Universidade Estadual de Londrina

FELIPE DALLMANN TOMAZELI

APRENDIZADO DE MÁQUINA ADVERSÁRIO CONTRA DETECTORES DE ANOMALIAS EM SÉRIES TEMPORAIS

LONDRINA
2023

FELIPE DALLMANN TOMAZELI

APRENDIZADO DE MÁQUINA ADVERSÁRIO CONTRA DETECTORES DE ANOMALIAS EM SÉRIES TEMPORAIS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação do Departamento de Computação da Universidade Estadual de Londrina como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof(a). Dr(a). Bruno Bogaz Zarpelão

LONDRINA

2023

AGRADECIMENTOS

Gostaria de expressar minha sincera gratidão ao professor Bruno Bogaz Zarpelão, pela sua orientação, conhecimento e apoio durante todo o desenvolvimento deste trabalho. Seu comprometimento e dedicação foram fundamentais para o sucesso deste projeto. Além disso, agradeço ao corpo docente da Universidade Estadual de Londrina e ao Departamento de Computação, bem como à administração da universidade, por todo o suporte oferecido ao longo do curso. Também não poderia deixar de agradecer à minha família e amigos por todo o apoio e incentivo durante esta etapa final de conclusão do meu curso.

"A teoria sem a prática de nada vale, a prática sem a teoria é cega.
(Lenin)

TOMAZELI, F. D.. **Aprendizado de Máquina Adversário contra Detectores de Anomalias em Séries Temporais**. 2023. 58f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

A crescente disponibilidade de dispositivos conectados à Internet das Coisas (*Internet of Things* - IoT) e também de sistemas ciberfísicos implica no aumento dos dados organizados em séries temporais, já que muitos desses dispositivos transmitem fluxos de dados contínuos ao longo do tempo. É fundamental analisar esses dados para extrair informações úteis, mas a ocorrência de eventos inesperados pode levar a anomalias nessas séries, tornando a análise mais complexa. A detecção de anomalias é importante porque pode representar falhas em dispositivos, interrupções de serviços ou ainda indicar atividades suspeitas ou maliciosas. Em aplicações do mundo real, a detecção de anomalias pode ser usada para prever problemas antes que ocorram, reduzir o tempo de inatividade, melhorar a qualidade do produto e até mesmo salvar vidas em situações críticas. Uma anomalia em uma série temporal pode significar um evento fora do comum ou também um problema sistêmico subjacente que precisa ser corrigido para evitar falhas futuras. Com isso em mente, estudos têm demonstrado que técnicas de Aprendizado de Máquina (AM) podem ser usadas com eficiência e precisão para detecção de anomalias em séries temporais. No entanto, é importante considerar que esses algoritmos são vulneráveis a ataques de Aprendizado de Máquina Adversário, o que pode fazer com que as anomalias passem despercebidas ou não sejam detectadas, comprometendo a segurança do sistema. Além disso, situações normais podem ser erroneamente detectadas como anomalias, gerando falsos positivos e prejudicando a análise. Diante dessa realidade, este trabalho propõe a observação de diferentes categorias de ataques e seus impactos contra detectores de anomalias em séries temporais, utilizando técnicas de Aprendizado de Máquina profundo. Adicionalmente, foram executados experimentos onde o ataque *Fast Gradient Sign Method* (FGSM) foi utilizado contra um detector de anomalias baseado em LSTM. Os resultados mostraram que esses modelos de AM apresentam algumas vulnerabilidades referentes aos ataques adversários nesse cenário. Isso é fundamental para garantir a confiabilidade e a segurança dos sistemas que utilizam detecção de anomalias em séries temporais com Aprendizado de Máquina.

Palavras-chave: Aprendizado de máquina. Aprendizado de máquina adversário. Séries temporais. Detecção de anomalias.

TOMAZELI, F. D.. **Adversarial Machine Learning against Anomaly Detectors in Time Series** . 2023. 58p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2023.

ABSTRACT

The increasing availability of devices connected to the Internet of Things (IoT) and cyber-physical systems leads to an increase in data organized in time series, as many of these devices transmit continuous data streams over time. In this scenario, it is fundamental to analyze this data to extract useful information, but the occurrence of unexpected events can lead to anomalies in these series, making the analysis more complex. The identification of anomalies is significant as these irregularities can indicate equipment malfunctions, service disruptions, or even imply dubious or malicious actions. In real-world applications, anomaly detection can be used to predict problems before they occur, reduce downtime, improve product quality, and potentially even preserve lives during critical situations. An anomaly in a time series can indicate an event outside the norm or even an underlying systemic problem that needs to be corrected to avoid future failures. Keeping this in perspective, research has demonstrated that the utilization of Machine Learning methodologies can lead to efficient and precise detection of anomalies in time series. Although Machine Learning techniques are efficient for anomaly detection in time series, it is important to consider that these algorithms are vulnerable to Adversarial Machine Learning attacks. An adversarial attack can be directed to make anomalies go unnoticed or not be detected, compromising the security of the system. Additionally, normal situations can be erroneously detected as anomalies, generating false positives and hindering the analysis. Given this reality, this work proposes the observation of different categories of attacks and their impacts against anomaly detectors in time series, using deep Machine Learning techniques. Additionally, in this work, experiments were carried out where the Fast Gradient Sign Method attack was used against an LSTM-based anomaly detector. The results showed that these Machine Learning models present some vulnerabilities regarding adversarial attacks in this scenario. This is fundamental to ensure the reliability and security of systems that use Machine Learning for anomaly detection in time series.

Keywords: Machine Learning. Adversarial Machine Learning. Time Series. Anomaly detection.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
AR	Autoregression
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
FP	Falsos Positivos
FN	Falsos Negativos
FGSM	Fast Gradient Sign Method
IA	Inteligência Artificial
JSMA	Jacobian Based Saliency Map Attack
LSTM	Long Short-Term Memory
MA	Moving Average
PGD	Projected Gradient Descent
TFP	Taxa de Falsos Positivos
TVP	Taxa de Verdadeiros Positivos
VN	Verdadeiros Negativos
VP	Verdadeiros Positivos

SUMÁRIO

1 INTRODUÇÃO

Desde o final do século XX, o crescimento acelerado da Internet trouxe diversos benefícios e comodidades para a humanidade. A Internet é essencial em vários campos, como sistemas ciber físicos e dispositivos conectados à Internet das Coisas, como eletrodomésticos, equipamentos de segurança e aparelhos relacionados à saúde. Esses sistemas são caracterizados pela capacidade de interagir e se comunicar com o mundo físico através da computação [4]. Normalmente, esses sistemas colhem dados sequencialmente ao longo do tempo, o que pode caracterizar essa coleção de dados como séries temporais.

Assim como a Internet, os sistemas de Inteligência Artificial (IA) estão se tornando cada vez mais comuns no cotidiano, auxiliando na automação de processos e na tomada de decisões. IA refere-se à capacidade dos computadores de realizar tarefas que normalmente são executadas por humanos, como o reconhecimento de objetos em uma imagem. Para alcançar esse nível de inteligência, é possível utilizar o método de análise de dados chamado Aprendizado de Máquina (AM), que, normalmente, passa por duas fases: treinamento e inferência. Durante a fase de treinamento, o sistema adquire conhecimentos e aprende com base em dados históricos [5]. Em seguida, baseado no que aprendeu durante a fase de treinamento, o algoritmo de AM pode realizar inferências sobre dados que não haviam sido apresentados anteriormente a ele.

As séries temporais são amplamente utilizadas em diversas áreas do conhecimento [6], incluindo finanças, medicina, ciência de dados, engenharia, meteorologia e muitas outras. A principal característica das séries temporais é a sua dependência temporal, ou seja, cada observação está relacionada à observação anterior e subsequente. Esse comportamento temporal torna as séries temporais um desafio interessante para o Aprendizado de Máquina [7], que pode ser usado para prever valores futuros e identificar padrões e anomalias. Em finanças, por exemplo, as séries temporais são usadas para prever preços de ações e commodities, ajudando a tomar decisões de investimento. Na medicina, as séries temporais são usadas para monitorar a evolução de doenças, avaliar a eficácia de tratamentos e prever a recorrência de sintomas. Em ciência de dados, as séries temporais são usadas para analisar o comportamento de usuários em plataformas digitais, identificar tendências de consumo e prever a demanda futura.

Embora as séries temporais tenham ampla funcionalidade, é importante destacar que elas estão sujeitas à ocorrência de anomalias, que podem representar eventos de interesse [8], como uma falha em um equipamento, uma mudança inesperada de comportamento ou um ataque. Essas anomalias podem ser causadas por erros na leitura de dados ou por imprevistos, e sua identificação e tratamento são essenciais para garantir a confiabilidade dos sistemas. A detecção de anomalias em séries temporais é um desafio

comum que pode ser abordado por meio de algoritmos de Aprendizado de Máquina. Esses algoritmos são treinados com dados históricos para aprender padrões normais e, assim, detectar quaisquer valores ou eventos que fogem desses padrões. É importante destacar que esses algoritmos podem se adaptar às mudanças no comportamento da série temporal, melhorando a precisão da detecção de anomalias com o tempo.

Neste cenário, evidencia-se a importância em garantir segurança e confiabilidade nos modelos de Aprendizado de Máquina [9]. No entanto, é importante destacar que existem vulnerabilidades que podem afetar a eficácia desses modelos, comprometendo sua funcionalidade e alterando os resultados esperados [10]. Essas vulnerabilidades podem ser exploradas por atacantes, os quais utilizam de diferentes técnicas de Aprendizado de Máquina Adversário, incluindo ataques de evasão. Os ataques de evasão consistem em inserir pequenas perturbações nos dados (denominados de exemplos adversários), a fim de enganar o algoritmo de detecção de anomalias. Essas perturbações podem ser imperceptíveis para os humanos, mas são suficientes para causar erros na detecção de anomalias pelo algoritmo de Aprendizado de Máquina. Como resultado, anomalias importantes podem ser ignoradas e eventos normais podem ser erroneamente detectados como anomalias. Diante disso, é importante que a comunidade de Aprendizado de Máquina trabalhe para desenvolver técnicas eficazes de defesa e segurança contra ataques adversários. Essas técnicas podem incluir a adição de camadas de segurança, a criação de algoritmos de detecção de evasão e a realização de testes de segurança abrangentes para identificar vulnerabilidades antes que elas possam ser exploradas por atacantes.

Diante do exposto, o presente trabalho tem como objetivo analisar os impactos de ataques adversários contra detectores de anomalias em séries temporais. Nesse sentido, espera-se avaliar a gravidade desses ataques, considerando a magnitude da perturbação aplicada pelo atacante e as diferentes técnicas utilizadas. Para alcançar o objetivo proposto, um modelo de detecção de anomalias baseado em LSTM será submetido aos ataques. Esse modelo é um algoritmo de Aprendizado de Máquina, que tem por objetivo realizar predições de uma série temporal. Com base nessas predições, um limiar será definido a partir do erro quadrático médio entre as predições e os valores reais, permitindo a detecção de anomalias na série temporal.

O ataque terá dois objetivos distintos: o primeiro será gerar falsos positivos e o segundo, falsos negativos. É importante salientar que o ataque será realizado em um ambiente de caixa branca, ou seja, o atacante terá acesso total aos parâmetros do modelo de detecção de anomalias, bem como aos dados e seus respectivos rótulos. Ademais, o atacante criará um segundo modelo de classificação para ser utilizado como linha de base e posteriormente comparado com o modelo de detecção de anomalias. Os ataques serão criado a partir das técnicas *Fast Gradient Sign Methods* e um método de ataque baseado no desvio padrão dos dados, denominado de *Naive*.

Este trabalho está organizado em cinco capítulos. O capítulo dois contém uma revisão bibliográfica sobre os principais conceitos abordados no trabalho, como séries temporais, Aprendizado de Máquina, detecção de anomalias em séries temporais e Aprendizado de Máquina adversário. No capítulo três, é descrita a metodologia utilizada para obter os resultados do trabalho. O capítulo quatro apresenta os dados obtidos de acordo com a metodologia descrita no capítulo anterior. Por fim, no capítulo cinco, são apresentadas as conclusões do estudo.

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA E ESTADO DA ARTE

2.1 Séries Temporais

Séries temporais são conjuntos de dados coletados sequencialmente ao longo do tempo, com intervalos equidistantes entre as observações. Esses dados são usados para identificar padrões e tendências, bem como para rastrear mudanças ao longo do tempo [11]. Na análise de séries temporais, as observações anteriores são consideradas para prever valores futuros. A modelagem matemática é fundamental na análise de séries temporais, já que os dados são coletados em diferentes pontos no tempo, criando um desafio estatístico para encontrar um modelo adequado. A modelagem busca capturar as relações entre as observações adjacentes no tempo e fornecer previsões precisas para os valores futuros. A Figura 1 apresenta um exemplo de série temporal referente à mudança da temperatura média anual nos Estados Unidos no período de 1880 até 2020. Dessa forma, a análise da série permite que os pesquisadores identifiquem padrões e tendências nas mudanças de temperatura ao longo do tempo, como períodos de aquecimento ou resfriamento.

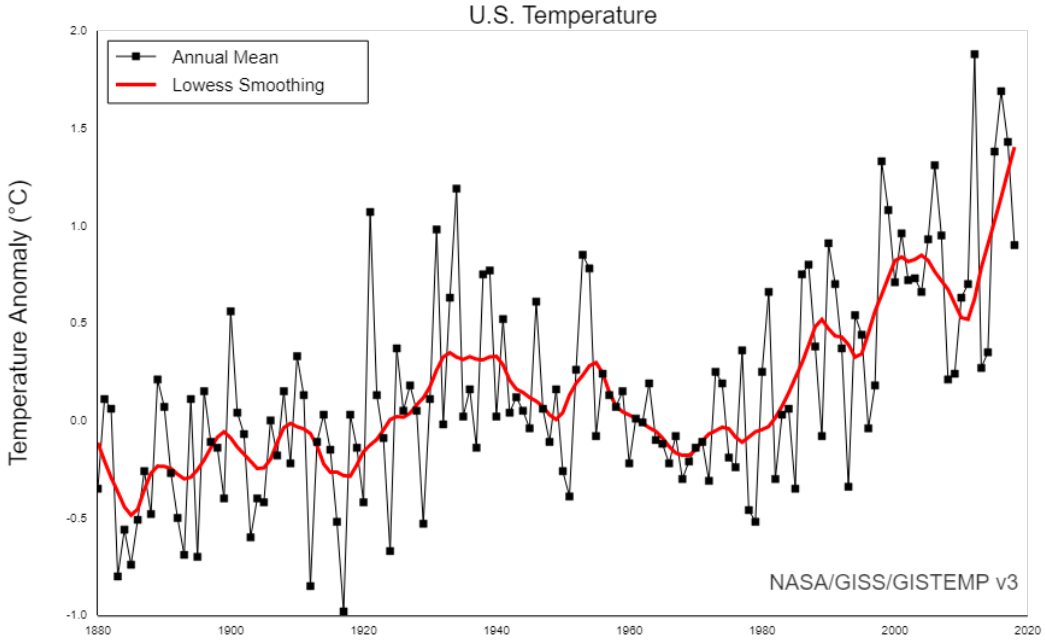

Figura 1 – Gráfico representando a mudança de temperatura média anual nos Estados Unidos. Fonte: [1, 2]

A análise de séries temporais é relevante cientificamente devido à sua aplicabilidade em diferentes áreas [12]. Por exemplo, na economia, são usadas as cotações diárias do mercado de ações; na epidemiologia, para acompanhar o crescimento de casos de uma

doença durante um determinado período; na medicina, para observar os batimentos cardíacos e auxiliar no diagnóstico; e em estudos estatísticos, para avaliar o crescimento populacional.

2.1.1 Análise de Séries Temporais

Análise de séries temporais dispõe de métodos matemáticos para extrair propriedades estatísticas dos dados [13]. Ela permite construir modelos de previsão de valores futuros a partir dos valores anteriores e utilizar modelos descritivos de regressão para quantificar a relação entre as variáveis da série temporal.

Um dos principais objetivos da análise de séries temporais é identificar tendências e sazonalidades. A extração de informações sobre o comportamento das séries temporais pode ser feita de modo tradicional, por meio da visualização dos dados em gráficos, ou ainda através de métodos estatísticos [14].

Identificar essas características é fundamental, pois, a partir delas, é possível prever valores futuros utilizando diferentes modelos de previsão de séries temporais, como o ARIMA (*Auto regressive integrated moving average*) [15] ou técnicas de Aprendizado de Máquina, como redes neurais artificiais. Além disso, a análise de séries temporais pode ser usada para detectar anomalias, ou seja, identificar padrões incomuns nos dados que podem representar possíveis problemas.

2.1.2 Propriedades Estatísticas das Séries Temporais

Um dos principais objetivos da análise de séries temporais é desenvolver modelos matemáticos que descrevam os dados. Para isso, algumas propriedades estatísticas são consideradas para definir se uma série temporal é estacionária ou não estacionária [16, 17]. As séries temporais podem ser descritas como uma sequência de observações $x_1, x_2, ..., x_n$, onde x_i é o valor da série no tempo t_i. Algumas características importantes a serem consideradas na construção de modelos são:

- **Tendência**: padrão de comportamento dos dados ao longo do tempo, ou seja, a direção geral dos valores da série. Ela pode ser de crescimento, quando os valores tendem a aumentar ao longo do tempo, de decrescimento, quando os valores tendem a diminuir, ou estacionária, quando os valores não apresentam variação significativa;
- **Sazonalidade**: característica das séries temporais quando há uma repetição cíclica de um comportamento que acontece periodicamente;
- **Autocorrelação**: característica das séries temporais que define a relação entre uma observação e as observações anteriores. Autocorrelação de primeira ordem é a relação

da observação atual com a anterior, de segunda ordem é a relação da atual com as duas anteriores, e assim por diante.

- **Estacionariedade**: medida de dispersão dos dados em relação ao valor médio;

Para que uma série temporal seja estacionária, é necessário que as suas propriedades estatísticas não variem ao longo do tempo. Especificamente, uma série temporal estacionária tem a média, variância e autocorrelação constantes. Caso uma série temporal não seja estacionária, é possível realizar o procedimento de diferenciação. Esse consiste em obter a diferença entre um valor da série temporal e o valor anterior. É utilizado para remover o efeito acumulado das séries temporais, transformando uma série não estacionária em estacionária.

2.1.3 Métodos de Previsão de Séries Temporais

Na análise de séries temporais, um dos principais objetivos é prever valores futuros da série com base em seus padrões e tendências históricos. Para isso, é comum utilizar modelos estatísticos como *Autoregression* (AR), *Moving Average* (MA), *Autoregressive Moving Average* (ARMA) e *Autoregressive Integrated Moving Average* (ARIMA). Esses modelos permitem modelar os dados de séries temporais e prever valores futuros com base nas relações estatísticas entre as observações passadas e presentes. Cada modelo tem suas próprias características e requisitos de ajuste, e a escolha do modelo mais adequado depende dos tipos de dados e das características da série temporal em questão.

- **Autoregression (AR)**: O funcionamento do método de Autoregressão(AR) é baseado em um conceito chamado *lag*. Esse especifica que a saída do modelo depende linearmente dos valores anteriores da série temporal, sendo a ordem p de $AR(p)$ o número de valores imediatamente anteriores [18]. Dessa forma, é conhecido como um modelo de longa memória, uma vez que pode considerar todos os valores anteriores até o ponto inicial;

- **Moving Average (MA)**: O método *Moving Avarage* baseia suas previsões nos erros passados [19] da série temporal, também conhecidos como "*white noise*". Esses erros são calculados como a diferença entre o valor real observado e o valor previsto pelo modelo no tempo anterior. O modelo MA é definido pela ordem q, que representa o número de erros passados que são considerados no modelo. Dessa forma, um modelo MA com ordem q é escrito como $MA(q)$. A escolha do valor de q depende das características da série temporal em questão e pode ser determinada por meio de análise exploratória dos dados.

- **Autoregressive Moving Average (ARMA)**:ARMA é um dos modelos clássicos utilizados para predição de séries temporais. Consiste em combinar os modelos

de $AR(p)$ e $MA(q)$. Dessa forma, os parâmetros p e q de $ARMA(p, q)$ devem ser definidos. Existem alguns métodos de definir a melhor combinação, por exemplo: *Minimum info criteria* ou *Squared canonical correlation*;

- **Autoregressive Integrated Moving Average (ARIMA)**: O modelo ARIMA é uma extensão do modelo ARMA. Também consiste em combinar os modelos AR e MA, porém, conta com o processo de diferenciação. É matematicamente descrito como $ARIMA(p, d, q)$, onde p e q seguem o mesmo padrão do modelo ARMA, e o parâmetro d é a quantidade de vezes que a diferenciação é aplicada;

2.2 Aprendizado de Máquina

Desde os primórdios, a capacidade humana tem evoluído significativamente com a criação de ferramentas e técnicas que facilitam vários aspectos do cotidiano. De maneira geral, o campo de estudo da Inteligência Artificial objetiva resolver problemas de maneira semelhante, de forma adaptativa e evolutiva [20]. Dessa forma, os algoritmos de Aprendizado de Máquina são componentes importantes da Inteligência Artificial. Em geral, esses algoritmos passam por um processo de aprendizado que permite a resolução de problemas, acumulando conhecimentos com base em propriedades estatísticas [21] e, assim, evoluindo ao longo do tempo.

Em contraste com os problemas resolvidos com algoritmos programados explicitamente, os algoritmos de Aprendizado de Máquina dão aos computadores a capacidade de resolver problemas sem que a solução tenha sido explicitamente programada anteriormente. Em vez disso, o objetivo dos algoritmos de Aprendizado de Máquina é aprender a solução com base nos dados fornecidos [22].

Esses algoritmos geralmente passam por duas fases principais: treinamento e inferência. Na fase de treinamento, o algoritmo recebe os dados disponíveis e, com base em propriedades estatísticas, gera a própria função para computar esses dados. Na fase de inferência, o algoritmo recebe novos dados ainda não vistos e analisa a capacidade de generalização obtida durante o treinamento.

Com tantas aplicações, não há uma solução única para todos os problemas de Aprendizado de Máquina. As implementações dos modelos variam de acordo com o problema, os tipos de dados disponíveis e a maneira de aprendizado, entre outros fatores. Dessa forma, é possível dividir os algoritmos de Aprendizado de Máquina em vários tipos de aprendizado [23], como demonstra a Figura 2.

2.2.1 Tipos de Aprendizado de Máquina

O primeiro tipo de Aprendizado de Máquina que será abordado é o supervisionado. Este tipo de aprendizado é um processo baseado em dados rotulados, ou seja, dados asso-

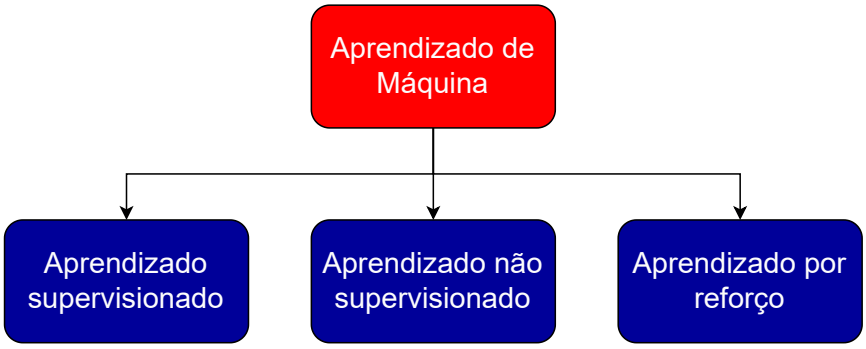

Figura 2 – Tipos de aprendizado.

ciados a uma classe ou valores contínuos conhecidos [24]. Durante a fase de treinamento, o algoritmo aprende a partir dos exemplos fornecidos e cria uma função que se aproxima dos dados rotulados. É uma das abordagens mais populares de Aprendizado de Máquina. Na fase de treinamento desse tipo de aprendizado, o algoritmo se ajusta comparando a saída obtida com a saída desejada. Posteriormente, os dados de inferência são utilizados para avaliar a capacidade do modelo de generalizar o problema proposto. É possível separar os problemas resolvidos por aprendizado supervisionado em classificação e regressão [20]. Regressão consiste em prever respostas contínuas, inferindo a relação entre uma variável dependente com uma ou mais variáveis independentes, tais como alterações de temperatura ao longo do tempo. Por outro lado, os problemas de classificação tem como objetivo classificar entradas em determinadas categorias, por exemplo, o reconhecimento de objetos em uma imagem.

Já o Aprendizado de Máquina não supervisionado é um processo em que o algoritmo é treinado sem um conjunto de dados rotulados. Em vez disso, os algoritmos aprendem a resolver o problema de forma autônoma, reconhecendo padrões e características dos dados [25]. Um dos principais benefícios do aprendizado não supervisionado é que os dados não precisam ser classificados ou rotulados, o que pode economizar tempo e esforço humano [26]. Além disso, essa abordagem é frequentemente utilizada em grandes conjuntos de dados, como em agrupamentos de dados [27] e redução de *features* [28]. Dessa forma, o objetivo principal do aprendizado não supervisionado é agrupar os dados de acordo com os padrões, similaridades e diferenças reconhecidos durante a fase de treinamento.

Por fim, o Aprendizado de Máquina por reforço é um tipo de algoritmo baseado em um sistema de recompensa, no qual um agente aprende a interagir com um ambiente de maneira a maximizar as recompensas [29]. Esse tipo de algoritmo é frequentemente utilizado em cenários onde o agente tem suas ações subordinadas a consequências, como jogar um jogo ou controlar um robô. Esse agente é responsável por melhorar a performance do algoritmo explorando as possibilidades de ações que ele pode realizar no ambiente.

As ações boas são recompensadas, enquanto as ruins são punidas. Um dos principais componentes desse tipo de aprendizado é a função que estima o valor da recompensa do agente ao longo do tempo [30]. É importante que essa função seja bem construída, já que ela é responsável pelo aprendizado do agente.

2.2.2 Aprendizado de Máquina Utilizado na Predição de Séries Temporais

A detecção de anomalias em séries temporais pode ser realizada por meio de técnicas de Aprendizado de Máquina supervisionado, como Redes Neurais Recorrentes [31], LSTM (Long-Short Term Memory) e Redes Neurais Convolucionais. Esses algoritmos, chamados de redes neurais artificiais, são modelos computacionais inspirados na estrutura e funcionamento do sistema nervoso humano [32], compostos em camadas de neurônios interconectados que aprendem representações abstratas dos dados de entrada. O processo de treinamento envolve ajustar os pesos das conexões entre os neurônios [33] para minimizar a diferença entre a predição da rede neural e o valor real dos dados de entrada. Esse processo é otimizado com o gradiente descendente [34], que é usado para atualizar os pesos do modelo de forma iterativa.

A escolha da função de perda depende do tipo de tarefa que o modelo está sendo treinado para realizar, como entropia cruzada binária para problemas de classificação binária e soma dos erros quadráticos para problemas de regressão. O objetivo do modelo é minimizar essa função de perda, ou seja, fazer com que essa diferença seja a menor possível. No entanto, se a configuração da rede não for apropriada, ela pode ser propensa ao *overfitting*, o que diminui sua capacidade de generalização para dados desconhecidos [35].

As Redes Neurais Recorrentes (RNR) são particularmente adequadas para análise de dados sequenciais, incluindo a predição de séries temporais. Essas redes mantêm um estado interno que permite que as informações sejam lembradas e usadas para processar novas entradas à medida que são apresentadas à rede. A rede LSTM é uma das arquiteturas de RNR mais populares para a predição de séries temporais. Ela contém células de memória que funcionam como uma memória de longo prazo para armazenar informações relevantes de entradas anteriores e possui portas de entrada, saída e esquecimento que controlam o fluxo de informações na célula de memória. Essas características permitem que a rede LSTM possa capturar dependências de longo prazo nas séries temporais, o que é particularmente importante em problemas de previsão de séries temporais.

Em resumo, o Aprendizado de Máquina utiliza redes neurais artificiais, que são ajustadas por meio do processo de treinamento e da atualização iterativa dos pesos com o gradiente descendente para prever séries temporais e detectar anomalias. A escolha do tipo de rede neural depende da tarefa a ser realizada, por exemplo, a rede LSTM é particularmente útil para a análise de dados sequenciais e previsão de séries temporais.

2.3 Detecção de Anomalias em Séries Temporais

A detecção de anomalias é uma tarefa crítica para a análise de séries temporais. Ela consiste no processo de identificação e tratamento de valores atípicos que se desviam do comportamento padrão da série temporal. Como as séries temporais têm uma ampla gama de aplicações, a detecção de anomalias é utilizada em várias áreas, como finanças, cibersegurança e saúde [36].

Existem vários métodos para detectar anomalias, incluindo os clássicos, que são baseados em propriedades estatísticas, como o cálculo da média da série temporal e a definição de um limite que define até onde os dados são normais ou anômalos [37]. Além disso, há também métodos mais recentes, como a aplicação de algoritmos de Aprendizado de Máquina [38].

A detecção de anomalias é um processo crucial na análise de séries temporais, pois valores anômalos podem ser indicativos de eventos indesejados ou de interesse. Quando se trata de eventos indesejados, como ruídos ou erros na coleta de dados, é fundamental detectá-los e corrigi-los para garantir que a análise não seja comprometida. Por outro lado, no caso de eventos de interesse, como leituras de batimentos cardíacos anômalos, é importante detectá-los e analisá-los cuidadosamente a fim de entender a situação e tomar medidas apropriadas.

Em resumo, a detecção de anomalias é um processo essencial na análise de séries temporais ao garantir que anomalias sejam detectadas e tratadas de forma adequada. Dessa forma, é possível garantir a precisão e confiabilidade dos resultados da análise, o que é fundamental em muitas aplicações críticas. Portanto, independentemente da natureza da anomalia, a detecção é crucial para garantir a precisão e confiabilidade da análise de séries temporais [37].

2.3.1 Aplicações da Detecção de Anomalias em Séries Temporais

Em finanças, séries temporais são amplamente utilizadas para previsões de preços e análise de mercado. A detecção de anomalias é fundamental para garantir a qualidade dos dados utilizados nessas análises e para identificar, por exemplo, fraudes em transações anormais. Anomalias não detectadas e tratadas podem afetar significativamente os resultados, levando a decisões equivocadas que podem ter impactos financeiros negativos.

Na área da saúde, a detecção de anomalias pode ser usada para identificar comportamentos fora do padrão esperado, que podem indicar um possível problema de saúde do paciente. Por exemplo, a análise de batimentos cardíacos pode ajudar na identificação de arritmias e outras doenças cardíacas. Além disso, a detecção de anomalias também pode ser útil para identificar problemas técnicos no equipamento utilizado para coletar dados, o que ajuda a garantir a precisão das medições e dos diagnósticos.

Na manufatura, a análise de séries temporais é usada para monitorar o processo de produção e identificar defeitos em produtos ou equipamentos. Nesse cenário, a detecção de anomalias é uma técnica útil para encontrar falhas ou comportamentos incomuns que possam indicar problemas de qualidade, aumentando a eficiência e confiabilidade no processo de manufatura.

2.3.2 Tipos de anomalias

Entre os diferentes tipos de anomalias, as anomalias pontuais são instâncias de dados que se destacam significativamente do resto devido à sua natureza incomum ou anormal [39]. Essas anomalias normalmente são mais simples de detectar, pois são evidentes em comparação com o restante dos dados. Por exemplo, em um conjunto de dados de transações bancárias, uma transação com um valor significativamente mais alto do que o restante das transações pode ser considerada uma anomalia pontual.

Além das anomalias pontuais, existem as anomalias contextuais, que são instâncias de dados anormais dentro de um contexto específico [39]. Elas podem não ser evidentes quando os dados são analisados individualmente, desconsiderando o contexto, mas podem ser incomuns quando vistas em um contexto específico. Por isso, esse tipo de anomalia é mais difícil de detectar do que as anomalias pontuais, que são significativamente diferentes do resto dos dados. Por exemplo, considerando, novamente, o monitoramento de uma conta bancária, uma transação pode não parecer uma anomalia se ela não tiver um valor fora do comum. No entanto, a transação pode ser considerada uma anomalia contextual se for feita em horário ou local muito diferentes do comportamento esperado. Nesse caso, a transação é anormal dentro do contexto de horário ou local das outras transações.

Por fim, as anomalias coletivas ocorrem quando há um grupo de instâncias de dados relacionadas que são incomuns ou anormais [39]. Essas anomalias podem não ser óbvias quando os dados são considerados individualmente, mas podem ficar evidentes quando visualizados como um grupo. Uma das áreas de aplicação das anomalias coletivas é a segurança cibernética. As redes de computadores são frequentemente monitoradas para detectar atividades anômalas que possam indicar ameaças cibernéticas. Por exemplo, um conjunto de acessos a um sistema a partir de endereços IP diferentes pode ser considerado normal, mas, se esses acessos forem feitos simultaneamente e com o mesmo objetivo, podem indicar um ataque de negação de serviço distribuído.

2.3.3 Métodos de detecção de anomalias

Alguns métodos de detecção de anomalias baseiam-se na análise de propriedades estatísticas das séries temporais para identificar mudanças no comportamento padrão dos dados [40] e podem ser divididos em métodos paramétricos e não paramétricos [41].

Os métodos paramétricos envolvem a modelagem dos dados como uma distribuição estatística conhecida, como a distribuição normal ou a distribuição de Poisson. A detecção de anomalias é baseada na identificação de pontos que se desviam significativamente da distribuição modelada. Esses métodos são adequados para conjuntos de dados com distribuições bem definidas e conhecidas, mas podem falhar em detectar anomalias em dados com distribuições desconhecidas ou não padronizadas

Por outro lado, os métodos não paramétricos não exigem que os dados sejam modelados como uma distribuição estatística conhecida. Em vez disso, eles usam técnicas como distância euclidiana, densidade local, vizinhos mais próximos ou agrupamento hierárquico para detectar anomalias. Esses métodos são adequados para conjuntos de dados com distribuições desconhecidas ou não padronizadas, mas podem gerar muitos falsos positivos em conjuntos de dados com alta dimensionalidade ou com muitos dados.

Além dos métodos paramétricos e não paramétricos, também existem métodos de detecção de anomalias baseados em Aprendizado de Máquina. Esses métodos utilizam técnicas de AM, como classificação, agrupamento e regressão, para identificar anomalias em dados onde é possível treinar um modelo para identificar anomalias em novos dados de entrada. Uma das vantagens dessa abordagem é que os algoritmos de AM são mais apropriados para lidar com conjuntos de dados complexos ou grandes [42].

Existem várias abordagens para a detecção de anomalias utilizando Aprendizado de Máquina, incluindo aprendizado supervisionado e aprendizado não supervisionado. No aprendizado supervisionado, o modelo é treinado com dados já rotulados como normais ou anômalos. Modelos como *Support Vector Machine*, *Decision Trees* ou *Multi Layer Perceptron* podem ser utilizados para essa tarefa. No aprendizado não supervisionado, o modelo aprende a agrupar os dados de acordo com suas características semelhantes, permitindo reconhecer padrões anômalos.

Além disso, ainda é possível combinar métodos estatísticos com métodos baseados no Aprendizado de Máquina. Utilizando, por exemplo, o algoritmo LSTM para realizar a predição da série temporal, é possível comparar o resíduo dessa predição com os dados reais e, com esses resíduos, detectar mudanças no comportamento da série temporal com base em propriedades estatísticas.

Em resumo, os algoritmos de Aprendizado de Máquina são uma ferramenta valiosa para a detecção de anomalias em séries temporais e em outros conjuntos de dados complexos ou grandes. No entanto, a construção de tais modelos pode ser mais complexa e requerer mais tempo computacional devido à fase de treinamento.

2.4 Aprendizado de Máquina Adversário

Devido à importância e à vasta aplicabilidade dos algoritmos de AM, um dos âmbitos de estudo que tem tomado grandes proporções é o Aprendizado de Máquina Adversário. Esse campo de estudo concentra-se em investigar técnicas para atacar esses algoritmos e formas de contramedida desses ataques, que são criados através da exploração de vulnerabilidades presentes na arquitetura dos algoritmos de Aprendizado de Máquina [43].

Um dos ataques mais comuns contra algoritmos de Aprendizado de Máquina é conhecido como "ataque de exemplo adversário". Esse tipo de ataque consiste em realizar modificações mínimas em um dado de entrada legítimo, resultando em uma saída completamente diferente do que era esperado pelo modelo. Especificamente, existem vários exemplos de ataques adversários em algoritmos de classificação de imagem. Tomando como exemplo um modelo de classificação de imagens capaz de reconhecer animais, é possível manipular uma imagem de um panda de maneira sutil e imperceptível aos humanos, de forma que o modelo erroneamente classifique-a em uma categoria diferente, como demonstra a Imagem 3.

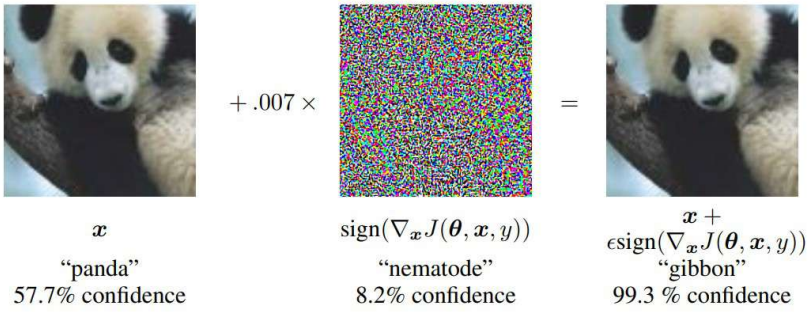

Figura 3 – Demonstração de um exemplo adversário.Fonte: [3]

A possibilidade de ataques adversários em algoritmos de Aprendizado de Máquina é uma questão extremamente relevante, uma vez que muitos desses modelos são utilizados em aplicações do mundo real. A vulnerabilidade de um modelo a esse tipo de ataque pode levar a graves consequências, especialmente em áreas sensíveis, como carros autônomos, onde um erro desse tipo pode colocar em risco vidas humanas. A preocupação com os ataques adversários é agravada pelo fato de que eles podem ser realizados mesmo que o

atacante não tenha acesso direto ao modelo treinado ou ao conjunto de dados de treinamento [44]. Isso significa que os modelos precisam ser projetados para serem robustos a esses tipos de ataques, independentemente da sua origem.

É importante destacar que a pesquisa sobre ataques adversários em algoritmos de Aprendizado de Máquina é uma área em constante evolução, e novas técnicas de ataque e defesa são descobertas com frequência. Por esse motivo, é fundamental que os pesquisadores e desenvolvedores de algoritmos de Aprendizado de Máquina estejam sempre atualizados e preparados para lidar com as ameaças emergentes.

2.4.1 Cenários de Conhecimento do Atacante

No campo do Aprendizado de Máquina adversário, o conceito de "conhecimento" refere-se à compreensão do atacante sobre as informações do alvo e suas características, incluindo acesso aos dados e aos parâmetros do modelo. De maneira geral, existem duas configurações de conhecimento: caixa branca e caixa preta.

Na configuração de caixa branca, o atacante tem conhecimento completo dos parâmetros e configurações do modelo alvo, bem como do conjunto de dados utilizado na sua fase de treinamento [45]. Nesse cenário, as possibilidades de ataque são mais amplas e, em alguns casos, mais efetivas.

Por outro lado, o cenário de caixa preta ocorre quando o atacante tem acesso limitado aos parâmetros, configurações e conjunto de dados do modelo alvo. Dessa forma, a construção de ataques pode ser mais complicada. No entanto, o atacante pode observar o comportamento do modelo para entender algumas de suas características e, posteriormente, transferir ataques a partir de um modelo cópia do alvo. A transferibilidade dos ataques adversários é um problema relevante em ambos os cenários, uma vez que um ataque bem-sucedido em um modelo de Aprendizado de Máquina pode ser transferido para outros modelos com características semelhantes, mesmo que o atacante tenha acesso limitado às informações sobre o modelo.

A existência de várias configurações de conhecimento e a possibilidade de transferência de ataques tornam a tarefa de defender um modelo contra as técnicas de Aprendizado de Máquina adversário bastante desafiadora. Para superar esses desafios, pesquisadores desenvolveram uma variedade de possíveis métodos de defesa, incluindo o treinamento adversário, que consiste em adicionar exemplos adversários na fase de treinamento do algoritmo para aumentar a sua robustez. Além disso, outras técnicas têm sido propostas, tais como a utilização de vários modelos em conjunto [46], o que pode aumentar a capacidade de detecção de exemplos adversários. No entanto, a busca por soluções mais robustas e eficazes para lidar com as ameaças dos ataques adversários continua sendo um desafio importante na área de Aprendizado de Máquina.

2.4.2 Características dos Ataques

Ataques adversários em Aprendizado de Máquina são técnicas que exploram vulnerabilidades nos modelos ou nos dados de entrada, e são divididos em dois tipos principais: ataques de evasão e ataques de envenenamento.

Os ataques de evasão, ou exemplos adversários, envolvem a aplicação de pequenas perturbações em novos dados de entrada do algoritmo na fase de inferência, levando o modelo a produzir saídas erradas. Esses ataques são usados para evadir mecanismos de defesa e podem ser preocupantes em áreas como filtragem de spam, cibersegurança, detecção de fraude e carros autônomos. [3, 47]. Alguns dos ataques mais conhecidos incluem o *Fast Gradient Sign Method* (FGSM), *Jacobian Based Saliency Map Attack* (JSMA) e o *Projected Gradient Descent* (PGD).

- **Fast Gradient Sign Method (FGSM):** O ataque FGSM é amplamente conhecido devido à sua facilidade de construção e eficiência em criar exemplos adversários para algoritmos de Aprendizado de Máquina. Ele é tipicamente utilizado em cenários de caixa branca, no qual, como já destacado, o atacante não só tem conhecimento do modelo alvo e seus parâmetros, mas também possui acesso ao conjunto de dados de treinamento. O ataque é construído a partir do gradiente da função de perda do modelo em relação ao dado de entrada, manipulando-o de tal forma a tentar maximizar a perda [3].

- **Jacobian Based Saliency Map Attack (JSMA)**: O ataque JSMA é comumente utilizado para criar exemplos adversários em modelos de redes neurais artificiais, especialmente em cenários de caixa branca. Diferente de outros ataques que buscam maximizar a perda global, o JSMA é projetado para atacar uma classe específica. A construção desse ataque envolve o cálculo das derivadas parciais das saídas do modelo em relação a uma entrada e a adição de pequenas perturbações nessa entrada de acordo com a dimensão que apresenta a maior magnitude de gradiente.

- **Projected Gradient Descent (PGD):** O ataque PGD é uma variação do ataque FGSM e é baseado no gradiente descendente. A construção dos ataques com essa técnica consiste em aplicar, iterativamente, perturbações no dado de entrada de acordo com o gradiente descendente da função de perda do modelo, enquanto o FGSM aplica essa perturbação apenas uma vez. Dessa forma, é possível aplicar perturbações maiores nos dados, tornando a defesa mais difícil.

Por outro lado, os ataques de envenenamento são um tipo de ataque adversário em que o atacante manipula o conjunto de dados de treinamento para introduzir amostras maliciosas. Essas amostras podem levar o modelo alvo a dar respostas incorretas quando recebe novos dados de entrada. Esses tipos de ataques são especialmente preocupantes

porque são difíceis de serem detectados, já que as amostras maliciosas são inseridas no conjunto de treinamento junto com os dados legítimos.

De maneira geral, os ataques adversários funcionam como problemas de otimização, que tem por objetivo minimizar a diferença entre o dado de entrada original e o dado manipulado, mantendo a perturbação grande o suficiente para torná-lo um exemplo adversário. Além disso, a facilidade em construir ataques adversários, muitas vezes com acesso limitado ao modelo, é uma preocupação crescente na área.

2.4.3 Técnicas de Defesa

Dadas as aplicações dos algoritmos de Aprendizado de Máquina no mundo real e suas vulnerabilidades a ataques adversários, é nítido o esforço de pesquisadores para desenvolver técnicas de defesa contra esses ataques. Algumas das técnicas comumente utilizadas incluem: treinamento adversário, transformação e manipulação dos dados de entrada, combinação de modelos e métodos de otimização de robustez.

Treinamento adversário é uma técnica de defesa comum contra ataques adversários. Consiste em incluir exemplos adversários [46], ou seja, dados de entrada modificados intencionalmente, durante a fase de treinamento. Isso permite que o modelo aprenda a reconhecer dados corretamente, mesmo diante de pequenas perturbações, aumentando sua robustez contra ataques adversários.

A técnica de transformação e manipulação dos dados de entrada consiste em modificar os dados de entrada antes de fornecê-los ao modelo de Aprendizado de Máquina, de forma a dificultar a construção de exemplos adversários. Essas transformações incluem, por exemplo, adição de ruído, pré-processamento e técnicas específicas para processamento de imagem, como redimensionamento, corte e modificações de cores. Em processamento de texto, é possível utilizar técnicas como lematização e sistematização para modificar os dados de entrada.

A combinação de modelos é uma técnica de defesa contra ataques de Aprendizado de Máquina adversário que envolve a combinação de diferentes modelos de Aprendizado de Máquina, cada um com uma arquitetura distinta. Dessa forma, é mais difícil para os ataques atingirem todos os modelos combinados [48].

Por fim, a técnica de otimização de robustez do modelo consiste em treiná-lo para ser resistente a pequenas perturbações nos dados [49], já que a maioria dos ataques tem como objetivo aplicar a menor perturbação possível nos exemplos adversários. O próprio método de treinamento adversário é uma aplicação de otimização de robustez. Embora essas técnicas de otimização sejam eficazes, elas não são suficientes para resolver todas as vulnerabilidades dos algoritmos de Aprendizado de Máquina, pois mesmo modelos treinados com essas técnicas podem ser afetados por ataques adversários. Além disso, é

importante lembrar que essas técnicas podem ser computacionalmente custosas.

2.4.4 Trabalhos Correlatos

O trabalho apresentado em [50] fala sobre sistemas ciber-físicos (CPSs), que são sistemas em que componentes de software estão profundamente entrelaçados com processos físicos. Esses sistemas são muito comuns em infraestruturas públicas críticas e a possível interrupção que poderia resultar de um sistema comprometido motivou a pesquisa em vários mecanismos de detecção de ataque em CPSs, incluindo técnicas baseadas em verificação de invariantes, atestação e identificação de impressão digital. Uma solução popular é construir detectores de anomalias em que um modelo de Aprendizado de Máquina é treinado em uma série temporal de dados físicos do sistema para julgar quando valores futuros estão se desviando do normal. Muitos estudos foram realizados nos últimos anos para explorar a eficácia dessas abordagens de detecção de anomalias de aprendizado profundo em CPSs.

Na literatura, é possível notar que algoritmos de ataque adversário foram aplicados em vários domínios de classificação (incluindo imagens, áudio e malware), mas enfrentam uma série de desafios adicionais no contexto dos CPSs em infraestruturas críticas. Em particular, os estados dos CPSs consistem em dados de sensores e atuadores contínuos e discretos, o que leva a uma interação complexa entre a influência do ruído e a sensibilidade aos ataques. Além disso, detectores baseados em redes neurais raramente são o único mecanismo de defesa em sistemas reais: normalmente, os CPSs também são equipados com verificadores de regras embutidos que monitoram violações de relacionamentos conhecidos entre sensores e atuadores específicos (expressos na forma de invariantes). Como consequência, os ataques adversários existentes são insuficientes por si só, pois os detectores de anomalias e os verificadores de regras devem ser enganados simultaneamente.

Assim sendo, o trabalho [50] apresenta um ataque adversário para testar detectores de anomalias de redes neurais recorrentes em CPSs, que são assumidos estar equipados com verificadores de regras adicionais. A abordagem utiliza um algoritmo baseado em gradiente de caixa branca, mas adaptado para enfrentar os desafios impostos pela infraestrutura crítica. Em particular, a solução cria ruído nos domínios contínuos e discretos de sensores e atuadores. Por fim, o trabalho em questão conseguiu como resultado uma redução de 50% na precisão, sugerindo que detectores de anomalias baseados em redes neurais recorrentes podem ser vulneráveis contra ataques adversários mesmo na presença de outros mecanismos de defesa.

3 MATERIAIS E MÉTODOS

3.1 Visão Geral

O objetivo deste trabalho é desenvolver e avaliar ataques adversários contra um modelo de detecção de anomalias baseado no algoritmo LSTM. Na fase de inferência, o modelo de detecção de anomalias compara o resíduo entre a predição resultante e o valor real e gera alertas quando o resíduo for maior que um limiar predeterminado. Além disso, um modelo de classificação baseado no algoritmo MLP será criado pelo atacante. Dessa forma, serão criados exemplos adversários individualmente para cada um dos modelos, como mostrado na Figura 4. É importante fazer essa distinção porque a natureza de cada modelo é diferente, e, portanto, suas metodologias de criação de exemplos adversários também serão. Posteriormente, os impactos dos exemplos adversários serão comparados.

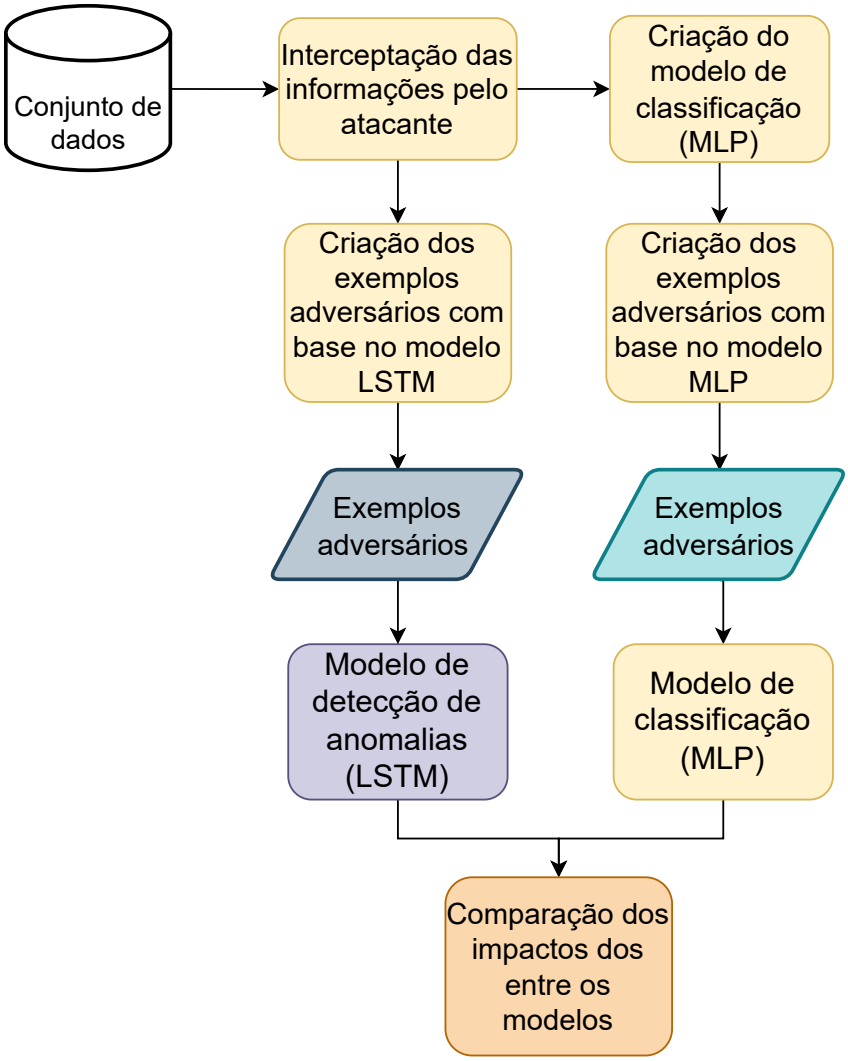

Figura 4 – Representação das etapas dos ataques.

O atacante realizará dois tipos de ataques: um baseado na técnica FGSM e outro no desvio padrão. No ataque FGSM, o atacante usará um gradiente descendente para encontrar a perturbação máxima que pode ser adicionada aos dados de entrada, de modo que ambos os modelos realizem predições erroneamente. Já no ataque baseado no desvio padrão, o atacante adicionará um ruído obtido a partir do desvio padrão referente aos dados de treinamento.

A metodologia adotada neste trabalho será detalhada nas próximas seções. Inicialmente, na Seção 3.2, serão fornecidas informações gerais sobre o conjunto de dados selecionado e suas características. Em seguida, na Seção 3.3, as métricas utilizadas para avaliar a detecção de anomalias dos modelos e o impacto dos ataques serão apresentados. As seções 3.4 e 3.5 abordarão, respectivamente, a arquitetura do modelo de detecção de anomalias e do modelo de classificação. É importante notar a diferença entre os dois modelos, sendo que o modelo de detecção funciona com base nas predições e na comparação com o limiar, enquanto o modelo de classificação avalia individualmente cada observação para classificá-la como anômala ou normal, portanto, não há comparação com o limiar. Por fim, na Seção 3.6 será explicada a criação dos exemplos adversários.

3.2 Conjunto de Dados

O conjunto de dados utilizado neste experimento foi retirado de um *benchmark* gratuito [51] de detecção de anomalias e pontos de mudanças em séries temporais. Os dados provêm da leitura de sensores posicionados em um sistema de circulação de água, contendo 8 atributos, cada um representando um dos sensores, além do rótulo de anomalia:

- **Accelerometer1RMS**: Mostra a aceleração de vibração.
- **Accelerometer2RMS**: Mostra a aceleração de vibração.
- **Current**: Mostra a amperagem do motor elétrico.
- **Pressure**: Representa a pressão no circuito depois da bomba de água.
- **Temperature**: Mostra a temperatura do corpo do motor.
- **Thermocouple**: Representa a temperatura do fluido no circuito de circulação.
- **Voltage**: Mostra a tensão no motor elétrico.
- **RateRMS**: Representa a vazão de circulação do fluido dentro do circuito.
- **anomaly**: Mostra se a observação é anômala (0 ou 1).

Existem mais de 30 subconjuntos, os quais são divididos em: dados sem anomalias, dados com anomalias geradas pelo fechamento da válvula na saída do fluxo da bomba, dados com anomalias geradas pelo fechamento da válvula na entrada de vazão da bomba e outros tipos de anomalias, incluindo comportamento exponencial do desequilíbrio do rotor. O subconjunto referente ao comportamento exponencial do desequilíbrio do rotor foi selecionado para o desenvolvimento do trabalho, uma vez que não se objetiva a construção de um detector de anomalias altamente eficaz, mas a utilização de técnicas já consolidadas na literatura e o uso de um conjunto de dados onde elas tenham bons resultados para, então, realizar os ataques. Conforme a Figura 5, o conjunto selecionado apresenta um total de 751 observações, sendo 557 rotuladas como normais e outras 194 rotuladas como anomalias.

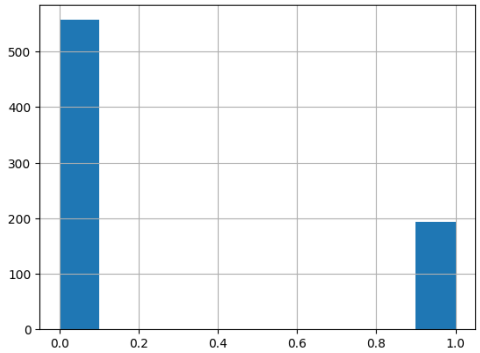

Figura 5 – Distribuição dos rótulos.

3.3 Métricas de Avaliação

As métricas utilizadas para avaliar a eficácia da detecção de anomalias neste trabalho são derivadas de contagens resumidas em uma matriz de confusão. Portanto, primeiramente, pretende-se apresentar quais são estas contagens:

- **Verdadeiros positivos (VP):** Quantidade de dados corretamente detectados como anomalia.
- **Verdadeiros negativos (VN):** Quantidade de dados normais não detectados como anomalia.
- **Falsos positivos (FP):** Quantidade de dados normais erroneamente detectados como anomalia.
- **Falsos Negativos (FN):** Quantidade de dados anômalos erroneamente detectados como normais.

Com base nestas contagens, foram calculadas as seguintes métricas:

- **Taxa de Falso Negativo (TFN) e Taxa de Falso Positivo (TFP):** TFP avalia a proporção de amostras normais que foram erroneamente detectadas como anomalias, enquanto TFN avalia a proporção de anomalias não detectadas.

$$TFN = \frac{FN}{(VP + FN)} \quad TFP = \frac{FP}{(FP + VN)}$$

- **Precisão e Revocação:** Precisão avalia a proporção de anomalias corretamente detectadas em relação à quantidade de amostras genuinamente anômalas. Revocação avalia a proporção de anomalias corretamente detectadas em relação ao total de anomalias existentes.

$$Precisão = \frac{VP}{(VP + FP)} \quad Revocação = \frac{VP}{(VP + FN)}$$

- **F1-Score:** Essa métrica é uma combinação das métricas Precisão e Revocação. De maneira geral, mede o grau de assertividade, realizando uma média harmônica entre as duas.

$$F1 - Score = 2 \cdot \left(\frac{Precisão \cdot Revocação}{Precisão + Revocação} \right)$$

- **MCC:** Essa métrica é considerada mais balanceada e pode ser utilizada nos casos em que as classes tenham tamanhos muito diferentes. O MCC varia de -1 até 1, sendo que 1 que as classificações realizadas são perfeitas, 0 indica que elas assumem um comportamento aleatório, e -1 indica que há uma inversão das classes.

$$MCC = \frac{VP \times VN - FP \times FN}{\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}}$$

3.4 Arquitetura e Funcionamento do Modelo de Detecção de Anomalias Baseado em LSTM

Inicialmente, os dados devem ser tratados para o treinamento do modelo alvo. Tratando-se de uma série temporal, a ordem original dos dados deve ser mantida, pois as observações adjacentes são dependentes umas das outras. Dessa forma, uma porção inicial do conjunto de dados é selecionada de modo que não haja nenhuma anomalia, já que esta será utilizada para o treinamento do modelo, que deve aprender a realizar predições de valores futuros de acordo com o comportamento padrão do sistema de circulação de água. Além disso, uma pequena parte desses dados, de 50 amostras, será aproveitada na fase de definição do limiar. O restante dos dados será disposto na fase de inferência, onde serão submetidos ao detector de anomalias, e os resultados serão comparados com os rótulos. A Figura 6 representa o funcionamento do sistema de detecção de anomalias baseado em LSTM.

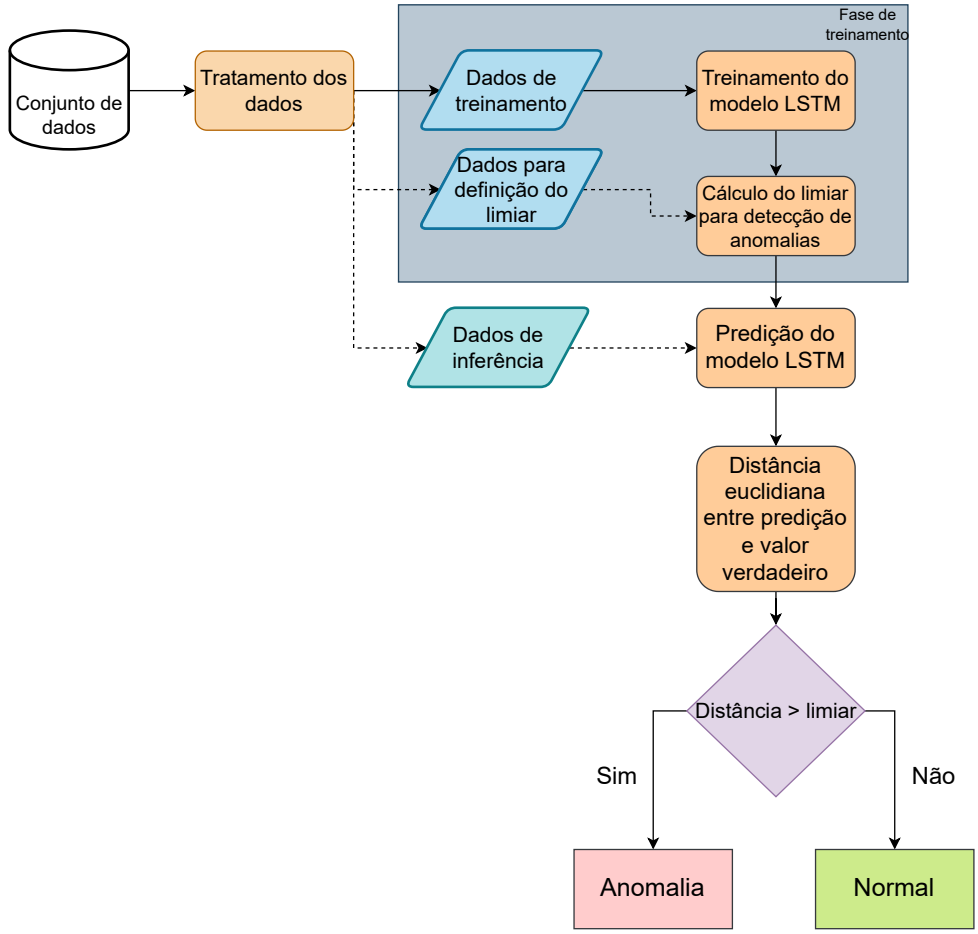

Figura 6 – Fluxo de detecção de anomalias.

A série temporal é dividida em janelas de tempo (t) de tamanho 5. Cada janela de tempo contém 5 observações consecutivas como entrada para o modelo. Por exemplo, se a série temporal fosse $[t_0, t_1, t_2, t_3, t_4, t_5, t_6 ...]$, a primeira janela de tempo de entrada seria $[t_0, t_1, t_2, t_3, t_4]$ sendo a saída do modelo a janela de tempo $[t_5]$, a segunda janela seria $[t_2, t_3, t_4, t_5, t_6]$ com a saída $[t_7]$ e assim por diante. Essas 5 observações passadas são usadas pelo modelo para aprender padrões temporais e capturar informações relevantes para fazer a previsão da próxima janela de tempo, conforme a Figura 7.

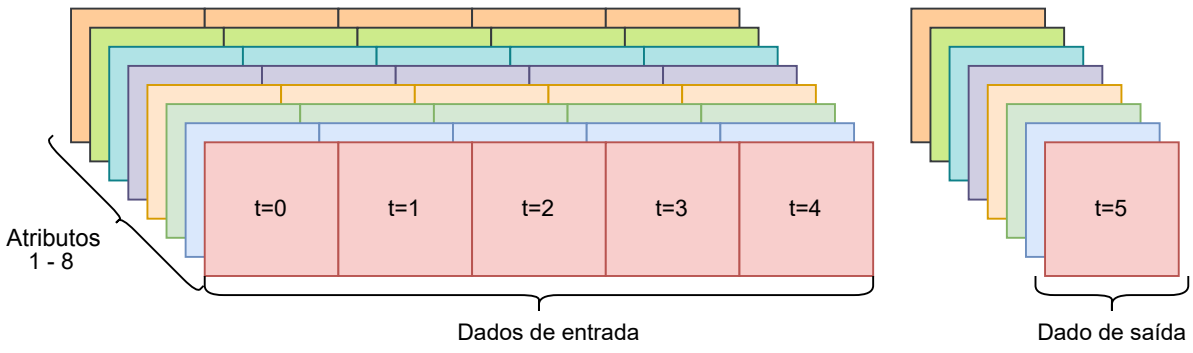

Figura 7 – Janela de dados.

Além disso, os dados são padronizados com o objetivo de transformá-los em uma escala limitada e comum, evitando que atributos com intervalos de valores maiores tenham mais importância durante o treinamento do algoritmo. Dessa forma, a normalização faz com que todos os atributos sejam igualmente importantes.

Tabela 1 – Descrição dos dados de treinamento.

	Accelerometer1RMS	Accelerometer2RMS	Current	Pressure	Temperature	Thermocouple	Voltage	Volume Flow RateRMS
count	400	400	400	400	400	400	400	400
mean	0,492537	0,465410	0,552681	0,494375	0,629966	0,558685	0,525644	0,547760
std	0,256604	0,205667	0,223295	0,217065	0,235306	0,187378	0,226363	0,179947
min	0,005378	0,000678	0,000000	0,000000	0,000000	0,131773	0,000000	0,326441
25%	0,268353	0,299001	0,398275	0,250000	0,450150	0,405172	0,398532	0,333209
50%	0,471788	0,438082	0,574248	0,500000	0,680163	0,544951	0,539604	0,659854
75%	0,708799	0,632221	0,719686	0,500000	0,805138	0,702586	0,672405	0,666519
max	1,000000	1,000000	1,000000	1,000000	1,000000	1,000000	1,000000	1,000000

Tabela 2 – Descrição dos dados de definição do limiar.

	Accelerometer1RMS	Accelerometer2RMS	Current	Pressure	Temperature	Thermocouple	Voltage	Volume Flow RateRMS
count	50	50	50	50	50	50	50	50
mean	0,226864	0,350378	0,536180	0,555000	0,699608	0,209286	0,475720	0,530253
std	0,134271	0,217033	0,253374	0,153945	0,145768	0,102542	0,238635	0,176963
min	0,000000	0,000000	0,029620	0,250000	0,432266	0,000000	0,039277	0,000000
25%	0,131798	0,177198	0,307716	0,500000	0,608333	0,116379	0,328654	0,333209
50%	0,196853	0,325724	0,590703	0,500000	0,692826	0,197044	0,500356	0,659803
75%	0,325792	0,519693	0,731688	0,750000	0,774007	0,278325	0,568622	0,666519
max	0,533736	0,758259	0,973520	0,750000	0,988355	0,445813	0,934861	0,666519

Tabela 3 – Descrição dos dados de inferência.

	Accelerometer1RMS	Accelerometer2RMS	Current	Pressure	Temperature	Thermocouple	Voltage	Volume Flow RateRMS
count	301	301	301	301	301	301	301	301
mean	0,028279	-0,082805	0,582008	0,508306	-0,504427	0,359704	0,522372	-0,384191
std	2,183022	1,539864	0,214114	0,208000	0,420320	0,218795	0,202989	3,622569
min	-15,226006	-10,776893	0,004430	-0,250000	-1,068569	-0,086207	0,011358	-25,519503
25%	0,241811	-0,036761	0,447594	0,500000	-0,762896	0,172414	0,397760	-0,353715
50%	0,330952	0,123751	0,630000	0,500000	-0,642276	0,370690	0,532847	0,000170
75%	0,426499	0,278079	0,737503	0,500000	-0,330831	0,525862	0,648007	0,659752
max	0,671475	0,623158	0,995290	1,000000	0,505572	0,908867	0,984764	0,999830

A padronização dos dados foi feita pelo algoritmo Min-Max, limitando os valores entre 0 e 1 com base nos dados de treinamento. Dessa forma, todos os dados da fase de inferência, possivelmente, serão menores que 0 ou maiores que 1. As Tabelas 1, 2 e 3 representam, respectivamente, a descrição dos dados de treinamento, dados utilizados na definição do limiar e dados de inferência. Cada uma das tabelas apresenta informações de todos os atributos em cada um dos conjuntos, mostrando a quantidade de dados (count), a média dos dados (mean), o desvio padrão (std), o menor valor (min), o quartil inferior (25%), mediana (50%), quartil superior (75%) e o maior valor (max). Além disso, vale ressaltar que, no conjunto de dados separado para a fase de inferência, os 107 primeiros são rotulados como normais, e, o restante dos 189 dados, rotulados como anômalos.

Após o tratamento dos dados, o modelo de detecção de anomalias é criado contendo duas camadas LSTM empilhadas cada uma com 100 células. Essa arquitetura é adequada para predição de séries temporais, sendo eficiente para lidar com dependências dos dados a longo prazo, uma vez que contém células que funcionam como uma memória, as quais guardam informações de longos períodos. Além disso, após as camadas LSTM, há uma camada densa com 16 neurônios e função de ativação ReLu, utilizada para extrair informações úteis das camadas anteriores e reduzir a dimensionalidade dos dados. Por último, há uma camada *Dropout* para reduzir o sobreajuste do modelo e aumentar a generalização, seguido de uma camada densa, que resultará na saída do modelo. Por fim, o modelo é submetido ao treinamento com as amostras tratadas anteriormente. A Figura 8 representa a arquitetura descrita.

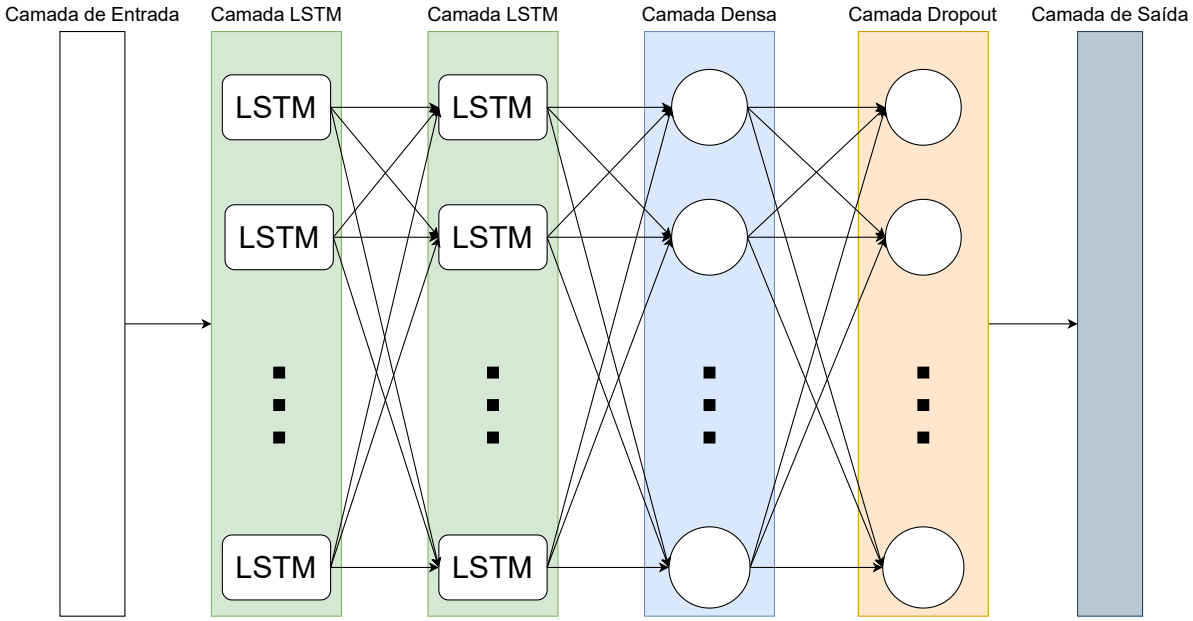

Figura 8 – Arquitetura do modelo de detecção de anomalias LSTM.

O modelo alvo é treinado utilizando o *framework TensorFlow*. Os experimentos foram conduzidos em um computador com as especificações: Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz 4.00 GHz, 16 GB memória RAM e uma unidade de processamento gráfico NVIDIA GeForce GTX 1080.

Uma vez que o modelo de detecção foi criado e passou pelo treinamento, é possível utilizá-lo sobre os dados de inferência para avaliar a detecção de anomalias. Tendo em vista que o esse modelo foi treinado com dados livres de anomalias, a ideia principal do detector de anomalias é que seja previsto o comportamento normal do sistema baseado em dados históricos. Posteriormente, a predição (y_p) do modelo será comparada com o valor real (y_v), calculando a distância euclidiana (DE) entre os dois. Posteriormente, esse resíduo é comparado com um limiar para verificar se a observação representa ou não uma anomalia.

Assim, sendo x os dados de entrada, o modelo de detecção de anomalias f resultará em uma predição $f(x) = y_p$. Na fase de inferência do modelo, o resultado da predição é comparado com os valores reais y_v, calculando a distância euclidiana (DE) entre os dois e, posteriormente, esse resíduo é comparado com um limiar para verificar se o sistema está em um estado anormal.

$$\begin{cases} \text{Anômalo (1)}, & \text{se } DE(y_p, y_v) >= limiar, \\ \text{Normal (0)}, & \text{se } DE(y_p, y_v) < limiar. \end{cases}$$

Para definir o limiar, calcula-se o Erro Quadrático Médio (EQM) entre as predições (y_p) e os valores reais (y_v) dos dados de definição do limiar. Esse valor representa a diferença entre as previsões do modelo e os valores reais, elevados ao quadrado e divididos pelo número de observações. Depois de calcular o EQM, adiciona-se o desvio padrão (σ) dos Erros Quadráticos (EQ) das previsões do modelo com os valores reais. Finalmente, o limiar é definido como a soma do EQM com o desvio padrão dos erros quadráticos conforme a Equação 3.1.

$$limiar = EQM(y_p, y_v) + z \times \sigma(EQ(y_p, y_v)) \tag{3.1}$$

3.5 Arquitetura e Treinamento do Modelo de Classificação MLP

O modelo de classificação binária é um tipo de modelo de Aprendizado de Máquina que tem como objetivo classificar os dados em duas classes distintas, neste caso, normal ou anomalia. Para isso, ele utiliza uma função matemática para mapear os dados de entrada em uma classe específica.

Além do modelo alvo LSTM, mais um modelo será criado, nesse caso, pelo atacante. Tratando-se de um cenário de caixa branca, é possível que ele crie um *Multi Layer Perceptron* (MLP), o qual atua como classificador binário. O objetivo de criar esse modelo classificador é para que o atacante possa criar os exemplos adversários utilizando a técnica FGSM, uma vez que essa técnica é baseada no gradiente descendente da função de perda e precisa dos parâmetros do modelo.

O atacante também aplicará ataques contra esse modelo classificador, a fim de avaliar seus impactos, comparando os resultados obtidos pelo MLP com os resultados do modelo de detcção de anomalias baseado em LSTM. Essa comparação permitirá que o atacante entenda melhor as vulnerabilidades do modelo de detecção e desenvolva estratégias mais eficazes para violá-lo.

As características de continuidade da série temporal não precisam ser exploradas, já que cada observação é analisada individualmente em relação ao seu rótulo. Portanto,

os dados devem ser preparados de maneira diferente, sendo que 60% deles serão utilizados para o treinamento, e, o restante, na fase de inferência. Nesse caso, é importante a presença de amostras anômalas tanto nos dados de treinamento quanto nos dados de inferência, com a mesma proporção, já que o modelo precisa aprender a diferenciar amostras anômalas de amostras normais. O funcionamento desse modelo é representado na Figura 9.

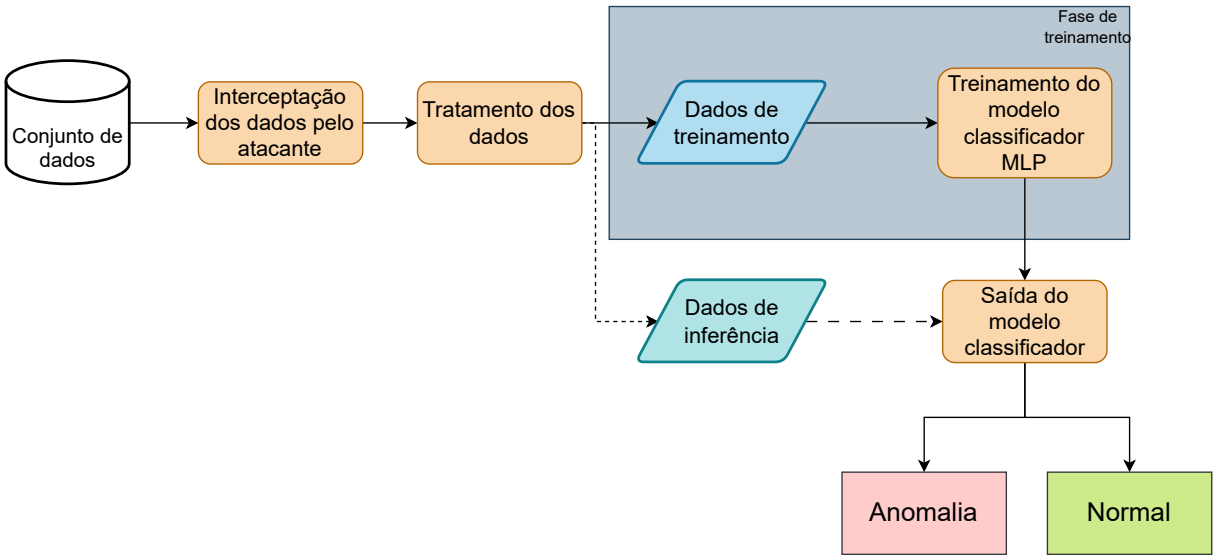

Figura 9 – Classificação de anomalias.

A arquitetura do modelo, representada na Figura 10, conta com uma camada de entrada com 8 nós para cada atributo. Após a camada de entrada, existem duas camadas densas, sendo que a primeira contém 64 neurônios e, a segunda, 32 neurônios. Finalmente, uma camada de saída que representa a classificação da observação (0 ou 1).

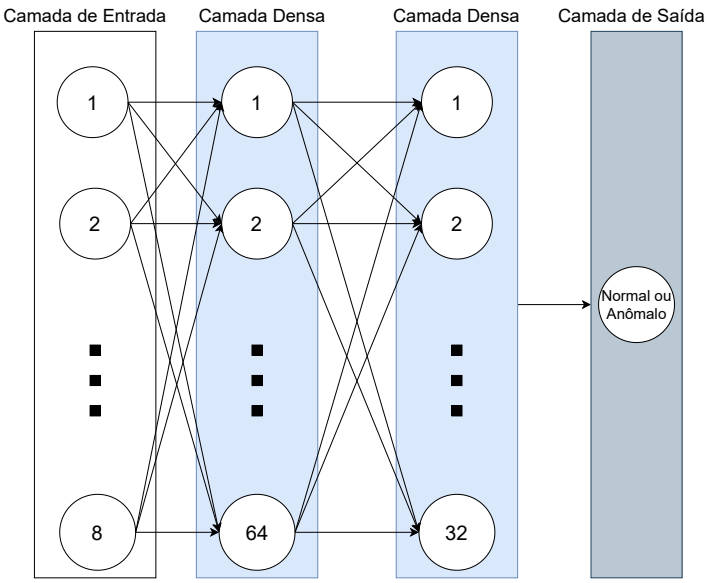

Figura 10 – Arquitetura do modelo de classificação MLP.

3.6 Exemplos Adversários

Exemplos adversários são os dados de entrada manipulados intencionalmente com o intuito de enganar o modelo alvo para fazer predições equivocadas. Normalmente, esses dados são criados adicionando a menor perturbação possível aos dados originais, de forma que atrapalhe o desempenho do modelo alvo. Vale relembrar que os exemplos adversários foram criados para ambos os modelos apresentados anteriormente.

Frisa-se que, tratando-se de um cenário de ataque de caixa branca, o atacante possui acesso às informações dos dados coletados e sua rotulação como anômalo ou normal. Assim sendo, os ataques foram divididos em dois objetivos principais: manipular amostras para gerar falsos positivos e falsos negativos.

Os exemplos adversários foram criados com base em duas técnicas. A primeira técnica é utilizada como uma linha de base, sendo essa uma técnica baseada no desvio padrão dos dados da fase de treinamento do modelo de detecção de anomalias, ou seja, dos dados apresentados na Tabela 1. Para realizar esse ataque é necessário calcular o desvio padrão de cada atributo e adicioná-los na observação que será atacada. Essa técnica foi denominada de ataque *Naive*.

Já a segunda técnica é o ataque *Fast Gradient Sign Method* (FGSM), que envolve o gradiente da função de perda do modelo e, em seguida, utiliza esse gradiente para manipular o dado na direção que maximize a perda. Portanto, a perturbação δ será calculada conforme a equação 3.2, onde ϵ é uma constante que representa a magnitude do ataque, $\nabla_x J(\theta, x, y_v)$ é o gradiente da função de perda, $J(\theta, x, y_v)$ e θ são os parâmetros do modelo.

$$\delta = \epsilon \cdot sign(\nabla_x J(\theta, x, y_v)) \tag{3.2}$$

Além disso, destaca-se que, para criar falsos positivos, é necessário perturbar apenas as amostras rotuladas como normais, com o objetivo de inverter a classificação após o ataque, gerando alarmes falsos no sistema. Já no caso dos falsos negativos, o princípio é o mesmo, mas com as amostras rotuladas como anômalas, mascarando possíveis problemas no sistema.

Tendo em vista as técnicas de ataque e os principais objetivos pontuados anteriormente, algumas perguntas foram formuladas para direcionar a análise e o entendimento dos diferentes cenários de ataque:

- **Utilizando o ataque *Naive* é possível criar amostras adversárias que resultem em falso positivo e falso negativo no modelo de detecção de anomalias (LSTM)?**

Com o objetivo de criar falsos positivos, utilizando o ataque *Naive*, é necessário aumentar a Distância Euclidiana (DE) entre a predição do modelo ($f(x)$) e o valor real (y_v) adicionando uma perturbação σ em x, onde σ é o desvio padrão das *features* dos dados de treinamento, x^* o exemplo adversário resultante do ataque *Naive* e k uma constante que representa a magnitude do ataque. Dessa forma, é possível criar alarmes falsos no sistema:

$$x^* = x + k \times \sigma \tag{3.3}$$

$$DE(f(x^*), y_v) >= limiar \tag{3.4}$$

Já no caso dos falsos negativos utilizando o ataque *Naive*, basta inverter a perturbação σ calculada anteriormente:

$$x^* = x - k \times \sigma \tag{3.5}$$

$$DE(f(x^*), y_v) < limiar \tag{3.6}$$

- **Utilizando o ataque *FGSM* é possível criar amostras adversárias que resultem em falso positivo e falso negativo no modelo de detecção de anomalias (LSTM)?**

Visando responder ao segundo questionamento, o objetivo inicial do atacante é gerar falsos positivos, ou seja, criar exemplos adversários a partir de amostras rotuladas como normais. Assim sendo, é necessário criar um exemplo adversário x^* modificando a entrada original x. Para isso, é necessário calcular a pertubação δ que será aplicada a x. Essa perturbação será construída a partir do ataque *FGSM* [3]. Dessa forma, será possível aumentar a Distância Euclidiana (DE) onde $f(x^*)$ é a predição do modelo com o exemplo adversário e y_v o valor real, ultrapassando o valor do limiar:

$$x^* = x + \delta \tag{3.7}$$

$$DE(f(x^*), y_v) >= limiar \tag{3.8}$$

Além disso, para responder completamente ao segundo questionamento, também é necessário criar amostras adversárias que resultem em falsos negativos, ou seja, amostras rotuladas como anômalas devem ser classificadas como normais. Portanto, o intuito desse ataque é diminuir a Distância Euclidiana (DE). Nesse caso, a perturbação é adicionada no

sentido contrário, minimizando a perda da função de custo ao invés de maximizar, sendo x^* o exemplo adversário resultante para gerar falsos negativos. A Distância Euclidiana (DE) entre a saída $f(x^*)$ e o valor verdadeiro y_v deve diminuir com esse ataque, a ponto de ficar menor que o limiar:

$$x^* = x - \delta \tag{3.9}$$

$$DE(f(x^*), y_v) < limiar \tag{3.10}$$

- **Entre os ataques *Naive* e FGSM, qual deles gera mais impacto em cada um dos cenários?**

Com base nos resultados dos últimos dois questionamentos, é possível avaliar qual ataque é mais eficiente para prejudicar o modelo alvo, considerando as métricas de avaliação antes e depois dos ataques. Além disso, é importante levar em consideração a magnitude das perturbações para cada um dos ataques, uma vez que seu objetivo é minimizar a perturbação aplicada, maximizando o impacto causado.

- **Considerando o resultado dos ataques no modelo de detecção de anomalias, qual a diferença no impacto do ataque Naive contra um modelo de classificação (MLP)?**

Já no caso do ataque *Naive*, por ser um ataque mais simples que não leva em consideração algumas informações úteis, como, por exemplo, a direção do gradiente, foram necessários alguns testes para determinar a direção da perturbação aplicada no exemplo adversário x^*. Dessa forma a geração de falsos positivos segue Equação 3.11 enquanto a geração de falsos negativos segue a Equação 3.12.

$$x^* = x - k \times \sigma \tag{3.11}$$

$$x^* = x + k \times \sigma \tag{3.12}$$

- **Considerando o resultado dos ataques no modelo de detecção de anomalias, qual a diferença no impacto do ataque FGSM contra um modelo de classificação (MLP)?**

Nesse caso, o objetivo é criar ataques para o modelo de classificação, sendo esse um baseado em MLP. Dessa forma, é possível comparar com os resultados dos ataques contra o modelo de detecção de anomalias baseado em LSTM. Ambas as técnicas de ataque serão

utilizadas novamente para a criação dos exemplos adversários. Diferente do caso anterior, o modelo em questão não passa por uma comparação com um limiar.

O processo de aplicação da perturbação FGSM no modelo classificador é bastante simples. Primeiro, é necessário determinar se a observação é classificada como normal ou anormal. Em seguida, a perturbação é adicionada à entrada original, na direção contrária à classificação do modelo. Se a observação for classificada como normal, a perturbação será adicionada na direção contrária à normalidade. Caso a entrada seja classificada como anômala, a perturbação será adicionada na direção contrária à anomalia. Vale ressaltar que essa direção é calculada com base no gradiente da função de perda em relação à entrada com sua respectiva saída. Portanto, a criação de falsos positivos e falsos negativos seguem a equação 3.13, onde δ é obtido através da técnica FGSM, como demonstra a Equação 3.14.

$$x^* = x + \delta \tag{3.13}$$

$$\delta = \epsilon \cdot sign(\nabla_x J(\theta, x, y_v)) \tag{3.14}$$

4 RESULTADOS

Primeiramente, é necessário avaliar o resultado do modelo alvo sem a interferência de nenhum tipo de ataque. Para isso, foi necessário definir o melhor ajuste para o limiar através da variação de z entre 15 e 30, conforme a equação 3.1 referente ao cálculo do limiar. Para definir esse valor, o modelo de detecção de anomalias foi treinado 5 vezes com os mesmos parâmetros e dados. Em cada uma delas, a capacidade de detecção foi avaliada para os valores de z e calculada a média das métricas entre as 5 iterações. A Figura 11 mostra os resultados obtidos, onde é possível notar que os melhores valores de z estão entre 21 e 22. Portanto, o valor da constante $z = 21$ foi definido.

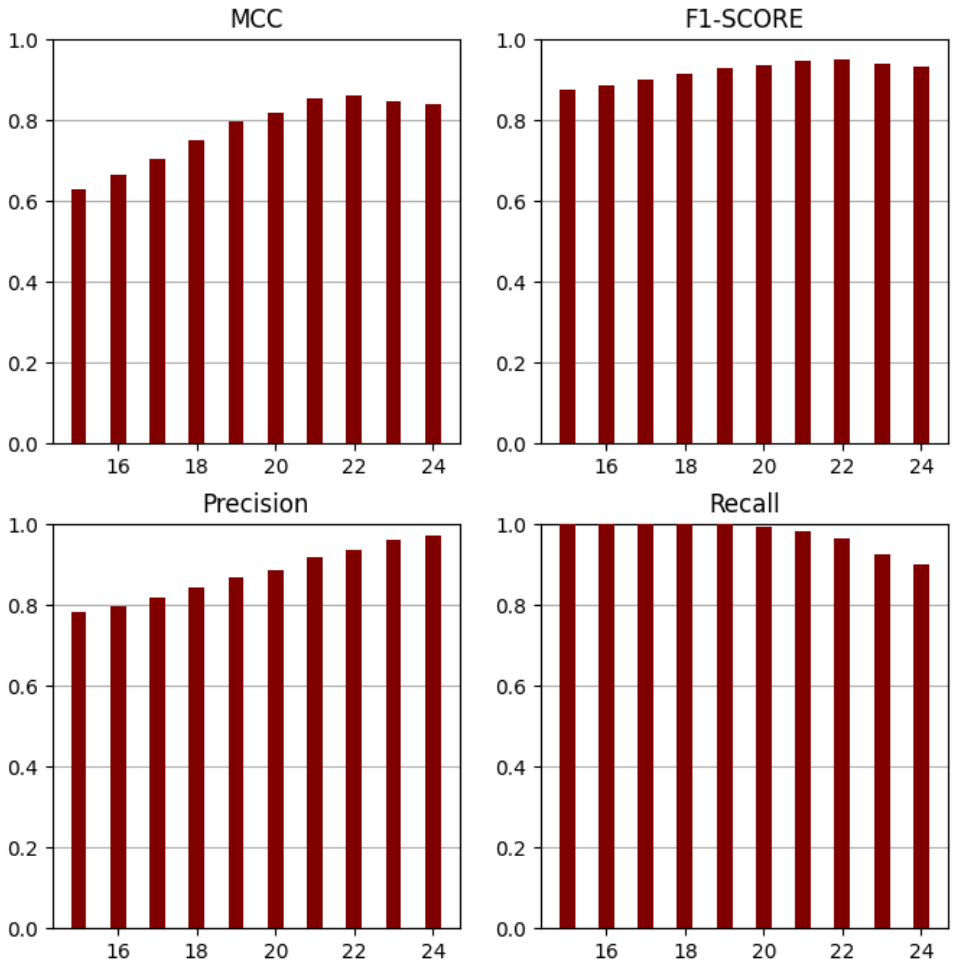

Figura 11 – Resultados obtidos para cada variação de z na definição do limiar.

Além disso, os resultados apresentados nas Seções 4.1, 4.2, 4.3, 4.4 e 4.5, que respondem aos questionamentos propostos, foram coletados de maneira semelhante, onde os modelos foram treinados 5 vezes. No caso do modelo baseado LSTM, os mesmos parâmetros e dados foram utilizados em cada uma das vezes. Já no caso do modelo de

classificação, os parâmetros foram os mesmos, mas a divisão dos dados de treinamento e inferência foi feita aleatoriamente em cada uma das iterações, mantendo sempre a proporção na quantidade de observações anômalas e normais. Ao final, foi calculada a média aritmética simples para cada uma das iterações.

4.1 Utilizando o ataque *Naive* é possível criar amostras adversárias que resultem em falso positivo e falso negativo no modelo de detecção de anomalias (LSTM)?

Sendo esse ataque baseado no desvio padrão, o valor de σ na equação 3.9, que representa o desvio padrão de cada uma das características dos dados de treinamento, é representado na Tabela 4. Após definir o valor de σ, espera-se que, em um primeiro cenário, os exemplos adversários criados a partir do ataque *Naive* resultem em um aumento na taxa de falsos positivos.

Tabela 4 – Desvio padrão dos atributos do conjunto de treinamento

Feature	Desvio Padrão
Accelerometer1RMS	0,256604
Accelerometer2RMS	0,205667
Current	0,223295
Pressure	0,217065
Temperature	0,235306
Thermocouple	0,187378
Voltage	0,226363
Volume Flow RateRMS	0,179947

Conforme a Tabela 5, é possível notar o aumento da taxa de falsos positivos de acordo com a magnitude do ataque. Neste cenário, houve um aumento máximo de aproximadamente 85% na taxa de falsos positivos, com a magnitude $\epsilon = 0,5$. Esses resultados servem como uma linha de base para avaliar os resultados do ataque FGSM. Apesar do aumento da taxa de falsos positivos, ainda não é um cenário tão preocupante quando consideradas as outras métricas de avaliação, onde o MCC sofreu uma redução de 5,36% e o F1-score redução de 1,58%.

Ao contrário, no segundo caso, a intenção do atacante é gerar falsos negativos, perturbando apenas as amostras rotuladas como anômalas para mascará-las. Os resultados obtidos são apresentados Tabela 6, com um aumento de 66% na taxa de falsos negativos. Assim como no caso anterior, o impacto nas métricas que avaliam a capacidade de detecção de anomalias não foi muito alto, com uma redução de, aproximadamente, 6% para o MCC e 2,42% para o F1-Score.

Tabela 5 – Média das métricas aplicando o ataque *Naive* para gerar falsos positivos no Modelo de detecção de anomalias.

ϵ	**MCC**	**F1-score**	**Precisão**	**Revocação**	**TFN**	**TFP**
Sem ataque	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,01	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,05	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,1	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,2	0,8551	0,9465	0,9567	0,9365	0,0634	0,0747
0,3	0,8551	0,9465	0,9567	0,9365	0,0634	0,0747
0,4	0,8397	0,9414	0,9465	0,9365	0,0634	0,0934
0,5	0,8166	0,9340	0,9315	0,9365	0,0634	0,1214

Tabela 6 – Média das métricas aplicando o ataque *Naive* para gerar falsos negativos no Modelo de detecção de anomalias.

ϵ	**MCC**	**F1-score**	**Precisão**	**Revocação**	**TFN**	**TFP**
Sem ataque	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,01	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,05	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,1	0,8495	0,9433	0,9615	0,9259	0,0740	0,0654
0,2	0,8429	0,9405	0,9613	0,9206	0,0793	0,0654
0,3	0,8364	0,9376	0,9611	0,9153	0,0846	0,0654
0,4	0,8235	0,9318	0,9606	0,9047	0,0952	0,0654
0,5	0,8109	0,9260	0,9602	0,8941	0,1058	0,0654

4.2 Utilizando o ataque *FGSM* é possível criar amostras adversárias que resultem em falso positivo e falso negativo no modelo de detecção de anomalias (LSTM)?

Inicialmente, o objetivo é aumentar a taxa falsos positivos, portanto, se o ataque for bem sucedido, além do aumento dessa taxa, ocorrerá uma redução nas métricas MCC e F1-Score.

Com base nos resultados observados na Tabela 7, é possível notar um aumento máximo de 757% na taxa de falsos positivos, que é significativamente maior que no ataque *Naive*. Dessa forma, conclui-se o objetivo do ataque nesse primeiro cenário com uma redução de 47,64 % na métrica MCC e 12,44% na métrica F1-Score. Tais resultados demonstram um impacto muito maior em comparação com o ataque anterior.

A Figura 12 é um exemplo gráfico desse ataque, onde os pontos representam as distâncias euclidianas das saídas do modelo LSTM em relação aos dados verdadeiros. Os

Tabela 7 – Média das métricas aplicando o ataque FGSM para gerar falsos positivos no Modelo de detecção de anomalias.

ϵ	**MCC**	**F1-score**	**Precisão**	**Revocação**	**TFN**	**TFP**
Sem ataque	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,01	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,05	0,8551	0,9465	0,9567	0,9365	0,0634	0,0747
0,1	0,8166	0,9340	0,9315	0,9365	0,0634	0,1214
0,2	0,7633	0,9170	0,8984	0,9365	0,0634	0,1869
0,3	0,6722	0,8894	0,8468	0,9365	0,0634	0,2990
0,4	0,5646	0,8592	0,7937	0,9365	0,0634	0,4299
0,5	0,4518	0,8309	0,7468	0,9365	0,0634	0,5607

pontos em azul representam as distâncias que ficaram abaixo do limiar, ou seja, consideradas normais pelo sistema. Os pontos em vermelho são as distâncias tidas como anômalas pelo algoritmo por ficarem acima do limiar. Por fim, os pontos em verde representam os falsos positivos. Essa separação é feita pela linha vermelha horizontal, que representa o limiar. Além disso, a linha preta vertical representa a separação entre os rótulos reais das amostras, ou seja, pontos antes da linha preta são verdadeiramente rotulados como normais, enquanto os pontos após a linha preta são verdadeiramente rotulados como anômalos. Com essa representação, é possível notar que, de acordo com o aumento da magnitude ϵ, a distância euclidiana entre as saídas produzidas a partir das amostras atacadas e os valores verdadeiros também aumenta, ficando com valores acima do limiar previamente estabelecido.

A Figura 12.A é uma representação do estado normal do algoritmo de detecção de anomalias, onde é possível notar a presença de alguns falsos positivos, representados pelos pontos em verde, mas não causados pelos exemplos adversários, são observações em que o próprio modelo não foi capaz de detectar corretamente. Já as Figuras 12.B, 12.C e 12.D representam o resultado do sistema quando submetido a exemplos adversários criados, respectivamente, com as magnitudes $\epsilon = 0,1$, $\epsilon = 0,3$ e $\epsilon = 0,5$. É possível notar que, conforme o valor da magnitude aumenta, os pontos anteriores à linha vertical preta são elevados acima do limiar e há um aumento na quantidade de pontos verdes, demonstrando o sucesso do ataque na geração de falsos positivos.

Já em um segundo cenário, o objetivo é aplicar a perturbação do ataque FGSM em amostras rotuladas como anômalas, ou seja, é esperado o aumento da taxa de falsos negativos. Em um cenário real, essa abordagem pode ser bastante prejudicial ao sistema, já que, se bem sucedida, ele entraria em um estado defeituoso sem gerar nenhum alerta.

A Tabela 8 representa os resultados obtidos nesse cenário, com um aumento má-

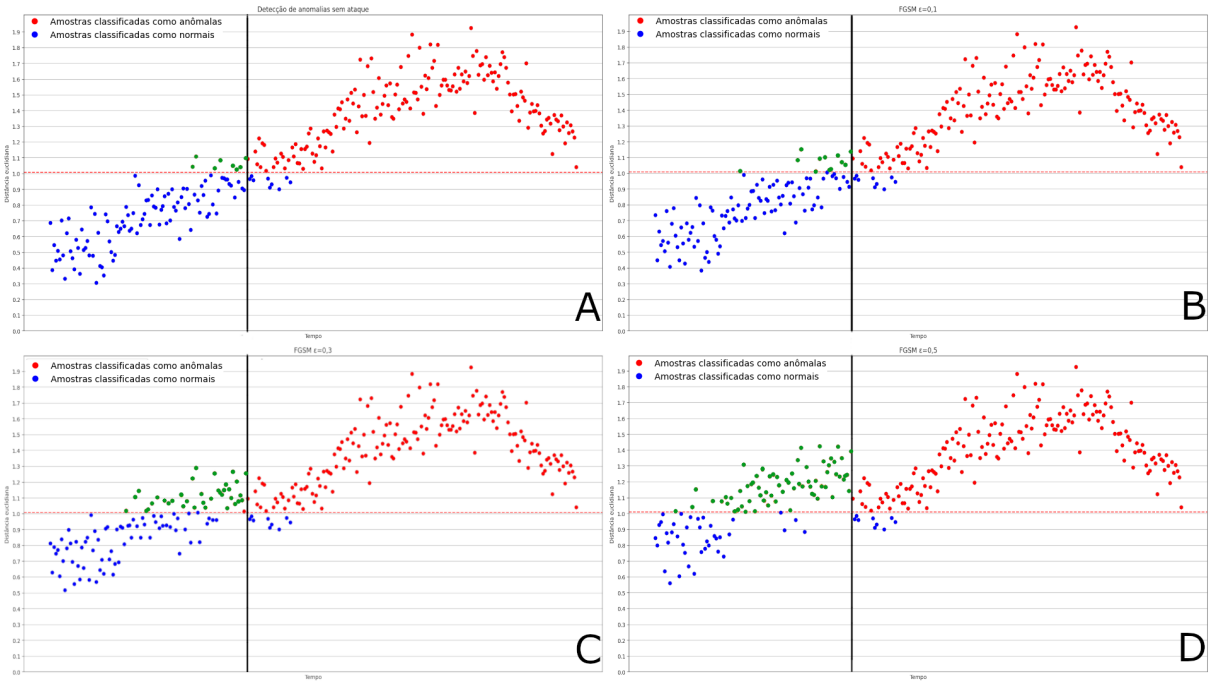

Figura 12 – Representação gráfica do ataque FGSM para gerar falsos positivos contra o modelo de detecção de anomalias.

Tabela 8 – Média das métricas aplicando o ataque FGSM parar gerar falsos negativos no Modelo de detecção de anomalias.

ϵ	**MCC**	**F1-score**	**Precisão**	**Revocação**	**TFN**	**TFP**
Sem ataque	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,01	0,8629	0,9490	0,9619	0,9365	0,0634	0,0654
0,05	0,8429	0,9405	0,9613	0,9206	0,0793	0,0654
0,1	0,8047	0,9230	0,96	0,8888	0,1111	0,0654
0,2	0,7863	0,9141	0,9593	0,8730	0,1269	0,0654
0,3	0,7744	0,9080	0,9588	0,8624	0,1375	0,0654
0,4	0,7924	0,9171	0,9595	0,8783	0,1216	0,0654
0,5	0,8561	0,9462	0,9617	0,9312	0,0687	0,0654

ximo de, aproximadamente, 116% na taxa de falsos negativos, uma redução de 10,25% na métrica MCC e 4,32% de redução para o F1-Score. Com base nos resultados, é possível notar que gerar impactos ao modelo com falsos negativos é mais complicado do que com falsos positivos. Isso ocorre porque o método de detecção de anomalias leva em consideração um resíduo entre a predição e o valor real. Nesse cenário, gerar falsos negativos significa diminuir esse resíduo para que seja menor que o limiar. No entanto, pode não ser possível alcançar esse objetivo com um ataque simples como o FGSM, que aplica uma perturbação com a mesma magnitude para todos os atributos de cada observação. Sabendo que o cálculo dos resíduos leva em consideração esses atributos, seria necessário aplicar

magnitudes diferentes para cada um deles. Essa dificuldade é perceptível pela Figura 13 nos resíduos mais distantes que necessitam de uma magnitude maior para o sucesso do ataque. Esses resultados também demonstram o bom ajuste do limiar, já que apenas algumas amostras com resíduos mais próximos do limiar resultam em falsos negativos. Nessa figura, os pontos em azul, representam as distâncias que ficaram abaixo do limiar, os vermelhos são as distâncias tidas como anômalas pelo algoritmo por ficarem acima do limiar, separados pela linha vermelha horizontal que representa o limiar e, por fim, a linha preta vertical representa a separação entre os rótulos reais das amostras.

A Figura 13.A é um exemplo do estado normal do algoritmo de detecção de anomalias sem a presença de nenhum exemplo adversário. Já as Figuras 13.B, 13.C e 13.D representam o sistema quando submetido a exemplos adversários criados, respectivamente, com as magnitudes $\epsilon = 0,1$, $\epsilon = 0,3$ e $\epsilon = 0,5$, onde o ponto verde demonstra até onde a distância euclidiana foi reduzida pelo ataque. Com base nessa representação gráfica, é possível perceber que, conforme o aumento da magnitude até $\epsilon = 0,3$, os pontos em verde ficam cada vez mais próximos do limiar, ou seja, há uma diminuição da distância euclidiana. Ao mesmo tempo, é possível perceber a limitação desse tipo de ataque, tomando o gráfico da Figura 13.D onde alguns pontos continuam diminuindo enquanto outros pontos começam a apresentar uma distância euclidiana maior do que anterior ao ataque.

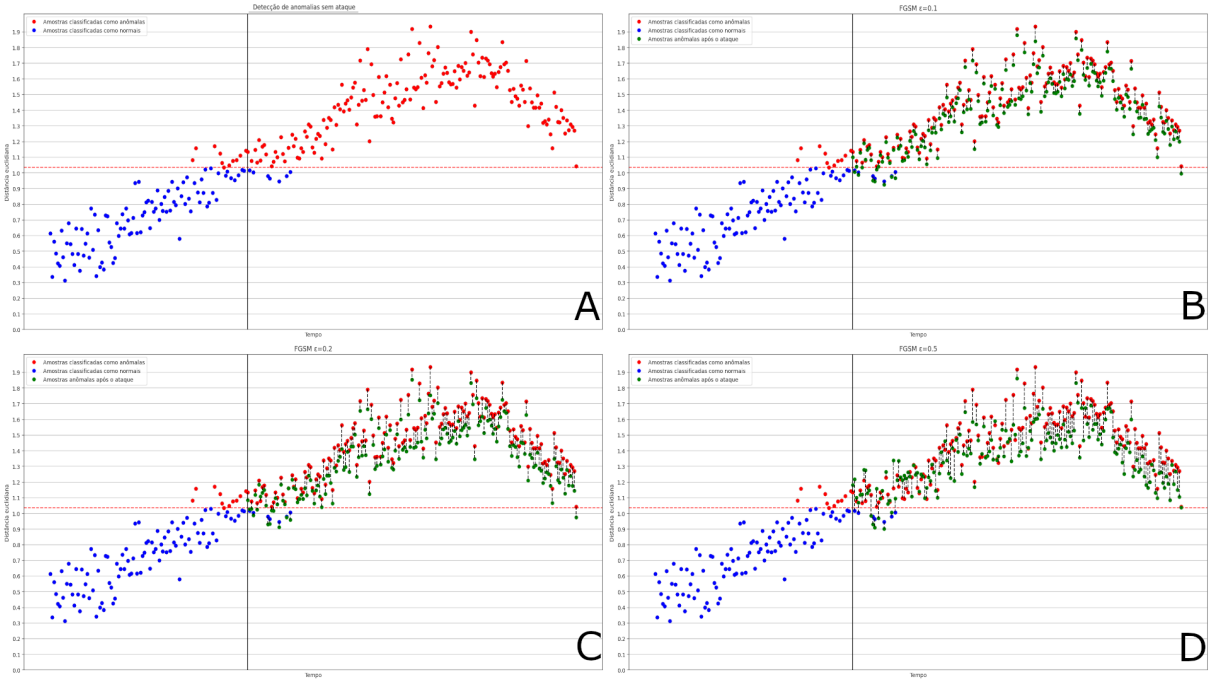

Figura 13 – Representação gráfica do ataque FGSM para gerar falsos nositivos contra o modelo de detecção de anomalias.

Para analisar mais detalhadamente esse cenário, a Figura 14 representa um recorte da Figura 13.D, sendo que a Figura 14.A apresenta as distâncias que estão aumentando e a Figura 14.B representa as distâncias que estão diminuindo. Torna-se notável que, para $\epsilon = 0,5$, algumas amostras aumentaram a distância ao invés de diminuí-la, sendo essas, em sua maioria, amostras mais próximas do limiar, enquanto as amostras mais distantes do limiar diminuíram a distância, como demonstra a Figura 14.B, mas não o suficiente para serem mascaradas. Vale ressaltar que o objetivo dos ataques adversários é adicionar a menor perturbação possível que cause o maior impacto, portanto, utilizar uma perturbação maior que $\epsilon = 0,5$, possivelmente, reduz ainda mais as distâncias da Figura 14.B, mas a custo de um aumento significativo na magnitude, o que tornaria o ataque mais fácil de ser detectado.

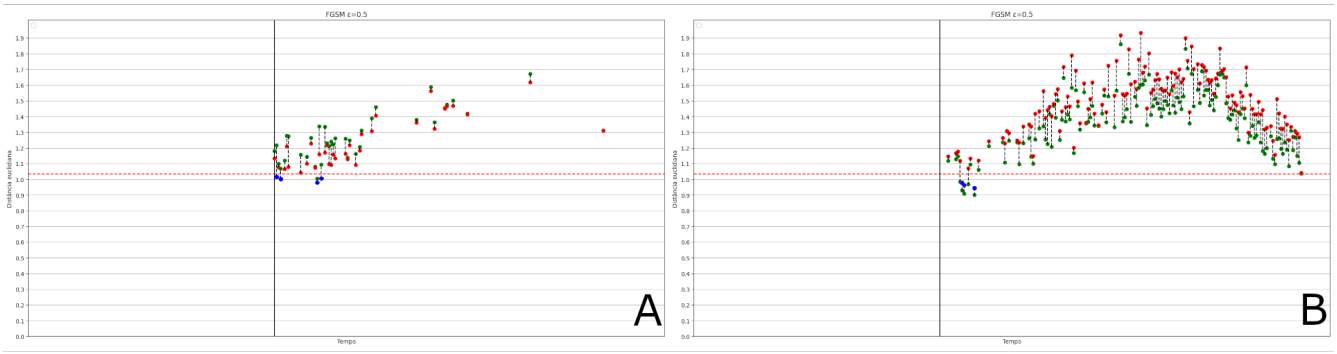

Figura 14 – Limitação do ataque FGSM para gerar falsos negativos no modelo de detecção de anomalias.

Em geral, os resultados apresentados demonstram um aumento na taxa de falsos negativos até um determinado ponto, e, a partir desse ponto, essa taxa começa a diminuir. Isso evidencia que aplicar a mesma magnitude de perturbação para todas as observações e seus atributos é insuficiente nesse cenário, já que um determinado atributo pode ter uma influência maior no cálculo do resíduo do que os outros.

4.3 Entre os ataques *Naive* e FGSM, qual deles gera mais impacto em cada um dos cenários ?

É possível notar que ambas as técnicas obtiveram sucesso nos cenários propostos, sendo que a criação de falsos positivos obteve resultados melhores do que a criação de falsos negativos. Primeiramente, na questão da geração de falsos positivos, o ataque *Naive* teve um aumento máximo de 85%, enquanto o ataque FGSM obteve um aumento de 757%. Já no cenário de geração de falsos negativos, enquanto a primeira técnica obteve um aumento de 66%, a segunda resultou em um aumento de 116%. Diante disso, é possível notar que a técnica FGSM gerou maior impacto contra o modelo de detecção de anomalias do que a técnica *Naive*.

A Figura 15 representa o crescimento da taxa de falsos positivos para ambos os ataques, de acordo com a magnitude da perturbação aplicada. Observa-se que a taxa de falsos positivos aumenta rapidamente sob o ataque FGSM em comparação com o ataque *Naive*. Isso evidencia a facilidade do ataque FGSM em gerar falsos positivos em regiões críticas do modelo, conforme aumenta a magnitude da perturbação.

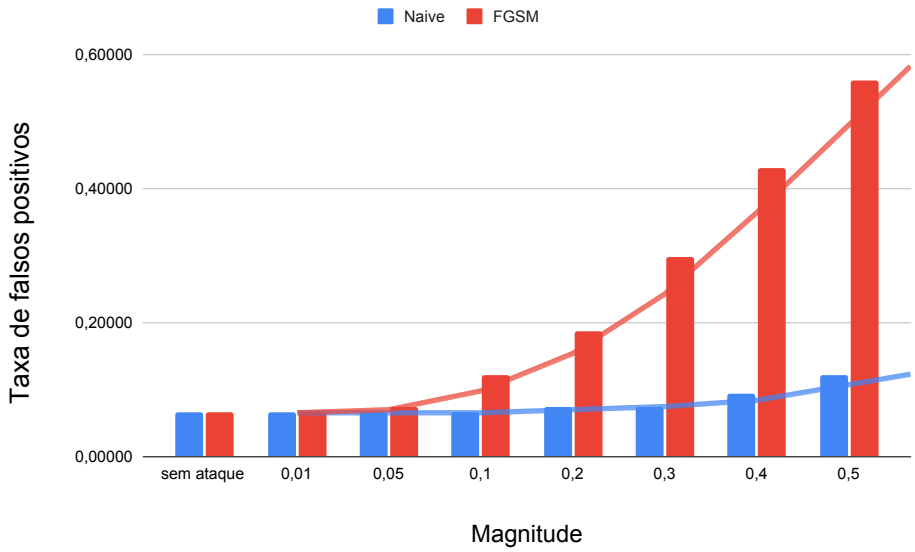

Figura 15 – Resultados da criação de falsos positivos utilizando os ataques *Naive* e FGSM.

Na Figura 16, é possível observar a mudança na taxa de falsos negativos para ambos os ataques. Verifica-se que o ataque FGSM apresenta o resultado máximo em $\epsilon = 0,3$, enquanto o ataque *Naive* apresenta uma baixa variação na taxa de falsos negativos, sendo incapaz de alcançar o resultado do FGSM. De qualquer forma, é importante destacar que o ataque FGSM não foi tão eficiente em gerar falsos negativos quanto em gerar falsos positivos.

Isso ocorre porque o ataque FGSM aplica a mesma magnitude de perturbação

para todos os atributos de uma observação, o que pode ser insuficiente para capturar a particularidade de cada atributo e sua influência na previsão do modelo. Portanto, uma abordagem mais individualizada na escolha da magnitude de perturbação poderia ser adotada para cada atributo, levando em conta suas particularidades e importância na previsão do modelo.

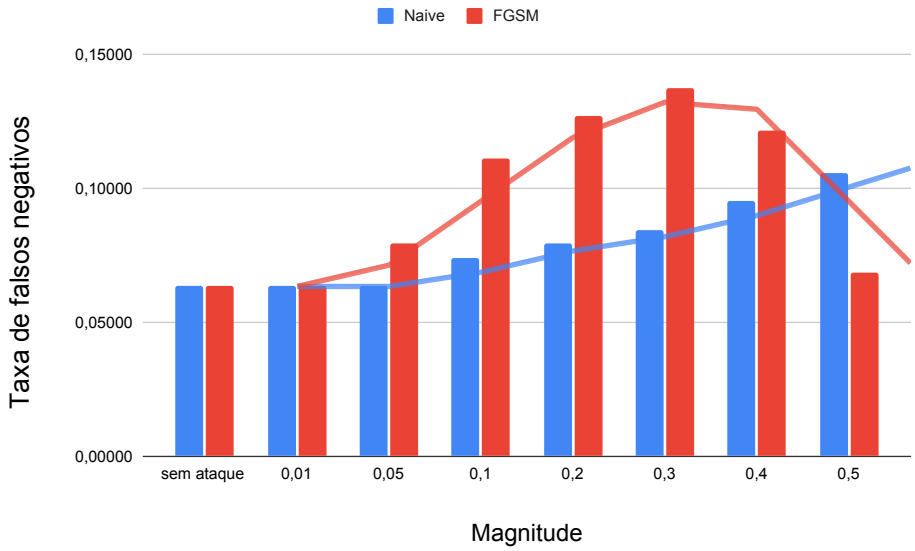

Figura 16 – Resultados da criação de falsos negativos utilizando os ataques *Naive* e FGSM.

A diferença no desempenho entre os dois ataques pode ser explicada pelo fato de que o ataque FGSM utiliza o gradiente da função de perda para encontrar a direção de perturbação, permitindo que ele gere perturbações mais significativas. Por outro lado, o ataque *Naive* aplica perturbações uniformes a todos os atributos, o que pode não ser tão eficaz para enganar o modelo.

4.4 Considerando o resultado dos ataques no modelo de detecção de anomalias, qual a diferença no impacto do ataque Naive contra um modelo de classificação (MLP)?

Os resultados obtidos mostram que o modelo de classificação baseado em MLP, sem ataques, apresenta um desempenho melhor em comparação ao modelo de detecção de anomalias. No entanto, quando submetido ao ataque *Naive*, o modelo em questão teve um aumento máximo de 605,08% na taxa de falsos positivos enquanto o outro apresentou um aumento de apenas 85%. Isso indica que, embora o modelo de classificação tenha um desempenho melhor no geral, ele é muito mais vulnerável a ataques adversários. É importante destacar que a metodologia utilizada para criar exemplos adversários com o ataque *Naive* é uma abordagem simplificada e não leva em consideração as particularidades do

modelo. Dessa forma, é possível que um ataque mais elaborado tenha um impacto ainda maior.

Tabela 9 – Média das métricas aplicando o ataque *Naive* para gerar falsos positivos no Modelo de classificação.

ϵ	**MCC**	**F1-score**	**Precisão**	**Revocação**	**TFN**	**TFP**
Sem ataque	0,9533	0,9649	0,9821	0,9482	0,0517	0,0059
0,01	0,9533	0,9649	0,9821	0,9482	0,0517	0,0059
0,05	0,9533	0,9649	0,9821	0,9482	0,0517	0,0059
0,1	0,9533	0,9649	0,9821	0,9482	0,0517	0,0059
0,2	0,9304	0,9482	0,9482	0,9482	0,0517	0,0178
0,3	0,9304	0,9482	0,9482	0,9482	0,0517	0,0178
0,4	0,8875	0,9166	0,8870	0,9482	0,0517	0,0416
0,5	0,8875	0,9166	0,8870	0,9482	0,0517	0,0416

No segundo caso, a Tabela 10 apresenta os resultados do ataque *Naive* visando gerar falsos negativos para o modelo de classificação. Nesse caso, o modelo de classificação apresentou uma aumento muito maior na taxa de falsos negativos, de 1100%, quando comparado com o outro modelo, que obteve um aumento de 66%.

Tabela 10 – Média das métricas aplicando o ataque *Naive* para gerar falsos negativos no Modelo de classificação.

ϵ	**MCC**	**F1-score**	**Precisão**	**Revocação**	**TFN**	**TFP**
Sem ataque	0,9533	0,9649	0,9821	0,9482	0,0517	0,0059
0,01	0,9416	0,9557	0,9818	0,9310	0,0689	0,0059
0,05	0,9181	0,9369	0,9811	0,8965	0,1034	0,0059
0,1	0,9063	0,9272	0,9807	0,8793	0,1206	0,0059
0,2	0,8708	0,8971	0,9795	0,8275	0,1724	0,0059
0,3	0,7870	0,8199	0,9761	0,7068	0,2931	0,0059
0,4	0,6878	0,7173	0,9705	0,5689	0,4310	0,0059
0,5	0,5393	0,5432	0,9565	0,3793	0,6206	0,0059

Em resumo, os resultados demonstram que o modelo de classificação baseado em MLP é superior ao modelo LSTM em condições normais, sem a presença de ataques. No entanto, quando submetido a ataques usando a técnica *Naive*, apresentou um impacto mais significativo do que o modelo de detecção de anomalias. É crucial destacar que essa técnica é uma abordagem simplificada que não leva em consideração seus parâmetros, o que sugere que um ataque mais sofisticado pode causar um impacto ainda mais expressivo.

4.5 Considerando o resultado dos ataques no modelo de detecção de anomalias, qual a diferença no impacto do ataque FGSM contra um modelo de classificação (MLP)?

Com base nos resultados apresentados na Tabela 11, pode-se notar que o modelo de classificação foi significativamente mais afetado pelo ataque FGSM em comparação com o modelo de detecção de anomalias. O ataque FGSM é mais sofisticado, portanto, espera-se que seja mais efetivo do que o ataque *Naive*.

Ao comparar os dois modelos, pode-se observar que o modelo de classificação (MLP) foi mais afetado pelo ataque FGSM, apresentando um aumento exorbitante de 15.940% na taxa de falsos positivos. Em contrapartida, o modelo de detecção de anomalias (LSTM) teve um aumento considerável de 757% na taxa de falsos positivos sob o mesmo ataque. Esses resultados sugerem que o modelo MLP é mais vulnerável aos ataques do que o modelo LSTM.

Tabela 11 – Média das métricas aplicando o ataque FGSM para gerar falsos positivos no Modelo de classificação.

ϵ	**MCC**	**F1-score**	**Precisão**	**Revocação**	**TFN**	**TFP**
Sem ataque	0,9533	0,9649	0,9821	0,9482	0,0517	0,0059
0,01	0,9533	0,9649	0,9821	0,9482	0,0517	0,0059
0,05	0,8978	0,9243	0,9016	0,9482	0,0517	0,0357
0,1	0,8773	0,9090	0,8730	0,9482	0,0517	0,0476
0,2	0,8386	0,8799	0,8208	0,9482	0,0517	0,0714
0,3	0,5425	0,6547	0,5	0,9482	0,0517	0,3273
0,4	0,3018	0,5	0,3395	0,9482	0,0517	0,6369
0,5	0,0035	0,4044	0,2570	0,9482	0,0517	0,9464

Já na segunda parte do questionamento, no qual o objetivo consiste em aumentar a taxa de falsos negativos, os resultados podem ser observados na Tabela 12. Neste cenário, o modelo de classificação apresentou um aumento máximo de 1734% na taxa de falsos negativos, enquanto o modelo de detecção de anomalias teve um aumento de apenas 116%. Isso se deve ao fato de que o processo de geração de exemplos adversários para o modelo de detecção de anomalias é baseado na distância euclidiana entre a predição e o valor real, tornando mais difícil gerar um ataque que diminua essa distância. Já no modelo de classificação, a geração de falsos negativos e falsos positivos é semelhante, bastando inverter a classificação conforme a direção do gradiente descendente da função de perda.

Tabela 12 – Média das métricas aplicando o ataque FGSM para gerar falsos negativos no Modelo de classificação.

ϵ	**MCC**	**F1-score**	**Precisão**	**Revocação**	**TFN**	**TFP**
Sem ataque	0,9533	0,9649	0,9821	0,9482	0,0517	0,0059
0,01	0,9416	0,9557	0,9818	0,9310	0,0689	0,0059
0,05	0,6750	0,7032	0,9696	0,5517	0,4482	0,0059
0,1	0,5247	0,525	0,9545	0,3620	0,6379	0,0059
0,2	0,2180	0,1562	0,8333	0,0862	0,9137	0,0059
0,3	0,1516	0,0967	0,75	0,0517	0,9482	0,0059
0,4	0,1516	0,0967	0,75	0,0517	0,9482	0,0059
0,5	0,1516	0,0967	0,75	0,0517	0,9482	0,0059

É importante ressaltar que a diferença na eficácia do ataque FGSM em ambos os modelos pode ser atribuída às diferenças fundamentais entre a natureza dos modelos. Enquanto o modelo de classificação é projetado para categorizar e rotular dados em classes distintas, o outro modelo é projetado para fazer previsões numéricas em um conjunto contínuo de valores. Portanto, a partir desses resultados, pode-se concluir que o ataque FGSM é mais eficaz contra o modelo de classificação.

5 CONCLUSÃO

Este trabalho teve como objetivo destacar a importância dos algoritmos de predição de séries temporais e sua interação com os algoritmos de Aprendizado de Máquina, especialmente no que se refere à detecção de anomalias em séries temporais. Com base nessa relação, foram analisados os impactos dos ataques adversários de Aprendizado de Máquina contra detectores de anomalias em séries temporais em diferentes cenários. Em vista da aplicabilidade dos algoritmos envolvidos neste trabalho, é de extrema importância que estudos nessa área sejam realizados.

Para obter resultados e realizar análises sobre o tema proposto, foram utilizados dados de um sistema ciberfísico de circulação de água, que contém informações de 8 sensores que realizam leituras a cada segundo, formando uma série temporal, que é rotulada como normal ou anômala para cada observação no tempo. Para lidar com esses dados, foram implementados dois modelos de Aprendizado de Máquina, sendo um modelo de detecção de anomalias e um modelo de classificação. O modelo de detecção realiza predições da série temporal e, a partir dessas predições, é possível calcular o resíduo entre o dado real e o previsto e compará-lo com um limiar, onde os dados que geram um resíduo maior que o limiar são tidos como anomalias. Já o segundo, sendo esse um modelo de classificação, foi criado pelo atacante e será utilizado como base para comparação dos impactos dos ataques em ambos os modelos.

Os resultados dos ataques *Naive* e FGSM revelaram que é mais fácil gerar falsos positivos do que falsos negativos no modelo de detecção de anomalias. Isso se deve ao fato de que a detecção de anomalias leva em consideração o resíduo entre a predição do modelo e o valor real obtido do sistema de água. Dessa forma, reduzir esse resíduo por meio dos ataques utilizados não foi suficiente para causar um grande impacto. Nesse cenário, o ataque FGSM resultou em um aumento máximo de 757% na taxa de falsos positivos e 116% na taxa de falsos negativos.

No caso do modelo de classificação, ambos os ataques demonstraram ter facilidade em gerar falsos positivos e falsos negativos. Isso ocorre porque, em ambas as situações, é suficiente inverter a classificação com base no gradiente descendente. Os resultados dos ataques contra o modelo de classificação foram bastante significativos, com um aumento máximo de 15.940% na taxa de falsos positivos utilizando o ataque FGSM e 1.734% na taxa de falsos negativos utilizando o mesmo ataque. Esses números mostram que o modelo de classificação é muito mais suscetível a ataques adversários do que o modelo de detecção de anomalias.

Ao comparar os resultados antes e após o ataque, observou-se que, embora o mo-

delo de detecção de anomalias apresentasse resultados inferiores quando não submetido a ataques, o modelo de classificação é mais prejudicado, tanto para geração de falsos positivos quanto de falsos negativos. Isso sugere que o primeiro modelo é mais robusto contra esses tipos de ataques. Além disso, o ataque FGSM obteve melhores resultados em ambos os modelos em comparação com o ataque *Naive*, por ser mais elaborado. É importante destacar que, mesmo com a utilização de ataques mais eficientes, ambos os modelos ainda são vulneráveis a exemplos adversários. Portanto, espera-se desenvolver em trabalhos futuros técnicas mais avançadas de detecção de anomalias, bem como métodos para distinguir exemplos adversários de dados normais, ataques mais elaborados e mecanismos de defesa para o modelo.

REFERÊNCIAS

[1] GISS Surface Temperature Analysis (GISTEMP v4). <https://data.giss.nasa.gov/gistemp/>. [Accessado em: 06/09/2022].

[2] HANSEN, J. E. et al. A closer look at united states and global surface temperature change. *J. Geophys. Res.*, v. 106, 2001.

[3] GOODFELLOW, I. J.; SHLENS, J.; SZEGEDY, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[4] ALGULIYEV, R.; IMAMVERDIYEV, Y.; SUKHOSTAT, L. Cyber-physical systems and their security issues. *Computers in Industry*, v. 100, 2018. ISSN 0166-3615. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0166361517304244>.

[5] HASTIE, T. et al. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: Springer, 2009. v. 2.

[6] KIM, K.-D.; KUMAR, P. R. Cyber–physical systems: A perspective at the centennial. *Proceedings of the IEEE*, v. 100, n. Special Centennial Issue, 2012.

[7] FAWAZ, H. I. et al. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, Springer Science and Business Media LLC, v. 33, n. 4, mar 2019. Disponível em: <https://doi.org/10.1007%2Fs10618-019-00619-1>.

[8] CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 41, n. 3, jul 2009. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/1541880.1541882>.

[9] CARLINI, N.; WAGNER, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In: *Proceedings of the 10th ACM workshop on artificial intelligence and security*. [S.l.: s.n.], 2017.

[10] HUANG, L. et al. Adversarial machine learning. In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. [S.l.: s.n.], 2011.

[11] ESLING, P.; AGON, C. Time-series data mining. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 45, n. 1, dec 2012. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/2379776.2379788>.

[12] DURBIN, J.; KOOPMAN, S. J. *Time series analysis by state space methods*. [S.l.]: OUP Oxford, 2012. v. 38.

[13] BROCKWELL, P. J.; DAVIS, R. A. *Introduction to time series and forecasting*. [S.l.]: Springer, 2002.

[14] FU, T. chung. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, v. 24, n. 1, 2011. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197610001727>.

[15] BOX, G. E. et al. *Time series analysis: forecasting and control*. [S.l.]: John Wiley & Sons, 2015.

[16] HYNDMAN, R. J.; ATHANASOPOULOS, G. *Forecasting: principles and practice*. [S.l.]: OTexts, 2018.

[17] MONTGOMERY, D. C.; JENNINGS, C. L.; KULAHCI, M. *Introduction to time series analysis and forecasting*. [S.l.]: John Wiley & Sons, 2015.

[18] SHIBATA, R. Selection of the order of an autoregressive model by akaike's information criterion. *Biometrika*, Oxford University Press, v. 63, 1976.

[19] HYNDMAN, R. et al. *Forecasting with exponential smoothing: the state space approach*. [S.l.]: Springer Science & Business Media, 2008.

[20] MITCHELL, T. M.; MITCHELL, T. M. *Machine learning*. [S.l.]: McGraw-hill New York, 1997. v. 1.

[21] MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole, v. 1, n. 1, 2003.

[22] VAPNIK, V. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, v. 10, n. 5, p. 988–999, 1999.

[23] DOMINGOS, P. A few useful things to know about machine learning. *Communications of the ACM*, ACM New York, NY, USA, v. 55, n. 10, p. 78–87, 2012.

[24] BISHOP, C. M.; NASRABADI, N. M. *Pattern recognition and machine learning*. [S.l.]: Springer, 2006. v. 4.

[25] CHAPELLE, O.; SCHOLKOPF, B.; ZIEN, A. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, IEEE, v. 20, n. 3, 2009.

[26] ZHOU, K.; QIAO, Y.; XIANG, T. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2018. v. 32, n. 1.

[27] KAUFMAN, L.; ROUSSEEUW, P. J. *Finding groups in data: an introduction to cluster analysis*. [S.l.]: John Wiley & Sons, 2009.

[28] GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *Journal of machine learning research*, v. 3, n. Mar, 2003.

[29] SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018.

[30] KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, v. 4, 1996.

[31] PASCANU, R. et al. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[32] MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

[33] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

[34] RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[35] GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. [S.l.], 2010. p. 249–256.

[36] HAWKINS, D. M. *Identification of outliers*. [S.l.]: Springer, 1980. v. 11.

[37] CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 41, n. 3, 2009.

[38] GUPTA, M. et al. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and data Engineering*, IEEE, v. 26, n. 9, 2013.

[39] NIELSEN, A. *Practical time series analysis: Prediction with statistics and machine learning*. [S.l.]: O'Reilly Media, 2019.

[40] PATCHA, A.; PARK, J.-M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, Elsevier, v. 51, n. 12, 2007.

[41] HODGE, V.; AUSTIN, J. A survey of outlier detection methodologies. *Artificial intelligence review*, Springer, v. 22, n. 2, p. 85–126, 2004.

[42] CHALAPATHY, R.; CHAWLA, S. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[43] PAPERNOT, N. et al. The limitations of deep learning in adversarial settings. In: IEEE. *2016 IEEE European symposium on security and privacy (EuroS&P)*. [S.l.], 2016.

[44] SZEGEDY, C. et al. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[45] BIGGIO, B.; NELSON, B.; LASKOV, P. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[46] MIYATO, T.; DAI, A. M.; GOODFELLOW, I. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.

[47] BIGGIO, B. et al. Evasion attacks against machine learning at test time. In: SPRINGER. *Joint European conference on machine learning and knowledge discovery in databases*. [S.l.], 2013.

[48] KURAKIN, A.; GOODFELLOW, I.; BENGIO, S. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[49] CARLINI, N.; WAGNER, D. Towards evaluating the robustness of neural networks. In: IEEE. *2017 ieee symposium on security and privacy (sp)*. [S.l.], 2017. p. 39–57.

[50] JIA, Y. et al. Adversarial attacks and mitigation for anomaly detectors of cyber-physical systems. *International Journal of Critical Infrastructure Protection*, Elsevier, v. 34, p. 100452, 2021.

[51] KATSER, I. D.; KOZITSIN, V. O. *Skoltech Anomaly Benchmark (SKAB)*. [S.l.]: Kaggle, 2020. <https://www.kaggle.com/dsv/1693952>.