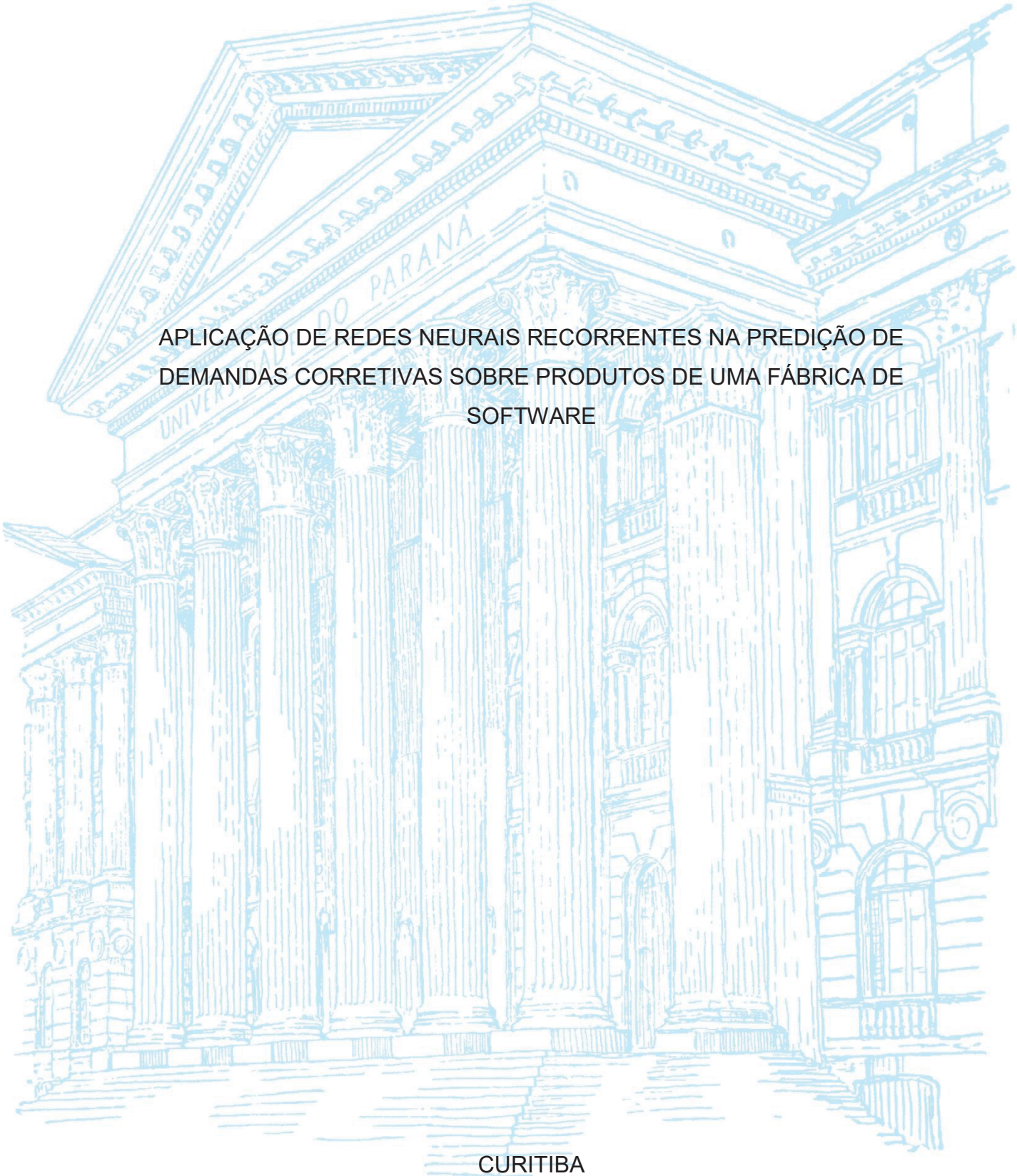
UNIVERSIDADE FEDERAL DO PARANÁ

THIAGO RIBEIRO VIEIRA



APLICAÇÃO DE REDES NEURAIS RECORRENTES NA PREDIÇÃO DE DEMANDAS CORRETIVAS SOBRE PRODUTOS DE UMA FÁBRICA DE SOFTWARE

CURITIBA

2022

THIAGO RIBEIRO VIEIRA

APLICAÇÃO DE REDES NEURAIS RECORRENTES NA PREDIÇÃO DE DEMANDAS CORRETIVAS SOBRE PRODUTOS DE UMA FÁBRICA DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao curso de Pós-Graduação em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

CURITIBA

2022

MINISTÉRIO DA EDUCAÇÃO
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
CURSO DE PÓS-GRADUAÇÃO INTELIGÊNCIA ARTIFICIAL APLICADA - 40001016348E1

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INTELIGÊNCIA ARTIFICIAL APLICADA da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **THIAGO RIBEIRO VIEIRA** intitulada: **Aplicação De Redes Neurais Recorrentes Na Predição De Demandas Corretivas Sobre Produtos De Uma Fábrica De Software**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 20 de Outubro de 2022.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Aplicação de Redes Neurais Recorrentes para Predição de Demandas Corretivas sobre Produtos de uma Fábrica de Software

Thiago Ribeiro Vieira
Especialização em Inteligência Artificial Aplicada
Universidade Federal do Paraná (UFPR)
Curitiba, Brasil
rv.thiagovieira@gmai.com

Razer Anthom Nizer Rojas Montaño
Especialização em Inteligência Artificial Aplicada
Universidade Federal do Paraná (UFPR)
Curitiba, Brasil
razer@ufpr.br

***Resumo*—Esse trabalho apresenta uma aplicação das Redes Neurais Recorrentes (RNN) capaz de prever as demandas corretivas sobre os produtos de uma fábrica de software. Sendo que, a demanda corretiva tem como objetivo evidenciar defeitos e erros encontrados sobre o software. A previsão de demandas corretivas em um determinado produto pode contribuir com informações importantes para a gestão de produto ou projeto na tomada de decisões. O foco do trabalho está em uma das classes de redes neurais artificiais (RNA), as redes neurais recorrentes. São utilizados três tipos de arquitetura dessa área: o modelo de redes neurais recorrentes Simples RNN, o Long Short-Term Memory (LSTM) e o Gated Recurrent Unit (GRU). Foram aplicados o método de aprendizado supervisionado para treinamento e validação dessas redes. Ao longo do trabalho foi criado um modelo de cada uma dessas arquiteturas para três produtos diferentes da fábrica de software, mais especificamente das áreas de Nota Fiscal Eletrônica de Serviço (NFS-e), da Administração de Receitas (AR) e de Prestação de Contas (PC). Com o resultado foi possível constatar a viabilidade no uso de sistemas de previsão que utilizam técnicas de Inteligência Artificial (IA) para a implementação de modelos de previsão de demandas corretivas sobre o portfólio de uma fábrica de software.**

Palavras-Chave—redes neurais recorrentes, previsão, demanda corretiva.

Abstract*—This work presents an application of Recurrent Neural Networks (RNN) capable of predicting the corrective demands on the products of a software factory. Since the corrective demand aims to highlight defects and errors found on the software. The forecast of corrective demands on a given product can contribute with important information for product or project management in decision making. The focus of the work is on one of the classes of artificial neural networks (ANN), the recurrent neural networks. Three types of architecture are used in this area: the Simple RNN recurrent neural network model, the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU). The supervised learning method was applied to train and validate these networks. Throughout the work, a model of each of these architectures was created for three different products of the software factory, more specifically in the areas of Electronic Service Invoice (NFS-e), Revenue Administration (AR) and Accountability. (PRAÇA). As a result, it was possible to verify the feasibility of using forecasting systems that use Artificial Intelligence (AI) techniques for the implementation of predictive models of corrective demands on the portfolio of a software factory.

Keywords— recurrent neural networks, forecasting, corrective demand.

I. DESENVOLVIMENTO

A manutenção do software é uma das fases do processo de desenvolvimento de software. Isso se deve à necessidade de ajustar e melhorar o produto de acordo com as mais diversas necessidades do usuário. O ciclo de vida da manutenção de software é um processo de mudanças, que tem como finalidade corrigir erros de codificação, correção de falhas de projeto e especificação [1]. Esse processo é reconhecido como a atividade que demanda o maior volume de esforço dentre todas as atividades de Engenharia de Software [2]. Portanto, ela é definida como a modificação de um produto de software depois de sua entrega (ao cliente) para corrigir erros, melhorar sua performance ou qualquer outro atributo, ou para adaptar o produto a um ambiente modificado [3].

Após a implantação do sistema em produção, inicia-se a fase de manutenção. Nesta fase, a correção de um erro ou o melhoramento funcional obriga que a modificação seja analisada, implementada, testada, documentada e entregue. Por este motivo, cada manutenção acaba sendo sempre um processo trabalhoso e delicado [4]. Conforme Magela [5] página 15, "a manutenção de software custa no mínimo 90% do valor original do projeto, sem ter um limite para o valor máximo".

Esse processo de manutenção de software deve ser definido e compreendido pelos envolvidos, ou seja, é necessário que suas etapas sejam estabelecidas e difundidas em toda a organização desde a identificação de uma demanda até a sua resolução e liberação para o cliente [6]. Uma vez que, a Inteligência Artificial (IA) segue a mesma linha de conceitos que a Inteligência Humana, sendo possível sua adaptação para um sistema computacional, com o intuito de fazer com que máquinas realizem tarefas que as pessoas desenvolvem melhor até o momento. Entre os conceitos desenvolvidos por IA, inclui-se a capacidade de fazer com que uma máquina possa adquirir e aplicar conhecimentos de aprendizagem [7].

A predição de demandas corretivas fornece a organização informações que as ajuda a se preparar a demanda futura de defeitos e erros sobre seu produto. Portanto, o presente trabalho tem como objetivo apresentar uma análise da predição de demandas corretivas a partir de dados disponíveis pela fábrica de software com os mais diversos produtos em seu portfólio. Para isso, foram utilizadas redes neurais recorrentes com três tipos de arquitetura, sendo elas: modelo de redes neurais recorrentes Simples RNN, o Long Short-Term Memory (LSTM) e o Gated Recurrent Unit (GRU). Buscou-se, realizar um estudo sobre os modelos propostos de RNN para estruturar e desenvolver uma rede capaz de prever as

demandas corretivas sobre os produtos por determinado período. Assim, é possível concluir qual tipo RNN provê melhores resultados.

A. Descrição dos Dados

Os dados utilizados para o treinamento e aplicação das redes neurais foram obtidos a partir das informações disponibilizadas em 2022 no banco de dados de demandas corretivas e evolutivas de uma fábrica de software.

Pela quantidade de informações disponíveis optou-se por realizar o treinamento das redes neurais recorrentes com a base de dados por produto nas áreas de manutenção do software, mais especificamente sobre os produtos de Nota Fiscal Eletrônica de Serviço (NFS-e), da Administração de Receitas (AR) e de Prestação de Contas (PC). No qual, foram coletados os dados de demandas corretivas e transformados em formato *comma separated value* (.csv) para cada produto supracitado. O conjunto de dados extraído continha dados de 01 de janeiro de 2018 até 31 de dezembro de 2021, possuindo informações da competência (mes/anos) e o quantitativo de demandas corretivas sobre o respectivo produto.

TABELA I
EXEMPLO DO CONJUNTO DE DADOS NFS-E EXTRAÍDO DO BANCO, MONSTRANDO OS DADOS DO EXERCIO DE 2021.

Ano-Mês	Quantidade Demanda
2021-01	36
2021-02	17
2021-03	25
2021-04	30
2021-05	30
2021-06	27
2021-07	37
2021-08	47
2021-09	35
2021-10	24
2021-11	17
2021-12	10

Lembrando que o objetivo desse trabalho é fazer a predição quantitativa de demandas corretivas sobre o produto da fábrica de software.

B. Métodos

A tarefa de predição dos dados exige uma preparação dos dados antes de aplicar os algoritmos de inteligência artificial, envolvendo pré-processamento dos dados e definição de parâmetros de melhor desempenho e resultado. Sendo assim, a seguir são descritos os métodos utilizados neste trabalho.

Primeiramente, antes de utilizar o conjunto de dados para treinamento dos modelos de redes neurais, é necessário fazer um pré-processamento com o objetivo de retirar do conjunto qualquer tipo de informação que possa atrapalhar o aprendizado das redes neurais, e adicionar informações relevantes que melhorem o desempenho dos modelos. Portanto, as linhas que continham valores nulos (null) foram ignoradas, limpando as informações irrelevantes do conjunto de dados (por exemplo, competência em que não houve abertura de demandas corretivas sobre um determinado produto, o que não traria nenhuma informação relevante para o treinamento dos modelos). Em seguida, todo o conjunto de dados é normalizado para valores entre 0 e 1 através da função de normalização Máx-Mín. Essa normalização é importante pois aumenta a acurácia de vários modelos de aprendizado de máquina [8], e é feita através da equação 1:

EQUAÇÃO 1 – NORMALIZAÇÃO MAX-MIN

$$X_{norm} = \frac{(x - x_{min})}{(x_{max} - x_{min})}$$

Com os dados necessários para o aprendizado dos modelos já tratados, foram construídos os modelos de rede neural Simples RNN, LSTM e GRU em linguagem Python utilizando a interface de implementação para algoritmos de aprendizado de máquina TensorFlow. Os modelos foram construídos utilizando os mesmos hiperparâmetros e número de camadas, e os parâmetros ótimos a serem utilizados foram definidos com o modelo LSTM servindo de teste. O conjunto de dados foi separado, sendo 67% utilizado para treinamento, e 33% utilizado para teste. A definição dos parâmetros ótimos é descrita detalhadamente a seguir.

Os hiperparâmetros são parâmetros utilizados para controlar o processo de aprendizado de redes neurais. Diferentemente dos parâmetros de peso dos neurônios, que é calculado através do treinamento dos modelos, os hiperparâmetros devem ser definidos pelo programador [9]. Neste trabalho, foram realizados testes para valores diferentes dos parâmetros de número de camadas e número de neurônios de cada camada.

Ao construir um modelo, é necessário definir o número de camadas que o modelo irá utilizar e trabalhar. Não há uma definição exata para qual o número de camadas ou neurônios que um modelo deve ter para obter uma melhor performance, portanto esse tipo de parâmetro deve ser testado dependendo da finalidade do modelo e das características do conjunto de dados [10].

Como métrica de qualidade, o Erro Absoluto Médio (EAM) é utilizada para calcular o erro das previsões dos modelos. Dessa forma, o erro absoluto médio é calculado através da diferença do valor real e do valor predito dividido pelo número de dados. O Erro Absoluto Médio mede a dispersão dos erros e, portanto, se o seu valor for pequeno significa que a previsão está próxima à demanda real e se for alto indica problemas com o método de previsão empregado, o erro absoluto médio é representado da seguinte maneira [11].

EQUAÇÃO 2 – ERRO ABSOLUTO MÉDIO (EAM)

$$EAM = \frac{\sum_{i=1}^{n} |x_i - y_i|}{n}$$

No qual, x_i é o i-ésimo valor real, y_i é o i-ésimo valor predito e n é o número de dados utilizados.

Erro Quadrático Médio (EQM) é um método de calcular a acurácia e o erro nos modelos preditivos utilizados nesse trabalho. Na equação 3 é ilustrado a formula do EQM, sendo que o valor de x_i é o i-ésimo valor real, y_i é o i-ésimo valor predito e n é o número de dados utilizados [11].

EQUAÇÃO 3 – ERRO QUADRÁTICO MÉDIO (EQM)

$$EQM = \frac{1}{n}\sum_{i=1}^{n} (x_i - y_i)^2$$

Otimizadores são algoritmos utilizados em redes neurais com o intuito de minimizar o erro, identificando os valores de peso ótimos a serem empregados na rede neural baseado nos dados de treinamento através do valor mínimo local da função. Os otimizadores mais comuns utilizados para redes neurais

são o Adam, o Stochastic Gradient Descent (SGD) e o RMSProp [12].

Para redes neurais, a função de ativação é uma função não-linear que é aplicada à saída de um neurônio e ajuda no balanceamento do vetor de peso que será passado aos próximos neurônios na rede. Para redes neurais recorrentes, as funções de ativação mais utilizadas são a sigmoide, a tangente hiperbólica (tanh) e a unidade linear retificada (ReLU, do inglês Rectified Linear Unit [13].

A época é uma iteração de quando o conjunto de dados de treinamento inteiro é passado pela rede neural uma vez. Portanto, o número de épocas é considerado o número de passagens do conjunto de dados através da rede neural [14]. É um parâmetro importante para se definir, pois quanto maior o número de épocas, mais o modelo se adapta ao conjunto de dados, sendo assim deve ser escolhido com cautela. Se o número de épocas for muito pequeno, o modelo pode não se ajustar corretamente ao treinamento, causando valores altos de erros, agora se o número for muito grande, o modelo pode se ajustar excessivamente ao conjunto, causando sobreajuste.

C. *Tecnologias*

Para criação de soluções que lidem com linguagem natural, existem diversas tecnologias para auxiliar no processo de desenvolvimento. A seguir são apresentadas algumas das tecnologias que foram usadas para o trabalho proposto.

Python na versão 3.6 foi a linguagem de programação prevalente para obtenção dos resultados, no qual a linguagem é de fácil aprendizado, simples e poderosa. A sintaxe elegante promove a rápida prototipação e teste de programas em diversas plataformas, sendo ideal para scripts e desenvolvimento rápido [15]. Desta forma, essa linguagem é predominante na área de aprendizado de máquina, diversas bibliotecas especializadas em *machine learning* surgiram para acelerar o desenvolvimento. Uma das principais bibliotecas, que foi desenvolvida pela Google, é a TensorFlow [16], que foi inicialmente desenvolvida para Python e posteriormente ganhou outras implementações em diversas linguagens. Ela auxilia na criação e treinamento de redes neurais, assim como estruturas de dados em tensores que são otimizadas para operações matriciais e execução em GPUs.

O Tensorflow é uma das bibliotecas mais dominantes na área de aprendizado profundo desde 2015, quando deixou de ser um projeto interno do Google e foi liberado para uso comunitário na modalidade de open source [17]. Embora atualmente existem implementações para outras linguagens será utilizada a versão original como biblioteca da linguagem Python para ser possível aproveitar as demais bibliotecas disponíveis aprendizado de máquina. Devido a sua facilidade na criação, treinamento e avaliação de redes neurais e bom desempenho ela foi uma escolha segura para o desenvolvimento desse trabalho.

Outro fator que impactou a escolha dessa biblioteca entre as alternativas disponíveis é a sua boa integração com o ambiente de execução utilizado, o Google Colaboratory. Por serem fornecidos pela mesma empresa a biblioteca consegue utilizar eficientemente os recursos disponíveis nas máquinas virtuais disponíveis, podendo executar em CPU, GPU ou TPU sem que haja necessidade de grandes configurações por parte dos desenvolvedores. Assim, a biblioteca Tensorflow na versão 2.3 se apresentou ideal para as condições de uso encontradas neste trabalho.

Todos os experimentos desenvolvidos foram criados em um ambiente de desenvolvimento colaborativo que facilita a reprodução dos resultados obtidos. Contudo, o ambiente utilizado foi o Google Colaboratory, onde é disponibilizado um acesso a uma máquina virtual preparada para executar tarefas de aprendizado de máquina com disponibilidade de acesso a GPUs para aceleração dos algoritmos paralelos. Essa solução provê uma facilidade na unificação do ambiente de execução e compartilhamento de resultados.

II. RESULTADOS E DISCUSSÕES

A seguir são apresentados os resultados obtidos com as Redes Neurais Recorrentes (RNN) capaz de prever as demandas corretivas sobre os produtos de uma fábrica de software. Foram explorados diferentes conjuntos de dados e diferentes arquiteturas de modelos de redes neurais e parâmetros, com o objetivo de encontrar qual configuração produz o melhor resultado e compreender como treinar modelos de aprendizado de máquina para tarefas de predição.

Para esse trabalho, um modelo utilizando 2 camadas (Simples RNN/GRU/LSTM e uma Camada Densa) se mostrou mais preciso. Cada camada de uma rede neural recorrente tem um número de neurônios. A camada de saída foi fixada em apenas 1 neurônio, por usa vez, a outra camada tiveram seu número de neurônios variado entre 5, 25, 50 e 100 para cada uma das camadas. A definição dessa configuração é feita de forma empírica, em seguida, é executado o treinamento da rede, cada treino leva em torno de 5 minutos para ser concluído com a execução no modo GPUs no Google Colaboratory. O processo de treinamento é interrompido e finalizado quando a rede apresenta uma boa capacidade de generalização e quando a taxa de erro for suficientemente pequena, ou seja, o mais próximo do 0 (zero).

Na Figura 1 e Figura 2, pode-se observar que os que menos apresentam valores de erro quadrático médio (EQM) e erro absoluto médio (EAM) foram obtidos com as camadas de 5 neurônios cada, portanto esse valor é utilizado como novo padrão para os testes subsequentes.

FIGURA 1 - ERRO QUADRÁTICO MÉDIO (EQM) - VALORES DE ERRO OBTIDOS PARA DIFERENTES NÚMEROS DE NEURÔNIOS. FONTE: ELABORADO PELO AUTOR.

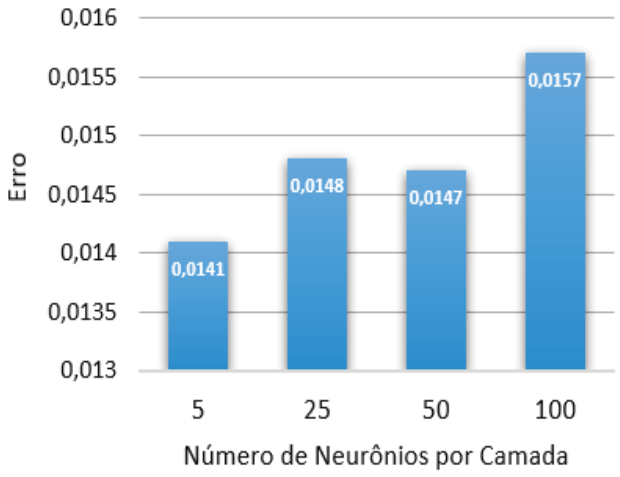
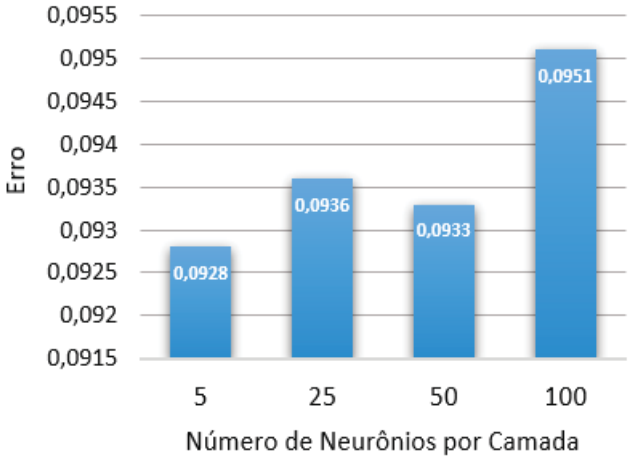
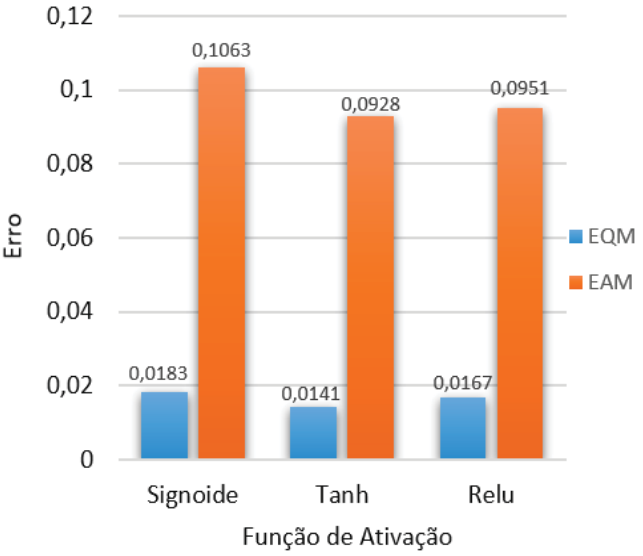


FIGURA 2 - ERRO ABSOLUTO MÉDIO (EAM) - VALORES DE ERRO OBTIDOS PARA DIFERENTES NÚMEROS DE NEURÔNIOS. FONTE: ELABORADO PELO AUTOR.



No que se refere aos otimizadores, o Adam obteve resultados melhores, tanto para os erros mínimos quanto levando em conta a média dos resultados de todos os testes realizados, e por isso continua sendo usado para os testes. Por sua vez, a função de ativação utilizada para o modelo será a tangente hiperbólica (tanh), devido ao erro menor entre os demais casos, mostrado na Figura 3.

FIGURA 3 - VALORES MÍNIMOS DE ERRO OBTIDOS PARA DIFERENTES FUNÇÕES DE ATIVAÇÃO. FONTE: ELABORADO PELO AUTOR.



As redes neurais que utilizam aprendizagem usando lote de padrões precisam de uma época de treinamento para realizar um passo de ajuste em seus pesos e limiares [15]. Com os resultados da Figura 4 e Figura 5, pode-se observar que o menor valor de erro para o modelo treinado com 250 de épocas, e para maior número de épocas, esse valor para de diminuir. Sendo assim, o modelo que têm a melhor predição é na verdade o de 250 épocas, portanto, aplicado nas etapas seguintes do trabalho.

FIGURA 4 - VALORES DE ERRO EQM OBTIDOS VARIANDO O NÚMERO DE ÉPOCAS. FONTE: ELABORADO PELO AUTOR.

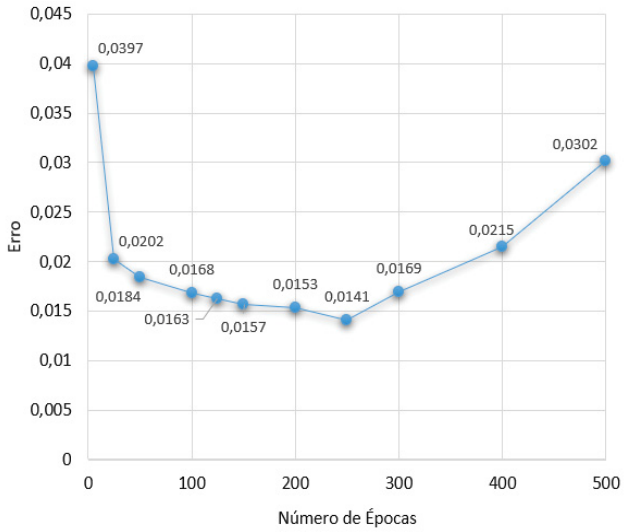
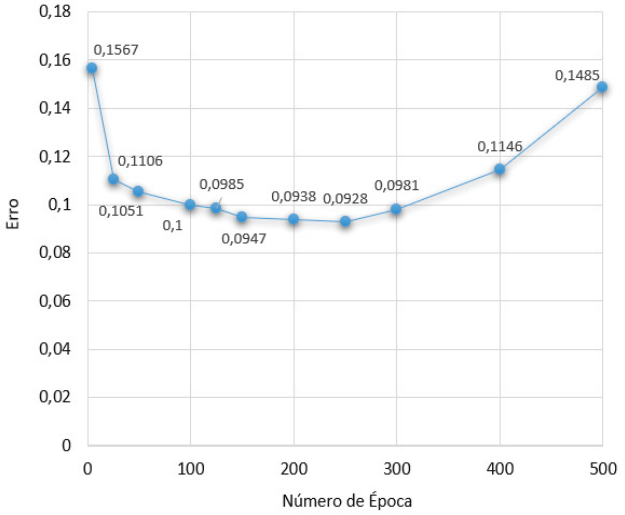


FIGURA 5 - VALORES DE ERRO EAM OBTIDOS VARIANDO O NÚMERO DE ÉPOCAS. FONTE: ELABORADO PELO AUTOR.



Todos os parâmetros definidos neste trabalho e utilizados sobre os modelos de redes neurais recorrentes diferentes (Simples RNN, LSTM e GRU) são mostrados na Tabela II.

TABELA II – PARÂMETROS UTILIZADO NOS MODELOS DE REDES NEURAIS RECORRENTES

Parâmetro	Valor
Número de Neurônios	5
Otimizador	Adam
Função de Ativação	Tangente Hiperbólica (tanh)
Número de Épocas	250

Como mencionado anteriormente, para obtenção dos resultados é realizado testes em 3 produtos diferentes usando 3 modelos de redes neurais recorrentes supracitados. Os gráficos resultantes dos testes é ilustrado a seguir de acordo com o produto.

TABELA III - VALORES MÍNIMOS DE EQM OBTIDOS DURANTE OS TESTES PARA PRODUTO E MODELO DE REDE NEURAL

Rede Neural / Produto	Simples RNN	LSTM	GRU
AR	0,0283	0,0169	0,0030
NFS-e	0,0241	0,0141	0,0031
PC	0,0251	0,0205	0,0017

Como pode-se observar na Tabela III, quando calculado o erro quadrático médio (EQM) dos modelos, o Simples RNN tem o erro relativamente maior do que dos outros dois para os três produtos. Enquanto, o erro do GRU é o menor de todos os modelos em ambos os produtos, no qual, a diferença é expressiva.

TABELA IV - VALORES MÍNIMOS DE EAM OBTIDOS DURANTE OS TESTES PARA PRODUTO E MODELO DE REDE NEURAL

Rede Neural / Produto	Simples RNN	LSTM	GRU
AR	0,1385	0,1168	0,0403
NFS-e	0,1278	0,0928	0,0404
PC	0,1291	0,1202	0,0309

Comprovando o que foi visto na Tabela III, a Tabela IV mostra também como o modelo de rede neural recorrente Simples RNN e LSTM são os menos precisos na predição do que o GRU. No caso do produto AR, a rede neural recorrente Simples RNN (Figura 6) conseguiu identificar o padrão relativamente bem no começo, mas a partir de uma alta expressiva que o produto teve não foi tão bem-sucedida quanto antes. A GRU (Figura 8) conseguiu identificar bem o padrão e teve bons resultados durante todo o treinamento, e a LSTM (Figura 7) no começo teve um desempenho semelhante à GRU mas também não conseguiu identificar tão bem o padrão depois de um aumento significante.

FIGURA 6 - VALORES REAIS E PREDITOS DO PRODUTO AR UTILIZANDO SIMPLES RNN. FONTE: ELABORADO PELO AUTOR.

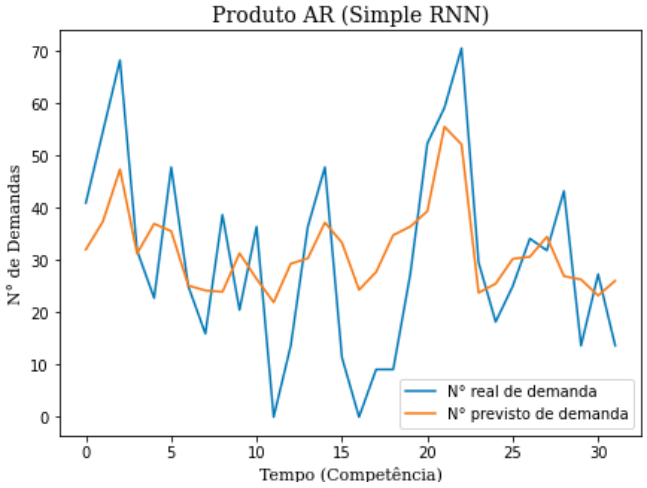


FIGURA 7 - VALORES REAIS E PREDITOS DO PRODUTO AR UTILIZANDO LSTM. FONTE: ELABORADO PELO AUTOR.

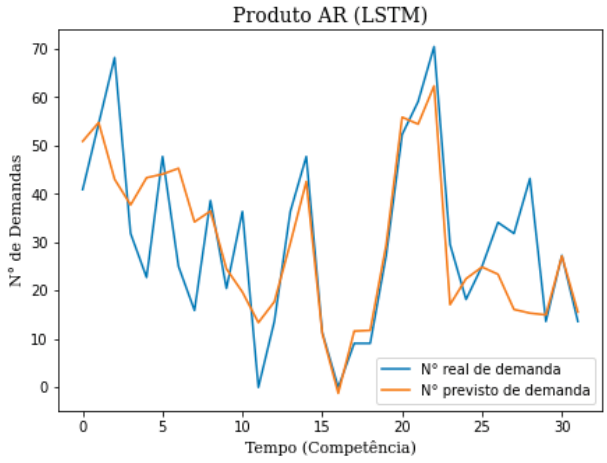
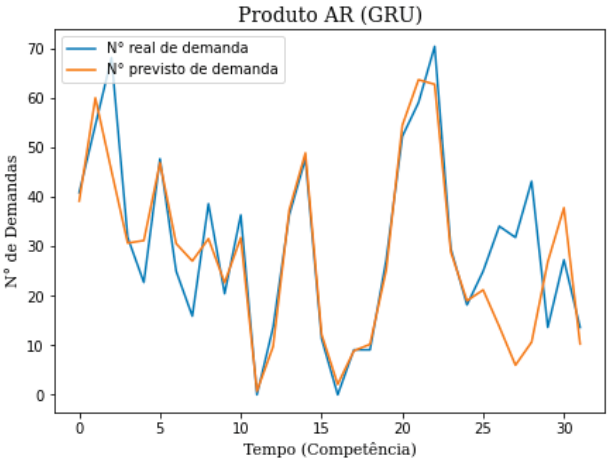


FIGURA 8 - VALORES REAIS E PREDITOS DO PRODUTO AR UTILIZANDO GRU. FONTE: ELABORADO PELO AUTOR.



Por sua vez, o produto de NFS-e, os modelos do Simples RNN (Figura 9) e LSTM (Figura 10) conseguem identificar bem o padrão, mas tem dificuldade para se ajustar mais precisamente após a competência 15. Já o modelo GRU foi o que apresentou mesmo erros conforme mencionados anteriormente, o que pode ser observado principalmente da competência 0 até a competência 25.

FIGURA 9 - VALORES REAIS E PREDITOS DO PRODUTO NFS-E UTILIZANDO SIMPLES RNN. FONTE: ELABORADO PELO AUTOR.

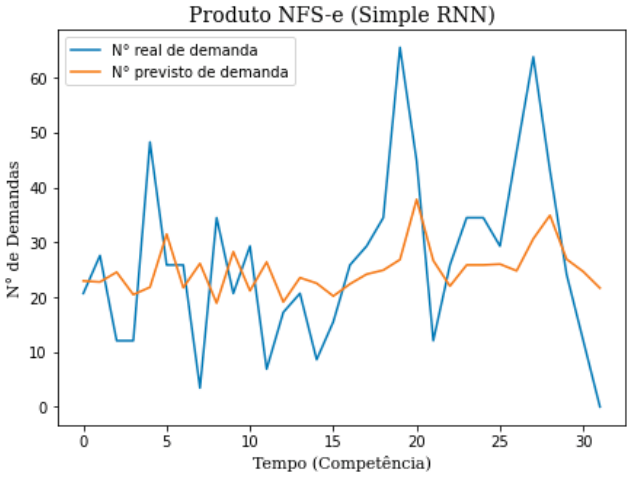


FIGURA 10 - VALORES REAIS E PREDITOS DO PRODUTO NFS-E UTILIZANDO LSTM. FONTE: ELABORADO PELO AUTOR.

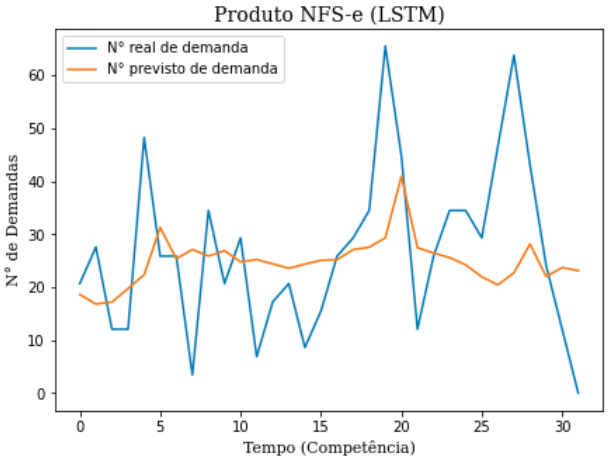
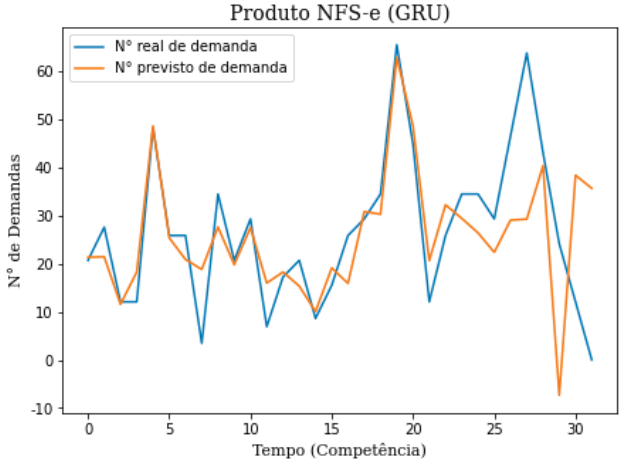


FIGURA 11 - VALORES REAIS E PREDITOS DO PRODUTO NFS-E UTILIZANDO GRU. FONTE: ELABORADO PELO AUTOR.



Por fim, o produto PC, o modelo GRU conseguiu se encaixar melhor ao conjunto de teste, obtendo erros menores do que quando comparado aos demais produtos e ainda obtendo desempenho superior ao do Simples RNN e LSTM (respectivamente, Figura 12 e Figura 13).

FIGURA 12 - VALORES REAIS E PREDITOS DO PRODUTO PC UTILIZANDO SIMPLES RNN. FONTE: ELABORADO PELO AUTOR.

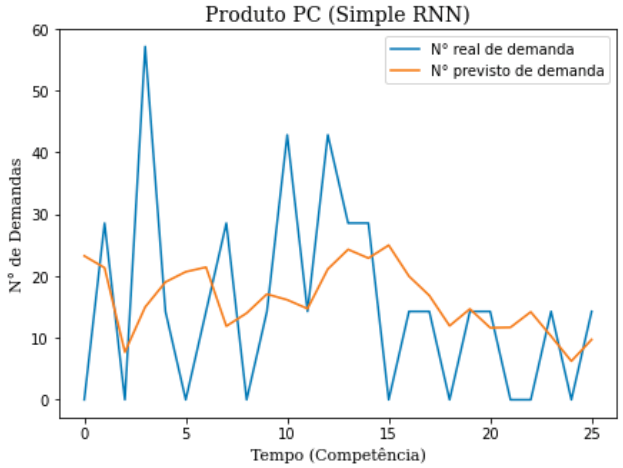


FIGURA 13 - VALORES REAIS E PREDITOS DO PRODUTO PC UTILIZANDO LSTM. FONTE: ELABORADO PELO AUTOR.

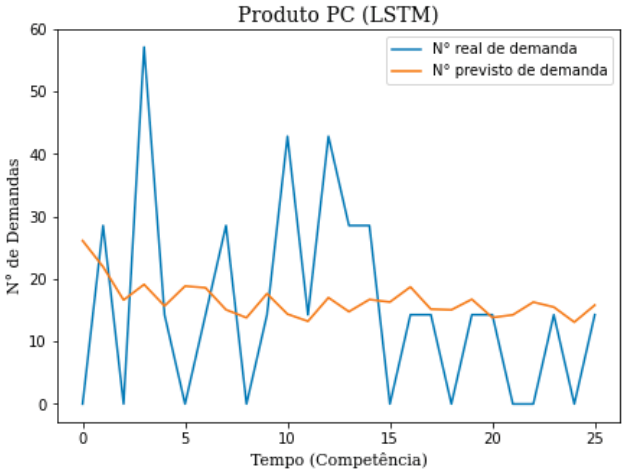
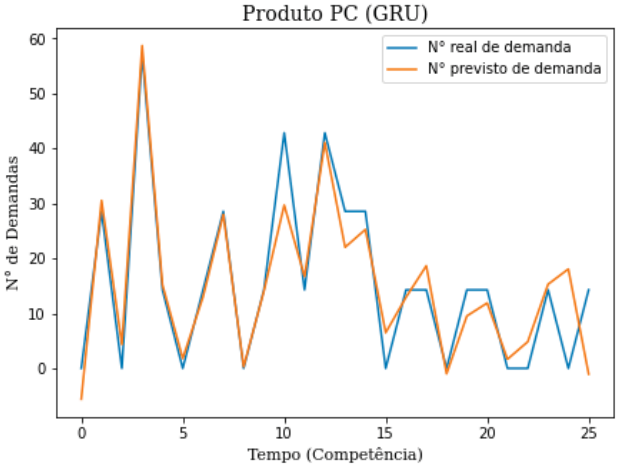


FIGURA 14 – VALORES REAIS E PREDITOS DO PRODUTO PC UTILIZANDO GRU. FONTE: ELABORADO PELO AUTOR.



III. CONSIDERAÇÕES FINAIS

Este trabalho de conclusão de curso abordou o desafio de utilizar técnicas de redes neurais recorrentes para prever as demandas corretivas sobre os produtos de uma fábrica de software. Para predizer o valor de três produtos diferentes (AR, NFS-e e PC), foram utilizadas abordagens com três tipos redes neurais recorrentes para cada uma: a rede neural recorrente Simples, a rede Long Short-Term Memory (LSTM), e a rede Gated Recurrent Unit (GRU).

A acurácia de cada modelo foi medida através do cálculo do erro quadrático médio (EQM) e do erro absoluto médio (EAM). Sendo que, a primeira abordagem, com o modelo Simples RNN, conseguiu captar os padrões de dados de teste, mas para nenhum dos produtos teve um resultado bom quando comparado aos outros modelos. O valor de EQM obtido nos testes ficou em torno de 0,0250, e os valores de EAM ficaram acima de 0,1250.

A segunda abordagem proposta, com o modelo LSTM, obteve resultados melhores com da primeira (Simples RNN), com valores de EQM e EAM perto de 0,0169 e 0,1168 respectivamente, sendo que a acurácia foi menor para o produto do AR. Apesar disso, o desempenho melhor para o AR pode ser resultado da otimização dos parâmetros, pois foi um dos modelos e produto em questão a ser utilizados para definir os parâmetros para todos os modelos.

A terceira e última abordagem, com o modelo GRU, obteve o melhor resultado para todos os produtos. O EQM desse modelo teve o menor valor de todos os testes (0,0017 com os dados de teste do produto PC), e também o menor valor de EAM com valor de 0,0309. Como o modelo GRU teve o melhor resultado em ambos os produtos da fábrica de software, pode-se dizer que foi o modelo que obteve melhor desempenho. Por sua vez, o LSTM teve um desempenho significativo, ocupando a segunda colocação em comparação aos modelos. Já o modelo Simples RNN, apesar de conseguir identificar padrões, teve o pior resultado para esse tipo de predição, com diferenças grandes de erro.

Com o resultado é possível explorar a viabilidade de uso de sistemas de predição que utilizam técnicas de Inteligência Artificial (IA) para a implementação de modelos de previsão de demandas corretivas sobre o portfólio de uma fábrica de software, com o intuito de contribuir com informações importantes para a gestão de produto ou projeto na tomada de decisões.

REFERÊNCIAS

[1] SOMMERVILLE, Ian. Engenharia de software. 8. ed. São Paulo: Pearson Education do Brasil, 2007. 552p.

[2] ROCHA, Ana Regina Cavalcanti da; MALDONADO, José Carlos; WEBER, Kival Chaves. Qualidade de software: teoria e prática. São Paulo: Prentice Hall, 2001. 303 p.

[3] IEEE, Institute. IEEE Standard for Software Maintenance. New York: Institute of Electrical and Eletronic Engineers. Inc., 1998, 52p.

[4] CORDEIRO, Marco Aurélio. Manutenibilidade de Software. Curitiba, 2009. Disponível em: http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=132. Acesso em: 22 de maio 2022.

[5] WEBER, Pedro Anselmo. Taskboarddev - ferramenta para monitoramento e rastreabilidade de atividades de manutenção de software baseada em conceitos ágeis. 2014. 72 f. Trabalho de Conclusão de Curso (Bacharelado em Sistema da Informação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau-SC.

[6] APRIL, A. Studying Supply and Demand of Software Maintenance and Evolution Services, 2010.

[7] ARTERO, A. O. (2009), "Inteligência Artificial Teórica e Prática", 1º ed., São Paulo, SP, Livraria da Física.

[8] JAMES, G. et al. An Introduction to Statistical Learning: with Applications in R. 2018. Disponível em: https://qubeshub.org/publications/847/1 . Citado na página 32.

[9] REIS, Carlos Henrique. Otimização de Hiperparâmetros em Redes Neurais Profundas. 2018. 72 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Federal de Itajubá- UNIFEI, Itajubá-MS.

[10] STATHAKIS, D. How many hidden layers and nodes? International Journal of Remote Sensing, Taylor Francis, v. 30, n. 8, p. 2133–2147, 2009. Disponível em: https://www.tandfonline.com/doi/full/10.1080/01431160802549278. Citado na página 34.

[11] JUNIOR, Flávio Pietrobon. (2014). Aplicação dos métodos de amortecimento exponencial para previsão de demanda de clientes. Disponível em: http://anteriores.aprepro.org.br/conbrepro/2014/anais/artigos/pesquisa20op/9.pdf

[12] KINGMA, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. Disponível em: https://arxiv.org/abs/1412.6980

[13] OLIVEIRA, Eric Lau. Redes Neurais Artificiais e Reconhecimento de Caracteres Numéricos em Imagens. 2018. 105 f. Trabalho de Conclusão de Curso (Bacharelado em Análise e Desenvolvimento de Sistemas), Fundação Educacional do Município de Assis, Assis-SP.

[14] SILVA, João Victor Laitano Coelho. Machine Learning para Detecção de faltas em um Sistema Elétrico. 2019. 80 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) – Departamento de Engenharia Elétrica - Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis-SC.

[15] TORCH. PyTorch Documentation. 2019. Disponível em: https://pytorch.org/docs/stable/index.html. Acesso em: 07 set. 2022.

[16] ABADI, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Software available from tensorflow.org. Disponível em: https://www.tensorflow.org/ . Acesso em: 07 set. 2022.

[17] FARIA, Lucas Teles. Sistema Inteligente Híbrido Intercomunicativo para Detecção de Perdas Comerciais. 2012. 112 f. Dissertação, Universidade Estadual Paulista -UNESP, Ilha Solteira-SP.