa bibliográfica (GIL, 2007) em artigos, dissertações, teses. Para Rudio (2016), pesquisa é um processo de investigação que se interessa em descobrir as relações existentes entre os aspectos que envolvem os fatos, fenômenos, situações ou coisas, é um "procedimento reflexivo sistemático, controlado e crítico, que permite descobrir novos fatos ou dados, relações ou leis, em qualquer campo do conhecimento." O trabalho também é de caráter qualitativo, onde são analisados dados baseados nas técnicas citadas para detectar falhas, anomalias, padrões e seus desvios.

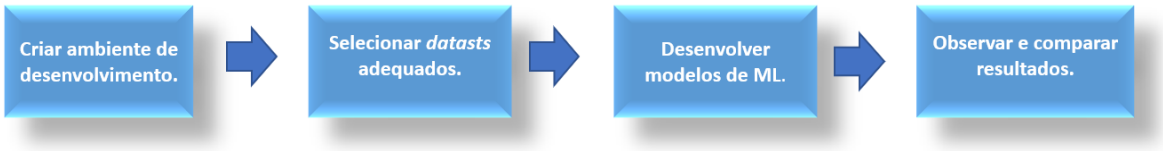
3.2 Métodos de Procedimentos

De acordo com Marconi e Lakatos (2011, p.223), os métodos de procedimentos "são as atividades práticas necessárias para a aquisição dos dados com os quais se desenvolverão os raciocínios (previsto nos objetivos específicos) que resultarão em cada parte do trabalho.". O método a ser utilizado no trabalho será uma revisão bibliográfica na qual segundo Marconi e Lakatos (2011, p.24), "... implantado a partir da análise de uma realidade segmentada, onde os procedimentos de coleta basearam-se em levantamento dos dados e informações". Em seguida foi feito uma análise experimental usando aprendizagem de máquina para detectar falhas, anomalias, padrões e seus desvios.

Desta forma a análise experimental foi dividida em algumas etapas descritas na Figura 9.

As mesma etapas são consideradas na implementação do modelo LSTM-*autoencoder*, e ainda nas fases experimentais serão obtidos os resultados com técnicas de medição de

Figura 9 - Etapas experimentais



Fonte: Elaborado pelo autor, 2019.

eficiência, comparações entre os modelos implementados e com outros modelos da literatura.

O trabalho considera as seguintes etapas:

(a) Criar ambiente de desenvolvimento adequado para o treinamento computacional e teste das soluções;

(b) Selecionar *datasets* adequados para a proposta;

(c) Desenvolver modelos de Machine Learning;

(d) Observar e comparar resultados.

3.3 **Revisão de literatura**

O levantamento bibliográfico foi realizado por meio de uma Leitura Exploratória de todo o material selecionado, Leitura Seletiva e Registro das informações. Quanto à análise e comparações estas foram compiladas e apresentadas na tabela 1.

Nesta pesquisa foi possível estabelecer uma base a fim de fundamentar o estudo em publicações recentes e correlatas ao tema central.

No estudo percebe-se o desenvolvimento de trabalhos em diferentes áreas: comércio, administração pública, bancos entre outros segmentos. Destaca-se o setor de comércio, que apresenta muitos estudos com soluções voltadas à atendimento e tecnologias (NOGOSEKE, 2019; CAPTANIO, 2019).

Alguns trabalhos estão voltados para a área financeira, em especial bancos. Os setores de energia e tecnologia vêm pautados por estudos de caso, com a adoção de sistemas e métodos bem definidos (NELSON, 2017; CASTELÃO, 2020). Porém, é possível identificar pesquisas em setores como hídrico e industrial (SANTOS, 2019). Os resultados desta pesquisa serviram de base para a estruturação da proposta de um modelo genérico para identificação de anomalias em *datasets*.

Tabela 1 - Análise de publicações

Autor	Ano	Sistema	Técnica	Resultados
SANTOS, G.	2019	Software MATLAB	Baseada em séries históricas	Resultados apontam maior precisão quando comparado com outro sistemas de simulação
VASSALI, L. C.	2018	Pacote Keras, Pandas para estruturação dos dados, Matplotlib para a construção de gráficos, scikit-learn para regressão linear e normalização dos dados e NumPy pela estrutura de vetor e funções matemáticas.	Baseada em séries históricas	Facilidade de reaplicação, demonstração da eficácia e ilustração dos resultados
CASTELÃO, R.	2016	Python	Preços de abertura de ações de empresas participantes da BOVESPA e NASDAQ	Resultados demonstram um melhor horizonte de previsão com melhor assertividade e tendências futuras mais plausíveis
NELSON, D. M. Q.	2017	-	Dados reais da Bolsa de Valores de São Paulo	Acurácia média de até 55,9% ao prever se o preço de uma determinada ação irá subir ou não no futuro imediato
Lodoli et al. (2016)	2016	Linguagem C++	Classificação de criptogramas	No uso de GPU mostrou-se eficaz, porém com dados de classificação pouco conclusivos.
NOGOSEKE, L. F.	2019	-	Melhoria nos sistema de gestão de atendimento ao cliente	Possível obter resultados relevantes em tarefas de respostas automáticas a perguntas de larga escala
Fiorio (2019)	2019	-	Foi utilizado um pré-amplificador valvulado para guitarra elétrica	O resultados apontam uma diminuição substancial do erro, somente acelera o processo de treinamento.
CAPTANIO, S.	2019	-	Dados históricos de empresas do polo moveleiro	O modelo conseguiu realizar a previsão de demanda com base em indicadores econômicos.

Fonte: Elaborado pelo autor,2020.

4 DESENVOLVIMENTO

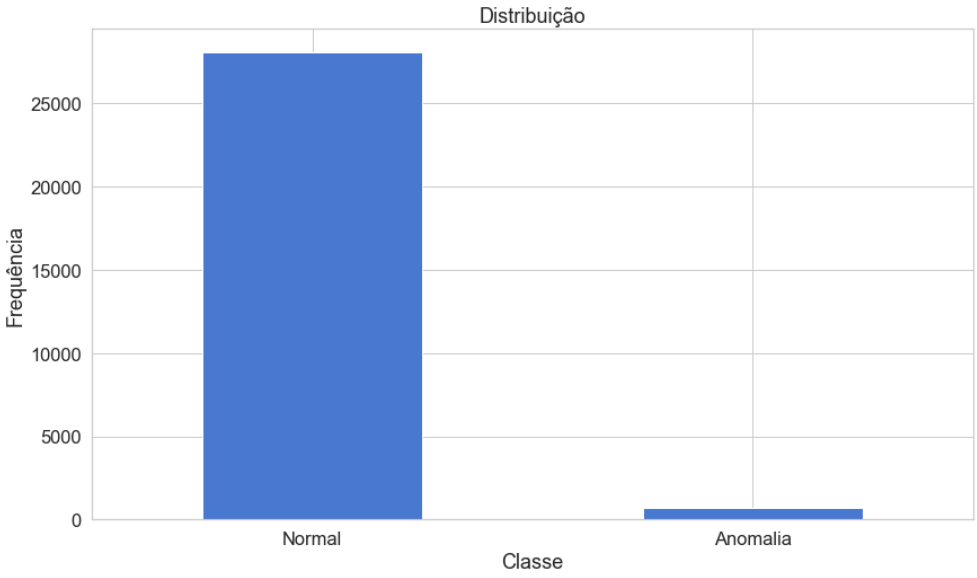
Neste trabalho, foi proposto um modelo genérico para detecção de anomalias, baseado no aprendizado de rede neural recorrente de memória (RNN), em específico na Memória de Longo Prazo (LSTM). O RNN LSTM e *Autoencoders* Redes Feedforward são treinados com dados de séries temporais de requisições de serviços para servidor Yahoo (RESEARCH, 2016), e são capazes de detectar anomalias em etapas de multiplexação à frente de uma entrada. O modelo de algoritmo foi adaptado para ser genérico (ERGEN; MIRZA; KOZAT, 2017) para possibilitar sua aplicação em outros conjuntos de dados, ou mesmo, para a resolução de diversos outros problemas, seja no campo da medicina detectando padrões anômalos em sinais cerebrais de EEG, seja em potenciais ataques terroristas usando análise de dados de sensores, ou ainda, analisando dados de dispositivo IoT em uma rede doméstica(CHAUHAN; VIG, 2015).

4.1 ***Dataset*** **selecionado**

O *dataset* selecionado foi o *S5-Labeled Anomaly Detection dataset* da Yahoo Webscope. Um conjunto de dados relacionando padrões normais e desvios anômalos foi usado nos experimentos. Esses dados são pedidos de serviço para um servidor yahoo (RESEARCH, 2016) este *dataset* é dividido em arquivos *.csv*, sendo que o primeiro grupo contém 67 arquivos. Todos contendo séries temporais com sinalização de interrupção e com sinalização boleana de anomalia. A natureza das afirmações e dados são desconhecidas o que torna excelente para a proposta, além de ser um conjunto de dados já consolidado e considerado confiável pela comunidade de pesquisa. No presente trabalho, foi utilizado 1/3 do primeiro pacote deste *dataset* nos testes.

Na figura 10 apresenta um gráfico de distribuição dos dados do *dataset*. Observa-se que o *dataset* é altamente desequilibrado. Entretanto, as marcações analisadas como normais compõem a maior parte dos dados.

Figura 10 - Gráfico de distribuição dos dados normais e os dados sinalizados como anômalos



Fonte: Elaborado pelo autor, 2019.

4.2 **Procedimentos**

Como o objetivo desta proposta é trabalhar com vários *datasets*, como os de eventos extremamente raros, usamos técnicas para sub amostrar exemplos de dados rotulados negativamente, com o intuito de ter um conjunto de dados mais balanceado. Essa abordagem já torna possível trabalhar com *datasets* de diferentes equilíbrios. Como temos 2,36% de dados rotulados positivamente, a sub amostragem resultará em um conjunto de dados irregular. A abordagem de aprendizado de máquina, por exemplo, SVM ou floresta aleatória, ainda funcionaria em um conjunto de dados desse tamanho. No entanto, ela teria limitações em sua precisão.

Para classificar eventos raros com o *autoencoder*, os dados foram divididos em duas partes: positivamente e negativamente. Os dados que foram marcados negativamente são tratados como estado normal do processo. Um estado normal é quando o processo não possui eventos. Iremos ignorar os dados rotulados positivamente e treinaremos um codificador automático apenas para os dados rotulados negativamente. Esse codificador automático aprende os recursos do processo normal. Um *autoencoder* bem treinado irá prever qualquer novo dado proveniente do estado normal do processo (pois terá o mesmo padrão ou distribuição), portanto, o erro de reconstrução será pequeno. No entanto, se tentarmos reconstruir os dados de um evento raro, o codificador automático terá problemas. Isso fará com que o erro de reconstrução seja alto durante o evento raro. Podemos detectar erros de reconstrução altos e rotulá-los como uma previsão de eventos raros. Essa

Tabela 2 - Dados Datasets

Formato do *dataset*	Anomalias	Dados Normais	Tipos de Dados	Algoritmo de classificação
28751, 3	679	2073	Numérico	1

Fonte: Elaborado pelo autor, 2019.

estratégia foi a adotada no método de detecção de anomalias.

Os dados contêm cerca de 28 mil linhas conforme pode ser visto na Tabela 2. Uma das colunas contém os rótulos binários, com 1 indicando uma anomalia. As demais colunas são preditores. Existem cerca de 680 amostras marcadas como positivas, sendo 2,36% do total.

A capacidade do LSTM de realizar previsões em séries temporais e poder ser aplicada na resolução dos mais variados tipos de problemas é conhecida, assim como sua eficiência em detectar anomalias do *autoencoder*. Neste trabalho, a aplicação das técnicas combinadas será usada na detecção de anomalias em datasets com requisição de serviços a servidores, pois o mesmo se comporta como uma série temporal. A ideia central será gerar tráfego a partir do treinamento da rede neural e compará-lo com um tráfego real, visando averiguar a existência de uma anomalia. Será utilizado um modelo híbrido *Autoencoder*-LSTM para melhorar a precisão e tornar a aplicação mais genérica. Foi escolhida a linguagem Python para a implementação deste sistema. Python é uma linguagem interpretada, interativa e orientada a objetos e tem se tornado uma das mais populares linguagens de programação usadas em Machine Learning e Data Science (NAGPAL; GABRANI, 2019).

Para medição da eficiência do modelo adotado e o devolvimento da solução foi utilizada a IDE Jupyter e as seguintes bibliotecas para o processamento e manipulação dos dados:

Pandas

Para manipulação e análise de dados. Em particular, oferece estruturas de dados e operações para manipular tabelas numéricas e séries temporais.

Numpy

Para suporte a arrays e matrizes multidimensionais, possuindo uma larga coleção de funções matemáticas para trabalhar com estas estruturas.

Pickle

Para converter hierarquia de objetos em um fluxo de bytes e para tornar esse fluxo de bytes em hierarquia de objetos novamente.

Matplotlib

Para plotagem gráfica de objetos a Matplotlib utiliza a Numpy e sua extensão de

matemática numérica. Ela fornece uma API orientada a objetos para incorporar gráficos em aplicativos usando kits de ferramentas GUI de uso geral, como Tkinter, wxPython, Qt, GTK e outros.

TensorFlow

Até o momento deste projeto o TensorFlow é uma das melhores bibliotecas de código aberto para aprendizado de máquina possui uma ampla variedade de tarefas. Utilizado para criação e treinamento de redes neurais para detectar e decifrar padrões e correlações, o TensorFlow é estruturado de forma análoga à forma como humanos aprendem e raciocinam (ABADI et al., 2016).

Seaborn

Seaborn é outra biblioteca de visualização de dados baseada no Matplotlib. Ela fornece uma interface de alto nível para desenhar gráficos estatísticos mais atraentes e melhora a exibição informativas.

Scipy

É uma biblioteca de código aberto especialmente feita para matemáticos, cientistas e engenheiros. A sua biblioteca central também é Numpy que fornece uma manipulação conveniente e rápida de um array N-dimensional.

Pylab

Permite gerar gráficos de duas dimensões de excelente qualidade, permitindo edição interativa, animações, inúmeros tipos de gráficos diferentes, anotações em sintaxe Latex e salvamento das imagens geradas em diversos formatos.

Sklearn

Biblioteca de aprendizado de máquina de código aberto.

Keras

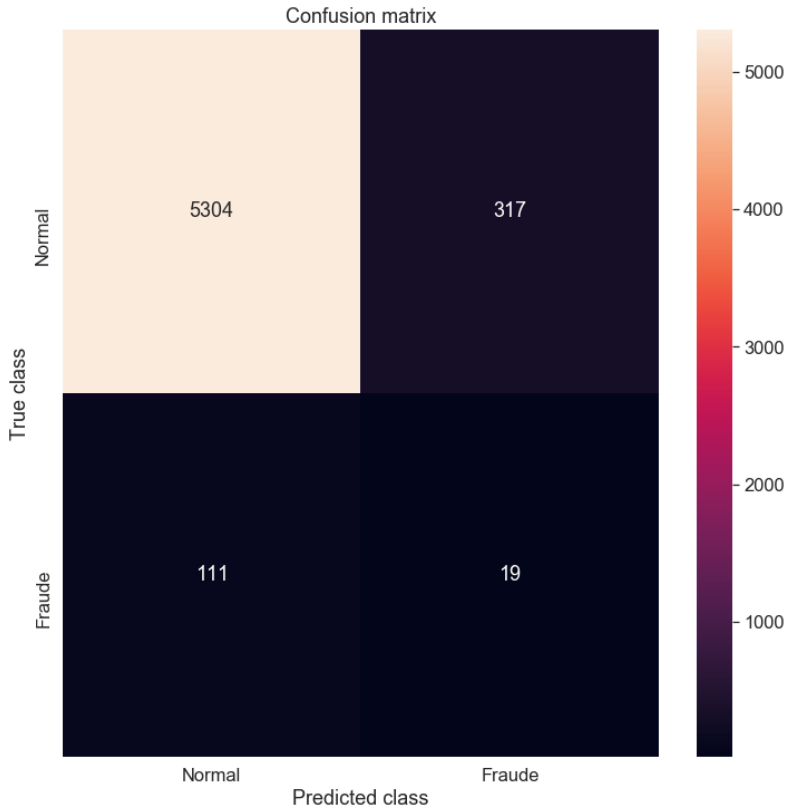
Biblioteca de rede neural de código aberto escrita em Python. É capaz de funcionar com o TensorFlow, projetado para permitir a experimentação rápida com redes neurais profundas, ela se propõe a ser fácil de usar, modular e extensível.

4.3 Métricas para medição de desempenho

Nesta seção são apresentadas as métricas usadas para medição do desempenho dos resultados das soluções propostas.

- Medir o Desempenho usando matriz de confusão;

Figura 11 - Matriz de Confusão



Fonte: Elaborado pelo autor, 2019.

- Medir e comparar desempenho usando: Recall, Precisão, Especificidade, Precisão e Curva AUC-ROC (figura 12);
- Comparar os modelos entre eles mesmos;
- Compartilhar os resultados com dados obtidos com material de referência e bibliografia do estado da arte em pesquisas relacionadas;
- E por fim, as considerações finais com base nos resultados e comparações.

4.3.1 Matriz de Confusão

Uma das métricas de análise utilizadas denomina-se matriz de confusão, ou matriz de classificação, ilustrada na Figura 11, que permite avaliar o desempenho do algoritmo empregado. Desta forma, permite avaliar os resultados de previsão e possibilidade de melhorar o modelo. A matriz permite analisar e comparar técnicas de forma a averiguar a eficiência das mesmas.

Tabela 3 - Matriz de confusão - legenda

TP	Verdadeiro Positivo	Prever **positivo** e ser **verdadeiro**. No nosso caso, um comportamento normal previsto corretamente nos dados, realmente não é uma anomalia, sendo **verdadeiro**
FP	Verdadeiro Negativo	Detectamos uma anomalia e isso é de fato **verdadeiro**
FN	Falso Positivo (erro tipo 1)	Prever normalidade nos dados porem é **falso**, há anormalidade
TN	Falso Negativo (erro tipo 2)	Prever **negativo** e é **falso**. Detectamos uma anomalia, mas realmente não é **falso**

Fonte: Elaborado pelo autor, 2020.

O número de transações normais classificadas como fraudes no modelo *Autoencoder* não necessariamente é um problema. Aumentar ou diminuir o valor do limite, dependendo do problema, pode ser válido. Além de vários parâmetros *epocs*, devem ser avaliados hiperparâmetros e combinações estabelecidos para treino e para testes, para que se possa definir com precisão a eficiência dos modelos. Aqui em especifico, foram usados 60% treino e 40% para testes. Foram descritos valores preditos como **Positivo** e **Negativo** e valores reais como **Verdadeiro** e **Falso**, que são apresentados na Tabela 3.

A Matriz de confusão é extremamente útil para medir Recall, Precisão, Especificidade e a Curva AUC-ROC, que são detalhados a seguir, assim como seus valores são calculados dispostos na Tabela 4.

Recall ou Sensibilidade:

De todas as classes positivas, quantas foram previstas corretamente.

Precisão:

Qual a proporção de identificações positivas que foi realmente correta.

Acurácia:

Todos os dados preditos corretante sobre todas a predições.

É a razão entre o somatório das previsões corretas (verdadeiros positivos com verdadeiros negativos) sobre o somatório das previsões.

Especificidade: A proporção de casos negativos que foram identificados corretamente, de forma semelhante à sensibilidade.

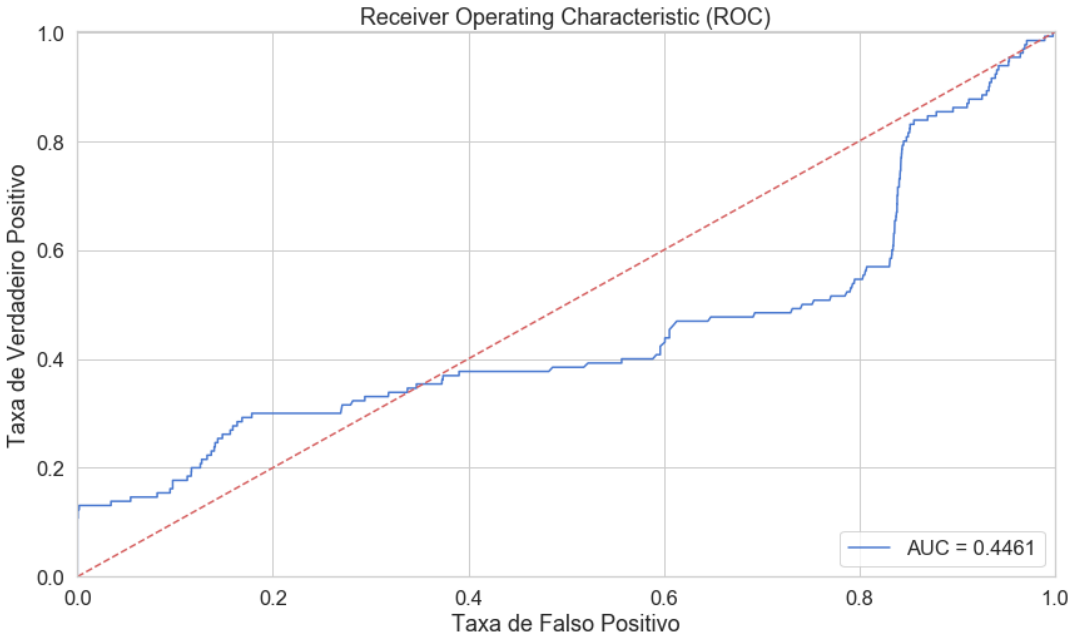
F-score:

É calculado por: 2*(Recall*Precisão)/(Recall+Precisão)

O F-score nos mostra o balanço entre a precisão e o recall do modelo.

É difícil comparar dois modelos com baixa precisão e alta recordação ou vice-versa. Então, para torná-los comparáveis, usamos o **F-Score**. O F-score ajuda a medir a Recall

Figura 12 - Curva De Roc *Autoencoder*



Fonte: Elaborado pelo autor, 2019.

e Precision ao mesmo tempo.

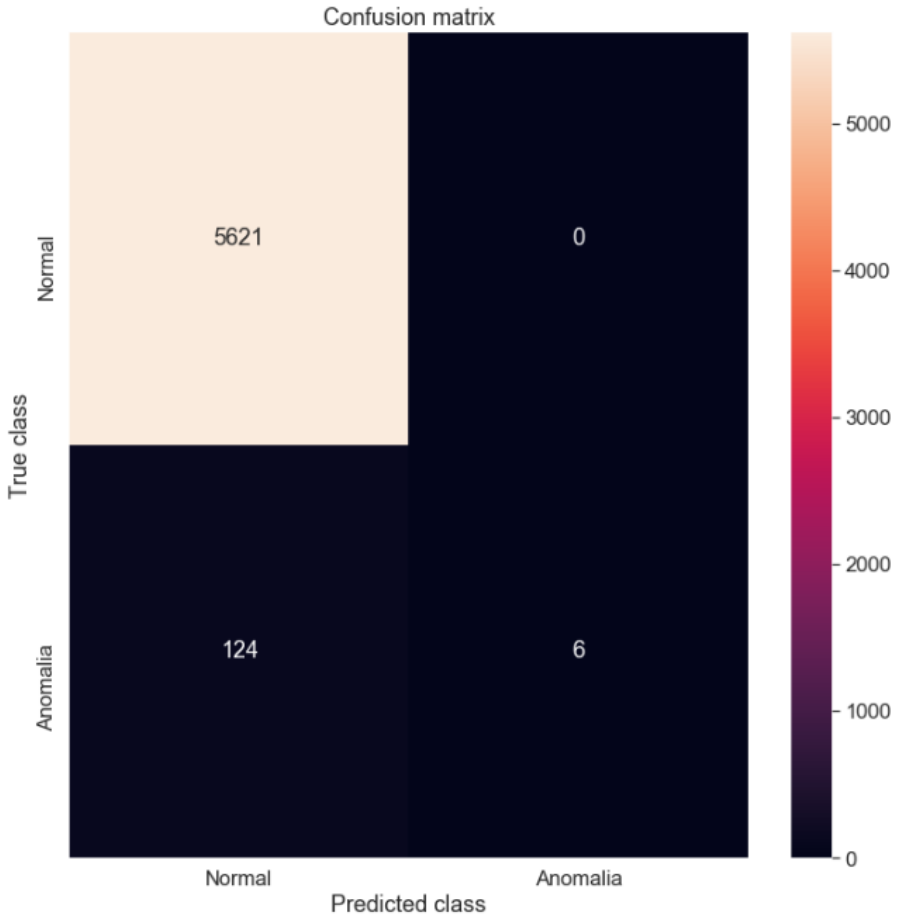
Root Mean Squared Error **(RMSE)**:

Essa é uma excelente métrica para modelos de regressão, além de ser muito fácil de interpretar. A Raiz Quadrada do Erro Quadrático Médio (ou simplesmente **RMSE** em inglês) nada mais é que a diferença entre o valor que foi previsto pelo seu modelo e o valor real que foi observado.

A curva **ROC** permite visualizar a taxa de verdadeiros positivos versus a taxa de falsos positivos, em diferentes valores do conjunto de dados. O **ROC** pode variar muito de acordo com a natureza e quantidades dos dados treinados, conforme Figura 12. Os pontos laranja acima da linha de corte laranja indicam anormalidades detectadas corretamente.

A Figura 13 apresenta uma matriz de confusão do modelo *Autoencoder* LSTM. Comparando a Figura 13 com a Figura 14, observa-se que, quando o verdadeiro negativo caiu um pouco de 19 para 6 casos, o número de transações normais classificadas como fraudes no modelo *Autoencoder-LSTM* caiu de 317 para 0. O falso negativo não sofreu mudança significativa indo de 111 para 124 casos e podemos notar que mais casos de verdadeiros positivos foram detectados: 317 casos a mais. Em geral, considerando outras análises além de alguns outros fatores, como quantidade de *epocs* e outros ajustes no modelo, tivemos uma melhora de cerca de 30% em relação ao modelo de *Autoencoder*. Neste caso, em especifico, foram usados 60% do *dataset* para treino e 40% para testes. Foi exigido muito mais poder computacional para processar este modelo, em decorrência

Figura 13 - Matriz de Confusão 2



Fonte: Elaborado pelo autor, 2019.

de mais complexidade, maior rede neural e mais camadas LSTM recorrentes.

4.4 Criação de ambiente de desenvolvimento para o treinamento computacional e teste das soluções

Os ambientes foram previamente preparados e estão detalhados na Tabela 5.

O valor da perda implica em quão bem ou mal, um determinado modelo se comporta, após cada iteração da otimização. Idealmente, espera-se a redução da perda após cada uma, ou várias iterações. Podemos observar na figura do *Model Loss*, figura 14, que o modelo esta coeso em cada interação.

No modelo *autoencoder* LSTM, a *Precisão e o Recall*, descritos na figura 15, foi utilizado um limite de 0,3 para fornecer uma compensação razoável entre precisão e *recall*, já que queremos um *recall* um pouco mais alto.

Tabela 4 - Classe Predita de Matriz de Confusão

	Positivo	**Negativo**	Sensibilidade: VP/(VP+FN)
Positivo	Verdadeiro Positivo(VP)	Falso Negativo(FN)	Precisão: VP/(VP+FP)
Negativo	Falso Positivo(FP)	Verdadeiro Negativo(VN)	Especificidade: VN/(VN+FP)

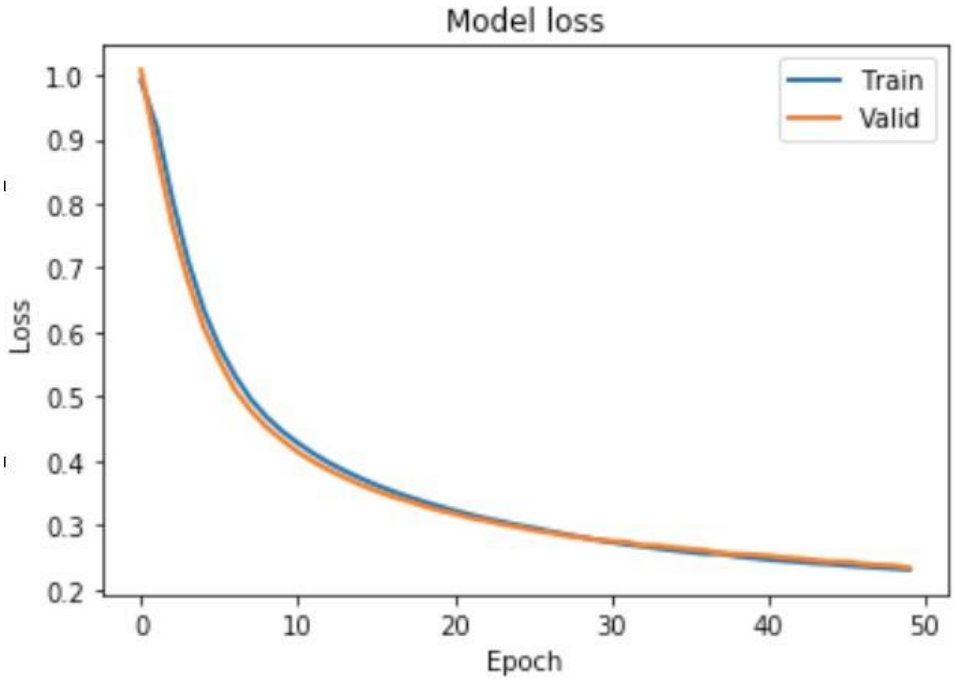
Fonte: Elaborado pelo autor, 2020.

Tabela 5 - Ambiente de desenvolvimento

Ambiente	Processador	S.O.	Tipo Proc.
Macbook Air	Core i5	IOS	CPU
Notebook HP	Core i3	Linux	CPU
Desktop	Core i2 Quad	Windows	CPU e GPU
Google Colab	Super Comp. Nuvem	Linux	CPU e GPU
Titan	Super Comp. Nuvem	Linux	CPU e GPU

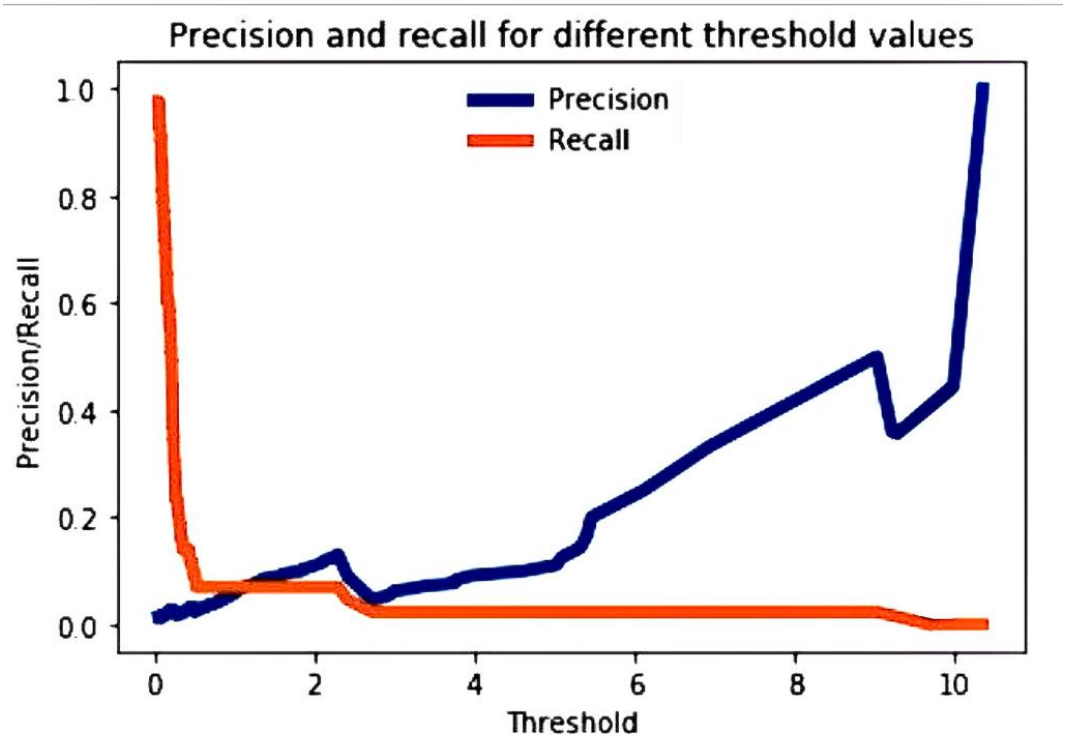
Fonte: Elaborado pelo autor, 2019.

Figura 14 - *Model Loss Autoencoder*-LSTM



Fonte: Elaborado pelo autor, 2019.

Figura 15 - Precisão e *Recall*



Fonte: Elaborado pelo autor, 2019.

5 RESULTADOS E DISCUSSÃO

Foram utilizados dois modelos finalistas para este trabalho:

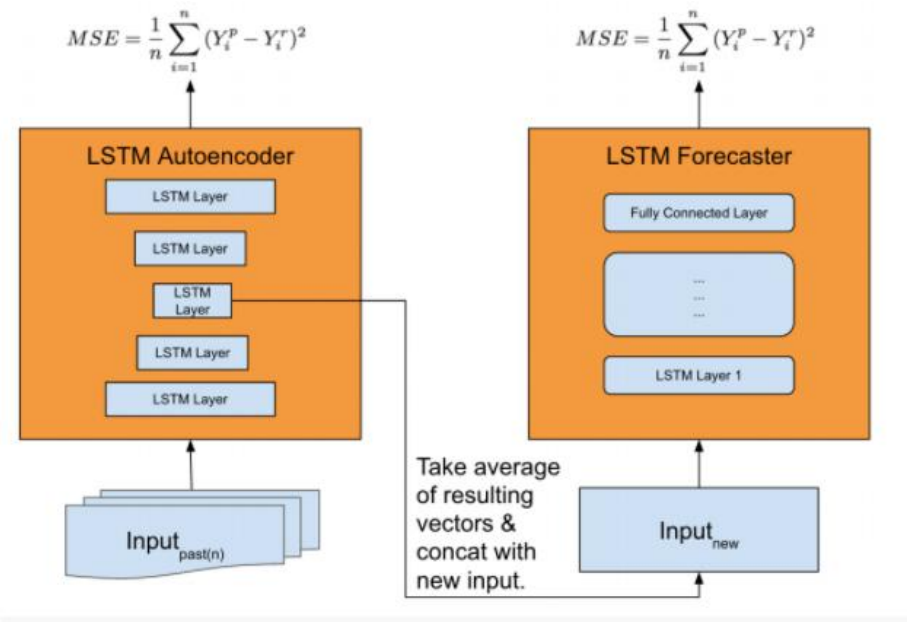
- Modelo utilizando *Autoencoder*;
- Modelo utilizando *Autoencoder*-LSTM;

Ambos os métodos são capazes de detectar com sucesso anomalias em diferentes *datasets* (CHANDOLA; BANERJEE; KUMAR, 2009). Neste experimento, falsos positivos e negativos foram encontrados em proporções diferentes e ajustes nos hiper parâmetros puderam alterar estes resultados. Neste caso, os modelos utilizando LSTM podem ser bons candidatos para comprovar ou não, sua capacidade de ser genérico. Com base nos modelos desenvolvidos neste trabalho, as comparações e as análises, esta suposição foi fortalecida. Além disso, podemos citar grandes vantagens do LSTM, como por exemplo, ser capaz de estabelecer um vínculo com os dados retrógrados e utilizá-los para as detecções de anormalidades, permitindo a manipulação com facilidade desta janela de tempo de dados.

Vários métodos foram sugeridos para problemas de detecção de anomalias (CHANDOLA; BANERJEE; KUMAR, 2009). No entanto, as RNNs são destacadas como mais promissoras para dados em séries temporais, devido à sua capacidade para capturar dependências de tempo (MA; PERKINS, 2003) (ZHANG et al., 2007). Neste caso, Ma e Perkins (2003) melhoraram o desempenho e a precisão desses algoritmos para dados em séries temporais. Os métodos convencionais (SCHÖLKOPF et al., 2001) (TAX; DUIN, 2004) para detectar anomalias não são capazes de processar sequências de dados de comprimento variável, uma vez que a estrutura baseada em LSTM, pode trabalhar com esses tipos de dados. O método proposto é preparado para detectar efetivamente anomalias em dados de séries temporais sem pré-processamento.

As arquiteturas RNN LSTM, cuja principal diferença em relação *multi-layer* é que em vez de conexões completas de *feed-forward*, a rede de repetição pode ter conexões de realimentação para as camadas anteriores (ou para a mesma camada). Esse *feedback* permite que os RNNs mantenham a memória das antigas entradas e os problemas do modelo no momento. Este modelo permite considerar uma janela do passado, que pode ter influenciado alguma anomalia existente nos dados, bem como, após uma possível verificação da sua existência. Neste caso, podemos ter uma máquina de aprendizagem mais forte para a próxima detecção. O LSTM com o uso da célula de memória, que pode manter seu valor para breve ou longo tempo como uma função de suas entradas, permite à célula lembrar o que é pertinente, de acordo com os dados e variações da modelagem e não apenas seu último valor computado. As variações do modelo RNN LSTM usados nesta proposta permite que o modelo possa fazer previsões de potenciais

Figura 16 - Arquitetura *Autoencoder* LSTM



Fonte: Laptev et al. (2017)

anomalias. Essas anomalias podem ser encontradas na detecção de fraudes ou intrusões e aplicadas a diferentes domínios como: ambientes de rede de computadores, segurança, saúde etc.

5.1 Implementação do *Autoencoder* LSTM

5.1.1 Código fonte da implementação do Autoencoder LSTM parte - 1

Nesta seção, um unico código referente à implementação do *Autoencoder* LSTM foi dividido em cinco partes selecionáveis para melhor explicação e apresentado . A Figura 16 ilustra a sua arquitetura.

Shift the data: Estes são dados de séries temporais em que temos que prever o evento (y = 1) com antecedência. Nestes dados, as linhas consecutivas têm um intervalo de 1 minuto. Os rótulos são alterados na coluna y por 2 linhas para fazer uma previsão de 2 minutos à frente.

```
1
2 ...
3
4 def curve_shift(df, shift_by):
```

Este primeiro bloco linhas 1 a 4 indica a chamada da função que mudará os rótulos

binários em um dataframa.

5.1.2 Código fonte da implementação do Autoencoder LSTM parte - 2

No segundo bloco de código selecionavel, linhas 1 a 32 podemos observar como é feito o deslocamento dos rótulos e que o deslocamento da curva será em relação aos 1s. Por exemplo, se o deslocamento for -2, o seguinte processo acontecerá: se a linha n for rotulada como 1, então

- Criar linha (n + shift_by) :(n + shift_by−1) = 1.
- Removendo a linha n.

ou seja, os rótulos serão deslocados para 2 linhas acima.

Entradas:

df - Um dataframe de pandas com uma coluna rotulada binária. Esta coluna rotulada deve ser nomeada como 'y'.

shift_by - Um número inteiro que indica o número de linhas a serem deslocadas.

df - Um quadro de dados com os rótulos binários deslocados por turno.

```
1
2  sign = lambda x: (1, -1)[x < 0]
3
4  def curve_shift(df, shift_by):
5
6  vector = df['is_anomaly'].copy()
7
8  for s in range(abs(shift_by)):
9      tmp = vector.shift(sign(shift_by))
10     tmp = tmp.fillna(0)
11     vector += tmp
12
13 labelcol = 'is_anomaly'
14
15 # Adicionar vetor ao df
16
17 df.insert(loc=0, column=labelcol+'tmp', value=vector)
18
19 # Remova as linhas com labelcol == 1.
20
21 df = df.drop(df[df[labelcol] == 1].index)
```

```
22
23 # Drop labelcol remomeia tmp col como labelcol
24
25 df = df.drop(labelcol, axis=1)
26 df = df.rename(columns={labelcol+'tmp': labelcol})
27
28 # Fazer labelcol binario
29
30 df.loc[df[labelcol] > 0, labelcol] = 1
31
32 return df
```

5.1.3 Código fonte da implementação do Autoencoder LSTM parte - 3

No terceiro bloco de código identado e selecionável linhas 1 a 18, Foram criadas as matrizes tridimensionais da forma: (amostras x timesteps x recursos). As amostras significam o número de pontos de dados. *Timesteps* é o número de etapas em que "olhamos para trás" a qualquer momento t, para fazer uma previsão. Isso também é conhecido como período de lookback. Os recursos são o número de recursos que os dados possuem, em outras palavras, o número de preditores em dados multivariados.

```
1  input_X = df.loc[:, df.columns != 'is_anomaly'].values
2  input_y = df['is_anomaly'].values
3
4  n_features = input_X.shape[1]
5
6
7  def temporalize(X, y, lookback):
8      output_X = []
9      output_y = []
10     for i in range(len(X)-lookback-1):
11         t = []
12         for j in range(1,lookback+1):
13             t.append(X[[(i+j+1)], :])
14
15         output_X.append(t)
16         output_y.append(y[i+lookback+1])
17
18     return output_X, output_y
```

5.1.4 Código fonte da implementação do Autoencoder LSTM parte - 4 e 5

No quarto e quinto blocos de código, linhas 1 a 22 e 1 a 11 respectivamente. A arquitetura do Autoencoder do modelo é construída através de um autoencoder simples. Arquiteturas mais complexas e outras configurações podem ser exploradas em trabalhos futuros.

```
timesteps =  X_train_y0_scaled.shape[1] # equal to the lookback
n_features =  X_train_y0_scaled.shape[2] # 59

epochs = 100
batch = 64
lr = 0.0003

lstm_autoencoder = Sequential(

# Encoder

lstm_autoencoder.add(LSTM(32, activation='relu', input_shape=(
    timesteps, n_features), return_sequences=True))
lstm_autoencoder.add(LSTM(16, activation='relu', return_sequences
    =False))
lstm_autoencoder.add(RepeatVector(timesteps))

# Decoder

lstm_autoencoder.add(LSTM(16, activation='relu', return_sequences
    =True))
lstm_autoencoder.add(LSTM(32, activation='relu', return_sequences
    =True))
lstm_autoencoder.add(TimeDistributed(Dense(n_features)))

lstm_autoencoder.summary()
```

```

# Otimizadores, loss, mse e outros parametros:

adam = optimizers.Adam(lr)
lstm_autoencoder.compile(loss='mse', optimizer=adam)

cp = ModelCheckpoint(filepath="lstm_autoencoder_classifier_yahoo.
    h5", save_best_only=True, verbose=0)
```

```
8
9  tb = TensorBoard(log_dir='./logs', histogram_freq=0, write_graph=
       True, write_images=True)
10
11 lstm_autoencoder_history = lstm_autoencoder.fit(X_train_y0_scaled
       , X_train_y0_scaled, epochs=epochs, batch_size=batch,
       validation_data=(X_valid_y0_scaled, X_valid_y0_scaled), verbose
       =1).history
```

5.2 Documentando as avaliações e comparando os modelos utilizados

Os dados preliminares coletados dos modelos utilizados foram:
F1-score = 0,114.
A precisão = 0,071.
Falso positivo = 0,026.
FFT = 0,099.

Os melhores valores atuais para o escore F1 são: 0,082 por duas unidades de tempo e 0,114 para previsão de quatro unidades de tempo à frente.

Isso pode ser melhorado treinando mais o modelo de previsão, aumentando a quantidade de dados, aumentando o poder computacional e melhorando o recurso de engenharia.

Observa-se que essas instâncias incluem previsões de 2 ou 4 minutos sendo a taxa de falsos positivos de cerca de 6%. Isso não é o ideal e acredita-se que pode ser melhorado para aumentar a taxa de recuperação, com menor taxa de falsos positivos.

Autoencoder: A AUC é 0,68 treinada com 100 *epocs*.
Autoencoder LSTM: A AUC é 0,70 treinada com 100 *epocs*.

Neste trabalho é demostrado o uso de um erro de reconstrução do *Autoencoder* para a classificação de eventos raros. Seguimos este conceito para que o *autoencoder* reconstrua um ruido, se o erro de reconstrução for alto e classificá-lo-em como uma anomalia. Neste caso, foi preciso determinar um limite para isso, sendo usado a precisão e *recall* (Figura 15) para ajudar a encontrar esse limite.

Foram testados limites entre 0,3 e 0,8, o que forneceu uma compensação razoável entre precisão e *recall*, pois um *recall* mais alto era necessário.

A Figura 17 apresenta um exemplo de uso do limiar = 0,8 para classificação. Os pontos laranja e azul acima da linha de limite representam o Positivo Verdadeiro e o Positivo Falso, respectivamente.

Já no *Autoencoder*-LSTM, teve uma melhoria de aproximadamente, de 10% na

AUC em comparação com o *Autoencoder* de camada densa em alguns dos testes. Nas matrizes de confusão das figuras 11 e 13, podemos observar algumas melhorias no segundo modelo. Esta melhoria pode ser significativa em alguns domínios. No entanto, a melhoria que alcançamos em comparação com o *Autoencoder* de camada densa, não é grande. O principal motivo é o modelo LSTM ter mais parâmetros para estimar, tornando importante usar a regularização com LSTMs. A regularização e outras melhorias no modelo podem ser implementadas em trabalhos futuros.

Sobre os dados rotulados binários de eventos raros extremos, considerando os padrões temporais, os *Autoencoders* LSTM foram usados para criar um classificador de eventos raros para um processo multivariado de séries temporais. Um modelo simples de *Autoencoder* LSTM foi treinado e usado para classificação e foram encontradas melhorias na precisão com relação *Autoencoder* denso. Acredita-se que, se usado um *Autoencoder* com *Dropout*, podemos obter ainda mais eficiência, logo também é sugerido em trabalhos futuros.

Em (HUANG et al., 2017), pode ser visto comparativos entre alguns nodelos. Uma tabela adaptada para os parametros usados neste trabalho e com os modelos usando Yahoo Lab como *dataset* é explicitada a seguir: Usado do dataset 19.222 linhas e 240 anomalias.

Os modelos da Tabela 6 são:

1 - Detecção de anomalias baseada em comportamento (BAD)

2 - Máquinas de vetor de suporte de uma classe (OCSVM)

3 - Isolation Forest (IForest)

4 - Previsão adaptativa de anomalias não supervisionadas em tempo real (Arena)

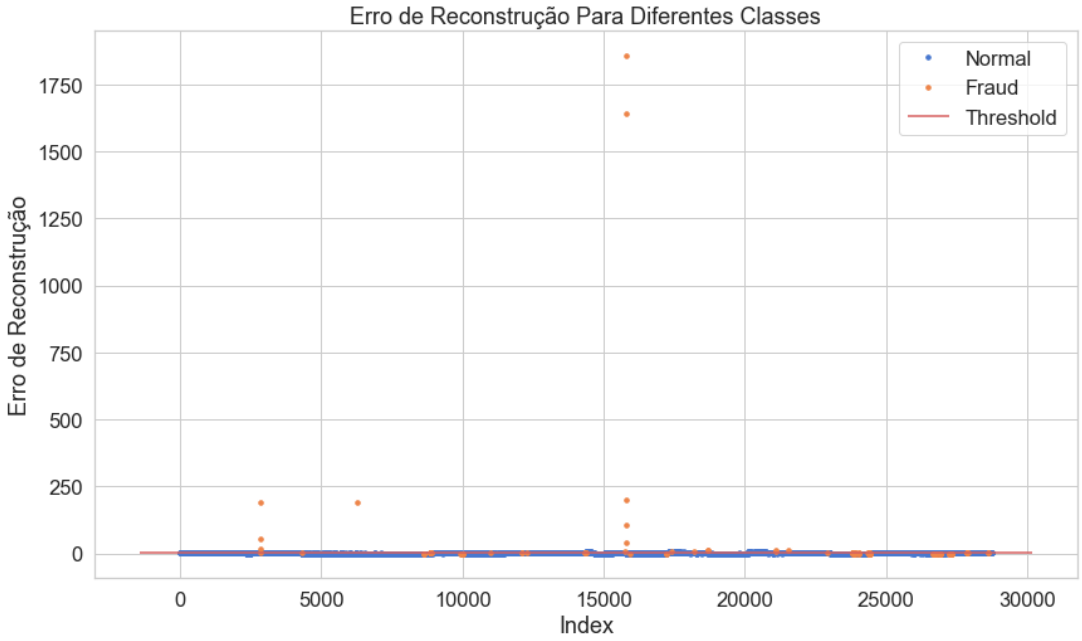
Pode-se observar a acurácia de alguns métodos na tabela 6 e comparar com nosso modelo proposto. Podemos destacar o modelo BAD que teve a pior performance por não ser capaz de detectar intersecções dos dados do *dataset*, acredita se que este resultado está relacionado ao dataset utilizado que era multivariável e continhas séries temporais, de alguma forma podemos compará-lo com o modelo autoencoder padrão, por também não permitir respostas a dados correlacionados temporalmente, contudo o autoencoder ainda pode permitir interseção entre os dados, mas ainda assim como a ideia geral éra atingir a maior generalidade possível ao se tratar em tipo de conjuntos de dados, sua aplicação assim com o autoencoder é limitada neste aspecto. Observando o modelo OCSVM que atinge 84% de precisão, e que fica atrás do modelo Arena. O motivo da alta precisão é que o SVM de uma classe precisa usar dados normais para treinar, o que ajuda o modelo a obter dados de monitoramento mais normais. Portanto, SVM de uma classe tem uma taxa de precisão maior, mas um menor taxa de recall. SVM não atualiza e ajusta o modelo em si. Se os dados de monitoramento mudaram, a precisão do o modelo diminuirá. A taxa de recall do modelo IFlorest é a melhor entre os quatro modelos da tabela, mas seu valor F e a taxa de falso positivo são inferiores ao modelo Arena. Isso ocorre porque o Iflorest

Tabela 6 - Tabela de acurácia de outros modelos

Método	Precision	Recall	F Value	FPR
BAD	82.31%	74.32%	0.781	4.3%
OCSVM	84.19%	79.27%	0.817	3.7%
IForest	81.43%	85.41%	0.834	3.3%
Arena	88.30%	82.79%	0.855	2.7%

Fonte: Adaptado de: Huang et al. (2017).

Figura 17 - Erro De Reconstrução *Autoencoder*



Fonte: Elaborado pelo autor, 2019.

no treinamento usa amostragem aleatória para classificar os dados de monitoramento em uma série de nós folha, então detecta os pontos de anomalia com base em sua densidade, portanto no processo de amostragem aleatória, algum monitoramento normal pode ser classificado aleatoriamente em nós de folha incorretos, o que leva à alta taxa de recall, mas não à taxa de acúmulo. O modelo Arena classifica automaticamente os pontos de anomalias com base nas informações de densidade. Segundo (HUANG et al., 2017), o Modelo Arena pode melhorar a precisão usando atualização em tempo real.

Precisão da previsão: Usamos para a avaliação dos nossos modelos, o Autoencoder e o Autoencoder-LSTM a precisão, *recall*, F Valor e taxa de falsos positivos (FPR) para avaliar a anomalia precisão de previsão.

6 CONSIDERAÇÕES FINAIS

Esta dissertação teve como objetivo desenvolver um modelo de RNN para detectar anomalias em *datasets* de forma genérica, usando aprendizagem de máquina. Para a realização da mesma foram necessários estudos prévios sobre intrusões, ameaças, *outliers* e anomalias em geral, e como podem ser detectados, além de estudos de várias técnicas e abordagens para este fim, e não somente considerando o aprendizado de máquina, para que uma visão mais abrangente e eficiente pudesse ser obtida. A primeira fase prática desta dissertação realizou o estudo do *dataset* fornecido pela Yahoo (RESEARCH, 2016), onde houve uma fase de exploração dos dados, seguida pela criação de atributos a serem usados posteriormente, em diferentes modelos de *Machine Learning*. As técnicas de Machine Learning usadas e implementadas ao longo de todo este trabalho foram: *Autoencoder*, e *Autoencoder*-LSTM.

O modelo de redes neuronais *Autoencoder* produziu bons resultados e apresentou uma percentagem de *outliers* muito boa no geral. Porém, no caso de *datasets* que contenham séries temporais, como o adotado neste trabalho, a capacidade do modelo foi ineficiente para propósitos genéricos.

O outro modelo testado foi o LSTM *autoencoder*, já que este modelo apresentou uma boa capacidade para previsão de cenários. Porém, foi observado que para melhores resultados, seria necessário um modelo híbrido para detecção de anomalias, que pudesse atender a mais tipos de *datasets*. O LSTM *autoencoder* foi o modelo usado para poder comparar a sua efetividade com o *Autoencoder* e com resultados de performance do estado da arte da área. Ou seja, a sua capacidade de guardar os dados em memória torna este modelo mais favorável e genérico. Os resultados obtidos parecem sugerir que há ganhos efetivos com o modelo LSTM *Autoencoder*, em especial, podemos dizer que o modelo LSTM *autoencoder* tem uma grande potencialidade. No entanto, seria preciso mais poder computacional, mais testes em outros *datasets*, para avaliar com melhor precisão o potencial deste modelo em relação ao propósito do trabalho.

Por fim, destacamos algumas das dificuldades encontradas na realização deste trabalho. Um dos problemas foi verificar se os dados teriam informação suficiente para detectar anomalias. Uma vez que o *dataset* continha poucas sinalizações positivas. Por esta razão, foi necessário criar técnicas de subamostragem para melhor equilibrar o *dataset* e desta forma, poder usar a mesma técnica com outros *datasets* desequilibrados, quando necessário. Uma outra dificuldade foi a existência de um número reduzido de dados classificados. Esta dificuldade não permitiu retirar conclusões mais fundamentadas acerca dos modelos desenvolvidos para a detecção de anomalias.

Considerando os dados e as sinalizações do *dataset* da Yahoo (RESEARCH, 2016) do treinamento do modelo LSTM e do modelo *autoencoder*, mostrados nas figuras 2 e 8, foi

utilizado para a detecção de anomalias, um limiar (*thresholding*) entre 03 e 08. A análise comparativa entre os dados foram feitas verificando se a anomalia real encontra-se dentro do *threshold* aplicado. A partir da análise gerada, em comparação com as anomalias utilizadas para o treinamento da rede, é possível concluir que a técnica implementada consegue realizar uma boa previsão, pois aplicando-se o cálculo do Erro Quadrático Médio Normalizado (*Normalized Mean Squared Error*, NMSE) para a comparação dos dados, o valor obtido é muito próximo de 0 para algumas das dimensões analisadas. Durante o processo de detecção de anomalias, o modelo obteve uma taxa de precisão acima de 93%, em alguns dos casos, a taxa de falso positivo menor do que 1% e uma acurácia de mais de 96%. As redes recorrentes LSTM com *autoencoder* obtiveram o melhor desempenho.

O estado da arte para detecção de anomalias deste conjunto de dados obteve resultados superiores e outros inferiores aos obtidos neste trabalho. Os resultados superiores espera-se estarem relacionados ao custo de maior esforço computacional e consequentemente, maior complexidade.

Os modelos propostos neste trabalho possuem arquitetura simplificada e visam atender minimamente ao compromisso de melhor desempenho e melhor classificação possível, atendendo de forma genérica a mais tipos de dados. Alguns dos resultados obtidos pelos modelos são próximos, ou equivalentes aos apresentados pela literatura.

Os resultados finais indicaram que o modelo é aplicável para a classificação de séries temporais em *datasets* mais simples e também, nos multivariáveis mais complexos, e que o seu desempenho é razoável, se comparado aos modelos mais caros computacionalmente.

Trabalhos futuros podem investigar novas abordagens, como arquiteturas híbridas diferentes das adotadas, vistas em [Bashivan et al. 2015]. Também é sugerido a experimentação de técnicas para melhorar as métricas dos modelos apresentados, como a de *data augmentation* por exemplo, já que esse tipo de abordagem tende a se beneficiar com o aumento do conjunto de dados [Goodfellow et al. 2016]. Além disso, pode ser explorada a extração de características do dado em si, como novas features, por exemplo, observando a relação entre custo computacional e desempenho do modelo.

REFERÊNCIAS

ABADI, M. et al. Tensorflow: a system for large-scale machine learning. In: *12th usenix symposium on operating systems design and implementation (osdi 16)*. [S.l.: s.n.], 2016.

ALTHUBITI, S. A.; JONES, E. M.; ROY, K. Lstm for anomaly-based network intrusion detection. in: 2018 28th international telecommunication networks and applications conference (itnac). p. 1–3, 2018. ISSN 2474-154x.

ANDO, Y.; GOMI, H.; TANAKA, H. Detecting fraudulent behavior using recurrent neural networks. 2016.

ANDRYCHOWICZ, M. et al. Learning to learn by gradient descent by gradient descent. p. 3981–3989, 2016.

BONTEMPS, L. et al. Collective anomaly detection based on long short-term memory recurrent neural networks. In: SPRINGER. *International Conference on Future Data and Security Engineering*. [S.l.], 2016. p. 14–152.

BORRAJO, M. L. et al. Hybrid neural intelligent system to predict business failure in small-to-medium-size enterprises. *International Journal of Neural Systems*, World Scientific, v. 21, n. 04, p. 277–296, 2011.

BROCKLEBANK, J.; DICKEY, D. SAS for forecasting time series. *Neural computation*, Springer, n. 2, 2002.

BROWNLEE, J. A gentle introduction to lstm autoencoder. 2018. Disponível em: ⟨https://machinelearningmastery.com/lstm-autoencoders/⟩. Acesso em: 29 ago. 2019.

CAPTANIO, S. Desenvolvimento de algoritmo adaptável utilizando redes neurais artificiais para previsão de demanda. 65 p. dissertação de mestrado. Universidade de Caxias do Sul, 2019.

CASTELÃO, R. Utilização de redes neurais para previsões no mercado de ações. 29 p. relatório técnico. Unicamp, 2020. Disponível em: ⟨https://www.ic.unicamp.br/~reltech/PFG/2018/PFG-18-01.pdf⟩. Acesso em: 2 fev. 2020.

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

CHAUHAN, S.; VIG, L. Anomaly detection in ECG time signals via deep long short-term memory networks. In: IEEE. *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*. [S.l.], 2015. p. 1–7.

DAMACENO, H. F. Detecção de intrusão em redes de computadores. trabalho de conclusão de curso. p. 54. IMESA, 2008.

DONAHUE, J. et al. Long-term recurrent convolutional networks for visual recognition and description. 2015.

EHLERS, R. S. Análise de séries temporais. 2009. Disponível em: ⟨www.webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70⟩. Acesso em: 18 dez. 2019.

ERGEN, T.; MIRZA, A. H.; KOZAT, S. S. Unsupervised and semi-supervised anomaly detection with LSTM neural networks. *arXiv preprint arXiv:1710.09207*, 2017.

ESKIN, E. et al. A geometric framework for unsupervised anomaly detection. In: *Applications of data mining in computer security*. [S.l.]: Springer, 2002. p. 77–101.

FERREIRA, C. C. d. C. P. et al. Sistemas fuzzy complementam a detecção de socialbots por aprendizado de máquina. Brazilian Journal of Development, v. 5, n. 12, p. 32413–32426, 2019.

FIORIO, L. V. Emulação de um pré-amplificador valvulado utilizando redes neurais artificiais. 119 p. trabalho de conclusão de curso. Universidade do Estado de Santa Catarina, 2019.

GIL, A. C. Gestão de Pessoas: enfoque nos papéis profissionais. Atlas, São Paulo, 2007.

GREFF, K. et al. Lstm: a search space obyssey. IEEE transactions on neural networks and learning systems, 2016.

HAN, J.; PEI, J.; KAMBER, M. Data mining: concepts and techniques. Elsevier, 2011.

HOSMER, D. W. Applied logistic regression. Wiley, 2000.

HUANG, S. et al. Arena: Adaptive real-time update anomaly prediction in cloud systems. in: 2017 13th international conference on network and service management (cnsm). IEEE, p. 1–9, 2017.

JR., D. W. H.; LEMESHOW, S.; STURDIVANT, R. Applied logistic regression. John Wiley & Sons, 2013.

KING, S. et al. The use of novelty detection techniques for monitoring high-integrity plant. In: IEEE. *Control Applications, 2002. Proceedings of the 2002 International Conference on*. [S.l.], 2002. v. 1, p. 221–226.

LAPTEV, N. et al. Time-series extreme event forecasting with neural networks at uber. in: International conference on machine learning. p. 1–5, 2017.

LARZALERE, B. Lstm autoencoder for anomaly detection. 2019. Disponível em: ⟨https://towardsdatascience.com/lstm-autoencoder-for-anomaly-detection-e1f4f2ee7ccf⟩. Acesso em: 05 mai. 2019.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. nature, nature publishing group, a division of macmillan publishers limited. v. 521, p. 436, 2015. Disponível em: ⟨https://doi.org/10.1038/nature14539⟩. Acesso em: 2 fev. 2020.

LODOLI, B. C. et al. Ferramenta de geração de redes neurais para classificação de criptogramas. 58 p. projeto de conclusão de curso. IME, 2016.

LOPEZ-MARTIN, M. et al. Network traffic classifier with convolutional and recurrent neural networks for internet of things. IEEE Access, v. 7, p. 18042–18050, 2017. ISSN 2169-3536.

MA, J.; PERKINS, S. Time-series novelty detection using one-class support vector machines. In: IEEE. *Neural Networks, 2003. Proceedings of the International Joint Conference on*. [S.l.], 2003. v. 3, p. 1741–1745.

MARCONI, M. A.; LAKATOS, E. M. Metodologia científica. Atlas, São Paulo, 2011.

MIRZA, A. H.; COSAN, S. Computer network intrusion detection using sequential lstm neural networks autoencoders. in: 2018 26th signal processing and communications applications conference (siu). p. 1–4, 2018.

MOUSTAFA, N. et al. A holistic review of network anomaly detection systems: A comprehensive survey. Journal of Network and Computer Applications, v. 128, p. 33–55, 2019. ISSN 1084-8045. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1084804518303886⟩. Acesso em: 2 fev. 2020.

NAGPAL, A.; GABRANI, G. Python for data analytics, scientific and technical applications. in: 2019 amity international conference on artificial intelligence (aicai). p. 140–145, 2019.

NELSON, D. M. Q. Uso de redes neurais recorrentes para previsão de séries temporais financeiras. 73 p. dissertação de mestrado. Universidade Federal de Minas Gerais, 2017.

NOGOSEKE, L. F. Aplicação de differentiable neural computers na resolução de problemas de processamento de linguagem natural. dissertação de mestrado. Universidade Federal do Paraná, 2019.

OLIVEIRA, P. H. M. A. Detecção de fraudes em cartões: um classificador baseado em regra de associação e regressão logística. Universidade de São Paulo, São Paulo, 2015.

PACHECO, C. et al. Building reference datasets to support socialbots detection. Metrology for Industry 4.0 and IoT, 2018.

QIN, G.; CHEN, Y.; LIN, Y. Anomaly detection using lstm in ip networks. in: 2018 sixth international conference on advanced cloud and big data (cbd). p. 334–337, 2018.

RAMAKRISHNAN, N.; SONI, T. Network traffic prediction using recurrent neural networks. in: 2018 17th ieee international conference on machine learning and applications (icmla). p. 187–193, 2018.

RANJAN, C. et al. Dataset: rare event classification in multivariate time series. arXiv: 1809.10717, 2018.

RESEARCH, Y. *S5 - A Labeled Anomaly Detection Dataset*. 2016. Disponível em: ⟨https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70⟩.

RUDIO, F. V. Introdução ao projeto de pesquisa cientifica. Vozes, 2016.

SANTOS, G. Uma aplicação de redes neurais recorrentes do tipo lstm à previsão dos preços de curto prazo do mercado de energia elétrica brasileiro. 76 p. dissertação de mestrado. Fundação Getulio Vargas, 2019.

SANZ, I. J. et al. Um sistema de detecção de ameaças distribuídas de rede baseado em aprendizagem por grafos. 2018. Disponível em: ⟨https://www.gta.ufrj.br/ftp/gta/TechReports/SAR18.pdf⟩. Acesso em: 10 jun. 2020.

SCHÖLKOPF, B. et al. Estimating the support of a high-dimensional distribution. *Neural computation*, MIT Press, v. 13, n. 7, p. 1443–1471, 2001.

SILVA, G. C. Detecção de intrusão em redes de computadores: Algoritmo imunoinspirado baseado na teoria do perigo e células dendríticas. dissertação de mestrado. Universidade Federal de Minas Gerais, p. 119, 2009.

SOUZA, H. H. T. M. Metodologia qualitativa de pesquisa. educação e pesquisa. v. 30, n. 2, 2004.

SOUZA, P. S. S. et al. Descoberta de anomalias em dispositivos IoT usando Isolation Forest. 2016. Disponível em: ⟨https://www.researchgate.net/publication/310645987_Descoberta_de_anomalias_em_dispositivos_IoT_usando_Isolation_Forest/citation/download⟩. Acesso em: 20 jun. 2020.

SRIVASTAVA, N.; MANSIMOV, E.; SALAKHUDINOV, R. Unsupervised learning of video representations using lstms. in: International conference on machine learning. p. 843–852, 2015. Disponível em: ⟨https://towardsdatascience.com/lstm-autoencoder-for-anomaly-detection-e1f4f2ee7ccf⟩.

SÁ, A. O. de; CARMO, L. F. R. da C.; MACHADO, R. C. Covert attacks in cyber-physical control systems. IEEE Transactions on Industrial Informatics, v. 13, n. 4, p. 1641–1651, 2017.

TAN, P. N.; STEINBACH, M.; KUMAR, V. Introdução ao datamining: mineração de dados. Ciência Moderna, 2009.

TAX, D. M.; DUIN, R. P. Support vector data description. *Machine learning*, Springer, v. 54, n. 1, p. 45–66, 2004.

TEIXEIRA, M. Contributo da auditoria interna para uma gestão eficaz. Universidade Aberta, 2006.

VASSALI, L. C. Aplicação de redes neurais lstm para a previsão de curto prazo de vazão do rio paraíba do sul. p. 36. trabalho de conclusão de curso. Universidade Federal de Juiz de Fora, 2018.

VINAYAKUMAR et al. Applying deep learning approaches for network traffic prediction. in: 2017 international conference on advances in computing, communications and informatics (icacci). p. 2353–2358, 2017.

WOZNIAK, M.; GRANA, M.; CORCHADO, E. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, v. 16, n. 0, p. 3–17, 2014. ISSN 1566-2535. Special Issue on Information Fusion in Hybrid Intelligent Fusion Systems. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S156625351300047X⟩.

ZARPELÃO, B. B. Detecção de anomalias em redes de computadores. tese de doutorado. Universidade Federal de Campinas, p. 119, 2009.

ZHANG, L.; LIN, J.; KARIM, R. Adaptive kernel density-based anomaly detection for nonlinear systems. *Knowledge-Based Systems*, Elsevier, v. 139, p. 50–63, 2018.

ZHANG, R. et al. One class support vector machine for anomaly detection in the communication network performance data. In: CITESEER. *Proceedings of the 5th conference on Applied electromagnetics, wireless and optical communications*. [S.l.], 2007. p. 31–37.