

**DENTIFICAÇÃO AUTOMÁTICA DE PERDAS NÃO
TÉCNICAS EM SISTEMAS DE DISTRIBUIÇÃO ELÉTRICA
UTILIZANDO REDES BIDIRECIONAL GATED RECURRENT
UNIT**

Por

MÔNICA LUÍZA DOEGE

Trabalho de Conclusão de Curso aprovado para
obtenção dos créditos na disciplina de Trabalho
de Conclusão de Curso II pela banca
examinadora formada por:

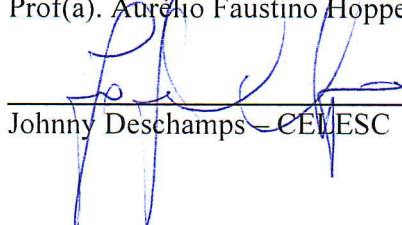
Presidente:


Prof(a). Andreza Sartori – Orientador(a), FURB

Membro:


Prof(a). Aurélio Faustino Hoppe – FURB

Membro:


Johnny Deschamps – CEE/ESC

Blumenau, 01 de julho de 2024

IDENTIFICAÇÃO AUTOMÁTICA DE PERDAS NÃO TÉCNICAS EM SISTEMAS DE DISTRIBUIÇÃO ELÉTRICA UTILIZANDO REDES BIDIRECIONAL GATED RECURRENT UNIT

Mônica Luíza Doege, Andreza Sartori – Orientadora

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

mdaege@furb.br, asartori@furb.br

Resumo: A concentração de fraudes no sistema de distribuição de energia no Brasil resulta em grandes perdas financeiras, além de econômicas e sociais. Visto isso, este trabalho consiste no desenvolvimento de um protótipo que utiliza Redes Neurais Artificiais Recorrentes Bidirecionais para a detecção de Perdas Não Técnicas. Este trabalho tem como motivação auxiliar as concessionárias de energia elétrica, que atualmente fazem a detecção de fraudes de forma manual. Para isto, foram realizados testes com diversos algoritmos de Aprendizado de Máquina, dentre eles, de Redes Neurais Artificiais Profundas. A linguagem de programação Python foi utilizada para a implementação, além de bibliotecas como Pandas, Scikit-learn e Keras. Dentre os modelos testados, a Rede Bidirecional Gated Recurrent Unit apresentou uma acurácia de 0.78, uma perda de 0.17, indicando um bom desempenho geral.

Palavras-chave: Detecção de Fraudes. Distribuição Elétrica. Redes Neurais Artificiais.

1 INTRODUÇÃO

O sistema de geração e distribuição de energia elétrica no Brasil enfrenta problemas de diversas naturezas, tais como: sociais, econômicos, políticos, ambientais e tecnológicos. De fato, a crise de energia elétrica no Brasil tomou força nos últimos anos e, como ponto principal, tem-se a forte dependência de hidrelétricas e a escassez de chuvas no país. Com essa falta de planejamento e organização geral quem mais sofre é a população. Com isso, como resultados surgem a possibilidade de racionamento e apagões, bem como o aumento significativo das tarifas de energia elétrica (Borges, 2021). Segundo Corsini (2022), os dados da Associação Brasileira dos Comercializadores de Energia (Abraceel) apontam que a tarifa de energia elétrica residencial teve um crescimento médio anual de 16,3% entre os anos de 2015 e 2021.

A crise hidrelétrica se tornou um adicional em problemas já presentes no sistema, como as perdas de energia. As perdas de energia elétrica se referem a energia gerada que é passada pelo sistema de distribuição, mas não chega a ser comercializada. Essas perdas podem ser divididas em duas categorias: Perdas Técnicas (PT) e Perdas Não Técnicas (PNT) (ANEEL, 2023). Segundo Piotrowski et al. (2021), as Perdas Técnicas (PT), ocorrem durante o processo de transporte e transformação de energia elétrica em energia térmica, mais conhecido como Efeito Joule. Já as Perdas Não Técnicas (PNT) são resultado da falta de faturamento da energia elétrica distribuída. Entre as causas mais comuns, estão presentes: o furto de energia, fraude na medição ou fornecimento de energia, falha ou falta na conferência dos medidores e erro no faturamento.

De acordo com dados da ANEEL (2023), em 2022 a taxa de perdas não técnicas no Brasil ficou por volta de 14%, dando um prejuízo de mais de R\$ 6 bilhões. Conforme levantamento realizado pela Associação Brasileira de Distribuidores de Energia Elétrica (ABRADEE), a energia perdida por furto ou desvio em 2022 seria suficiente para abastecer os estados de Santa Catarina, Paraná, Mato Grosso do Sul e Espírito do Santo. O valor dessas perdas é pago tanto pelas distribuidoras de energia quanto pelo consumidor final, com uma média de 10% desse custo sendo repassado para o consumidor. O cálculo das PNT é determinado pela diferença entre a energia entregue pela distribuidora de energia e a quantidade de energia contabilizada e paga pelos consumidores. Com essa diferença, é realizado o rateio do valor a pagar pelo consumidor e pela distribuidora. As PNT também impactam a possibilidade de crescimento e melhorias dentro das distribuidoras, na qual o valor pago poderia ser revertido em geração de energia renovável. Além de prejuízo, esse tipo de prática traz riscos a população, como aumento de possibilidade de curto-circuito, incêndio e choques elétricos (CEMIG, 2023).

As informações sobre o processo de identificação das PNT e sobre o sistema de distribuição de energia elétrica foram fornecidas pelo gerente da concessionária de energia elétrica que disponibilizou os dados para este estudo. Segundo ele, atualmente os dados de consumo são registrados pelos medidores presentes em cada UC. A leitura destes dados é feita pelos “leiturista”, que passam em cada casa registrando manualmente os consumos. As irregularidades são encontradas por meio de fiscalizações nas UCs, realizadas por inspeções de rotina. Essas fiscalizações de rotina ocorrem em dois casos. O primeiro é via análise manual pelos técnicos; caso nessa análise for encontrada alguma suspeita, é realizada a fiscalização no local. Nesta análise, observa-se principalmente a curva de consumo e o nível de consumo dado o valor da Carga Declarada pela UC. Qualquer incoerência nos dados gera uma fiscalização no local. O segundo caso

ocorre por meio de denúncias pelos canais de Ouvidoria da empresa, que também resultam em fiscalização no local. Esse processo de análise é demorado e custoso para a distribuidora de energia, e está sujeito a falhas. Além disso, parte do preço é pago pela população, causando aumento nas tarifas de energia elétrica.

Neste contexto, esse trabalho tem como objetivo disponibilizar uma solução automatizada para a detecção de PNT, utilizando uma base de dados fornecida pela distribuidora de energia por meio de Redes Neurais Artificiais Recorrentes. Os objetivos específicos são: (i) identificar os padrões que definem Perdas Não Técnicas; (ii) validar a acurácia do modelo de Aprendizado de Máquina; (iii) utilizar dados reais para o treinamento, validação e testes do modelo gerado.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção aborda os assuntos e técnicas utilizadas neste trabalho. Na seção 2.1 discorre sobre as perdas de energia elétrica. Na seção 2.2 são abordados os conceitos fundamentais das Redes Neurais Recorrentes e o funcionamento das Redes Neurais Artificiais do tipo bidirecional para detecção de anomalias. A seção 2.3 aborda o processo atual para detecção de PNT. Por fim, a seção 2.4 apresenta os trabalhos correlatos.

2.1 ENERGIA ELÉTRICA E PERDAS DE ENERGIA ELÉTRICA

De acordo com Paulo, Villanueva e Braz (2020), Unidade Consumidora (UC) é definida com o conjunto de instalações e equipamentos para o recebimento de energia em um só ponto de entrega. Cada consumidor é responsável pelo requerimento de uma nova ligação ao sistema elétrico, bem como fornecer as informações necessárias para a instalação, incluindo a declaração de carga. A declaração de carga ou Carga Declarada é a estimativa máxima que a UC irá utilizar de energia. O consumidor é responsável também pelo zelo de todos os equipamentos mantidos sob lacre da UC, incluindo o relógio. Alterações na rede instalada são definidas como fraude, e estas fraudes podem resultar em Perdas Não Técnicas.

Segundo a ANNEL (2023), no sistema de distribuição existem 2 tipos de perda de energia: Perdas Técnicas (PT) e Perdas Não Técnicas (PNT). As PT estão relacionadas à transformação de energia elétrica em energia térmica nos condutores - mais conhecido como o efeito joule, perdas nos núcleos dos transformadores, entre outros. As PT são inevitáveis em qualquer sistema de distribuição. Essas variam conforme as características das redes de cada área de concessão, sendo reconhecidas, nas tarifas pela ANEEL, apenas os níveis eficientes.

De acordo com a Piotrowski *et al.* (2021), Perdas Não Técnicas (PNT) são descritas como a falta de faturamento na distribuição de energia elétrica. As PNT têm inúmeras situações, mas se resumem em fraude. Segundo dados da ANNEL (2023), em 2022 as PNT somam mais de R\$6 bilhões. O cálculo das PNT é feito com a diferença entre a energia entregue pela distribuidora de energia e a quantidade de energia contabilizada e paga pelos consumidores. Com essa diferença é feito o rateio do valor a pagar pelo consumidor e pela distribuidora. A Figura 1 demonstra o processo de geração de distribuição de energia elétrica, que se inicia na geração de energia elétrica nas usinas. Então, passa pela transmissão e distribuição. Nestes dois últimos passos é onde ocorre as PNT e PT. Também é demonstrado a forma de cálculo das PNT, no qual é feito a subtração das PT sob o valor de perdas total da rede de distribuição.

Figura 1 - Demonstração de PT e PNT



Fonte: CERBRANORTE (2024).

2.2 APRENDIZADO DE MÁQUINA E REDES NEURAIS ARTIFICIAIS

Aprendizado de Máquina (AM) é uma área da Inteligência Artificial (IA) que fornece a capacidade de aprendizagem para computadores sem ser explicitamente programado (SAMUEL, 1959). O AM pode ser dividido em: Aprendizado Supervisionado, Aprendizado Não Supervisionado e Aprendizado por Reforço. Segundo Goodfellow, Bengio e Courville (2016), Aprendizado Não Supervisionado consiste em observar o conjunto de dados e aprender as propriedades mais significativas da distribuição do conjunto. Aprendizado Supervisionado envolve a observação do conjunto de dados, tal que cada item do conjunto tenha um valor ou vetor associado, e em seguida a máquina aprende a prever a partir do conjunto de dados o valor ou vetor associado.

A estrutura das Redes Neurais Artificiais (RNA) é formada por diversas unidades de processamento, chamadas neurônios artificiais, que se encontram interligadas. Essa rede de neurônios artificiais se comunica através de sinais e são capazes de representar comportamentos complexos (SOARES, 2021). Segundo Ludemir (2021), a implementação mais simples de RNA é uma rede Perceptron. O algoritmo de aprendizado do Perceptron utiliza a correção de erros como base, isto é, a diferença entre a resposta desejada e a resposta da rede. Na fase de treinamento do Perceptron os exemplos rotulados são apresentados ao algoritmo. Os parâmetros da rede (pesos) são modificados a cada apresentação de um novo exemplo à rede. Depois do ajuste dos parâmetros, na fase de teste, o sistema é avaliado. Segundo Goodfellow, Bengio e Courville (2016), as Redes Neurais Profundas do tipo *feedforward* propõem um modelo no qual a informação flui através da função que está sendo avaliada a partir do valor de x , ou seja, a informação flui somente uma vez, não havendo conexões de feedback.

2.2.1 Redes Neurais Recorrentes

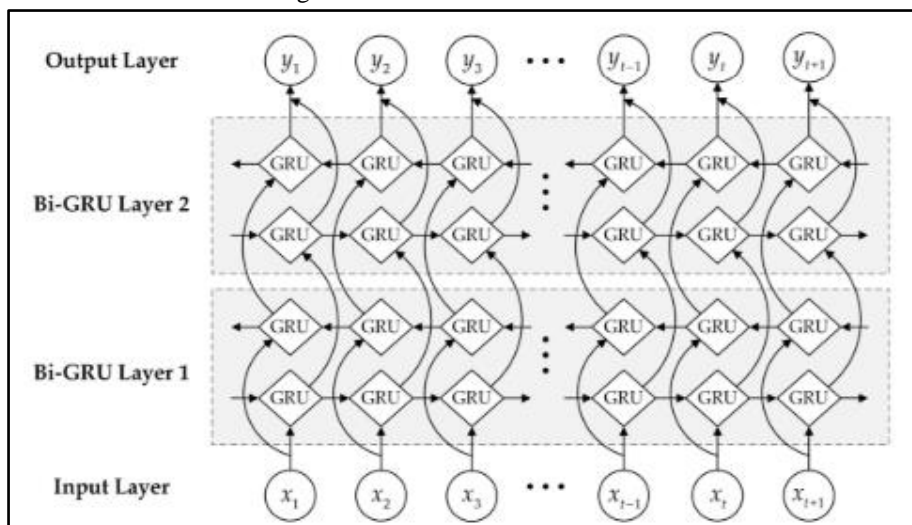
Dentre as Redes Neurais Profundas, estão as Redes Neurais Recorrentes (RNN), os quais foram inspiradas pelas conexões cíclicas dos neurônios no cérebro, por isso utilizam funções de realimentação iterativas para armazenar informações. As RNNs foram um importante foco de pesquisa e desenvolvimento durante a década de 1990, sendo projetadas para aprender padrões sequenciais ou variantes no tempo por intermédio da recorrência do sinal alimentado. Esse tipo de rede preenche a lacuna deixada pela rede neural de arquitetura *perceptron* multicamadas (*Multi Layer Perceptron* - MLP), que não possui capacidade de aprender esse tipo específico de classe por ser do tipo *feedforward*. Uma MLP do tipo *feedforward* é uma rede que possui camadas que se conectam, passando os dados em uma direção. A grande diferença entre uma RNN e uma MLP *feedforward* é que a primeira possui, pelo menos, um feedback loop, ou seja, a saída de um neurônio é utilizada como sua entrada em algum instante futuro (SOARES, 2021).

Um problema muito comum nas RNNs é o problema da dissipação do gradiente (*vanishing gradient problem*). Este problema acontece quando a sensibilidade da rede a erros diminui gradativamente ao longo do tempo, à medida que novas entradas substituem as ativações das camadas ocultas. Consequentemente, a rede ‘esquece’ as informações das primeiras entradas, limitando sua capacidade de aprendizado. Para resolver esse problema é introduzida a Rede Neural *Gated Recurrent Unit* (GRU) (SOARES, 2021).

A arquitetura GRU utiliza a recorrência para armazenar e acessar informações ao longo do tempo. Entretanto, o seu desempenho prático pode não corresponder totalmente, devido à limitação de acessar apenas informações passadas. A Rede Neural Bidirecional GRU inclui uma camada adicional que processa a sequência de dados na direção oposta, permitindo que a rede explore tanto o passado quanto o futuro. Isso é alcançado através de duas camadas ocultas conectadas à mesma camada de saída. Este mecanismo possibilita à rede explorar informações tanto do passado quanto do futuro, melhorando assim a capacidade preditiva e a eficiência do processo de aprendizado (SOARES, 2021).

A Figura 2 ilustra a estrutura de uma rede com duas camadas de Bidirecional *Gated Recurrent Unit* (Bi-GRU). A entrada dos dados é representada pela “*Input Layer*” e cada registro é representado pelo valor de x . Esta estrutura apresenta duas camadas de Bi-GRU, chamadas de “*Bi-GRU layer 1*” e “*Bi-GRU layer 2*”. Cada camada de Bi-GRU representa vários “nós” da arquitetura GRU, e cada nó é ligado ao próximo de maneira bidirecional. A saída dos dados é representada pela camada chamada “*Output Layer*”, e cada predição é representada pelo valor de y .

Figura 2 - Estrutura da rede BiGRU



Fonte: Li *et al.* (2020).

2.3 TRABALHOS CORRELATOS

Nesta seção serão apresentados três trabalhos correlatos. O trabalho de Niu e Zhang (2021), apresentado no Quadro 1 relata um modelo utilizando rede neural convolucional de unidade recorrente fechada juntamente com o algoritmo de *K-means* para detectar roubo de energia elétrica. No Quadro 2, o trabalho de Markovska *et al.* (2023), apresenta um modelo utilizando rede neural convolucional temporal com *Self-Attention* para a detecção de roubo de energia. Por fim, Korba e Karabadi (2019) implementaram uma solução para fraudes de energia elétrica em uma infraestrutura com medição avançada, utilizando máquina de vetores de suporte, apresentado no Quadro 3.

Quadro 1 – A Data-Driven Method for Electricity Theft Detection Combining CONVGRU and K-Means Clustering

Referência	Niu e Zhang (2021)
Objetivos	Deteção de roubo de energia elétrica utilizando um modelo de redes neurais
Principais funcionalidades	Aplicar um modelo de rede neural chamado <i>Convolutional Gated Recurrent Unit</i> (ConvGRU) para a deteção de roubo de energia elétrica. A base de dados utilizada contém registros de mil unidades consumidoras durante um período de 504 dias, 20% desses dados foram alterados para representar os consumidores irregulares
Ferramentas de desenvolvimento	Para a implementação foi utilizada a técnica de encoder-decoder, onde a CNN é a parte codificadora e a GRU é a parte decodificadora. Como técnica de pré-processamento, foi utilizado o K-means.
Resultados e conclusões	O modelo atingiu uma acurácia de 98.9%, com taxa de falso negativo igual a 4% e a taxa de positivo verdadeiro de 96%.

Fonte: elaborado pela autora.

O trabalho de Niu e Zhang (2021) está relacionado a este estudo devido ao objetivo de detectar roubo de energia elétrica, o que é a principal causa de Perdas Não Técnicas no sistema de distribuição. Os registros irregulares foram criados a partir de registros regulares, mudando apenas a curva de consumo. A técnica utilizada para essa criação de novos registros foi baseada no “*Hadamard Product*”. A técnica utilizada por Niu e Zhang (2021) é utilizada como base para gerar novos registros sintéticos neste trabalho. Assim como este trabalho, o estudo de Niu e Zhang (2021) utilizou técnicas de aprendizado de máquina do tipo supervisionado, especificamente uma junção de Redes Neurais Convolucionais e *Gated Recurrent Unit*.

Quadro 2 – Electricity Theft Detection Based on Temporal Convolutional Networks with Self-Attention

Referência	Markovska <i>et al.</i> (2023)
Objetivos	Deteção de roubo de energia elétrica utilizando TCN e <i>Self-Attention</i>
Principais funcionalidades	Implementar um modelo de rede neural utilizando camadas de <i>Temporal Convolutional Network</i> (TCN) e <i>Self-Attention</i> . O conjunto de dados possui 42.372 registros, de 2014 a 2016. Cerca de 8% desses registros eram irregulares.
Ferramentas de desenvolvimento	Para o preparo dos dados foram utilizadas diversas técnicas, tais como: Min-Max, interpolação de Hermite e balanceamento da base de dados. Para a criação do modelo, foi adicionado 3 camadas de TCN e uma de <i>Self-Attention</i> .
Resultados e conclusões	O modelo proposto alcançou uma acurácia de 94,66%, F1-Score de 95% e AUC de 0.946.

Fonte: elaborado pela autora.

O trabalho de Markovska *et al.* (2023) se assemelha a este estudo em diversos aspectos, como o objetivo em comum e a implementação de modelo semelhante. No trabalho de Markovska *et al.* (2023) foram utilizadas algumas técnicas de preparação dos dados, tais como: interpolação de Hermite e técnica de Min-Max. Diferente deste estudo, o trabalho de Markovska *et al.* (2023) apresenta uma grande quantidade de dados, que contém 42.372 registros/unidades consumidoras, de janeiro de 2014 a outubro de 2016. Destes, cerca de 8% são consumidores irregulares.

Quadro 3 – Smart Grid Energy Fraud Detection Using SVM

Referência	Korba e Karabadji (2019)
Objetivos	Detecção de fraude na Infraestrutura de Medição Avançada (<i>Advanced Metering Infrastructure</i> - AMI)
Principais funcionalidades	Aplicar o modelo de SVM para a detecção de fraude no sistema de distribuição de energia elétrica, por meio dos dados recebidos pela Infraestrutura de Medição Avançada (AMI). A base de dados utilizada possui 5 mil registros.
Ferramentas de desenvolvimento	O modelo proposto tem 4 etapas: preparação da base de dados, seleção de atributos, adição de registros irregulares, construção do modelo SVM para a classificação.
Resultados e conclusões	O modelo obteve uma acurácia de 91,062% e uma taxa de falso positivo de 9%.

Fonte: elaborado pela autora.

O trabalho de Korba e Karabadji (2019) tem como objetivo detectar fraude de energia elétrica em um sistema de distribuição que possui AMI, diferenciando-se deste estudo, que utiliza uma base de dados com valores coletados manualmente/presencialmente. A *Advanced Metering Infrastructure* (AMI) é um sistema que faz a medição de cada unidade consumidora de forma automática, sem a necessidade do registro feito pelos técnicos. Além disso, o foco específico do trabalho de Korba e Karabadji (2019) está em três casos de fraude: (i) quando a unidade consumidora diminui o valor da medição em uma certa porcentagem, (ii) quando a unidade consumidora zera o valor da medição durante um período específico, (iii) quando a unidade consumidora diminui o valor da medição por certa porcentagem apenas durante um período determinado. Um ponto de semelhança entre os estudos é a utilização de algoritmos de aprendizado de máquina supervisionado.

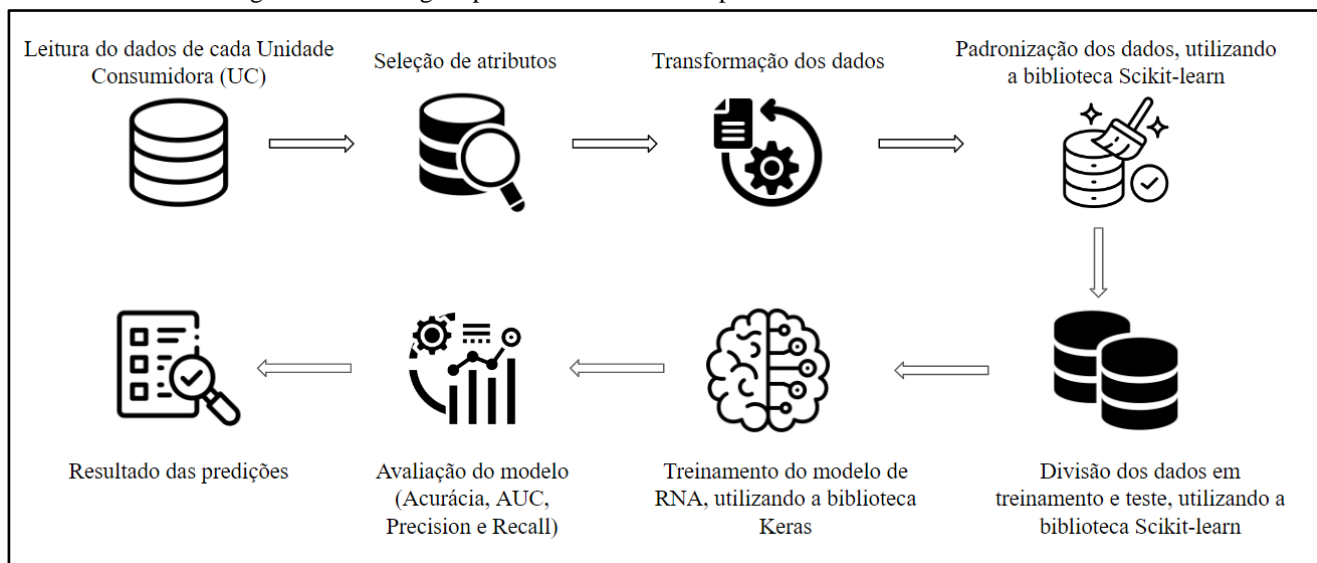
3 DESCRIÇÃO DO MODELO DE APRENDIZADO DE MÁQUINA

Nesta seção serão descritas as etapas do desenvolvimento dos modelos de RNA, bem como os requisitos do protótipo, a especificação e desenvolvimento do protótipo para detecção de Perdas Não Técnicas no sistema de distribuição elétrica. Por fim, serão demonstradas as métricas utilizadas para testar e avaliar cada um dos modelos.

3.1 ESPECIFICAÇÃO

Com o objetivo de desenvolver um modelo de RNA capaz detectar Perdas Não Técnicas no sistema de distribuição elétrica, a fim de auxiliar os técnicos responsáveis pela fiscalização das unidades consumidoras, optou-se pela abordagem ilustrada no diagrama de fluxo da Figura 3. Primeiramente os arquivos da base de dados são carregados, e, em seguida, é realizada uma seleção de atributos. Esta seleção é feita com base nas informações que são utilizadas pelos técnicos para fazer a detecção manual, que englobam os atributos básicos sobre a infraestrutura da UC e o histórico de consumo. Todos os atributos importantes são transferidos para novos arquivos, separados por Unidade Consumidora. Esses dados são convertidos para o formato .CSV, que pode ser interpretado pelos modelos desenvolvidos. Após essa transformação, os dados foram padronizados e divididos aleatoriamente em dois conjuntos de dados (treino e teste) utilizando a biblioteca *Scikit-learn*. Os modelos de RNAs desenvolvidos foram: *Long Short-Term Memory (LSTM)*, *Gated Recurrent Unit (GRU)*, *Deep Neural Network (DNN)*, *Convolutional Gated Recurrent Unit (CNN+GRU)* e *Bidirecional Gated Recurrent Unit (Bi-GRU)*. Após o treinamento de cada modelo, é realizada uma avaliação utilizando as métricas de acurácia, *Precision* e *Recall*. Por fim, os resultados são armazenados para serem utilizados na etapa de predição.

Figura 3 - Abordagem para treinar e validar a performance do modelo de RNA



Fonte: elaborado pela autora.

A partir do fluxo definido na Figura 3, foram estabelecidos os Requisitos Funcionais (RF), apresentados no Quadro 4, e os Requisitos Não Funcionais (RNF) no Quadro 5.

Quadro 4 - Requisitos funcionais

Requisitos funcionais
RF01 - Extrair as características de unidades consumidoras regulares e irregulares
RF02 - Realizar o tratamento dos dados como normalização, limpeza e organização da base de dados
RF03 - Realizar a detecção de possíveis consumidores irregulares utilizando modelos de aprendizado de máquina
RF04 - Exibir o valor da acurácia e perda do modelo proposto

Fonte: elaborado pela autora.

Quadro 5 – Requisitos não funcionais

Requisitos não funcionais
RNF01 - Utilizar a linguagem de programação Python para fazer a implementação do protótipo
RNF02 - Utilizar base de dados cedida pela concessionária de energia elétrica
RNF03 - Utilizar a biblioteca Pandas e Scikit-learn para fazer a leitura, pré-processamento e divisão dos dados
RNF04 - Utilizar a biblioteca Keras para implementar os modelos de RNA
RNF05 - Utilizar o Google Colab para o desenvolvimento da implementação e exibição dos resultados
RNF06 - Utilizar a arquitetura de Rede Bidirecional <i>Gated Recurrent Unit</i> (Bi-GRU) no desenvolvimento dos modelos preditores

Fonte: elaborado pela autora.

3.2 IMPLEMENTAÇÃO

Esta seção apresenta a implementação do protótipo. Para o desenvolvimento do protótipo foi utilizada a linguagem Python juntamente com o framework *Keras* com o Google Colab como ambiente de desenvolvimento. A seção 3.2.1 aborda a construção da base de dados para treinamento e testes. A seção 3.2.2 apresenta as arquiteturas de Redes Neurais Artificiais utilizadas.

3.2.1 Base de Dados de Faturas de Energia Elétrica

O conjunto de dados foi cedido por uma concessionária de energia elétrica, contendo 428 arquivos, onde cada arquivo representa um relatório de uma Unidade Consumidora (UC). Destes 428 arquivos, 246 são registros com irregularidades, 120 são registros regulares, e 61 são registros regulares onde a unidade consumidora também utiliza energia solar. Cada arquivo contém informações básicas sobre a unidade consumidora, como valor da carga declarada e tipo da unidade consumidora. Além dessas informações, o arquivo contém o histórico de consumo e os dados de leitura do relógio. Em conformidade com as leis da Lei Geral de Proteção de Dados Pessoais (LGPD), nenhum dado sensível foi utilizado neste trabalho.

Para a extração dos dados e criação dos arquivos .CSV, foi necessário implementar um algoritmo utilizado a linguagem de programação Python a fim de estruturar a base de dados. Para cada Unidade Consumidora (UC), foi criado um arquivo com 13 colunas, conforme demonstrado na Figura 4. A coluna “id” se refere a um número identificador gerado

para cada UC. A coluna “carga_declarada” se refere a carga de energia declarada para UC do arquivo. A coluna “origem” se refere a origem do dado, na imagem temos o tipo “ANL” que se refere a registro manual da medição. A coluna “data_ref” é composta pelo mês e ano do valor que se refere na coluna “faturado”. A coluna “dias” informa quantos dias tem no mês referido na coluna “data_ref”. A coluna “faturado” informa o valor da energia faturada. A coluna “ftp” se refere ao fator de potência. Valores na coluna “ftp” só são vistos em UCs que possuem medidor eletrônico. A coluna “irregular” informa se, durante o registro manual da medição pelos “leitores”, foi visto irregularidade. A coluna “data_leitura” informa a data que foi realizada a leitura. A coluna “COM/AHR” se refere ao valor da leitura do relógio. A coluna “dias_2” informa a quantidade de dias no mês da data da leitura. A coluna “id_relogio” é um número gerado para cada relógio que a UC teve. Pode-se perceber a mudança do equipamento do relógio após a descoberta de irregularidades, ou quando o relógio apresentou problemas ou precisou ser atualizado. Durante esse processo de transformação, foi verificada a presença de valores nulos (*null*) em colunas importantes, como por exemplo a coluna “carga_declarada”, como demonstrado no quadro em vermelho na Figura 5. Tais relatórios foram manualmente removidos do conjunto de dados devido a importância dessa informação para uma predição correta.

Figura 4 - Exemplo de um arquivo de uma UC

id	carga_declarada	tipo_consumidor	origem	data_ref	dias	faturado	ftp	irregular	data_leitura	CON/AHR	dias_2	id_relogio
0	28000	RESIDEN	ANL	03/2021	30	188	.	0	10/03/2021	53969	30	1
0	28000	RESIDEN	ANL	04/2021	30	154	.	0	09/04/2021	54123	30	1
0	28000	RESIDEN	ANL	05/2021	31	204	.	0	10/05/2021	54327	31	1
0	28000	RESIDEN	ANL	06/2021	30	195	.	0	09/06/2021	54522	30	1
0	28000	RESIDEN	ANL	07/2021	30	216	.	0	09/07/2021	54738	30	1
0	28000	RESIDEN	ANL	08/2021	32	141	.	0	10/08/2021	54879	32	1
0	28000	RESIDEN	ANL	09/2021	30	144	.	0	09/09/2021	55023	30	1
0	28000	RESIDEN	ANL	10/2021	29	197	.	0	08/10/2021	55220	29	1
0	28000	RESIDEN	ANL	11/2021	32	233	.	0	09/11/2021	55453	32	1
0	28000	RESIDEN	ANL	12/2021	30	144	.	0	09/12/2021	55597	30	1
0	28000	RESIDEN	ANL	01/2022	32	201	.	0	10/01/2022	55798	32	1
0	28000	RESIDEN	ANL	02/2022	29	191	.	0	08/02/2022	55989	29	1
0	28000	RESIDEN	ANL	03/2022	30	194	.	0	10/03/2022	56183	30	1
0	28000	RESIDEN	ANL	04/2022	29	131	.	0	08/04/2022	56314	29	1
0	28000	RESIDEN	ANL	05/2022	32	119	.	0	10/05/2022	56433	32	1
0	28000	RESIDEN	ANL	06/2022	30	122	.	0	09/06/2022	56555	30	1
0	28000	RESIDEN	ANL	07/2022	32	137	.	0	11/07/2022	56692	32	1
0	28000	RESIDEN	ANL	08/2022	30	124	.	0	10/08/2022	56816	30	1
0	28000	RESIDEN	ANL	09/2022	30	213	.	0	09/09/2022	57029	30	1
0	28000	RESIDEN	ANL	10/2022	31	199	.	0	10/10/2022	57228	31	1
0	28000	RESIDEN	ANL	11/2022	30	177	.	0	09/11/2022	57405	30	1
0	28000	RESIDEN	ANL	12/2022	30	213	.	0	09/12/2022	57618	30	1
0	28000	RESIDEN	ANL	01/2023	31	248	.	0	09/01/2023	57866	31	1
0	28000	RESIDEN	ANL	02/2023	30	281	99.05	0	30/01/2023	58074	21	1

Fonte: elaborado pela autora.

Figura 5 - Exemplo com "carga_declarada" nulo

id	carga_declarada	tipo_consumidor	origem	data_ref	dias	faturado	ftp	irregular	data_leitura	CON/AHR	dias_2	id_relogio
4	null	RESIDEN	ANL	03/2021	29	188	.	0	18/03/2021	53880	29	1
4	null	RESIDEN	ANL	04/2021	32	152	.	0	19/04/2021	54032	32	1
4	null	RESIDEN	ANL	05/2021	29	88	.	0	18/05/2021	54120	29	1
4	null	RESIDEN	ANL	06/2021	30	55	.	0	17/06/2021	54175	30	1
4	null	RESIDEN	ANL	07/2021	32	42	.	0	19/07/2021	54217	32	1
4	null	RESIDEN	ANL	08/2021	30	58	.	0	18/08/2021	54275	30	1
4	null	RESIDEN	ANL	09/2021	30	61	.	0	17/09/2021	54336	30	1
4	null	RESIDEN	ANL	10/2021	32	43	.	0	19/10/2021	54379	32	1
4	null	RESIDEN	ANL	11/2021	30	187	.	0	18/11/2021	54566	30	1
4	null	RESIDEN	ANL	12/2021	29	196	.	0	17/12/2021	54762	29	1
4	null	RESIDEN	ANL	01/2022	32	241	.	0	18/01/2022	55003	32	1

Fonte: elaborado pela autora.

Para a implementação do modelo da RNA, foi necessário transformar todos os arquivos .CSV em *dataframes*, e, para isso, foi utilizada a biblioteca Pandas. Durante a fase de montagem dos *dataframes*, foi necessário aplicar duas alterações, que são apresentadas respectivamente nas Figura 6 e Figura 7. Quando os valores referentes a coluna “data_ref” se repetem e há valor na coluna “ftp.”, conforme apresentado nos quadros em vermelho na Figura 6, o valor de “faturado” deve ser considerado para aquele mês, e o outro registro do mesmo mês deve ser ignorado. Essa regra foi aplicada ao criar o *dataframe*, removendo as linhas com datas duplicadas que não tinham valor na coluna “Ftp.”. Dado que a coluna “Ftp.” não era mais necessária, foi então removida do *dataframe*.

Figura 6 - Exemplo para remoção do dado com data_ref repetida

id	carga_declarada	tipo_consumidor	origem	data_ref	dias	faturado	ftp	irregular	data_leitura	CON/AHR	dias_2	id_relogio
0	28000	RESIDEN	ANL	02/2023	30	281	99.05	0	30/01/2023	58074	21	1
0	28000	RESIDEN	ANL	02/2023	30	39		0	30/01/2023	0	0	2

Fonte: elaborado pela autora.

Outra coluna que apresenta repetição na base de dados é a “data_leitura”. Para padronizar uma linha por mês, foi considerado apenas o último valor da coluna “CON/AHR” para cada mês, isto é, apenas a última leitura do relógio. No final, cada UC apresentava um registro/linha por mês. Dada a diferença na quantidade de relatórios irregulares e regulares, os relatórios regulares com energia solar foram adicionados à lista de relatórios regulares. A Figura 7 ilustra um exemplo onde essa regra deve ser aplicada, mantendo somente a primeira linha.

Figura 7 - Exemplo para remoção do dado com data_leitura repetida

id	carga_declarada	tipo_consumidor	origem	data_ref	dias	faturado	ftp	irregular	data_leitura	CON/AHR	dias_2	id_relogio
0	28000	RESIDEN	ANL	01/2023	31	248		0	09/01/2023	57866	31	1
0	28000	RESIDEN	ANL	02/2023	30	281	99.05	0	30/01/2023	58074	21	1

Fonte: elaborado pela autora.

Considerando que cada linha do arquivo de uma UC representa um mês, observou-se não ser necessário manter as colunas “dias” e “dias_2”, que indicam a quantidade de dias passados desde a última leitura. Portanto, essas duas colunas foram removidas. Além disso, a coluna “irregular” também foi removida, pois não representa com precisão as linhas irregulares.

Para a padronização de todos os dados do *dataframe*, foi utilizado a técnica *StandardScaler*, disponibilizada pela biblioteca *Sklearn*. A padronização serve para colocar todos os dados de entrada na mesma escala, tal técnica pode ajudar a atingir um melhor treinamento, principalmente nos algoritmos mais sensíveis as diferenças na escala dos dados. Outra observação importante é o período de cada relatório, os quais, a maioria dos relatórios possuem um período de 36-37 linhas/meses. A fim de uniformizar a quantidade de linhas por UC, todos os relatórios com mais de 36 linhas foram reduzidos para 36, sendo removidas as linhas 37 em diante. As UCs com menos de 36 linhas foram removidos. Após a aplicação de todas as alterações, foram criados um total de 295 *dataframes*, dos quais 171 são com registros irregulares e 124 regulares.

Como *ground truth* foi utilizado uma planilha de período de irregularidades disponibilizada pela concessionária de energia. Nesta planilha, apresentada na Figura 8, contém as datas do início e fim das irregularidades de cada UC irregular. Então, para cada *dataframe*, foi criada uma lista com 36 números, sendo estes 1 para irregular e 0 para regular. Para os relatórios regulares, a lista contém 36 números 0. Por exemplo, a unidade consumidora com id igual a 1, evidenciado em vermelho na Figura 8, teria apenas as leituras de março de 2022 a janeiro de 2023 como irregular (valor 1), as demais leituras serão categorizadas como regulares (valor 0).

Figura 8 - Planilha de período de irregularidades utilizada como *ground truth*

id unidade consumidora	data inicio irreg	data fim irregularidade
1	11/03/2022	30/01/2023
2	18/03/2023	23/05/2023
3	07/07/2022	28/03/2023
4	25/01/2023	28/03/2023
5	04/08/2022	09/03/2023
6	02/04/2020	08/02/2023
7	05/04/2022	14/06/2023
8	10/11/2020	10/10/2023
9	21/02/2023	26/04/2023
10	23/02/2023	07/08/2023
11	16/06/2023	01/12/2023
12	10/11/2020	18/10/2023
13	05/05/2022	15/02/2023
14	22/05/2020	26/04/2023
15	20/02/2020	15/02/2023
16	27/05/2020	09/05/2023
17	28/04/2021	13/02/2023

Fonte: elaborado pela autora.

3.2.1.1 Balanceamento da Base de Dados

Ao analisar a base de dados recebida, observou-se o pequeno número de registros, tanto de forma geral quanto por classe (regular e irregular). Para isto, utilizou-se uma técnica de multiplicação de matrizes chamada “Hadamard Product” proposto por Niu e Zhang (2021) para a criação de registros sintéticos. Registros sintéticos são novos registros criados com base nos registros existentes. A base de dados apresentada pelos autores mostra uma quantidade menor, e com essa técnica foi obtido resultados satisfatórios. A técnica Hadamard Product consiste em multiplicar os valores de cada registro por um número aleatório entre 0 e 1. Com isso, é possível criar cenários, porém com a mesma curva de consumo, mantendo o padrão de irregularidade/regularidade.

Essa técnica foi aplicada neste trabalho, onde cada relatório de UC foi usado como base para gerar novos relatórios semelhantes. Ao aplicar a técnica Hadamard Product neste trabalho, concluiu-se que, essa técnica seria válida somente nas colunas “faturado” e “COM/AHR”, pois alterar os demais dados criaria cenários incorretos. Devido a isso, para os novos registros, as demais colunas foram mantidas inalteradas. A Tabela 1 apresenta a quantidade de registros originais e registros sintéticos adicionados para cada classe de UC. A partir dos 295 registros originais foram gerados 1180 registros sintéticos, totalizando 1475 registros.

Tabela 1 - Relação de registros da base

Classe	Quantidade de UCs	Total de Registros
Irregular	171 (Registros originais) 684 (Registros sintéticos)	855 registros
Regular	124 (Registros originais) 496 (Registros sintéticos)	620 registros
Total	295 (Registros originais) 1180 (Registros sintéticos)	1475 registros

Fonte: elaborado pela autora.

3.2.2 Arquitetura das Redes Neurais Artificiais Aplicadas

Para este trabalho foram utilizadas cinco arquiteturas de Redes Neurais Artificiais: *Long Short-Term Memory (LSTM)*, *Gated Recurrent Unit (GRU)*, *Deep Neural Network (DNN)*, *Convolutional Gated Recurrent Unit (CNN+GRU)* e *Bidirecional Gated Recurrent Unit (Bi-GRU)*. As redes LSTM, GRU, CNN+GRU e Bi-GRU foram escolhidas por sua melhor capacidade de prever dados temporais e menção em trabalhos relacionados, tais como os correlatos citados anteriormente. Todos os modelos foram implementados utilizando a linguagem de Programação Python e a biblioteca *Keras*, com o Google Colab como ambiente de desenvolvimento. Como métricas de avaliação dos algoritmos foram escolhidas a acurácia, *Precision* e *Recall*. A acurácia serve para demonstrar a performance do modelo. O valor de *Precision* serve para demonstrar a proporção de previsões positivas que realmente são positivas. Já o *Recall* serve para demonstrar a proporção de elementos positivos que foram corretamente identificados. Com a *Precision* e o *Recall* será possível verificar a taxa de acerto de forma geral do modelo implementado. O Quadro 6 resume as arquiteturas das RNAs aplicadas neste trabalho.

Quadro 6 - Arquitetura das RNAs aplicadas

Técnica	Configuração das camadas	Tipo de Optimizer	Tipo de Loss
LSTM	1 camada de LSTM com 64 units 1 camada densa com o tipo de ativação <i>ReLU</i> com 64 units 1 camada de <i>Drop-out</i> com uma taxa de 0.3 1 camada densa com o tipo de ativação <i>Softmax</i> com 2 units	Adam com Learning Rate de 0.0001	Categorical Crossentropy
GRU	1 camada de GRU com 32 units 2 conjuntos de camada densa com o tipo de ativação <i>ReLU</i> + camada de <i>Drop out</i> , com uma taxa de 0.3 e 64 units 1 camada densa com o tipo de ativação <i>Softmax</i> com 2 units	Adam com Learning Rate de 0.0001	Categorical Crossentropy
DNN	1 camada densa com o tipo de ativação <i>Sigmoid</i> com 64 units 1 camada densa simples com 64 units 1 camada de <i>Drop-out</i> com uma taxa de 0.3 1 camada densa com o tipo de ativação <i>ReLU</i> com 32 units 1 camada com o tipo de ativação <i>Sigmoid</i> com 2 units	Adam com Learning Rate de 0.001	Binary Focal Crossentropy
CNN + GRU	1 camada do tipo convolucional 1D com o tipo de ativação <i>ReLU</i> com 32 units 1 camada de <i>MaxPooling</i> 1D 1 camada de GRU com 32 units 1 camada <i>Flatten</i> 1 camada densa com o tipo de ativação <i>Softmax</i>	Adam com Learning Rate de 0.00001	Categorical Crossentropy

Bi-GRU	1 camada Bidirecional do tipo GRU com 32 units 1 camada densa do tipo de ativação <i>ReLU</i> com 32 units 1 camada densa do tipo de ativação <i>Sigmoid</i> com 2 units	Adam com Learning Rate de 0.001	Binary Focal Crossentropy
--------	--	---------------------------------	---------------------------

Fonte: elaborado pela autora

No Apêndice A são apresentados os códigos das redes DNN (Quadro 8), LSTM (Quadro 10), GRU (Quadro 10) e CNN+GRU (Quadro 11). No Quadro 7 é apresentada a implementação do modelo da Rede Bidirecional *Gated Recurrent Unit*. A linha 1 apresenta a inicialização do modelo. Nas linhas 2 e 3 é adicionado uma camada de rede bidirecional do tipo GRU. A linha 4 é responsável pela adição de uma camada densa com o tipo de ativação *ReLU*. A linha 5 é responsável pela adição de uma camada com o tipo de ativação *Sigmoid*. Como função de perda, foi utilizado o `BinaryFocalCrossentropy` (linha 6), pois o problema apresentado se trata de um problema binário – regular ou irregular. A variável `from_logits=True`, na linha 6, indica que as saídas não passaram por uma função *softmax*, fazendo com que a função de perda seja aplicada diretamente as saídas do modelo. Essa variável é necessária, pois a camada com ativação do tipo *Sigmoid* que é responsável pela classificação do resultado.

Quadro 7 - Código de construção do modelo BiGRU e implementação do compile

1	<code>model = Sequential()</code>
2	<code>model.add(Bidirectional(keras.layers.GRU(32, input_shape=(X_train.shape[1],</code>
3	<code>X_train.shape[2]), return_sequences=True))</code>
4	<code>model.add(Dense(32, activation='relu'))</code>
5	<code>model.add(Dense(1, activation='sigmoid'))</code>
6	<code>model.compile('adam', loss=keras.losses.BinaryFocalCrossentropy(from_logits=True),</code>
7	<code>metrics=['accuracy'], keras.metrics.Precision(), keras.metrics.Recall())</code>
8	<code>X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)</code>
9	<code>X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train,</code>
10	<code>test_size=0.2)</code>
11	<code>model.fit(X_train, y_train, epochs=30, batch_size=10, validation_data=(X_validation,</code>
12	<code>y_validation))</code>
13	<code>_, accuracy = model.evaluate(X_test, y_test)</code>

Fonte: elaborado pela autora

Para o treinamento da rede, a base de dados foi dividida de forma aleatória com 75% para treinamento e 25% para testes. O conjunto de dados do treinamento foi dividido em 80% para o treinamento do modelo e 20% para a validação do treinamento. Para essa divisão foi utilizado o método `train_test_split` da biblioteca *Sklearn*, (linha 8). A linha 9 apresenta o treinamento do modelo, com 30 épocas, em cada época contém 10 lotes.

4 RESULTADOS

Esta seção são apresentados os testes realizados e os resultados. Na subseção 4.1 são detalhados os desafios encontrados no desenvolvimento do projeto juntamente com os testes realizados com diversos algoritmos de aprendizado de máquina. Na subseção 4.2 são apresentados os resultados obtidos com as Redes Neurais Artificiais, particularmente a Rede *Bidirecional Gated Recurrent Unit*.

4.1 TESTES REALIZADOS COM ALGORITMOS DE APRENDIZADO DE MÁQUINA

Antes da aplicação das RNAs, diversos algoritmos de Aprendizado de Máquina (AM) foram testados para identificar fraudes no consumo de energia. Dentre os quais, foram utilizados: *Support Vector Machine* (SVM), *Decision Tree*, *K-Nearest Neighbors Algorithm* (KNN), *Isolation Forest*, *Local Outlier Factor* (LOF) e *K-means*. Como o SVM foi utilizado em um dos correlatos citados anteriormente, decidiu-se testar esse modelo neste trabalho. Os modelos de *Decision Tree* e KNN foram adicionados a fim de comparação e testes para Aprendizado Supervisionado.

Além disso, observou-se neste trabalho que é possível interpretar os dados com uma abordagem de detecção de *outliers*. Para esse tipo de detecção, são comumente utilizados modelos Não Supervisionados. A partir desta observação, foram utilizados os modelos LOF e *K-means*, a fim de testar e comparar com as técnicas de Aprendizado Supervisionado. Como esses algoritmos exigem dados bidimensionais (2D), um novo conjunto de dados foi criado, contendo apenas os valores de faturamento e relógio, pois são considerados os atributos mais relevantes para a detecção de irregularidades no consumo. Em todos os modelos testados, foram também adicionados os novos registros sintéticos. A Tabela 2 apresenta a acurácia, *precision* e *recall* obtidos em cada modelo de AM, com e sem a adição de registros sintéticos. Para obter as métricas citadas foi utilizado a biblioteca *Scikit-learn*.

O modelo KNN obteve a melhor acurácia, de 0.73, porém o valor de recall deste foi apenas 0.33. Apesar do modelo KNN ter apresentado uma boa performance, o modelo teve dificuldades de identificar os registros irregulares corretamente. Nos testes realizados com os registros sintéticos, observou-se uma melhora nos resultados de *precision* (0.72) e *recall* (0.86), porém diminuiu a acurácia para 0.68. O modelo *Isolation Forest* apresentou uma acurácia mais baixa, de 0.68, porém apresentou *Precision* (0.59) e *Recall* (0.68) moderados, indicando que o modelo consegue identificar melhor os casos positivos, ou seja, os registros irregulares corretamente. O mesmo se observou para o SVM,

que apresentou uma acurácia moderada de 65% e um equilíbrio razoável entre precisão (67%) e recall (68%). Isso demonstra que o modelo alcançou uma performance e taxa de acertos equilibrada, apontando ser um modelo adequado para o conjunto de dados.

Tabela 2 - Resultados de Acurácia, Precision e Recall por técnica de AM

Tipo de Aprendizado	Técnica de AM	Testes Sem Registros Sintéticos			Testes Com Registros Sintéticos		
		Acurácia	Precision	Recall	Acurácia	Precision	Recall
Supervisionado	SVM	0.65	0.67	0.68	0.68	0.72	0.86
	Decision Tree	0.69	0.40	0.40	0.64	0.72	0.72
	KNN	0.73	0.49	0.33	0.68	0.74	0.81
Não Supervisionado	Isolation Forest	0.68	0.59	0.68	0.63	0.45	0.63
	LOF	0.45	0.57	0.45	0.51	0.51	0.51
	K-Means	0.56	0.60	0.56	0.30	0.30	0.55

Fonte: elaborado pela autora.

Os testes com registros sintéticos apresentaram uma pequena melhora no SVM na três métricas de avaliação, com uma acurácia de 0.68, *Precision* de 0.72 e *Recall* de 0.86. O modelo *K-means* apresentou uma piora com a adição dos registros sintéticos, mostrando uma acurácia e *Precision* de 0.30, e *Recall* de 0.55. O modelo LOF não apresentou melhora significativa com a adição dos novos registros, apresentando métricas abaixo dos demais. O modelo *Decision Tree* apresentou uma piora na acurácia (0.64) e uma melhora no *Precision* (0.72) e *Recall* (0.72).

4.2 RESULTADOS OBTIDOS COM AS REDES NEURAIIS ARTIFICIAIS

Neste trabalho, foram também realizados testes com diversas Redes Neurais Profundas, e com estas foi possível alcançar uma acurácia superior em comparação com outros modelos de Aprendizado de Máquina. Apesar da acurácia maior, os modelos *Long Short-Term Memory* (LSTM), *Gated Recurrent Unit* (GRU) e *Convolutional Gated Recurrent Unit* (CNN+GRU) apresentaram uma perda muito alta, em torno de 0.6-0.65. Durante a fase de testes, foram realizados ajustes na quantidade de camadas, tamanho do batch (*batch size*), número de épocas, tipo de aprendizado, taxa de aprendizado, tipo de perda e tipo da camada de classificação a fim de resolver o problema de perda. Dentre os modelos gerados, Bi-GRU se destacou por apresentar melhor curva de aprendizado no treinamento, menor perda e melhor acurácia, 0.16 e 0.77 respectivamente. Além do Bi-GRU, a *Deep Neural Network* também se mostrou promissora, apresentando uma acurácia de 0.75. Entretanto esse modelo apresentou valores muito próximos a zero para a *Precision* e *Recall*, onde ao passar pelo arredondamento de duas casas decimais, ambas resultaram em zero. A Tabela 3 apresenta os resultados da acurácia, perda, AUC, *Precision* e *Recall* de cada RNA utilizada.

Tabela 3 - Comparação dos resultados das RNAs utilizadas

Técnicas	Acurácia	Perda	Precision	Recall
LSTM	0.71	0.64	0.45	0.50
Deep Neural Network	0.75	0.17	0.00	0.00
GRU	0.49	0.63	0.45	0.50
CNN + GRU	0.56	0.62	0.37	0.50
BiGRU	0.77	0.16	0.89	0.18

Fonte: elaborado pela autora.

É importante ressaltar que os resultados obtidos foram alcançados utilizando apenas os registros recebidos, sem adição de registros sintéticos. Testes foram realizados com os registros sintéticos e, apesar do aumento do número de registros, percebeu-se uma pouca mudança no desempenho dos modelos de RNA. A Tabela 4 apresenta uma comparação entre as Redes Neurais utilizadas com os registros sintéticos: *Long Short-Term Memory* (LSTM), *Gated Recurrent Unit* (GRU), *Deep Neural Network* (DNN), *Convolutional Gated Recurrent Unit* (CNN+GRU) e Bidirecional *Gated Recurrent Unit* (Bi-GRU). A melhor acurácia foi alcançada pelo modelo GRU, com 0.85, entretanto, o valor da perda foi 0.63. As redes *Deep Neural Network* e BiGRU apontaram um valor de *Precision* e *Recall* muito próximo a zero, resultando em zero ao arredondar para duas casas decimais.

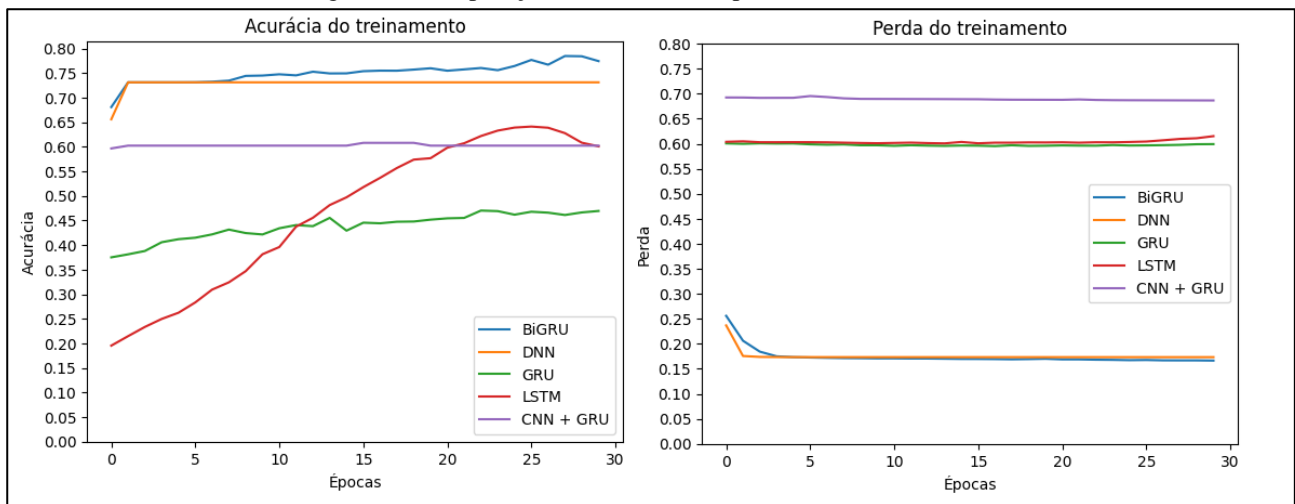
Tabela 4 - Comparação entre as RNAs utilizando registros sintéticos

Técnicas	Acurácia	Perda	<i>Precision</i>	<i>Recall</i>
LSTM	0.74	0.55	0.40	0.5
Deep Neural Network	0.73	0.17	0.0	0.0
GRU	0.85	0.63	0.46	0.5
CNN + GRU	0.74	0.53	0.38	0.5
BiGRU	0.73	0.17	0.0	0.0

Fonte: elaborado pela autora.

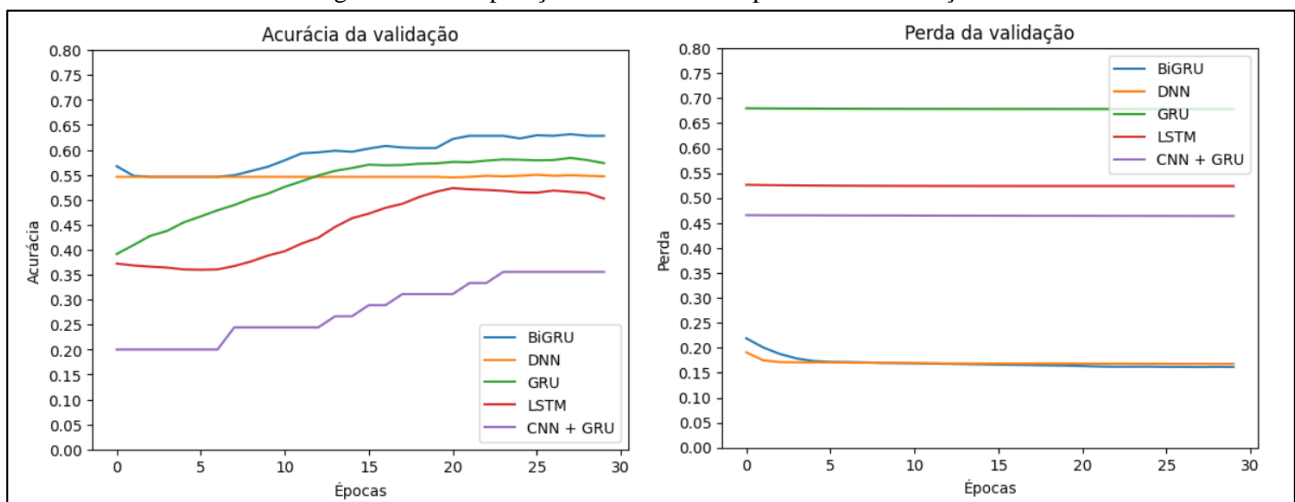
Sem considerar os dados sintéticos, o modelo atingiu uma acurácia de 0.77 e uma perda de 0.16 durante o treinamento. Durante a fase de teste, o modelo alcançou uma acurácia de 0.78 e uma perda de 0.17, indicando que o modelo tem um bom desempenho geral e está generalizando bem, sem sinais de overfitting. Além desses resultados, o modelo apresentou uma boa curva de aprendizado, na qual o melhor número de épocas encontrado foi 30. A partir da 15ª época, o valor da perda se estabilizou, demonstrando convergência do modelo. Em contraste com o modelo DNN, que atingiu valores de acurácia semelhantes (0.75), entretanto a curva de treinamento foi diferente, se estagnando sempre no mesmo valor de acurácia, indicando *overfitting*, como apresenta a Figura 9 e Figura 10. Foram realizados também testes de todas as redes neurais sem a adição dos registros solares como regulares, porém em nenhum caso houve melhora no desempenho e métricas.

Figura 9 - Comparação entre as RNAs aplicadas no treinamento



Fonte: elaborado pela autora.

Figura 10 - Comparação entre as RNAs aplicadas na validação



Fonte: elaborado pela autora.

A Tabela 5 apresenta os resultados atingidos durante o treinamento e teste do modelo BiGRU. O modelo atingiu uma Precisão de 0.89 no treinamento e 0.72 nos testes. A acurácia é semelhante nos conjuntos de treinamento e teste, indicando que o modelo tem um bom desempenho geral e está generalizando bem. A perda é baixa e quase igual nos

conjuntos de treinamento (0.16) e teste (0.17), sugerindo que o modelo está aprendendo adequadamente e as previsões são consistentes em ambos os conjuntos. Entretanto, o modelo atingiu um *Recall* de 0.18 no treinamento e 0.16 nos testes. Tanto o valor baixo do *Recall* quanto o desequilíbrio entre os valores de *Precision* e *Recall* pode ser causado pelo desbalanceamento do conjunto de dados utilizado. Isto é, existem poucos registros irregulares para muitos registros regulares.

Tabela 5 - Resultados do modelo BiGRU de treinamento e teste

Métrica	Treinamento	Teste
Acurácia	0.77	0.78
Perda	0.16	0.17
<i>Precision</i>	0.89	0.72
<i>Recall</i>	0.18	0.16

Fonte: elaborado pela autora.

5 CONCLUSÕES

Atualmente as concessionárias de energia no Brasil e a população perdem muito dinheiro em decorrência dessas irregularidades. Além disso, existe também o perigo que essas irregularidades têm sobre a segurança das ruas brasileiras, visto que as irregularidades podem causar curtos-circuitos e incêndios. Com isto, o presente trabalho consistiu no desenvolvimento de um protótipo para a detecção de Perdas Não Técnicas (PNT) no sistema de distribuição de energia elétrica. Dentre os modelos testados, a Rede Bidirecional *Gated Recurrent Unit* (Bi-GRU) apresentou o melhor resultado, alcançando uma acurácia de 0.78, um índice de perda de 0.17, uma *Precision* de 0.89 e *Recall* de 0.16. O modelo apresentou uma boa performance e baixa perda. A alta taxa de precisão aponta que quando o modelo faz a predição de casos positivos, grande parte das vezes está correto, isto é, quando o modelo prevê casos irregulares. Devido *Recall* baixo, o modelo classificou muitos registros irregulares como regulares. Acredita-se que o desequilíbrio entre os valores de *Precision* e *Recall* acontecem devido ao desbalanceamento da base de dados utilizada. Apesar das métricas atingidas não apresentarem um valor computacional elevado, o desempenho do modelo superou a meta estabelecida para as detecções de fraude manuais da região. De acordo com um dos gerentes da concessionária, a meta de detecção de fraudes para o ano de 2024 é de 32,7%, e o modelo proposto alcançou 78%.

As limitações do projeto estão relacionadas principalmente à qualidade e disponibilidade da base de dados. Primeiramente, a maneira que a base de dados foi fornecida, exigiu transformações no formato do arquivo, bem como verificações e alterações manuais. Outra limitação significativa foi que, apesar da base de dados conter mais UCs irregulares, o período de irregularidade em cada UC não é total. Isto é, mesmo a UC sendo irregular, esta contém períodos regulares. Analisando os dados, percebeu-se uma quantidade maior de número de registros regulares em comparação à quantidade de regulares. A partir desta observação, tentou-se resolver o problema de balanceamento e a baixa quantidade de dados por meio da geração de registros sintéticos. No entanto, ao analisar os resultados, observou-se pouco ganho na performance dos modelos.

Em comparação com os correlatos selecionados, pode-se observar uma diferença significativa na quantidade de dados recebida. Neste trabalho foi utilizado uma base com 295 UCs, já os correlatos apresentaram base de dados com no mínimo mil UCs. Acredita-se que com uma base de dados maior, poderiam ser alcançados melhores resultados.

Como trabalhos futuros, recomenda-se o aprimoramento da base de dados. Isso inclui aumentar a quantidade de dados na base de dados e melhorar o balanceamento das classes. Além disso, automatizar a extração direta da base de dados seria importante, eliminando a necessidade de conferências manuais. Outros testes também poderiam ser realizados, como a experimentação com diferentes modelos, como *Transformers* ou *XGBoost*, e a aplicação de técnicas de validação cruzada, como o *K-Fold Cross-Validation*. Poderia ser aplicada também técnicas de regularização, como camadas de L1 e L2 a fim de melhorar o desempenho do modelo de RNA. Sugere-se ainda, o desenvolvimento de um programa simples e amigável, para que os técnicos responsáveis possam utilizar o modelo com maior facilidade. Outra possibilidade é a implementação de modelos que utilizam regressão, tendo como resultado uma porcentagem indicando o quão provável aquele registro é irregular.

Com isto, o trabalho apresentado tem o seu valor na comunidade acadêmica, por propor uma solução a um problema presente na sociedade e mostrar os resultados obtidos. Os resultados obtidos não apresentam um valor alto o suficiente para a detecção de PNT. Apesar disto, o estudo apresenta bastante espaço para trabalhos futuros e melhorias, sendo possível atingir o objetivo com mais assertividade.

REFERÊNCIAS

ALBIERO, Beatriz et al. Employing Gradient Boosting and Anomaly Detection for Prediction of Frauds in Energy Consumption. *Anais do Encontro Nacional de Inteligência Artificial e Computacional (Eniac)*, Brasil, v. 1, n. 1, p. 1-10, out. 2019.

ANEEL (Brasil). Agência Nacional de Energia Elétrica. **Perdas de Energia**. 2023. Disponível em: <https://www.gov.br/aneel/pt-br/assuntos/distribuicao/perdas-de-energia>. Acesso em: 06 set. 2023.

CERBRANORTE. **Perdas Comerciais**. 2024. Disponível em: <https://cerbranorte.org/ferramentas-web/>. Acesso em: 24 jun. 2024

BORGES, Fabricio Quadros. Crise de energia elétrica no Brasil - uma breve reflexão sobre a dinâmica de suas origens e resultados. **Recima21 - Revista Científica Multidisciplinar - Issn 2675-6218**, [S.L.], v. 2, n. 10, p. 1-11, 2 nov. 2021. Recima21 - Revista Científica Multidisciplinar. <http://dx.doi.org/10.47820/recima21.v2i10.809>.

CEMIG (Brasil). Companhia Energética de Minas Gerais. **Furto de Energia**. 2023. Disponível em: <https://www.cemig.com.br/atendimento/furto-de-energia/>. Acesso em: 19 set. 2023.

CORSINI, Luri. Energia elétrica aumentou mais do que o dobro da inflação nos últimos anos. **CNN Brasil**, Rio de Janeiro, 18 jan. de 2022. Disponível em: <https://www.cnnbrasil.com.br/economia/energia-eletrica-aumentou-mais-do-que-o-dobro-da-inflacao-nos-ultimos-anos/>. Acesso em: 27 set. 2023.

GOLDSTEIN, Markus; UCHIDA, Seiichi. Unsupervised Anomaly Detection Benchmark. **Plus One**, Fukuoka, Japan, v. 1, n. 1, p. 1-31, 2015. <http://dx.doi.org/10.7910/DVN/OPQMVF>.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Machine Learning Basics. In: GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. Cambridge: Mit Press, 2016. Cap. 5. p. 1-800. Disponível em: <https://www.deeplearningbook.org>. Acesso em: 24 jun. 2024

HANDELMAN, Guy S. et al. Peering Into the Black Box of Artificial Intelligence: evaluation metrics of machine learning methods. **American Journal Of Roentgenology**, [S.L.], v. 212, n. 1, p. 38-43, jan. 2019. American Roentgen Ray Society. <http://dx.doi.org/10.2214/ajr.18.20224>.

KORBA, Abdelaziz Amara; KARABADJI, Nour El Islem. Smart Grid Energy Fraud Detection Using SVM. **2019 International Conference On Networking And Advanced Systems (Icnas)**, Annaba, v. 1, n. 1, p. 1-6, jun. 2019. IEEE. <http://dx.doi.org/10.1109/icnas.2019.8807832>.

LI, Pengpeng *et al.* Bidirectional Gated Recurrent Unit Neural Network for Chinese Address Element Segmentation. **Isprs International Journal Of Geo-Information**, [S.L.], v. 9, n. 11, p. 635, 26 out. 2020. MDPI AG. <http://dx.doi.org/10.3390/ijgi9110635>.

LUDERMIR, Teresa Bernarda. **Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências**. Estudos Avançados, Pernambuco, v. 35, n. 101, p. 85-94, abr. 2021. FapUNIFESP (SciELO). <http://dx.doi.org/10.1590/s0103-4014.2021.35101.007>.

MARKOVSKA, Marija et al. Electricity Theft Detection Based on Temporal Convolutional Networks with Self-Attention. **2023 30Th International Conference On Systems, Signals And Image Processing (IwSSIP)**, Skopje, v. 1, n. 1, p. 1-5, 27 jun. 2023. IEEE. <http://dx.doi.org/10.1109/iwSSIP58668.2023.10180294>.

NIELSEN, Michael A.. **Neural Networks and Deep Learning**. São Francisco, Ca, Usa: Determination Press, 2015. Disponível em: <http://neuralnetworksanddeeplearning.com/>. Acesso em: 24 jun. 2024.

NIU, Zhewen; ZHANG, Gengwu. A Data-Driven Method for Electricity Theft Detection Combing ConvGRU and K-means Clustering. **2021 Ieee 5Th Conference On Energy Internet And Energy System Integration (Ei2)**, Taiyuan, v. 1, n. 1, p. 1-6, 22 out. 2021. IEEE. <http://dx.doi.org/10.1109/ei252483.2021.9712851>.

NORVIG, Peter; RUSSELL, Stuart. Capítulo 8. In: NORVIG, Peter; RUSSELL, Stuart. **Inteligência Artificial: Uma Abordagem Moderna**. Rio de Janeiro: Grupo Editorial Nacional S.A, 2022. p. 1-1324.

PIOTROWSKI, Leonardo Jonas et al. Análise das Perdas de Energia no Sistema Elétrico de Distribuição Brasileiro. **Proceedings Of The 13Th Seminar On Power Electronics And Control (Sepoc 2021)**, Santa Maria, v. 1, n. 1, p. 1-6, 18 maio 2021. Sepoc. <http://dx.doi.org/10.53316/sepoc2021.012>.

PAULO, Fernanda Rodrigues; VILLANUEVA, Juan Moises Mauricio; BRAZ, Helon David de Macêdo. **Detecção de fraude em unidades consumidoras não telemedidas com uso de técnicas de aprendizado de máquina**. 2020. 111 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal da Paraíba, Paraíba, 2020. Disponível em: <https://repositorio.ufpb.br/jspui/handle/123456789/18759>. Acesso em: 24 jun. 2024.

SOARES, Lucas Duarte. **Redes neurais artificiais BIGRU_CNN aplicadas à previsão de demanda de energia elétrica de curto prazo**. 2021. 118 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Programa de Pós-Graduação em Engenharia Elétrica e Computação, Universidade Estadual do Oeste do Paraná, Foz do Iguaçu, 2021. Disponível em: <https://tede.unioeste.br/handle/tede/5712>. Acesso em: 20 jun. 2024.

APÊNDICE A – CÓDIGO DAS REDE NEURAIIS ARTIFICIAIS TESTADAS

Os quadros abaixo apresentam as arquiteturas utilizadas nas RNAs testadas neste trabalho, contendo os parâmetros e a sequência de treinamento. O Quadro 8 apresenta o código da rede neural *Deep Neural Network*. Segundo Nielsen (2015), a rede neural DNN possui mais de uma camada de neurônio oculta, onde o número de camadas e de neurônios por camada varia de acordo com a necessidade do modelo.

Quadro 8 - Código da *Deep Neural Network*

1	model = Sequential()
2	model.add(Dense(64, input_shape=(X.shape[1],X.shape[2]), activation='relu'))
3	model.add(Dense(64))
4	model.add(keras.layers.Dropout(0.3))
5	model.add(Dense(32, activation='relu'))
6	model.add(Dense(1, activation='sigmoid'))
7	model.compile('adam', loss=keras.losses.BinaryFocalCrossentropy(from_logits=True),
8	metrics=['accuracy',keras.metrics.Precision(), keras.metrics.Recall()])
9	X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)
10	X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train,
11	test_size=0.2)
12	model.fit(X_train, y_train, epochs=30, batch_size=10,
13	validation_data = (X_validation, y_validation))
14	model.evaluate(X_test, y_test)

Fonte: elaborado pela autora

O Quadro 9 apresenta o código da rede neural LSTM. Segundo Soares (2021), LSTM é uma arquitetura de RNN que possui células chamadas *long short-term memory*, que servem para resolver o problema de *backpropagation through time*. Esta arquitetura é amplamente utilizada por causa do seu superior desempenho na modelagem com dados de curto e longo prazo.

Quadro 9 - Código da Rede LSTM

1	model = Sequential()
2	model.add(keras.layers.LSTM(64,
3	input_shape=(X_train_2.shape[1], X_train_2.shape[2]), return_sequences=True))
4	model.add(Dense(64, activation='relu'))
5	model.add(keras.layers.Dropout(0.3))
6	model.add(Dense(2, activation='softmax'))
7	model.compile(keras.optimizers.Adam(learning_rate=0.0001),
8	loss='categorical_crossentropy',
9	metrics=['accuracy',keras.metrics.Precision(), keras.metrics.Recall()])
10	X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)
11	X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train,
12	test_size=0.2)
13	model.fit(X_train, y_train, epochs=30, batch_size=10,
14	validation_data = (X_validation, y_validation))
15	model.evaluate(X_test, y_test)

Fonte: elaborado pela autora

O Quadro 10 apresenta o código da rede neural GRU. A rede neural GRU é uma versão melhorada da arquitetura LSTM. Isto se deve pela rede GRU ser mais eficiente em alcançar a convergência e atualizar os pesos internos durante o treinamento, pois sua estrutura de portas internas é mais sucinta (SOARES, 2021).

Quadro 10 - Código da Rede GRU

1	model = Sequential()
2	model.add(keras.layers.GRU(32, input_shape=(X_train_2.shape[1], X_train_2.shape[2])
3	,return_sequences=True))
4	model.add(Dense(64, activation='relu'))
5	model.add(Dense(64, activation='relu'))
6	model.add(Dense(2, activation='softmax'))
7	model.compile('adam', loss='categorical_crossentropy',
8	metrics=['accuracy',keras.metrics.Precision(), keras.metrics.Recall()])
9	X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)
10	X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train,
11	test_size=0.2)
12	model.fit(X_train, y_train, epochs=30, batch_size=10,
13	validation_data = (X_validation, y_validation))
14	model.evaluate(X_test, y_test)

Fonte: elaborado pela autora

O Quadro 11 apresenta o código da rede neural CNN+GRU. Essa arquitetura apresenta uma junção das arquiteturas *Convolutional Neural Network* e *Gated Recurrent Unit*. Segundo Soares (2021), a CNN é uma RNA profunda do tipo *feedforward*, porém que não é formada por conexões cíclicas e não possui memória como entrada.

Quadro 11 - Código da Rede CNN+GRU

1	model = Sequential()
2	model.add(Conv1D(32, 3,
3	input_shape=(X_train_2.shape[1], X_train_2.shape[2]), activation='relu'))
4	model.add(MaxPooling1D())
5	model.add(keras.layers.GRU(32))
6	model.add(Dense(2, activation='softmax'))
7	model.compile(keras.optimizers.Adam(learning_rate=0.0001),
	loss='categorical_crossentropy',
8	metrics=['accuracy', keras.metrics.Precision(), keras.metrics.Recall()])
9	X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)
10	X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train,
11	test_size=0.2)
12	model.fit(X_train, y_train, epochs=30, batch_size=10,
13	validation_data = (X_validation, y_validation))
14	model.evaluate(X_test, y_test)

Fonte: elaborado pela autora