UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS RUSSAS
GRADUAÇÃO EM ENGENHARIA MECÂNICA

FELIPE BESSA DE SOUZA

ESTUDO DA APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS E DE REGRESSÕES PARA A PREDIÇÃO DE FORÇAS DE USINAGEM E RUGOSIDADE EM PROCESSOS DE TORNEAMENTO

RUSSAS
2021

FELIPE BESSA DE SOUZA

ESTUDO DA APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS E DE REGRESSÕES PARA A PREDIÇÃO DE FORÇAS DE USINAGEM E RUGOSIDADE EM PROCESSOS DE TORNEAMENTO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Mecânica do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Engenharia Mecânica.

Orientador: Prof. Dr. Pedro Helton Magalhães Pinheiro.

RUSSAS

2021

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

S238e Souza, Felipe Bessa de.
Estudo da aplicação de redes neurais artificiais e de regressões para a predição de forças de usinagem e rugosidade em processos de torneamento / Felipe Bessa de Souza. – 2021.
121 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia Mecânica, Russas, 2021.
Orientação: Prof. Dr. Pedro Helton Magalhães Pinheiro.

1. Torneamento. 2. Rede Neural Artificial. 3. Regressão. I. Título.

CDD 620.1

FELIPE BESSA DE SOUZA

ESTUDO DA APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS E DE REGRESSÕES PARA A PREDIÇÃO DE FORÇAS DE USINAGEM E RUGOSIDADE EM PROCESSOS DE TORNEAMENTO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Mecânica do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Engenharia Mecânica.

Orientador: Prof. Dr. Pedro Helton Magalhães Pinheiro.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____
Prof. Dr. Pedro Helton Magalhães Pinheiro (Orientador)
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. Edvan Cordeiro de Miranda
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. George Luiz Gomes de Oliveira
Universidade Federal do Ceará (UFC)

A Deus.

Aos meus pais, Jairo e Vilma.

AGRADECIMENTOS

Aos meus pais e à minha família, pelo apoio oferecido em todos os momentos e por acreditarem que a educação é o melhor caminho.

Ao Professor Dr. Pedro Helton Magalhães Pinheiro pela orientação e pela ajuda oferecida ao longo do Trabalho de Conclusão de Curso.

Aos meus amigos de graduação, que ofereceram momentos de alegria e de ajuda durante esses 5 anos.

A todos os professores que me acompanharam durante toda a minha graduação pela troca de conhecimentos e apoio para a minha evolução como profissional.

À instituição UFC Campos Russas pela oportunidade de realizar um curso de graduação de qualidade.

“O homem não é nada além daquilo que a educação faz dele.”
Immanuel Kant

RESUMO

Devido à alta competitividade do mercado e elevada exigências dos consumidores, a compreensão dos processos de manufatura torna-se cada vez mais importante para o aumento da produção e para a obtenção de produtos com características dentro de limites de tolerância aceitáveis. O torneamento é um dos processos de fabricação que possui maior relevância nas indústrias e a predição de características como as forças de usinagem e a rugosidade do material usinado possui uma fundamental importância para o planejamento da produção e para a escolha dos melhores parâmetros que afetam estas características. Com as estimativas das forças de usinagem é possível dimensionar de forma eficaz os motores dos tornos e a importância do estudo da rugosidade nessas operações de usinagem está relacionado a fatores como atrito e estética da peça final. Deste modo, este trabalho tem como objetivo o estudo da aplicação de modelos matemáticos como as redes neurais artificiais e modelos de regressão para diversos materiais e ferramentas de corte em processos de torneamento para encontrar funções que possam fornecer predições das forças e rugosidade da peça. Foi feito uma busca na literatura para encontrar dados experimentais que podiam ser utilizados nos modelos e também foi criado um programa em *Python* que aplica os modelos matemáticos e exibe todos os resultados. Foi possível concluir que a regressão linear apresentou o pior desempenho em relação aos dados de treinamento e houve uma tendência de sobreajuste aos dados de treinamento com a elevação do grau da regressão.

Palavras-chave: Torneamento. Rede Neural Artificial. Regressão.

ABSTRACT

Due to the high competitiveness of the market and high demands from consumers, understanding of manufacturing processes becomes increasingly important for increasing production and obtaining products with characteristics within acceptable tolerance limits. Turning is one of the manufacturing processes that has the greatest relevance in industries and the prediction of characteristics such as machining forces and the roughness of the machined material is of fundamental importance for the planning of production and for the choice of the best parameters that affect these characteristics. With the estimates of the machining forces it is possible to effectively dimension the lathes motors and the importance of studying the roughness in these machining operations is related to factors such as friction and aesthetics of the final part. Thus, this work aims to study the application of mathematical models such as artificial neural networks and regression models for various materials and cutting tools in turning processes to find functions that can provide predictions of the forces and roughness of the part. A literature search was made to find experimental data that could be used in the models and a Python program was also created that applies the mathematical models and displays all the results. It was possible to conclude that the linear regression presented the worst performance in relation to the training data and there was a tendency of overfitting the training data with the increase of the degree of regression.

Keywords: Turning. Artificial Neural Network. Regression.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ANOVA	Análise de variância
MAPE	Erro Absoluto Médio Padrão
MSE	Erro Quadrático Médio
RNA	Rede Neural Artificial
RL	Regressão Linear
RQ	Regressão Quadrática
R3	Regressão de 3° grau
R4	Regressão de 4° grau

LISTA DE SÍMBOLOS

v_c	Velocidade de corte
d	Diâmetro da peça
n	Rotação da peça
F_u	Força de usinagem
F_c	Força de corte
F_p	Força passiva
F_f	Força de avanço
k_s	Pressão específica de corte
A	Area da seção de corte
a_p	Profundidade de corte
f	Avanço
P_c	Potência de corte
P_f	Potência de avanço
P_m	Potência do motor
γ	Rendimento
R_a	Desvio aritmético médio
R_t	Altura total do perfil
R_z	Altura máxima do perfil
l_r	Comprimento de amostragem
l_n	Comprimento de avaliação
r_e	Raio de ponta da ferramenta
x	Sinal de entrada
w	Pesos
v	Combinação linear dos sinais de entrada;
b	Bias
φ	Função de ativação
E_m	Erro quadrático médio
E	Erro quadrático
η	Taxa de aprendizado
δ	Gradiente local
u	Resíduo

α Coeficiente linear

β Coeficiente angular

y_i Valor real medido

$\hat{y}_i$ Valor de predição

SUMÁRIO

1 INTRODUÇÃO

Ao longo da história, o avanço e o desenvolvimento da sociedade sempre estiveram ligados ao poder e habilidade de manipular os materiais para satisfazer as necessidades humanas (WILLIAM D. CALLISTER; RETHWISCH, 2013). Diante da elevada importância para a economia e da concorrência no meio industrial, a compreensão dos processos de manufatura torna-se cada vez mais importante para o aumento da produção e para a obtenção de produtos com características dentro de limites de tolerância aceitáveis.

Dentre a manufatura, a operação mais popular é a usinagem, que consiste em uma operação que produz cavaco enquanto confere à peça forma, acabamento e dimensões. Segundo Machado (2009), é um processo complexo, pois apresenta imprevisíveis condições ideais de corte que gerem a peça desejada. Porém, quando essas condições ideais são definidas, a usinagem é simplificada, pois o cavaco se forma corretamente, não necessitando de qualquer intervenção do operador e com o menor custo possível.

O torneamento é um dos processos de usinagem que possui maior relevância. Trata-se da retirada de material realizada em um torno no qual a peça gira em torno de seu eixo enquanto uma ferramenta de corte avança longitudinal e/ou transversalmente (MACHADO et al., 2009). O torneamento é utilizado para a criação de perfis arredondados e cilíndricos e possui aplicações básicas como o faceamento, perfilamento, recartilhamento, entre outras.

Diante do exposto, a predição de características como as forças de usinagem e a rugosidade do material usinado possui uma fundamental importância para o planejamento da produção e para a escolha dos melhores parâmetros que afetam estas características. Com as estimativas das forças de usinagem é possível dimensionar de forma eficaz os motores dos tornos e a importância do estudo da rugosidade nessas operações de usinagem está relacionado a fatores como atrito e estética da peça final. Direcionando os estudos para o torneamento, alguns dos principais parâmetros que afetam este processo são a velocidade de corte, a profundidade de corte e o avanço.

A utilização de redes neurais artificiais (RNA) para a resolução de problemas na usinagem vem ganhando espaço na literatura. As RNAs tratam-se de sistemas constituído por unidades de processamentos simples unidas por elevado número de conexões no qual podem calcular determinadas funções matemáticas (BRAGA; CARVALHO; LUDERMIR, 2000). Através desse sistema, é possível encontrar funções que expliquem o comportamento das variáveis do processo de torneamento. Além disso, é possível também encontrar estas funções

por meio de regressões polinomiais. Porém há poucos estudos na literatura que comparam esses métodos de estimativa para a rugosidade e forças de rugosidade.

Visando diminuir a imprevisibilidade dos parâmetros resultantes do torneamento e gerar comparações entre métodos de previsão, a proposta desse trabalho é o estudo da aplicação das RNAs e modelos de regressão para diversos materiais e ferramentas de corte em processos de torneamento para encontrar funções que possam fornecer predições das forças e rugosidade da peça.

1.1 Objetivo geral

O presente trabalho tem como objetivo geral o estudo da criação e implementação de modelos de redes neurais artificiais do tipo *perceptron* multicamadas e modelos de regressão para a predição de forças e da rugosidade de peças no processo de torneamento.

1.2 Objetivos específicos

- Elaboração de um algoritmo na linguagem de programação *Python* para a predição e para encontrar os parâmetros das RNAs que produzam maior precisão.
- Criação de uma ferramenta computacional com interface gráfica no qual os usuários possam obter as predições dos dados experimentais inseridos.
- Comparação dos modelos de RNAs e regressões através da implementação em dados experimentais encontrados na literatura.

2 REVISÃO BIBLIOGRÁFICA

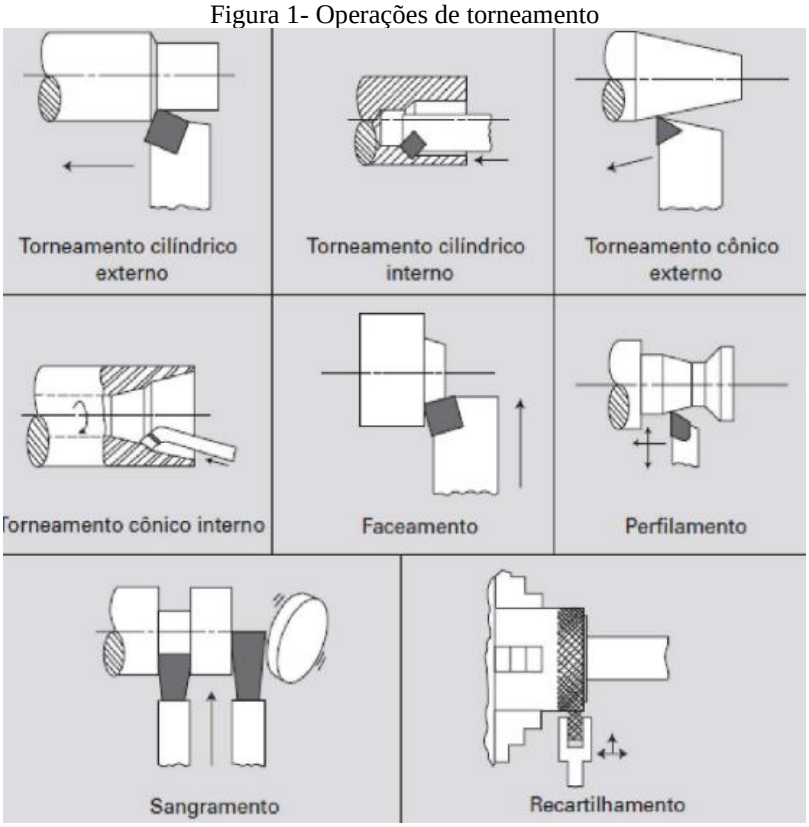
A revisão bibliográfica irá apresentar inicialmente o processo de torneamento, abordando os tipos de processo, parâmetros de corte e definindo as forças de usinagem e rugosidade da peça final. Em seguida será explicado os métodos de redes neurais artificiais e de regressão que foram utilizados para as previsões. Por último, há um tópico que exibe trabalhos relacionados.

2.1 Torneamento

Entende-se por usinagem as operações que produzem cavaco ao conceder à peça de trabalho as dimensões, ou a forma ou o acabamento, ou mesmo a combinação qualquer desses parâmetros. O cavaco trata-se da parte do material com forma geométrica irregular que é retirada pela ferramenta de corte. Dentre os processos mecânicos de usinagem, um dos que possui maior relevância é o torneamento. Nesta operação, a peça gira em torno de um eixo de rotação da máquina enquanto a ferramenta de corte se movimenta em uma trajetória coplanar ao eixo de rotação, obtendo superfícies de revolução.(FERRARESI, 1970)

Segundo Machado (2009), as principais operações que podem ser realizadas no torno são representadas na Figura 1:

Figura 1- Operações de torneamento



Fonte: Machado (2009)

2.1.1 Condições de corte

Abaixo serão expostos os principais parâmetros ou condições de corte utilizadas no torneamento.

2.1.1.1 Velocidade de corte

A velocidade de corte no torneamento é velocidade tangencial instantânea da rotação da ferramenta de corte em torno da peça de trabalho (DINIZ; MARCONDES; COPPINI, 2013). É calculada através da Equação 1:

$$v_c = \frac{\pi . d . n}{1000} \qquad Equação\ 1$$

Onde:

v_c = velocidade de corte (m/min)

d = diâmetro da peça (mm)

n = rotação da peça (rpm)

2.1.1.2 Avanço

O avanço ou $feed\ rate\ (f)$ refere-se à distância percorrida de avanço em cada volta ou em cada curso da ferramenta de corte. (DINIZ; MARCONDES; COPPINI, 2013)

2.1.1.3 Profundidade de usinagem

A profundidade de usinagem é a profundidade de penetração da ferramenta em comparação à peça, medida em uma direção perpendicular ao plano de trabalho. (MACHADO et al., 2009)

2.2 Forças durante a usinagem

O estudo para compreender o comportamento e a grandeza da força de usinagem que atua sobre a cunha cortante possui elevada importância, pois esse esforço afeta a potência necessária do motor da máquina, a temperatura de corte, o desgaste da ferramenta e na obtenção de tolerâncias adequadas para o projeto.(DINIZ; MARCONDES; COPPINI, 2013)

Segundo Machado (2009), a $força\ de\ usinagem\ (F_u)$ possui três componentes básicas que agem diretamente na cunha de corte da ferramenta:

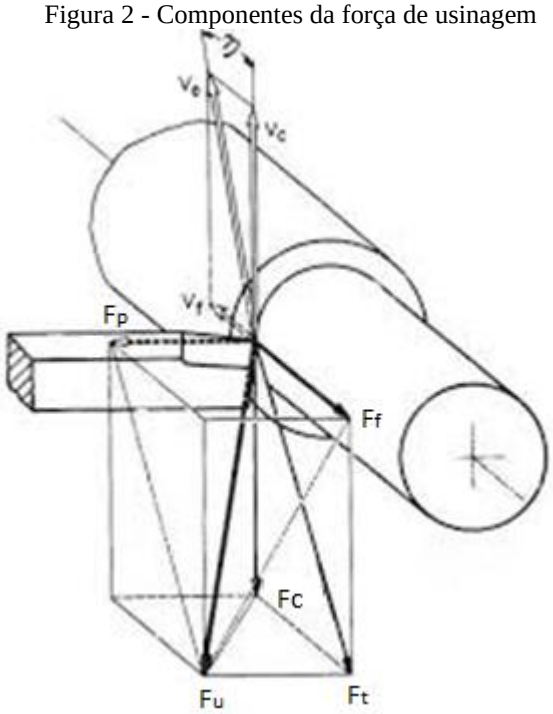
- $Força\ de\ corte\ ou\ força\ principal\ de\ corte\ (F_c)$: trata-se da projeção de F_u sobreo plano de trabalho na direção de corte, dada pela velocidade de corte.
- $Força\ passiva\ ou\ força\ de\ profundidade\ (F_p)$: projeção de F_u perpendicular ao plano de trabalho.
- $Força\ de\ avanço\ (F_f)$: projeção da força de usinagem sobre o plano de trabalho na direção de avanço.

Deste modo, a força de usinagem pode ser definida pela Equação 2:

$$F_u = \sqrt{F_c + F_p + F_f} \qquad Equação\ 2$$

Esses componentes da força de usinagem podem ser visualizados na Figura 2:

Figura 2 - Componentes da força de usinagem



Fonte: Diniz (2013)

2.2.1 Determinação teórica da força de corte

Segundo Machado (2009), a força de corte teórica pode ser determinada pela Equação 3.

$$F_c = k_s \cdot A = k_s \cdot a_p \cdot f \qquad Equação\ 3$$

No qual:

k_s= pressão específica de corte

A= área da seção de corte

a_p= profundidade de corte

f= avanço

A pressão específica de corte trata-se da força necessária para a remoção de uma área de corte equivalente a 1mm². É medida em laboratório para cada par ferramenta/peça em função dos parâmetros de corte. O seu valor varia segundo estes fatores:

- Material da peça;

- Material e geometria da ferramenta;
- Área da seção de corte;
- Velocidade de corte;
- Condições de lubrificação e refrigeração;
- Desgaste da ferramenta.

Entre as equações utilizadas para encontrar a pressão específica de corte, a mais utilizada é a de Kienzle, que é definida pela Equação 4.

$$k_s = k_{sl} \cdot k^{1-Z} \qquad Equação\ 4$$

No qual k_{sl} e $1 - Z$ são definidos experimentalmente para materiais específicos.

2.2.2 Potências de usinagem

Para executar os movimentos de corte e avanço, a máquina-ferramenta produz potência para girar o seu eixo-árvore. Segundo Diniz (2013), estas potências podem ser calculadas pelas Equações 5 e 6.

Potência de corte:

$$P_c = \frac{F_c \cdot v_c}{60 \cdot 75} \qquad Equação\ 5$$

Potência de avanço:

$$P_f = \frac{F_f \cdot v_f}{60 \cdot 75} \qquad Equação\ 6$$

No qual P_c e P_f são dados (CV), $F_c\ e\ F_f$ em (Kgf), v_c e v_f em (m/min).

Diniz (2013) afirma que a relação P_c/P_f apresenta valores muito elevados, e em exemplos extremos que buscam os menores valores dessa relação, foi encontrado que a potência de avanço é aproximadamente 140 vezes menor que a potência de corte. Deste modo, em

máquinas que apresentam somente um motor para realizar, tanto o movimento de avanço, quanto o movimento de corte, a potência de avanço pode ser desprezada no dimensionamento desse motor. A potência fornecida pelo motor é dada então pela Equação 7.

$$P_m = \frac{P_c}{\gamma} \qquad Equação\ 7$$

No qual γ é o rendimento da máquina operatriz. Pode variar entre 60 e 80% em máquinas convencionais que utilizam caixa de engrenagens para transmissão e é maior que 90% em máquinas CNC que apresentam variação contínua de rotação e possuem poucos elementos de transmissão de movimento.

2.2.3 Efeito dos parâmetros na força de usinagem

Utilizando um experimento fatorial completo com o aço EN31, Bartarya *et al.* (2012) utilizou a análise de variância para obter a influência dos parâmetros de usinagem nos esforços. Foi observado que para a força de corte, a profundidade de corte foi o parâmetro mais significativo, seguido pelo avanço. Para a força radial, o parâmetro que mais afetou foi a profundidade de corte com 71,56% de contribuição na variância.

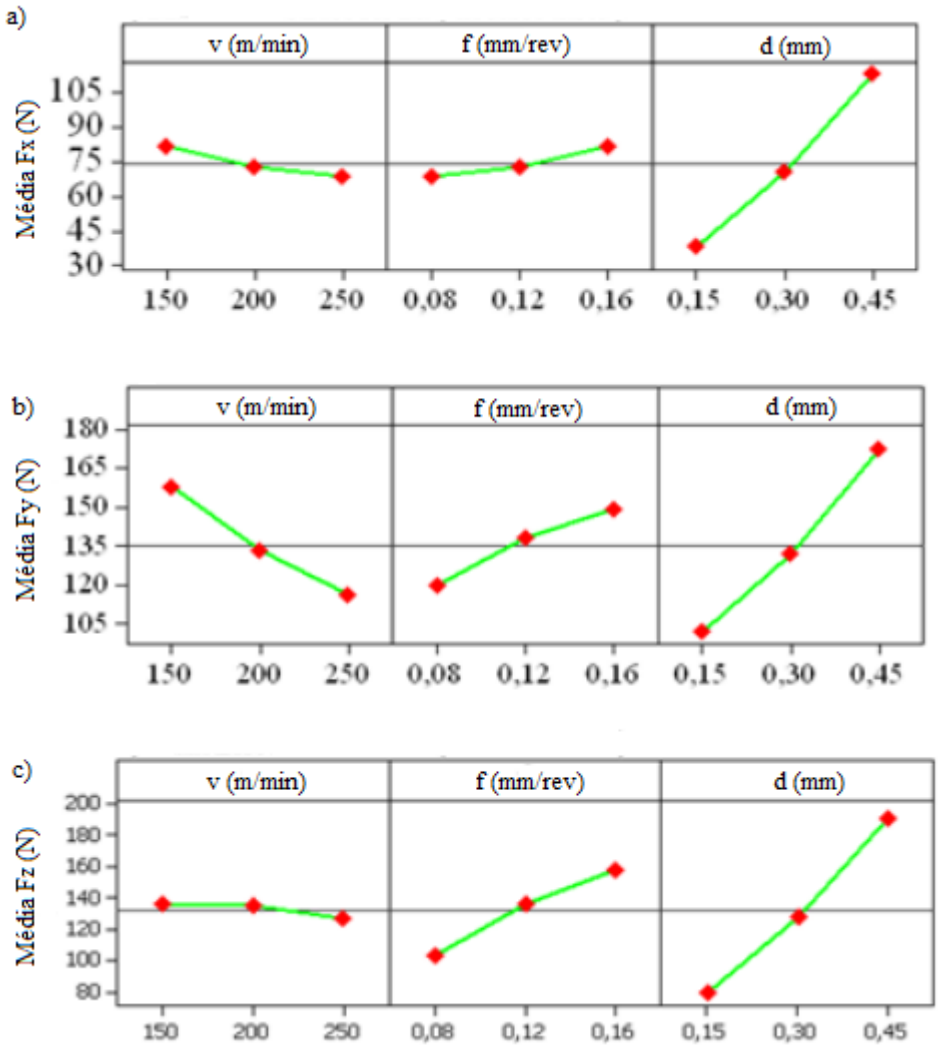
Chinchanikar *et al.* (2013) estudou o efeito da dureza e dos parâmetros de corte no aço AISI 4340 e concluiu que as forças de corte foram maiores no material de maior dureza. Além disso, observou que o parâmetro que mais afetou os esforços foram a profundidade com contribuição entre 60 e 70%, seguido pelo avanço com contribuição entre 25 e 30%.

Fnides (2011) analisou a usinagem do aço AISI H11 com a ferramenta de cerâmica CC650 e concluiu que a profundidade de corte foi o parâmetro com mais influência nas componentes da força de usinagem. As contribuições para F_f, F_p e F_c foram 94,22, 81,14 e 77,84%, respectivamente. O segundo fator foi o avanço com contribuições de 1,72, 10,69 e 16,15%, respectivamente. A velocidade de corte foi o fator menos efetivo.

Para Laouissi (2019) no estudo do ferro fundido EN-GJL-250 com insertos de cerâmicas revestidos e não revestidos, foi obtido que a profundidade de corte obteve maior contribuição com 64,73 e 69,63% para o inserto sem revestimento e com revestimento, respectivamente. Foi seguido pelo avanço e pela velocidade com (20,09 e 22,78%) e (1,3 e 3,74%), respectivamente.

Em Keblouti (2017) na análise do AISI 52100, a profundidade também foi o fator mas significante, contribuindo para as componentes F_f, F_p e F_c com (64,81%, 61,24% e 43,01%) e (88,23%, 55,09% e 71,73%) para os insertos com e sem revestimento, respectivamente. A Figura 3 exibe essa influência dos parâmetros nas componentes da força para o inserto sem revestimento.

Figura 3 - Influência dos parâmetros nas forças. a)Fx b) Fy c)Fz



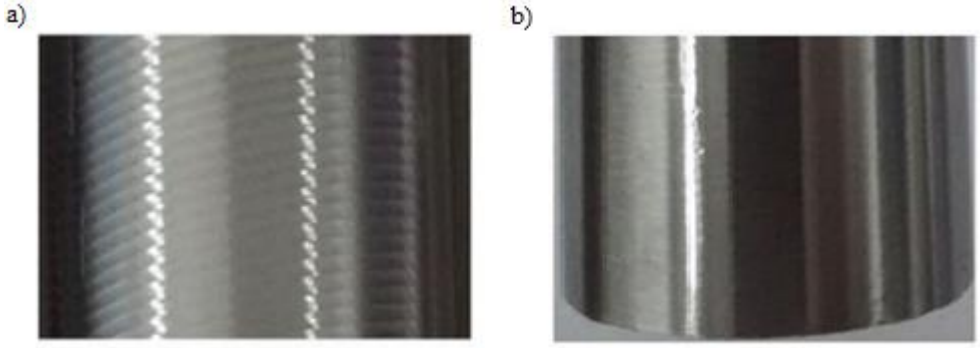
Fonte: Keblouti *et al.*(2017)

2.3 Rugosidade

Devido a exigências de acabamento de determinados projetos, é fundamental o estudo de rugosidade do produto final do processo de usinagem. Influenciada pela ação inerente ao processo de corte (desgaste da ferramenta, marcas de avanço, aresta postiça, etc) e pelos parâmetros de usinagem escolhidos para o processo, a rugosidade da superfície da peça irá

apresentar irregularidades ou erros geométricos de diferentes intensidades, como mostra a Figura 4, no qual a usinagem com diferentes condições de corte pode gerar acabamentos grosseiros ou lisos. Deste modo, a rugosidade torna-se um parâmetro de saída para controlar e selecionar os parâmetros de um processo de usinagem. (MACHADO et al., 2009).

Figura 4 - Acabamento em peças. a) Acabamento grosseiro b) Acabamento liso



Fonte: Biasibetti *et al*. (2019)

2.3.1 Tipos de rugosidade

Para a medição da rugosidade, há diversos parâmetros que podem ser utilizados. Os principais, segundo a norma ABNT NBR ISSO 4287 (2002), estão representados no Quadro 1.

Quadro 1 - Parâmetros de rugosidade

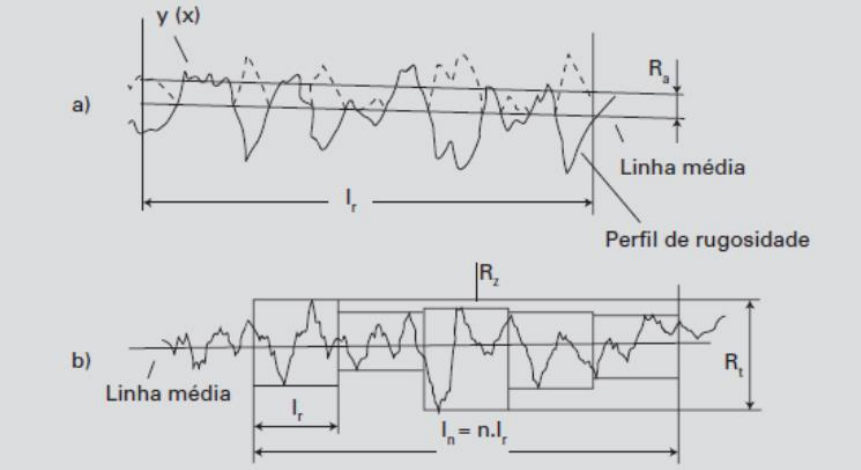
Símbolo	Nome	Definição
$\boldsymbol{R_a}$	Desvio aritmético médio	Média aritmética dos valores absolutos das ordenadas no comprimento de amostragem.
$\boldsymbol{R_q}$	Desvio médio quadrático	Raiz quadrada da média dos valores das ordenadas no comprimento de amostragem.
$\boldsymbol{R_t}$	Altura total do perfil	Soma da maior altura de pico do perfil e da maior profundidade de vale do perfil no comprimento de avaliação.
$\boldsymbol{R_z}$	Altura máxima do perfil	Soma da altura máxima dos picos e a maior das profundidades dos vales no comprimento de amostragem.
$\boldsymbol{R_{sk}}$	Fator de assimetria do perfil (skewness)	Quociente entre o valor médio dos valores das ordenadas e $\boldsymbol{R_q}$ ao cubo, no comprimento da amostragem.
$\boldsymbol{R_{ku}}$	Fator de achatamento do perfil	Quociente entre o valor médio dos valores das ordenadas à quarta potência no comprimento de amostragem.

Fonte: Machado (2009)

A Figura 5 apresenta a forma de medição dos parâmetros R_a, R_t e R_z, no qual o comprimento de amostragem (l_r) é o comprimento na direção do eixo X para identificar as

irregularidades da superfície sob avaliação, enquanto o comprimento de avaliação (l_n) é o comprimento que pode conter um ou mais comprimentos de amostragem.

Figura 5 - Representação dos parâmetros de usinagem



Fonte: Machado (2009)

2.3.2 Cálculo teórico da rugosidade no processo de torneamento

Segundo Machado (2009), é possível ter uma estimativa teórica para R_a e R_t, apesar de ter erros relacionados a vibração e desgaste das arestas de corte da ferramenta. Se o avanço (f) é menor que o raio de ponta da ferramenta (r_e), tem-se as Equações 8 e 9:

$$R_a = \frac{f^2}{31{,}2.r_e} \qquad Equação\ 8$$

$$R_t = \frac{f^2}{8.r_e} \qquad Equação\ 9$$

2.3.3 Efeito dos parâmetros de corte na rugosidade

Buscando compreender como os parâmetros de corte impactam na rugosidade dos produtos usinado no torneamento, diversos pesquisadores fizeram estudos para quantificar esta influência.

Através da análise da variância (ANOVA) em um experimento envolvendo a variação, três fatores, três níveis e dois tipos de insertos, Keblouti *et al.* (2017) encontrou a influência da velocidade de corte, avanço e profundidade de corte na rugosidade do aço AISI

52100. Foi observado que o avanço foi o parâmetro que mais afetou a rugosidade, contribuindo com 85,17% e 85,67% para o inserto com cobertura e sem cobertura, respectivamente. O segundo parâmetro mais efetivo foi a velocidade de corte, contribuindo com 4,91% e 7,55% para inserto com cobertura e sem cobertura, respectivamente. A profundidade de corte foi o fator com menos influencia, contribuindo com menos de 2%.

Ramanujam *et al.* (2011) também utilizou a análise variância para o estudo sobre o compósito Al-SiC (10p) –MMC e concluiu que o parâmetro que mais influenciou na rugosidade foi a profundidade de corte, contribuindo com 32,23%. O segundo mais importante parâmetro foi a velocidade de corte com 19,59%. O avanço contribui somente com 7,48%.

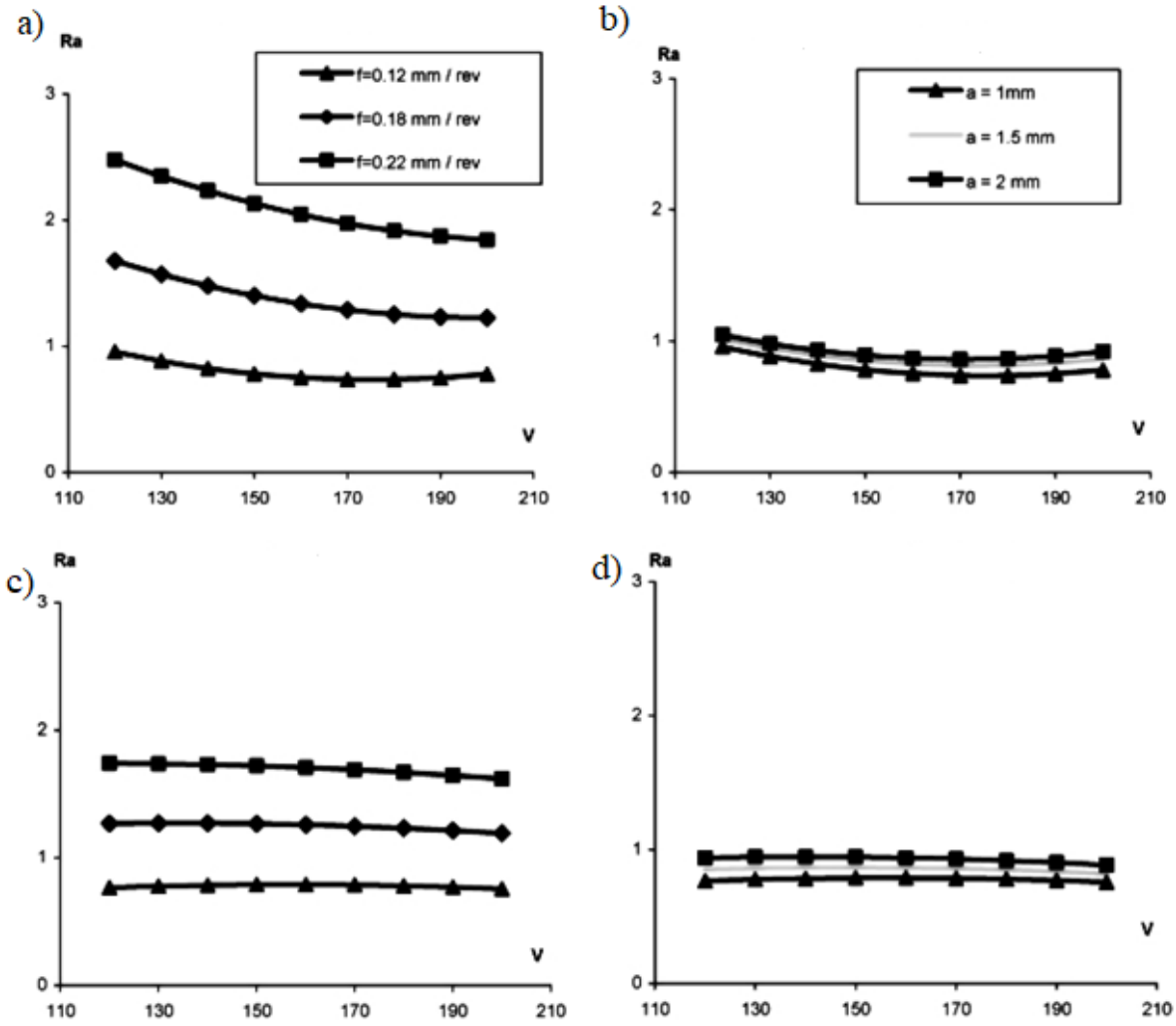
Aouci *et al.* (2013) investigou altura total (R_t) e a altura máxima do perfil (R_z) além da rugosidade média (R_a) para o aço X38CrMoV5-1. Foi observado que o avanço e a velocidade de corte foram os fatores que mais impactaram nos parâmetros de rugosidade, tendo influência de mais de 33% cada em todos os casos.

Lima *et al.* (2005) analisou o torneamento do aço AISI 4340 com duas durezas diferentes (42 e 50 HRC) e concluiu que o acabamento da superfície era melhorado quando a velocidade de corte era aumentada e deteriorada com o aumento do avanço. A profundidade de corte não apresentou efeito considerável.

Meddour *et al.* (2015) utilizou a análise de variância para o aço AISI 52100 e encontrou que os fatores que mais influenciaram na determinação da rugosidade foram o raio de ponta da ferramenta e o avanço com 41,83 e 35,39% de contribuição, respectivamente.

Cakir *et al.* (2009) desenvolveu um modelo de regressão para a rugosidade do aço AISI P20 usinado com o inserto CNMG 120408 com dois tipos diferentes de cobertura e investigou os efeitos dos parâmetros de corte. Foi observado, como demonstra a Figura 6, que o avanço foi o fator de mais efeito. Este fato pode ser explicado pela diminuição das forças de corte devido a diminuição do avanço, pois forças menores provocam menos vibrações e, consequentemente, superfícies melhores acabadas. Outro fator que afetou de forma considerável a rugosidade foi a velocidade de corte, seu aumento provoca a diminuição da rugosidade. O argumento para este fato também pode ser a diminuição das forças de corte com o aumento da velocidade de corte.

Figura 6 - Efeito dos parâmetros de corte na rugosidade. a) a=1mm para inserto 1
b) f=0.12mm/rev para inserto 1 c) a=1mm para inserto 2 d) f=0.12/rev para inserto 2



Fonte: Cakir *et al.* (2009)

2.4 Redes neurais

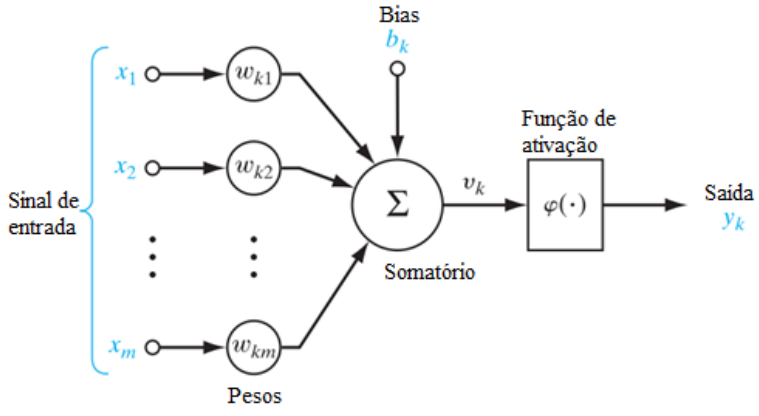
As redes neurais artificiais (RNA) são sistemas paralelos que utilizam unidades de processamentos simples, denominados nodos, que irão calcular determinadas funções matemáticas. Esses nodos são organizados em camadas e interligados por conexões no qual são associados a pesos que servem para ponderar o sinal de entrada em cada neurônio. O procedimento usual para a utilização das redes neurais na solução de problemas inicia-se na fase de aprendizagem, no qual um conjunto de exemplos é apresentado a rede, a qual consegue obter de forma automatizada as informações e características necessárias para explicar os dados fornecidos (BRAGA; CARVALHO; LUDERMIR, 2000).

Devido ao paralelismo relacionado ao arranjo dos nodos ao longo da rede, as RNAs possibilitam, em geral, um desempenho superior aos modelos convencionais. Além disso, a principal vantagem das RNAs trata-se de aprender com exemplos e generalizar a partir dos dados fornecidos. Esta generalização refere-se à capacidade do aprendizado com um pequeno conjunto de exemplos e a obtenção de respostas coerentes para dados que não foram utilizados no treinamento. Isto significa que as RNAs não somente mapeiam relações entre entrada e saída, elas são capazes de obter informações não apresentadas de forma explícita pelo conjunto de dados de treinamento. (BRAGA; CARVALHO; LUDERMIR, 2000)

2.4.1 Modelo de um neurônio

O neurônio é a unidade fundamental de processamento de informações da rede neural, que é demonstrado na Figura 7.

Figura 7 - Modelo de um neurônio



Fonte: Adaptado de Haykin (2008)

Segundo Haykin (2008), é possível identificar três elementos básicos em um neurônio:

- Conexões, também chamada de sinapses, são caracterizadas por um peso próprio. O sinal de entrada x_j da sinapse j conectada ao neurônio k é multiplicado pelo peso w_{kj}. A primeira letra do subscrito do peso refere-se ao neurônio e o segundo subscrito a sinapse do dado de entrada.
- Somatório para os sinais de entrada multiplicados pelos seus respectivos pesos.
- Função de ativação que irá determinar a amplitude do sinal de saída.

Além desses elementos, o neurônio pode conter o bias b_k, que irá aumentar ou diminuir o dado de entrada da função de ativação.

As equações que representam o neurônio são as Equações 10 e 11:

$$v_k = \sum_{j=1}^{m} w_{kj} x_j \qquad Equação\ 10$$

$$y_k = \varphi(v_k + b_k) \qquad Equação\ 11$$

onde:

x_1, x_2, ..., x_m= sinais de entrada;

w_{k1}, w_{k2}, ..., w_{km}= pesos do neurônio k;

v_k = combinação linear dos sinais de entrada;

b_k = bias;

$\varphi(\cdot)$ = função de ativação;

y_k = saída do neurônio.

2.4.2 Funções de ativação

A utilização da função de ativação no neurônio tem o objetivo de realizar uma transformação não linear do sinal de entrada e que será transmitido para as etapas posteriores da rede neural. Serão expostas abaixo duas importantes funções de ativação apresentadas em Nwankpa *et al.* (2018).

A função tangente hiperbólica (tanh) é centrada em 0 e apresenta uma extensão entre -1 e 1 e é representada pela Equação 12:

$$\varphi(x) = \left(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}\right) \qquad Equação\ 12$$

A função Rectified Linear Unit (ReLU) define os valores de entrada atribuiu o valor de 0 para valores menores que 0. É definida pela Equação 13:

$$\varphi(x) = \max(0, x) = \begin{cases} x_{m,} & se\ x_m \geq 0 \\ 0, & se\ x_m < 0 \end{cases} \qquad Equação\ 13$$

2.4.3 Arquiteturas

A escolha da arquitetura da RNA é um parâmetro fundamental, pois ela restringe o tipo de problema que a rede pode solucionar (BRAGA; CARVALHO; LUDERMIR, 2000). Segue abaixo alguns exemplos de propriedades da arquitetura.

Número de camadas:

- Rede de camada única: só existe um nó entre o sinal de entrada e saída;
- Rede com múltiplas camadas: há mais de um neurônio entre algum sinal de entrada e algum sinal de saída.

Conexões dos nodos:

- *Feedforward:* ou acíclica, o dado de saída de um neurônio na *i*-ésima camada não pode ser utilizados como entrada em camadas com índice menor ou igual a *i*;
- *Feedback:* ou cíclica, o dado de saída de um neurônio na *i*-ésima camada pode ser utilizados em camadas com índice menor ou igual a *i*.

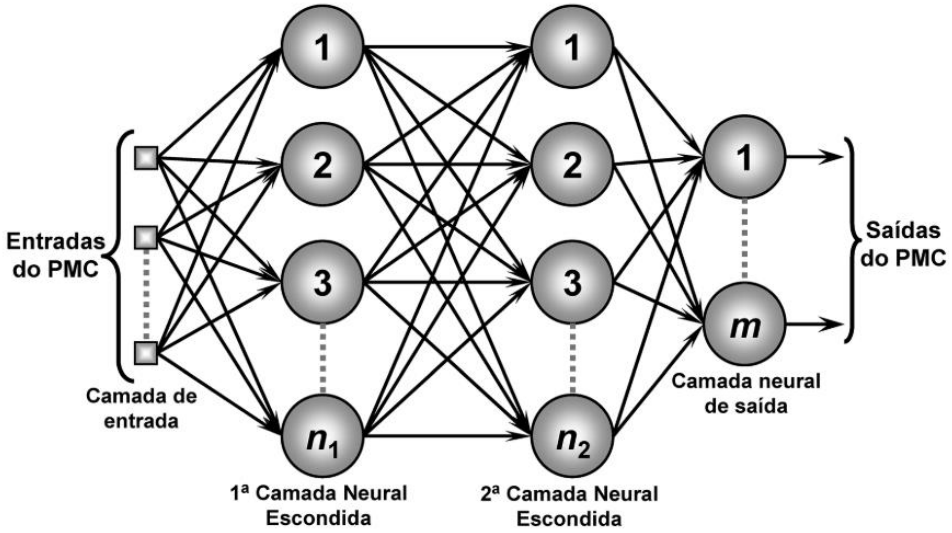
Conectividade:

- Rede fracamente conectada;
- Rede completamente conectada.

2.4.4 Rede neural artificial Perceptron de múltiplas camadas (PMC)

As redes Perceptron multicamadas possuem a presença de pelo menos uma camada intermediária de neurônios, denominada escondida, localizada entre a camada de entrada e a camada de saída. O fluxo de informações de dados e de informações inicia-se na camada de entrada, logo após segue nas camadas escondidas, sendo o último destino a camada de saída. Além disso, a propagação dos sinais de entrada da rede é realizada sempre no sentido da camada de entrada para a camada de saída (SILVA; SPATTI; FLAUZINO, 2010). A Figura 8 demonstra um exemplo de uma rede PMC com duas camadas intermediárias.

Figura 8 - Exemplo de rede PMC



Fonte: Silva, Spatti e Flauzino (2010)

2.4.5 Treinamento da rede

Segundo Braga *et al.* (2000), o algoritmo mais conhecido para o treinamento das redes PMC é o *back-propagation*. Neste algoritmo é utilizado pares de dados (entrada, saída desejada) para ajustar os pesos por meio da correção dos erros. O treinamento possui duas fases, na primeira, denominada *forward*, é definida uma saída para um determinado padrão de entrada. Já na segunda fase, chamada *backward*, a saída desejada e a saída dada pela rede são comparadas e utilizadas para atualizar os pesos das camadas.

O algoritmo segue os seguintes passos na fase *forward*:

1. O sinal de entrada é apresentado à primeira camada, C^0;

2. As camadas seguintes, C^i, calculam suas respectivas saídas e estas servem como entrada para as os nodos da camada C^{i+1};
3. As saídas da última camada são comparadas com as saídas desejadas.

A fase *backward* é definida por:

1. Seguindo o fluxo da última camada até a camada de entrada, os nodos da camada em analise ajustam os pesos para reduzir os erros. Já o erro das camadas escondidas é calculado utilizando os erros das camadas seguintes ponderados pelos pesos das conexões.

Para o ajuste dos pesos, Silva *et al.* (2010) expõe as equações destinadas à obtenção de uma rede PMC com duas camadas ocultas e que serão mostradas abaixo.

2.4.5.1 Função do erro

Primeiramente é necessário definir as matrizes de elementos da rede, representadas pelas Equações 14 e 15:

$$V_j^{(L)} = \sum_{i=0}^{n} W_{ji}^{(L)} \cdot x_i \qquad Equação\ 14$$

$$Y_j^{(L)} = \varphi\left(V_j^{(L)}\right) \qquad Equação\ 15$$

onde:

$W_{ji}^{(L)}$= matrizes de pesos que conectam o *j-ésimo* neurônio da camada (L) ao *i-ésimo* neurônio da camada (L-1);

$V_j^{(L)}$= entrada ponderada do *j-ésimo* neurônio da camada (L);

$Y_j^{(L)}$= saída do *j-ésimo* neurônio da camada (L).

O parâmetro utilizado para o ajuste dos pesos é o erro quadrático médio representado pela Equação 17, que é obtido através da média da Equação 16:

$$E(k) = \frac{1}{2}\sum_{j=1}^{n_3}\left(d_j(k) - Y_j^{(3)}(k)\right)^2 \qquad Equação\ 16$$

$$E_m = \frac{1}{p}\sum_{k=1}^{p} E(k) \qquad Equação\ 17$$

onde:

$Y_j^{(3)}(k)$= valor da saída do *j-ésimo* neurônio para a *k-ésima* amostra de treinamento;

$d_j(k)$= saída desejada para a *k-ésima* amostra de treinamento;

$E(k)$= erro quadrático para a *k-ésima* amostra de treinamento.

2.4.5.2 Ajuste dos pesos da camada de saída

Considerando como um procedimento interativo, o ajuste do peso da camada de saída é defino pela Equação 18:

$$W_{ji}^{(3)}(t+1) = W_{ji}^{(3)}(t) + \eta\delta_j^{(3)}Y_i^{(2)} \qquad Equação\ 18$$

no qual η é definido como a taxa de aprendizado e $\delta_j^{(3)}$ é o gradiente local relacionado a camada de saída definido pela Equação 19.

$$\delta_j^{(3)} = \left(d_j - Y_j^{(3)}\right)\cdot\varphi'\left(V_j^{(3)}\right) \qquad Equação\ 19$$

2.4.5.3 Ajuste dos pesos da segunda camada escondida

O ajuste dos pesos sinápticos para a segunda camada intermediaria e o gradiente local são expressos por pelas Equações 20 e 21, respectivamente:

$$W_{ji}^{(2)}(t+1) = W_{ji}^{(2)}(t) + \eta\delta_j^{(2)}Y_i^{(1)} \qquad Equação\ 20$$

$$\delta_j^{(2)} = -\left(\sum_{k=1}^{n_3}\delta_k^{(3)}\cdot W_{kj}^{(3)}\right)\cdot\varphi'\left(V_j^{(2)}\right) \qquad Equação\ 21$$

2.4.5.4 Ajuste dos pesos da primeira camada escondida

As Equações 22 e 23 representam o ajuste para o primeiro peso da camada intermediária e o gradiente local, respectivamente.

$$W_{ji}^{(1)}(t+1) = W_{ji}^{(1)}(t) + \eta\delta_j^{(1)}x_i \qquad Equação\ 22$$

$$\delta_j^{(2)} = \left(\sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)}\right) \cdot \varphi'\left(V_j^{(1)}\right) \qquad Equação\ 23$$

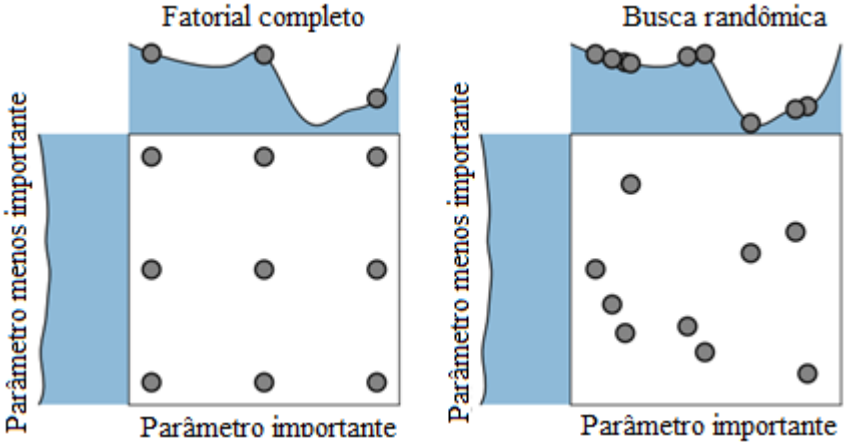
2.4.6 Seleção e otimização dos hiperparâmetros

Responsáveis por controlar o processo de treinamento, a escolha adequada dos hiperparâmetros são fundamentais para a criação de modelos eficientes. A busca pelo número de camadas e neurônios, taxa de aprendizado e funções de ativação, por exemplo, que minimizam o erro associado a rede neural é uma pratica que deve ser realizada para encontrar resultados coerentes.

Um dos métodos mais utilizados para a seleção dos hiperparâmetros é o experimento fatorial completo. É especificado um conjunto finito de valores para cada hiperparâmetro e são o produto cartesiano desses conjuntos são avaliados. O principal problema associado a esse método é que o número de experimentos cresce exponencialmente com o aumento do conjunto de valores dos hiperparâmetros (HUTTER; KOTTHOFF; VANSCHOREN, 2019).

Outro método para a otimização dos hiperparâmetros é a busca randômica. Neste método são testadas combinações aleatórias de valores que estão em um intervalo determinado pelo executor do experimento. A busca randômica consegue ter melhor performance do que o fatorial completo quando há hiperparâmetros mais importantes que outro, pois quando é considerado um número B de configurações, o número de diferentes valores que o fatorial completo consegue avaliar para um número N de categoria de hiperparâmetros é $B^{1/N}$, enquanto a busca randômica consegue avaliar B valores, como é demonstrado na Figura 9 (HUTTER; KOTTHOFF; VANSCHOREN, 2019).

Figura 9 - Comparação entre fatorial completo e busca randômica



Fonte: Hutter e Kotthoff (2019)

2.4.7 Sobre-ajuste e sub-ajuste

Fenômenos não desejados podem surgir após o treinamento de alguns modelos estatísticos. O sub-ajuste, também denominado *underfitting*, ocorre quando o modelo criado é simples demais para aprender e reconhecer os padrões do dados de treinamento, o que acarreta um erro elevado em relação ao dados de treinamento e de teste (GERON, 2017). Para a solução deste problema é sugerido:

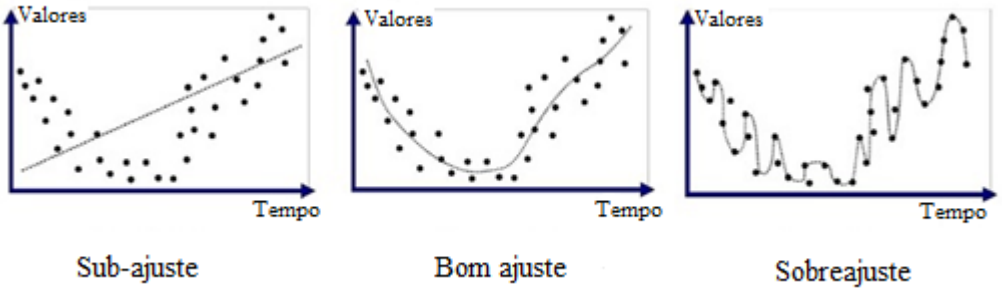
- Selecionar modelos mais complexos com mais parâmetros;
- Reduzir as restrições do modelo.

O oposto do sub-ajuste é o sobre-ajuste (*overfitting*), ocorre quando o modelo apresenta uma ótima performance em relação aos dados de treinamento, porém não consegue uma boa generalização (GERON, 2017). Deste modo, os modelos que apresentam sobre-ajuste apresentam performances precárias com dados de testes que não foram apresentados ao modelo durante o treinamento. São sugeridas as seguintes mudanças para haver uma maior generalização:

- Simplificação do modelo reduzindo o número de parâmetros;
- Reunir mais dados de treinamento;
- Reduzir o ruído dos dados de treinamento.

A Figura 10 demonstra estes dois fenômenos, além de um modelo que seria ideal, pois consegue bons resultados com os dados de treinamento e é generalista.

Figura 10 – Sub-ajuste e sobre-ajuste

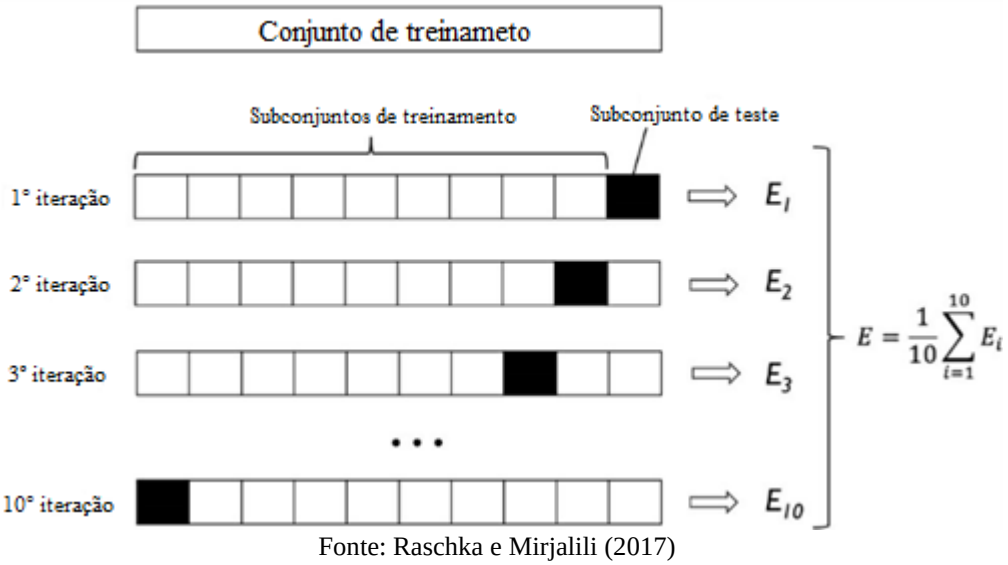


Fonte: Branco (2020)

2.4.8 Validação cruzada K-fold

A validação cruzada *K-fold* pode ser utilizada para encontrar os melhores hiperparâmetros para gerar um modelo com boa performance na generalização. Este método de amostragem consiste na separação de forma aleatória dos dados de treinamento em *k* divisões sem substituiçao, no qual (*k-1)* divisões serão utilizadas para o treinamento, enquando a divisão restante será utilizada para avaliar o erro do modelo (RASCHKA; MIRJALILI, 2017). Esta metodologia será repetida *k* vezes e no final será calculada a média dos erros como é demonstrado na Figura 11.

Figura 11 - Validação cruzada *K-fold*



Fonte: Raschka e Mirjalili (2017)

Após a obtenção de hiperparâmetros otimizados, todo o conjunto de dados de treinamento podem ser utilizados para o aprendizado e no final pode ser avaliado a performance

do modelo utilizando os dados de testes que não foram apresentados na fase de treinamento (RASCHKA; MIRJALILI, 2017).

2.5 Regressão linear múltipla

A regressão linear múltipla trata-se de um modelo matemático no qual o valor de uma variável dependente pode ser representada por uma função linear de duas ou mais variáveis independentes (HOFFMANN, 2016). A regressão pode ser representada por *k* variáveis explanatórias como exposto na Equação 24:

$$Y_j = \alpha + \sum_{i=1}^{k} \beta_i X_{ij} + u_j, \qquad j = 1, \dots, n \qquad Equação\ 24$$

onde:

Y_j= variável independente;

X_{ij}= variáveis dependentes;

u_j= resíduo;

α= coeficiente linear;

β_i= coeficiente angular.

Há também a representação com notação matricial, representado pela Equação 25:

$$\mathbf{y} = \mathbf{X\beta} + \mathbf{u} \qquad Equação\ 25$$

no qual:

$$\mathbf{y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 1 & X_{11} & X_{21} & \dots & X_{k1} \\ 1 & X_{12} & X_{22} & \dots & X_{k1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & X_{1n} & X_{2n} & \dots & X_{kn} \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

2.5.1 Método dos mínimos quadrados

Segundo Hoffmann (2016), é possível encontrar estimadores para os parâmetros α e β através do método dos mínimos quadrados. Esta metodologia adota estimadores que minimizam a soma dos quadrados dos desvios entre os valores estimados e os valores observados na amostra. A estimativa dos parâmetros da regressão é dada pelas Equações 26 e 27:

$$\mathbf{b} = \begin{bmatrix} a \\ b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix} \qquad Equação\ 26$$

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \qquad Equação\ 27$$

2.6 Regressão polinomial

Segundo Massart *et al.* (1997), a regressão múltipla pode também ser utilizada para resolver problemas de regressão polinomial. Utilizando $X_{1j} = X_j$, $X_{2j} = X_j^2$, ..., $X_{mj} = X_j^m$ na equação 24, é possível encontrar o polinômio com *m* grau, dado pela Equação 28:

$$Y_j = \alpha + \beta_1 X_j + \beta_2 X_j^2 + \cdots + \beta_m X_j^m, \qquad j = 1, \ldots, n \qquad Equação\ 28$$

Quando há mais de uma variável explanatória, o número de termos do polinômio cresce rapidamente como é possível ver na Equação 29 que representa um modelo quadrático, pois além dos termos lineares e quadráticos, existe os termos de cruzamento entre as variáveis como $b_{12}x_1x_2$. (MASSART et al., 1997)

$$y = \alpha + b_1x_1 + b_2x_2 + b_3x_3 + b_{11}x_1^2 + b_{22}x_2^2 + b_{33}x_3^2 + \\ + b_{12}x_1x_2 + b_{13}x_1x_3 + b_{23}x_2x_3 \qquad Equação\ 29$$

2.7 Indicadores de acurácia

Para avaliar a acurácia e adequação dos modelos de previsão ao dados fornecidos, podem ser utilizados os seguintes indicadores, representados pelas Equações 30, 31 e 32: (HANIEF; WANI; CHAROO, 2017)

$$Erro\ absoluto\ médio\ percentual\ (MAPE) = \frac{1}{N}\sum_{i=1}^{N}\left(\left|\frac{y_i - \hat{y}_i}{y_i}\right|\right) \qquad Equação\ 30$$

$$Erro\ médio\ quadrático\ (MSE) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad Equação\ 31$$

$$Coef.de\ determinação\ (R^2) = \left(\frac{n(\sum y_i\hat{y}_i) - (\sum y_i)(\sum \hat{y}_i)}{\sqrt{[n\sum {y_i}^2 - (\sum y_i)^2]\left[n\sum {\hat{y}_i}^2 - (\sum \hat{y}_i)^2\right]}}\right)^2 \quad Equação\ 32$$

No qual y_i é o valor real medido e $\hat{y}_i$ é o valor de predição dado pelos modelos matemáticos.

Quanto mais próximo de 1 for o valor de R^2, menor será a soma dos quadrados do resíduo, o que indica um melhor ajuste aos dados fornecidos. Porém, se o valor se aproximar de 0 indica que o somatório dos resíduos será elevado e o ajuste do modelo será precário. (FAHRMEIR et al., 2013)

2.8 Trabalhos relacionados

Visando a análise de alguns trabalhos que elaboraram modelos matemáticos para a predição e a influência dos parâmetros de usinagem nos esforços de corte e da rugosidade da superfície em diferentes condições experimentais, foram elaboradas as Tabelas 1 e 2. Nestas são apresentados os materiais, os tipos de insertos utilizados em cada estudo, a quantidade de dados experimentais utilizados e os coeficientes de determinação do modelo em relação aos dados utilizados para o treinamento, validação e teste do modelo.

Em relação aos tipos de modelos utilizados, é possível observar que a regressão quadrática e a rede neural artificial apresentam elevados coeficientes de determinação, todos

sendo superiores a 0,9 em relação aos dados utilizados em treinamento. A única abordagem analisada que utilizou a regressão linear foi em Deshpande *et al.* (2017), e os coeficientes apresentaram- se inferiores aos demais modelos.

Além disso, somente Hanief *et al.* (2017) e Asiltürk (2011) abordam a utilização de um grupo de dados somente para teste das redes neurais para avaliar a capacidade de generalização do modelo. Os coeficientes de determinação para este tipo de avalição apresentaram-se superiores a 0,99. Hanief *et al.* (2017) utilizou também um grupo de validação para a seleção dos parâmetros do modelo.

Tabela 1- Trabalhos relacionados (força)

Referência	Material da peça	Ferramenta de corte	Método	Grandeza	Qnt. dados treino	R^2 treino	Qnt. dados validação	R^2 validação	Qnt. dados teste	R^2 teste
(Aouci 2013)	X38CrMoV5-1	CBN7020	Regressão quadrática	Fx	27	0.9516	-	-	-	-
				Fy		0.9222	-	-	-	-
				Fz		0.9316	-	-	-	-
(Chinchanikar 2013)	AISI 4340 (35 HRC)	KC9110	Regressão quadrática	Fx	20	0.98	-	-	-	-
				Fy		0.98	-	-	-	-
				Fz		0.98	-	-	-	-
	AISI 4340 (45 HRC)	KC9110	Regressão quadrática	Fx	20	0.97	-	-	-	-
				Fy		0.97	-	-	-	-
				Fz		0.99	-	-	-	-
(Deshpande 2017)	Inconel 718	TNMG 160408 sem revestimento	Regressão linear	Fz	20	0.812	-	-	-	-
		TNMG 160408 com revestimento	Regressão linear	Fz	20	0.7073	-	-	-	-
(Keblouti 2017)	AISI 52100 steel	GC 1525 com revestimento	Regressão quadrática	Fx	27	0.9258	-	-	-	-
				Fy		0.955	-	-	-	-
				Fz		0.9252	-	-	-	-
		GC 1525 sem revestimento	Regressão quadrática	Fx	27	0.9513	-	-	-	-
				Fy		0.9119	-	-	-	-
				Fz		0.9379	-	-	-	-
(Laouissi 2019)	EN-GJL-250	CC1690	Regressão quadrática	Fz	27	0.98	-	-	-	-
			Rede neural	Fz	27	0.99	-	-	-	-
		CC6090	Regressão quadrática	Fz	27	0.96	-	-	-	-
			Rede neural	Fz	27	0.99	-	-	-	-
(Mata 2010)	PEEK CF30	TiN- coated	Regressão quadrática	Fx	27	0.99	-	-	-	-
				Fy		0.92	-	-	-	-
				Fz		0.99	-	-	-	-
(Hanief 2017)	C23000	HSS	Rede neural	Fr	19	1	4	0.99911	4	0.9995
(Lin 2001)	S55C	TNMG160404L2G	Regressão quadrática	Fz	27	0.9963	-	-	-	-

Fonte: Autor (2020)

Tabela 2 - Trabalhos relacionados (rugosidade)

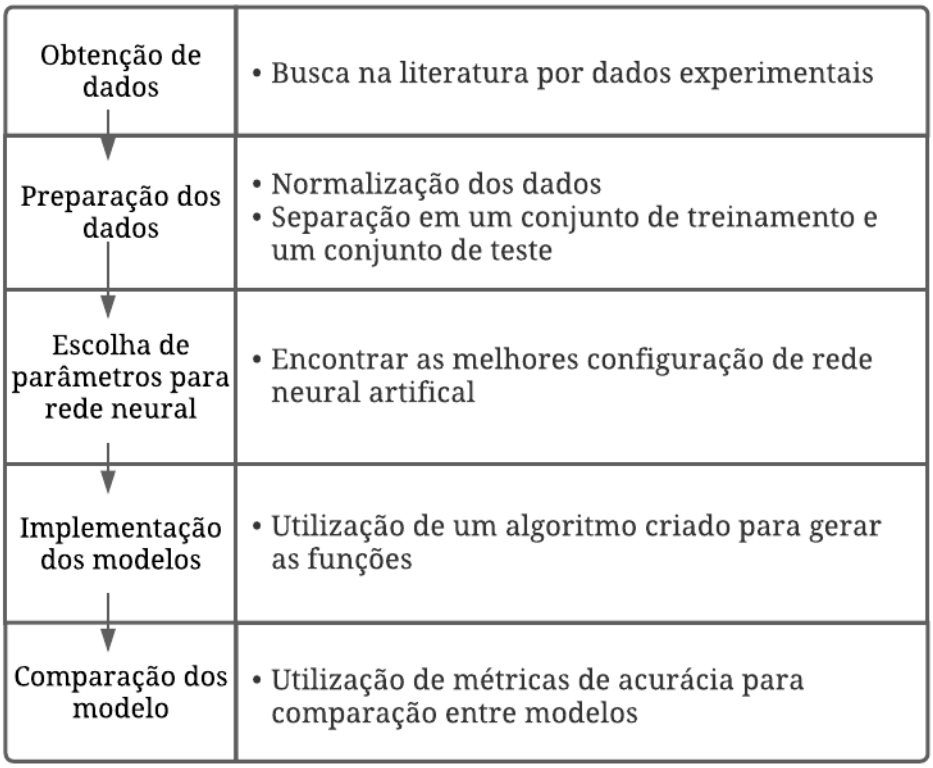
Referência	Material da peça	Ferramenta de corte	Método	Grandeza	Qnt. dados treino	R^2 treino	Qnt. dados validação	R^2 validação	Qnt. dados teste	R^2 teste
(Aouci 2013)	X38CrMoV5-1	CBN7020	Regressão quadrática	Ra	27	0.9813	-	-	-	-
				Rt		0.97	-	-	-	-
				Rz		0.9722	-	-	-	-
(Asiltürk 2011)	AISI 1040	MWLNR 25X25	Regressão quadrática	Ra	27	0.9892	-	-	-	-
			Rede neural	Ra	21	0.9989	-	-	6	0.9945
(Asiltürk 2012)	AISI 304	SNMG 120408-PP	Regressão quadrática	Ra	27	0.91	-	-	-	-
				Rz	27	0.91	-	-	-	-
(Laouissi 2019)	EN-GJL-250	CC1690	Regressão quadrática	Ra	27	0.96	-	-	-	-
			Rede neural	Ra	27	0.99	-	-	-	-
		CC6090	Regressão quadrática	Ra	27	0.98	-	-	-	-
			Rede neural	Ra	27	0.99	-	-	-	-
(Keblouti 2017)	AISI 52100 steel	GC 1525 com revestimento	Regressão quadrática	Ra	27	0.9466	-	-	-	-
		GC 1525 sem revestimento	Regressão quadrática	Ra	27	0.9601	-	-	-	-
(Lin 2001)	S55C high carbon steel	TNMG160404L2G	Regressão quadrática	Ra	27	0.9951	-	-	-	-

Fonte: Autor (2020)

3 METODOLOGIA

Neste capitulo serão apresentadas as etapas que foram utilizadas para o desenvolvimento do trabalho. Como é mostrado na Figura 12, a metodologia utilizada foi separada em 5 partes: obtenção de dados de estudos, tratamento destes dados através da normalização e separação, seleção dos melhores parâmetros para a rede neural, aplicação dos modelos apresentados e por último, a comparação da eficácia dos modelos.

Figura 12 - Metodologia



Fonte: Autor (2020)

3.1 Obtenção de dados

A primeira etapa consiste na busca de estudos que contenham experimentos no qual foram elaborados experimentos que mediram as forças e rugosidade em relação a alteração do avanço, velocidade e profundidade de corte. As Tabelas 3 e 4 apresenta todos os estudos encontrados que podem ser utilizados para a análise dos modelos de previsão.

Tabela 3 - Estudos com dados de força

Número	Referência	Tipo	Ferramenta de corte	Material da peça	Número de dados
1	(Lin 2001)	Fr	TNMG160404L2G	S55C	27
2	(Thangarasu 2018)	Fr	TiCN com revestimento	Aço macio	27
3	(Borsos 2017)	Fx	CNMG 120408KM	AISI 1045	27
4	(Lalwani 2008)	Fx	TNMA160408S01525	MDN250	20
5	(Fnides 2011)	Fx	CC650	AISI H11 (50 HRC)	27
6	(Bartarya 2012)	Fx	TNGA160408 S01525	EN31 (60±2 HRc)	27
7	(Chinchanikar2013)	Fx	KC9110	AISI 4340 (35 HRC)	20
8	(Chinchanikar2013)	Fx	KC9110	AISI 4340 (45 HRC)	20
9	(Keblouti 2017)	Fx	GC 1525 com revestimento	AISI 52100	27
10	(Keblouti 2017)	Fx	CT5015 sem revestimento	AISI 52100	27
11	(Mata 2010)	Fx	TiN com revestimento	PEEK CF30	27
12	(Lalwani 2008)	Fy	TNMA160408S01525	MDN250	20
13	(Fnides 2011)	Fy	CC650	AISI H11 (50 HRC)	27
14	(Borsos 2017)	Fy	CNMG 120408KM	AISI 1045	27
15	(Bartarya 2012)	Fy	TNGA160408 S01525	EN31 (60±2 HRc)	27
16	(Chinchanikar2013)	Fy	KC9110	AISI 4340 (35 HRC)	20
17	(Chinchanikar2013)	Fy	KC9110	AISI 4340 (45 HRC)	20
18	(Keblouti 2017)	Fy	GC 1525 com revestimento	AISI 52100	27
19	(Keblouti 2017)	Fy	CT5015 sem revestimento	AISI 52100	27
20	(Mata 2010)	Fy	TiN com revestimento	PEEK CF30	27
21	(Lalwani 2008)	Fz	TNMA160408S01525	MDN250	20
22	(Fnides 2011)	Fz	CC650	AISI H11 (50 HRC)	27
23	(Borsos 2017)	Fz	CNMG 120408KM	AISI 1045	27
24	(Bartarya 2012)	Fz	TNGA160408 S01525	EN31 (60±2 HRc)	27
25	(Chinchanikar2013)	Fz	KC9110	AISI 4340 (35 HRC)	20
26	(Chinchanikar2013)	Fz	KC9110	AISI 4340 (45 HRC)	20
27	(Keblouti 2017)	Fz	GC 1525 com revestimento	AISI 52100	27
28	(Keblouti 2017)	Fz	CT5015 sen revestimento	AISI 52100	27
29	(Deshpande 2017)	Fz	TNMG 160408 sem revestimento	Inconel 718	20
30	(Deshpande 2017)	Fz	TNMG 160408 com revestimento	Inconel 718	20
31	(Laouissi 2019)	Fz	CC1690	EN-GJL-250	27
32	(Laouissi 2019)	Fz	CC6090	EN-GJL-250	27
33	(Mata 2010)	Fz	TiN com revestimento	PEEK CF30	27
34	(Shankar 2015)	Fz	CNMG 120408	77 BHN	27

Fonte: Autor (2020)

Tabela 4 - Estudos com dados de rugosidade

Número	Referência	Tipo	Ferramenta de corte	Material da peça	Número de dados
1	(Lin 2001)	Ra	TNMG160404L2G	S55C	27
2	(Bartarya 2012)	Ra	TNGA160408 S01525	(60±2 HRc)	27
3	(Chinchanikar2013)	Ra	KC9110	AISI 4340 (35 HRC)	20
4	(Chinchanikar2013)	Ra	KC9110	AISI 4340 (45 HRC)	20
5	(Keblouti 2017)	Ra	GC 1525 com revestimento	AISI 52100	27
6	(Keblouti 2017)	Ra	CT5015 sem revestimento	AISI 52100	27
7	(Deshpande 2017)	Ra	TNMG 160408 sem revestimento	Inconel 718	20
8	(Deshpande 2017)	Ra	TNMG 160408 com revestimento	Inconel 718	20
9	(Laouissi 2019)	Ra	CC1690	EN-GJL-250	27
10	(Laouissi 2019)	Ra	CC6090	EN-GJL-250	27
11	(Asiltürk 2011)	Ra	WNMG 080408-TF	AISI 1040	27
12	(Asiltürk 2012)	Ra	SNMG 120408-PP	AISI 304	27
13	(Cakir 2009)	Ra	CNMG 120408 1	AISI P20	27
14	(Cakir 2009)	Ra	CNMG 120408 2	AISI P20	27
15	(Galanis 2010)	Ra	DNMG 110402-M3	AISI 316L	27
16	(Suhail 2010)	Ra	CNMG 432 TT5100	AISI 1020	27
17	(Asiltürk 2012)	Rz	SNMG 120408-PP	AISI 304	27

Fonte: Autor (2020)

A Tabela 5 apresenta um exemplo de um desses estudos no qual foram feitas 27 medições dos esforços e da rugosidade durante o torneamento do AISI 52100 para encontrar os efeitos de diferentes insertos e parâmetros de corte nas grandezas medidas, e estas medições podem ser utilizadas no presente trabalho para fazer as previsões através dos modelos matemáticos propostos no capitulo 2.

Tabela 5 - Exemplo de dados

V (m/ min)	*f* (mm/ rev)	*d* (mm)	Inserto GC1525				Inserto CT5015			
			Fx (N)	Fy (N)	Fz (N)	Ra (μm)	Fx (N)	Fy (N)	Fz (N)	Ra (μm)
150	0.08	0.15	48.78	106.65	92.37	0.47	26.74	98.50	60.33	0.96
		0.3	52.12	104.90	89.16	0.45	69.49	128.57	95.03	0.8
		0.45	76.83	131.45	129.35	0.51	113.01	165.04	148.22	0.6
	0.12	0.15	53.14	108.53	117.65	0.78	46.60	119.56	78.65	0.77
		0.3	65.34	114.25	136.24	0.89	76.57	160.59	138.04	0.9
		0.45	102.24	162.04	187.13	0.84	129.13	195.12	190.15	0.96
	0.16	0.15	70.34	127.91	137.54	1.15	43.52	132.53	88.89	1.39
		0.3	83.28	158.70	148.13	1.18	81.32	175.13	174.19	1.45
		0.45	127.96	187.70	211.59	1.42	146.82	244.63	241.24	1.46
200	0.08	0.15	43.05	88.41	78.30	0.52	37.12	88.44	53.55	0.4
		0.3	53.82	110.37	104.31	0.4	70.28	126.06	98.54	0.42
		0.45	69.13	143.15	103.67	0.51	111.24	142.98	149.41	0.47
	0.12	0.15	46.11	88.02	120.49	0.7	46.66	108.76	88.73	0.72
		0.3	54.67	107.77	118.49	0.82	65.15	144.21	145.84	0.74
		0.45	89.91	127.84	161.24	0.7	94.91	155.90	181.57	0.93
	0.16	0.15	48.27	108.91	140.93	1.06	41.06	102.04	89.19	1.37
		0.3	62.89	127.37	149.46	1.06	76.10	150.93	178.32	1.37
		0.45	110.81	169.54	212.82	1.3	108.12	176.50	225.57	1.44
250	0.08	0.15	47.52	107.29	69.76	0.49	37.59	95.14	69.75	0.32
		0.3	74.66	115.53	122.98	0.39	54.41	107.01	99.29	0.37
		0.45	89.87	141.56	138.00	0.5	95.51	124.58	144.59	0.37
	0.12	0.15	43.88	85.18	112.92	0.46	25.47	77.47	74.98	0.57
		0.3	46.50	102.21	124.55	0.75	57.64	117.12	125.01	0.72
		0.45	72.53	144.55	164.30	0.58	108.06	162.28	198.91	0.83
	0.16	0.15	50.35	108.29	127.92	1.02	39.22	92.78	97.91	1.26
		0.3	71.33	128.96	147.20	0.9	84.21	79.65	88.96	1.27
		0.45	123.36	155.66	220.28	1.15	111.50	186.04	233.37	1.3

Fonte: Borsos *et al.* (2017)

3.2 Normalização e separação dos dados

Para facilitar o treinamento e a convergência dos modelos, é adequado normalizar os dados. Será utilizada a Equação 33.

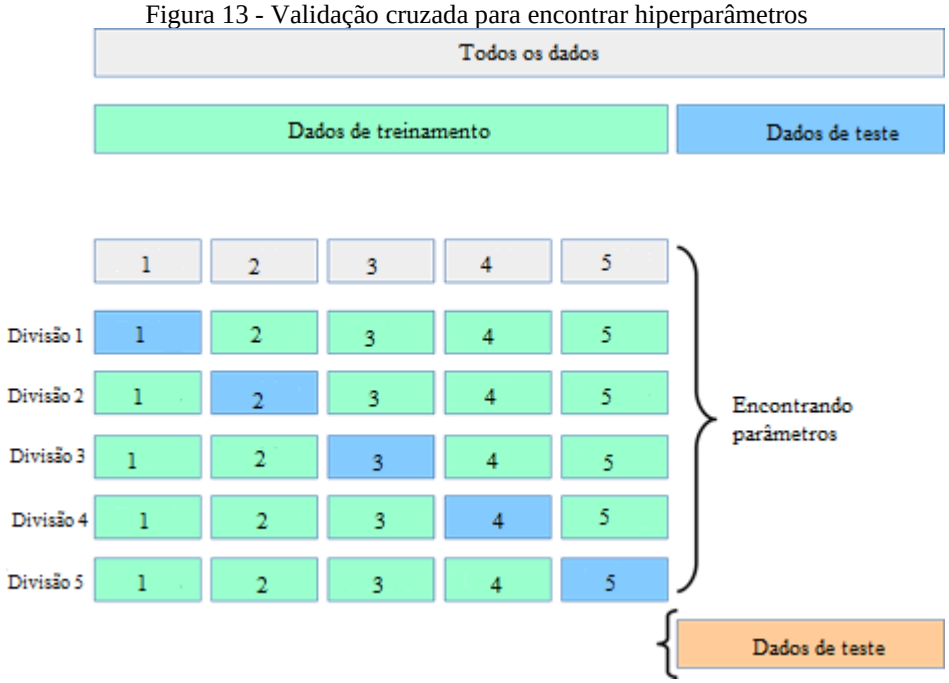
$$Y_n = \frac{Y - \bar{Y}}{\sigma} \qquad Equação\ 33$$

Além disso, foi feita a separação dos dados para treinamento e para teste. Segundo a recomendação de Geron (2017), 80% dos dados foram destinados para o treinamento e 20% para teste.

3.3 Escolha de parâmetros para a rede neural

Visando ter os as melhores configurações para as redes neurais artificiais para que exista uma melhor adequação aos dados e garantir uma melhor generalização, será utilizada a validação cruzada K-fold juntamente com a busca randômica. Como é demonstrado na Figura 13, o conjunto de dados de treinamento será separado em 5 subconjuntos (*folds)*.

Figura 13 - Validação cruzada para encontrar hiperparâmetros



Fonte: https://scikit-learn.org/stable/modules/cross_validation.html

Uma determinada configuração de hiperparâmetros será aplicada e será feita a média do MSE das 5 divisões (*splits).* Serão testadas 20 configurações de hiperparâmetros de forma aleatória que respeitem os seguintes valores e intervalos:

- Taxa de aprendizado (ln): 0.1, 0.01, 0.001, 0.0001;
- Número de neurônios (n): [1, 100];
- Número de camadas: 1,2;
- Número de épocas: [1,1000];
- Função de ativação: ReLu, Tanh.

A configuração que obter a menor média de MSE será escolhida para a análise posterior com os métodos de regressão. A Tabela 6 representa um exemplo no qual foram feitas as 20 iterações para obter a melhor configuração de hiperparâmetros.

Tabela 6 - Exemplo de busca de hiperparâmetros

Média (MSE)	Desvio padrão	n	ln	2° camada	Função	Épocas
-0.299	0.3763	10	0.1	False	relu	38
-0.3102	0.4229	17	0.1	True	relu	716
-0.3397	0.2624	7	0.01	True	tanh	130
-0.4803	0.1794	19	0.001	False	tanh	282
-0.1091	0.0356	29	0.001	False	relu	469
-0.4998	0.3606	88	0.1	False	tanh	926
-0.2493	0.3222	95	0.0001	True	relu	984
-0.2818	0.3028	10	0.01	True	tanh	865
-0.698	0.6259	58	0.001	True	relu	8
-0.3087	0.3267	9	0.01	False	tanh	514
-0.3092	0.4074	73	0.0001	True	relu	729
-0.2113	0.2873	22	0.001	True	relu	543
-0.179	0.1302	25	0.1	True	relu	562
-0.1635	0.0891	53	0.001	False	relu	498
-0.1805	0.2103	83	0.01	True	relu	337
-0.492	0.2703	99	0.01	False	tanh	16
-0.1775	0.1444	23	0.01	False	relu	472
-0.2534	0.2952	24	0.001	True	relu	778
-0.1831	0.1724	58	0.01	True	tanh	382
-0.472	0.3111	35	0.1	False	tanh	596

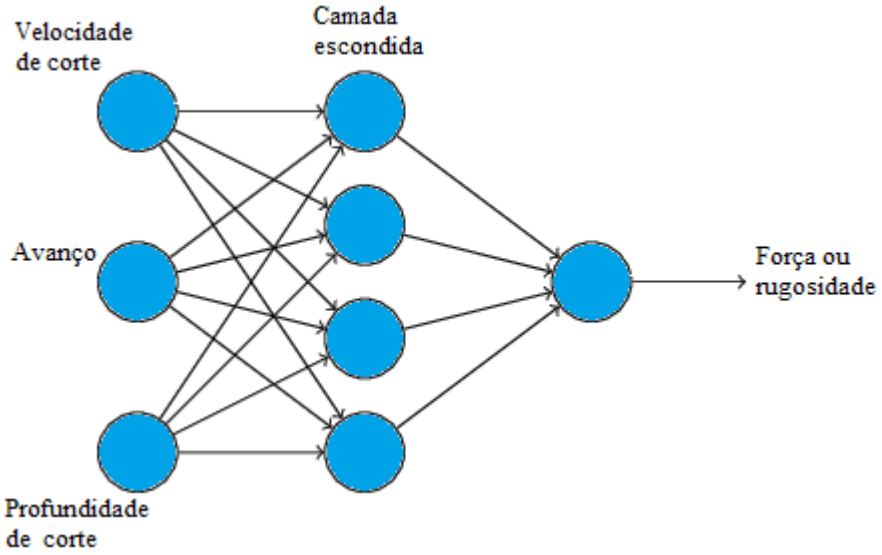
Fonte: Autor (2020)

3.4 Implementação dos modelos

Serão implementados a rede neural com a configuração obtida através do processo anterior e as regressões linear, quadrática, de 3º e 4º grau.

As redes neurais terão como dados de entrada a velocidade de corte, avanço e profundidade de corte, e a camada de saída irá fornecer a força ou a rugosidade, como mostra a Figura 14, e todos os hiperparâmetros que serão utilizados será os definidos na seção 3.3.

Figura 14 - Rede Neural Artificial aplicada ao torneamento



Fonte: Autor (2020)

Para os modelos de regressão, serão utilizadas as Equações 35, 36, 37e 38.

Regressão linear:

$$y_1 = \alpha + b_1 x_1 + b_2 x_2 + b_3 x_3 \qquad Equação\ 35$$

Regressão quadrática:

$$y_2 = y_1 + b_4 x_1^2 + b_5 x_2^2 + b_6 x_3^2 + b_7 x_1 x_2 + b_8 x_1 x_3 + b_9 x_2 x_3 \qquad Equação\ 36$$

Regressão 3º grau:

$$y_3 = y_2 + b_{10} x_1^3 + b_{11} x_1^2 x_2 + b_{12} x_1^2 x_3 + b_{13} x_1 x_2^2 + b_{14} x_1 x_3^2 + \\ + b_{15} x_2^3 + b_{16} x_2^2 x_3 + b_{17} x_2 x_3^2 + b_{18} x_3^3 + b_{19} x_1 x_2 x_3 \qquad Equação\ 37$$

Regressão 4º grau:

$$y_4 = y_3 + b_{20} x_1^4 + b_{21} x_1^3 x_2 + b_{22} x_1^3 x_3 + b_{23} x_1^2 x_2^2 + \\ + b_{24} x_1^2 x_2 x_3 + b_{25} x_1^2 x_3^2 + b_{26} x_1 x_3^3 + b_{27} x_1 x_2^2 x_3 + b_{28} x_1 x_2 x_3^2 + \\ + b_{29} x_1 x_3^3 + b_{30} x_2^4 + b_{31} x_2^3 x_3 + b_{32} x_2^2 x_3^2 + b_{33} x_2 x_3^3 + b_{34} x_3^4 \qquad Equação\ 38$$

Será atribuído a velocidade, a profundidade de corte e o avanço para as variáveis x_1, x_2 e x_3.

Para encontrar os coeficientes, será utilizado o método dos mínimos quadrados definido na Equação 27.

3.5 Comparação dos modelos

A comparação dos modelos será feita através dos indicadores de acurácia apresentados na seção 2.5 e de gráficos de dispersão. Será comparado a acurácia das previsões em relação aos dados de treinamento e de teste, sendo que a avaliação da previsão dos dados de teste é importante para avaliar a generalização do modelo.

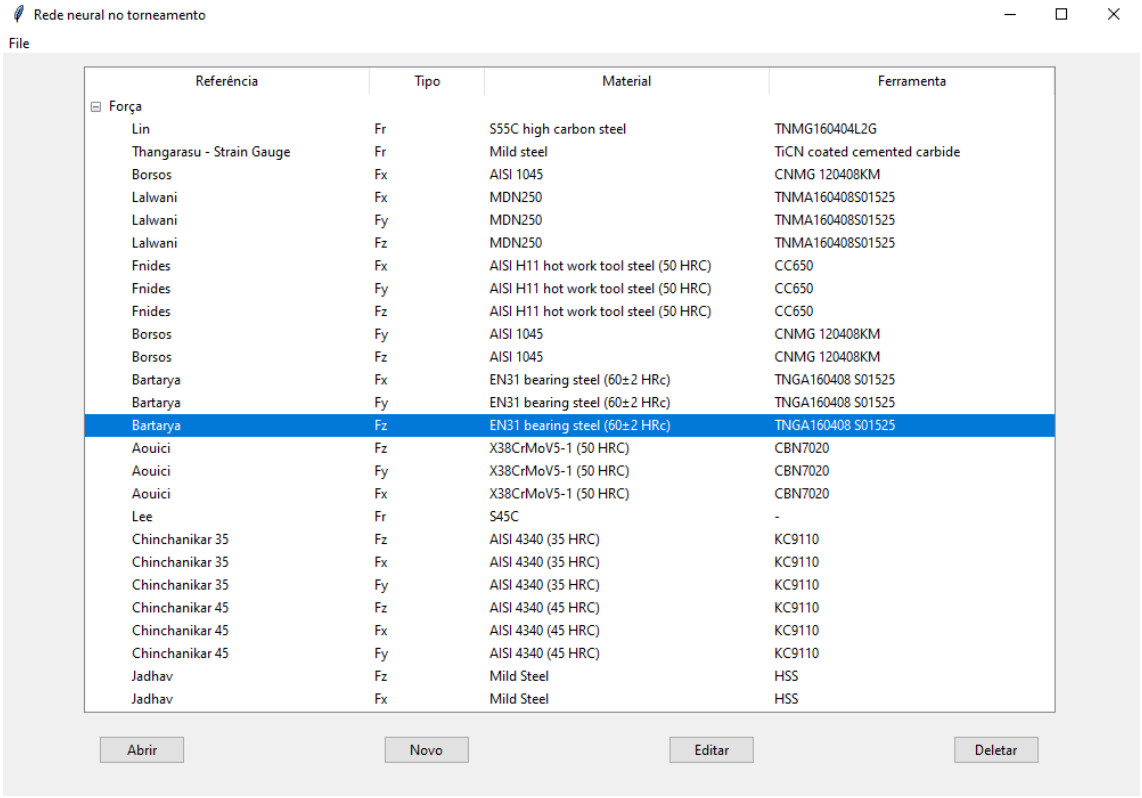
3.6 Programa desenvolvido

A linguagem *Python* foi utilizada no programa desenvolvido para a aplicação dos modelos. Para a criação e aplicação das redes neurais foi utilizado a biblioteca *Tensor Flow* e para os modelos de regressão empregou- se a biblioteca *Sklearn.* Visando a facilitação do uso do algoritmo foi desenvolvido uma interface gráfica com *Tkinter.* Todo o código do programa está presente no Apêndice C.

3.6.1 Página Inicial

Na página inicial, mostrada na Figura 15, são visualizados os estudos adicionados e possui as opções de abrir o estudo selecionado, deletá-lo, editá-lo ou criar um novo.

Figura 15 - Pagina inicial do programa

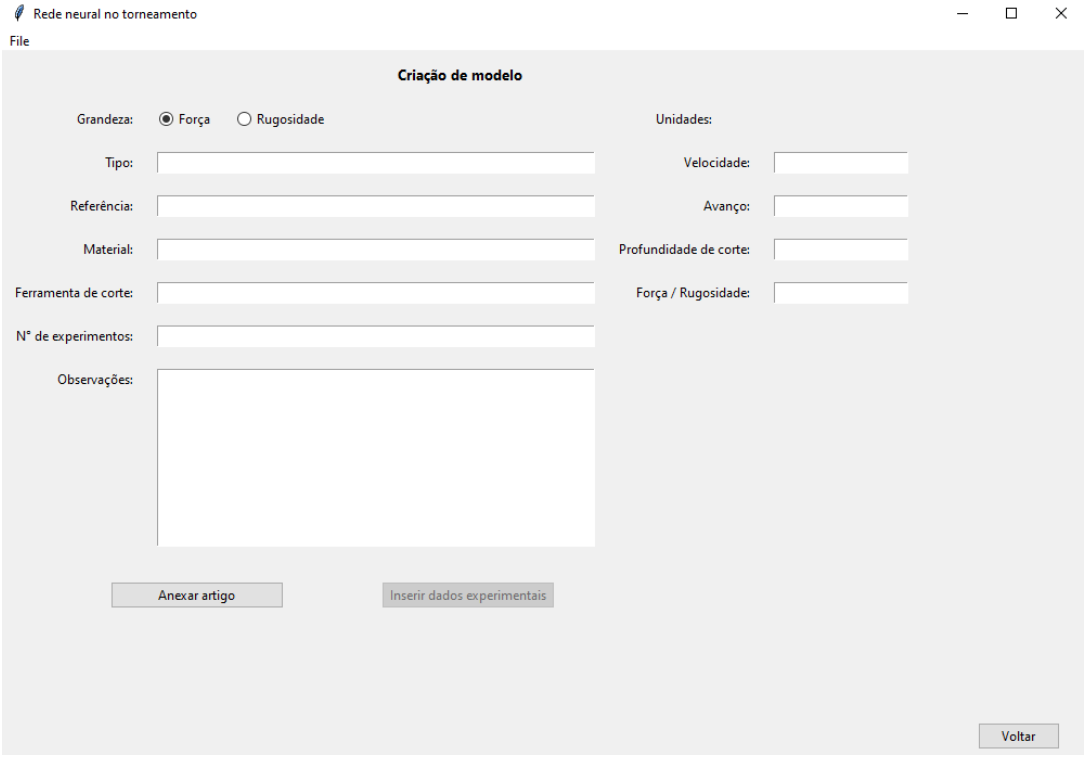


Fonte: Autor (2020)

3.6.2 Inserir novo estudo

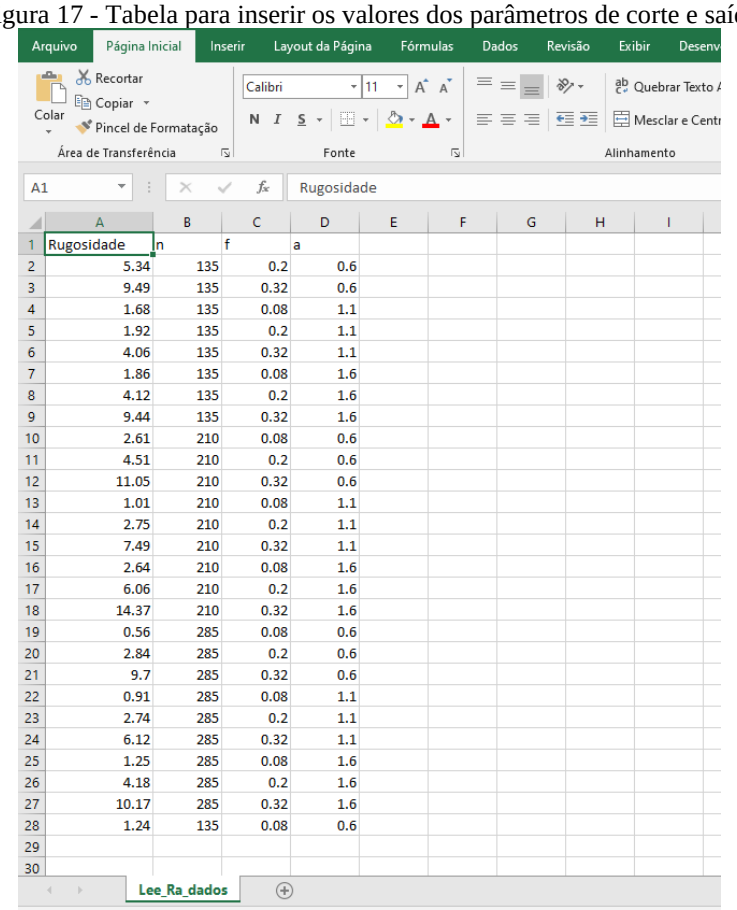
Para inserir um novo estudo e aplicar os modelos de previsão, é necessário cadastrar as informações da Figura 16 e o programa irá direcionar o usuário para um arquivo .csv no excel, mostrados na Figura 17, para inserir os valores dos parâmetros de usinagem e da rugosidade ou força como, por exemplo, os dados da Tabela 5.

Figura 16 - Página para cadastrar um novo estudo



Fonte: Autor (2020)

Figura 17 - Tabela para inserir os valores dos parâmetros de corte e saída



	A	B	C	D
1	Rugosidade	n	f	a
2	5.34	135	0.2	0.6
3	9.49	135	0.32	0.6
4	1.68	135	0.08	1.1
5	1.92	135	0.2	1.1
6	4.06	135	0.32	1.1
7	1.86	135	0.08	1.6
8	4.12	135	0.2	1.6
9	9.44	135	0.32	1.6
10	2.61	210	0.08	0.6
11	4.51	210	0.2	0.6
12	11.05	210	0.32	0.6
13	1.01	210	0.08	1.1
14	2.75	210	0.2	1.1
15	7.49	210	0.32	1.1
16	2.64	210	0.08	1.6
17	6.06	210	0.2	1.6
18	14.37	210	0.32	1.6
19	0.56	285	0.08	0.6
20	2.84	285	0.2	0.6
21	9.7	285	0.32	0.6
22	0.91	285	0.08	1.1
23	2.74	285	0.2	1.1
24	6.12	285	0.32	1.1
25	1.25	285	0.08	1.6
26	4.18	285	0.2	1.6
27	10.17	285	0.32	1.6
28	1.24	135	0.08	0.6
29				
30				

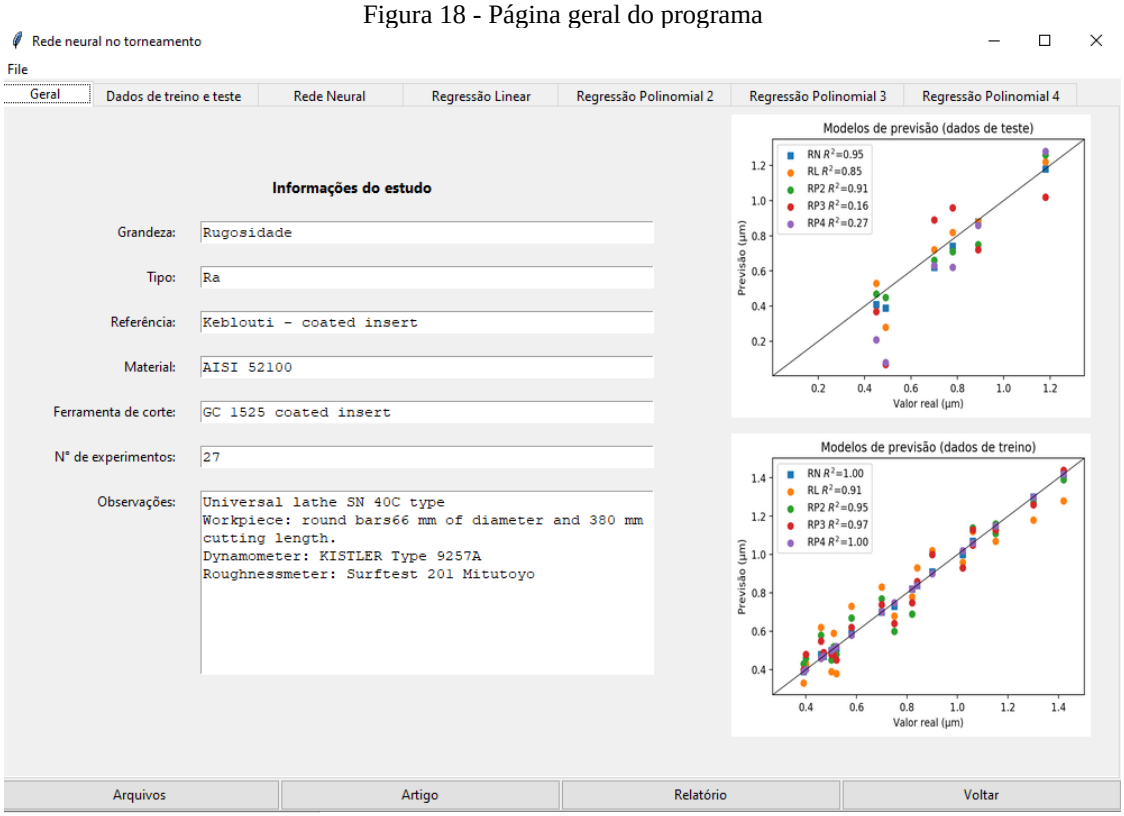
Fonte: Autor (2020)

Após inserir todas as informações, haverá a busca dos hiperparâmetros da rede neural e a aplicação dos modelos de previsão para descobrir os pesos, coeficientes e erros de previsão.

3.6.3 Aba Geral

Quando abrir o estudo selecionado na página inicial, o programa irá para a aba geral representada na Figura 18, no qual será apresentada as informações cadastradas anteriormente, além dos gráficos de dispersão com todos os modelos para os conjuntos de dados de treino e teste. Estes gráficos apresenta os eixos de valores previstos pelos modelos e os valores medidos nos estudos analisados. Quanto mais próximo da reta inclinada os pontos estiverem dispostos, melhor será a previsão e o ajuste do modelo aos dados.

Figura 18 - Página geral do programa



Fonte: Autor (2020)

A interface gráfica apresenta os seguintes botões:

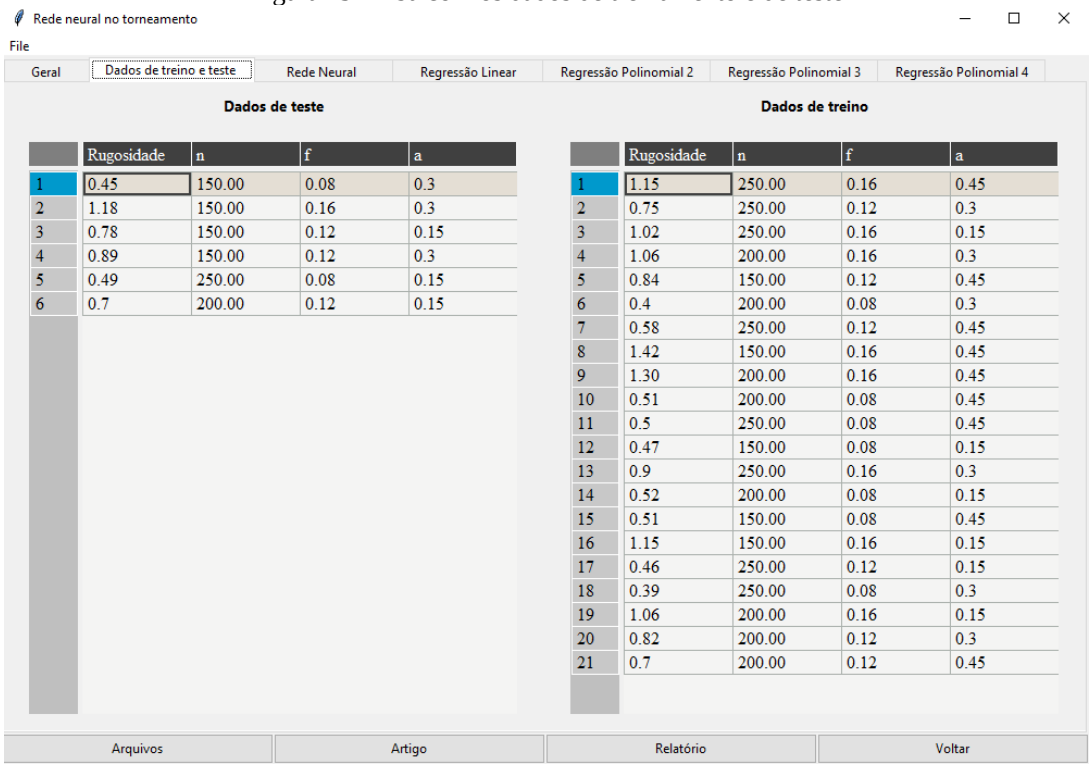
- Arquivos: abre uma pasta com todos os modelos e resultados;
- Artigo: irá abrir o artigo acadêmico do qual foram retirados os dados experimentais;

- Relatório: irá gerar um relatório no formato word com todas as informações cadastradas e dos modelos.

3.6.4 Aba Dados de treino e teste

Há a aba com os dados dos experimentos representado na Figura 19 que irá separar todos os valores em um conjunto para o treinamento e outro para teste.

Figura 19 - Aba com os dados de treinamento e de teste



	Rugosidade	n	f	a
1	0.45	150.00	0.08	0.3
2	1.18	150.00	0.16	0.3
3	0.78	150.00	0.12	0.15
4	0.89	150.00	0.12	0.3
5	0.49	250.00	0.08	0.15
6	0.7	200.00	0.12	0.15

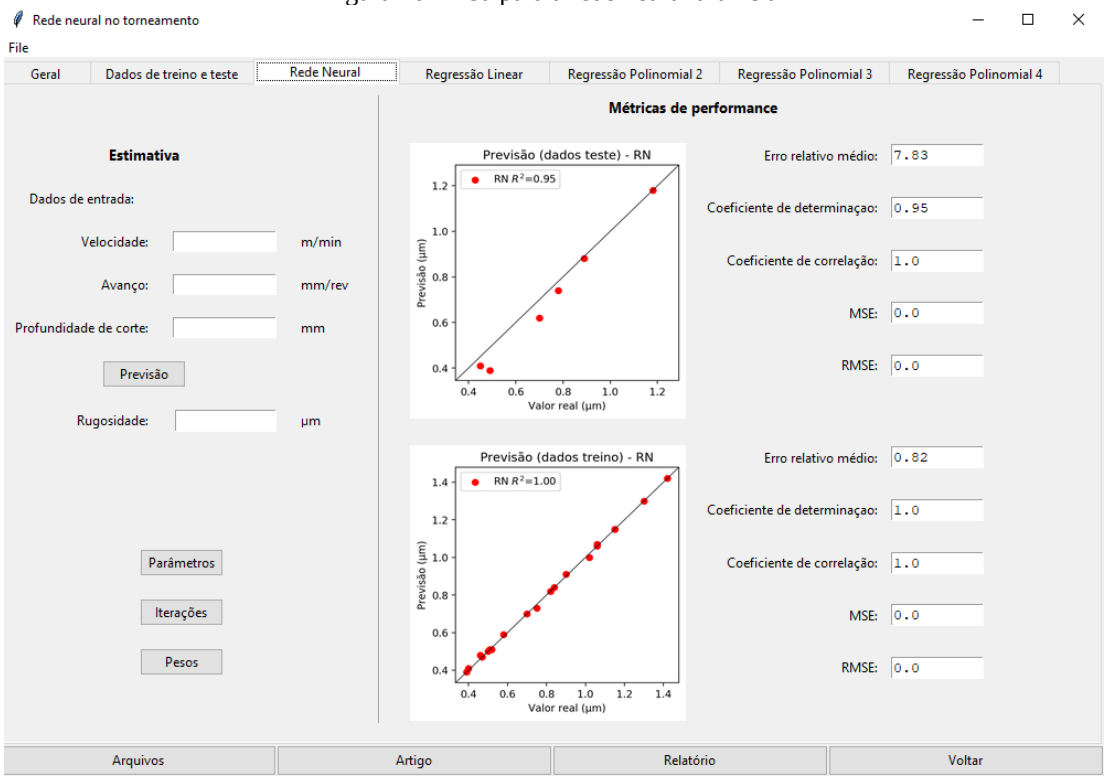
	Rugosidade	n	f	a
1	1.15	250.00	0.16	0.45
2	0.75	250.00	0.12	0.3
3	1.02	250.00	0.16	0.15
4	1.06	200.00	0.16	0.3
5	0.84	150.00	0.12	0.45
6	0.4	200.00	0.08	0.3
7	0.58	250.00	0.12	0.45
8	1.42	150.00	0.16	0.45
9	1.30	200.00	0.16	0.45
10	0.51	200.00	0.08	0.45
11	0.5	250.00	0.08	0.45
12	0.47	150.00	0.08	0.15
13	0.9	250.00	0.16	0.3
14	0.52	200.00	0.08	0.15
15	0.51	150.00	0.08	0.45
16	1.15	150.00	0.16	0.15
17	0.46	250.00	0.12	0.15
18	0.39	250.00	0.08	0.3
19	1.06	200.00	0.16	0.15
20	0.82	200.00	0.12	0.3
21	0.7	200.00	0.12	0.45

Fonte: Autor (2020)

3.6.5 Aba Rede Neural

Aba destinada ao modelo de RNA, representada pela Figura 20, há uma parte destinada para o usuário inserir a velocidade, profundidade de corte e avanço e será retornado uma estimativa para força ou rugosidade. Além disso, há os gráficos de dispersões para análise visual e os valores dos indicadores de acurácia para cada conjunto de dados.

Figura 20 - Aba para a rede neural artificial



Fonte: Autor (2020)

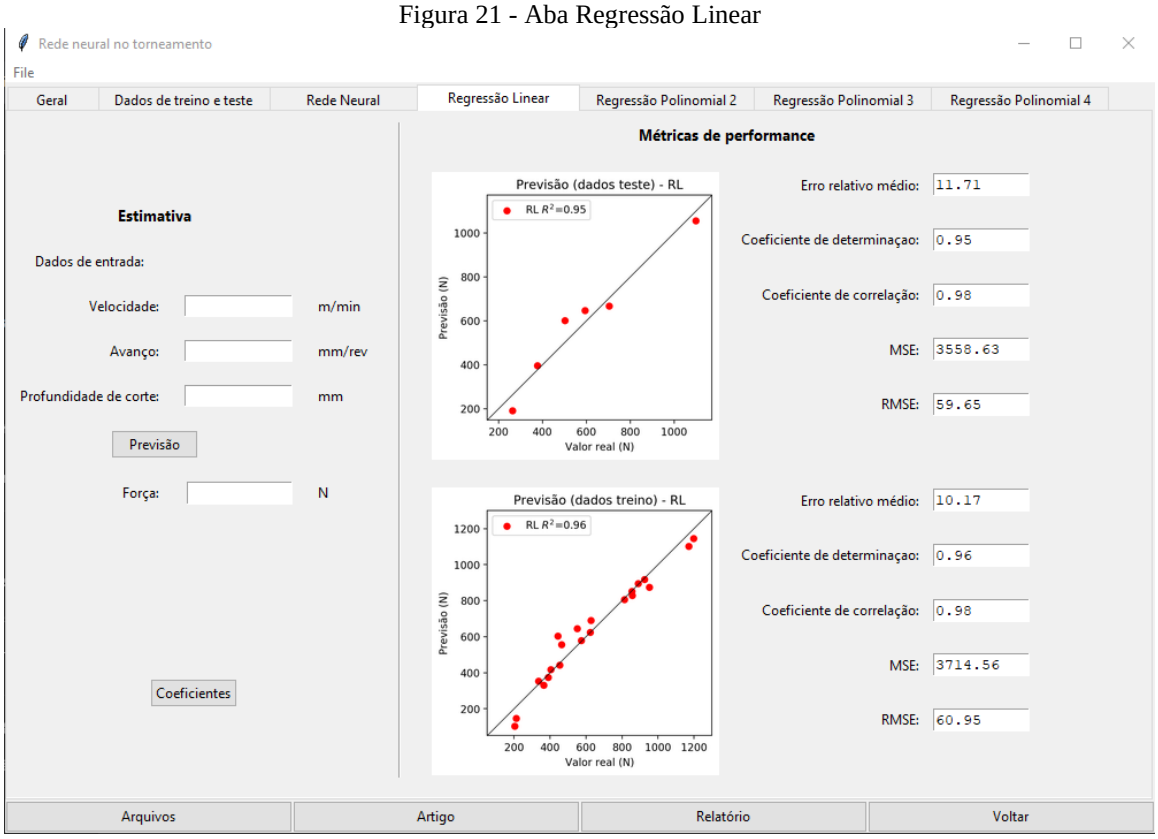
Nesta aba há os seguintes botões:

- Parâmetros: apresenta os melhores hiperparâmetros da busca;
- Iterações: apresenta as iterações para encontrar os melhores hiperparâmetros, como foi apresentado na Tabela 6;
- Pesos: irá mostrar os pesos da rede neural artificial.

3.6.6 Abas para as regressões

Semelhante à aba de RNA, há também as abas para os modelos de regressão. Nestas haverá também uma parte para fazer as estimativas, exibição dos gráficos de dispersão e métricas de performance. A Figura 21 apresenta a aba para o modelo de regressão linear.

Figura 21 - Aba Regressão Linear



Fonte: Autor (2020)

Há o botão “Coeficiente” que irá exibir os coeficientes encontrados para todos os modelos de regressão.

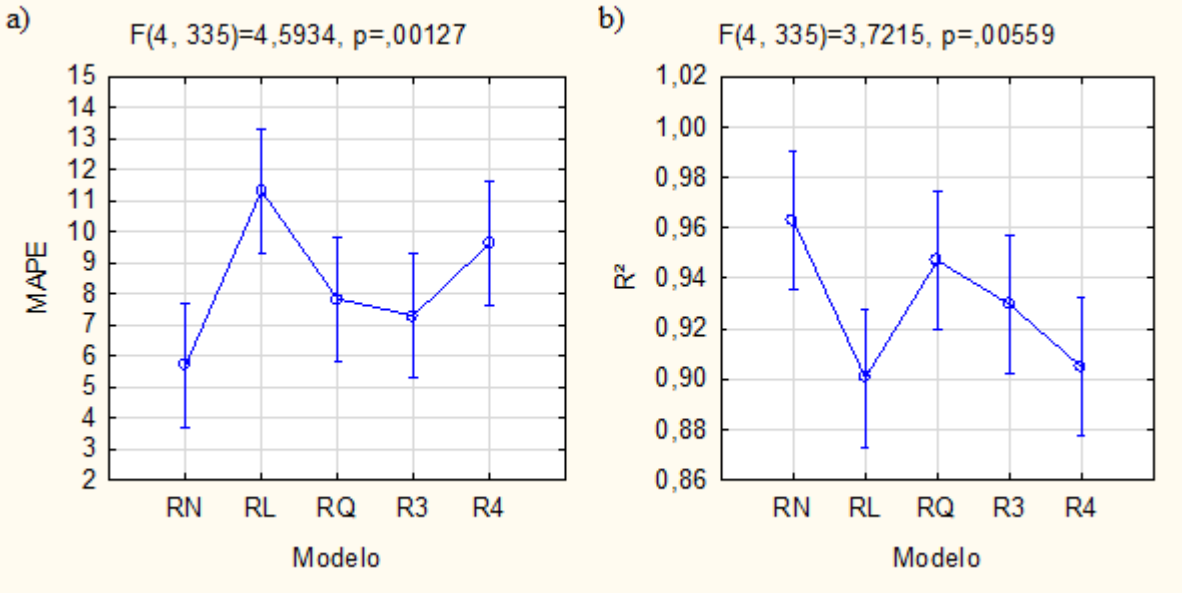
4 RESULTADOS E DISCUSSÃO

Após a aplicação de todos os métodos, foram calculadas as métricas de performance, presentes no Apêndices A e B, que serão utilizadas para comparação entre os métodos. Todos os gráficos a seguir utilizaram intervalos de confiança com nível 95%.

4.1 Resultados para força

Considerando primeiro a análise da força, observa-se pela Figura 22 que o modelo matemático afetou significativamente os indicadores MAPE e R^2. Em relação ao coeficiente de determinação, é possível observar uma tendência de piores valores com o aumento do grau da regressão a partir da regressão quadrática.

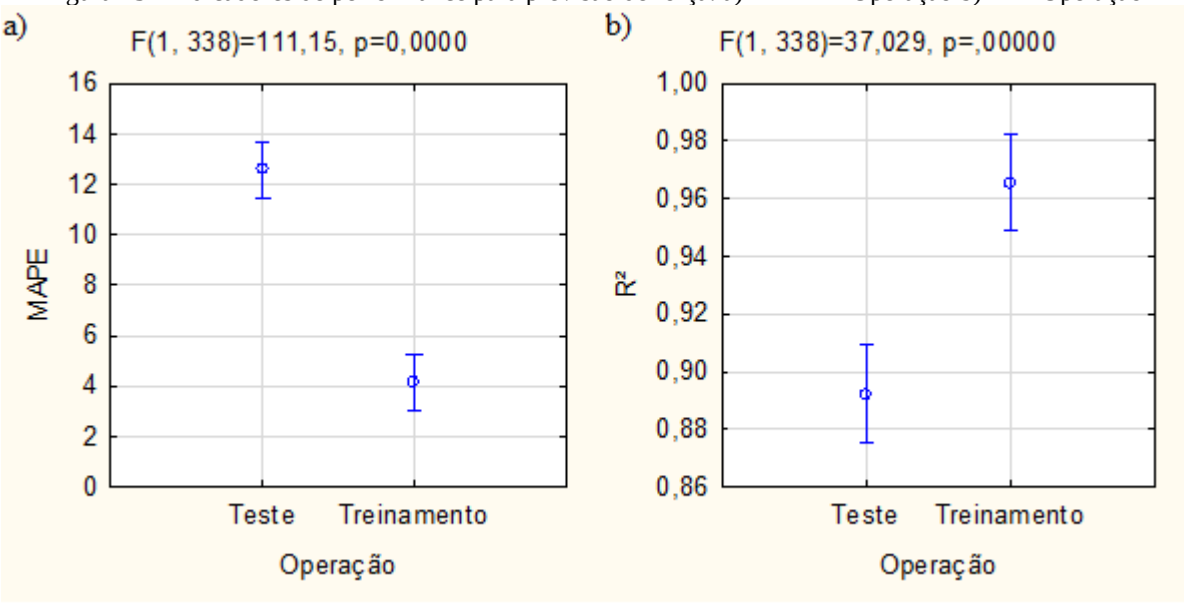
Figura 22 – Indicadores de performance para previsão de força. a) MAPE x Modelo b) R^2 x Modelo



Fonte: Autor (2020)

Fazendo a análise dos indicadores considerando os conjuntos de dados, é possível perceber pela Figura 23 que os modelos apresentaram melhores resultados em relação aos dados de treinamento, como o esperado, pois os modelos foram ajustados para esse conjunto.

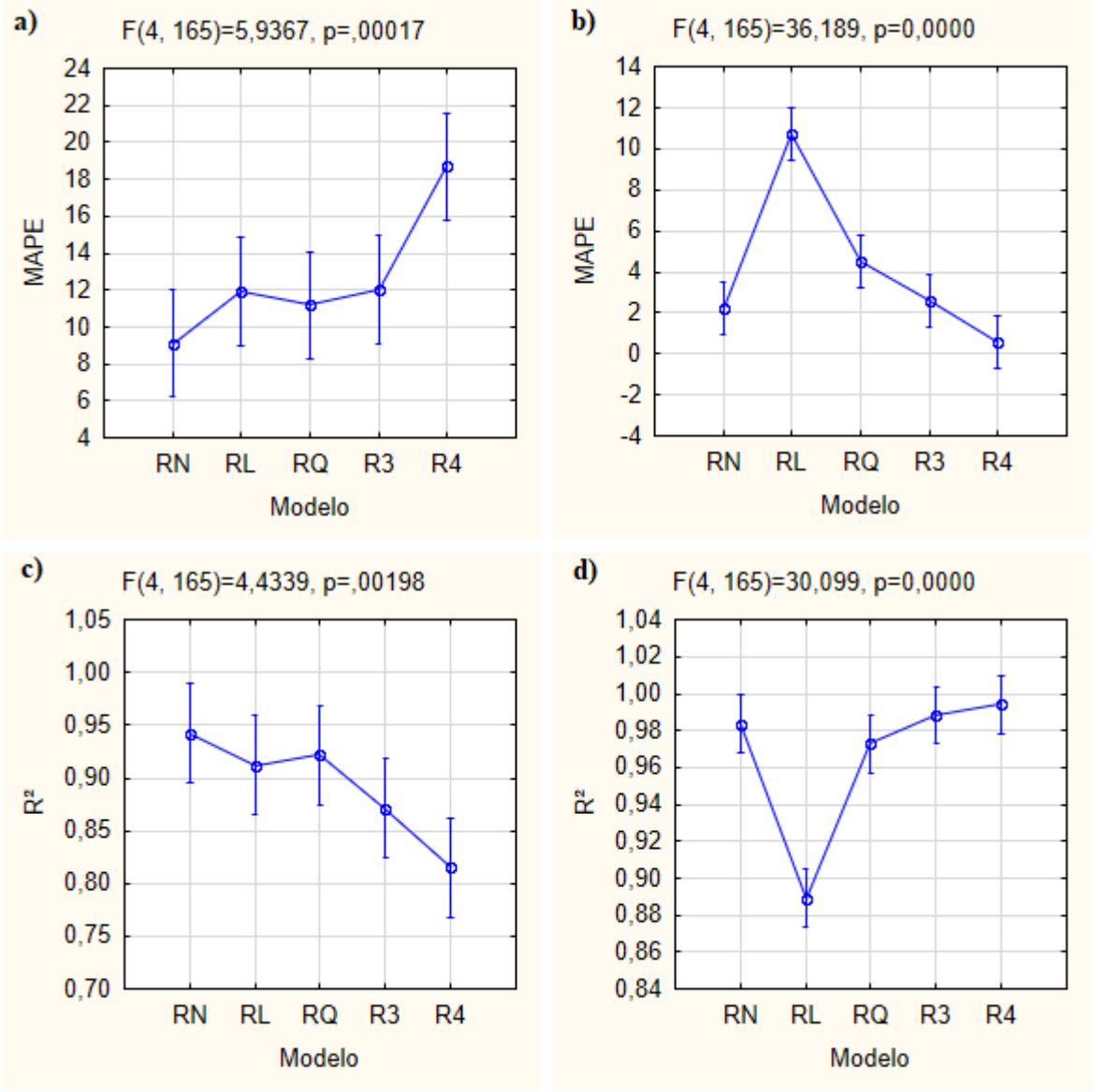
Figura 23 - Indicadores de performance para previsão de força. a) MAPE x Operação b) R^2 x Operação



Fonte: Autor (2020)

Visando fazer a análise dos indicadores em relação aos modelos de maneira separada para cada tipo de conjuntos de dados, foi feita a ANOVA demonstrada na Figura 24. Observa-se no gráfico a) que a regressão de 4° grau apresentou o pior MAPE entre os modelos em relação ao conjunto de teste. Tal fato pode indicar um sobre-ajuste aos dados de treinamento, não possibilitando uma boa previsão para dados não conhecidos pelo modelo. Também se nota pelos gráficos b) e d) da Figura 24 que a regressão linear apresentou os piores indicadores em relação ao conjunto de treinamento, tal fato pode ser explicado pela pouca quantidade de termos do método e baixa complexidade, gerando um sub-ajuste. Além disso, observa-se que, a partir da regressão quadrática, há uma tendência de que o aumento do grau da regressão promove melhores resultados para indicadores no conjunto de treinamento e piores resultados para o conjunto de teste. Isto pode estar associado a um sobre-ajuste dos modelos com a elevação do grau, o que pode ser explicado pelo aumento de termos da equação, tornando as funções de previsão adequadas somente para os dados conhecidos.

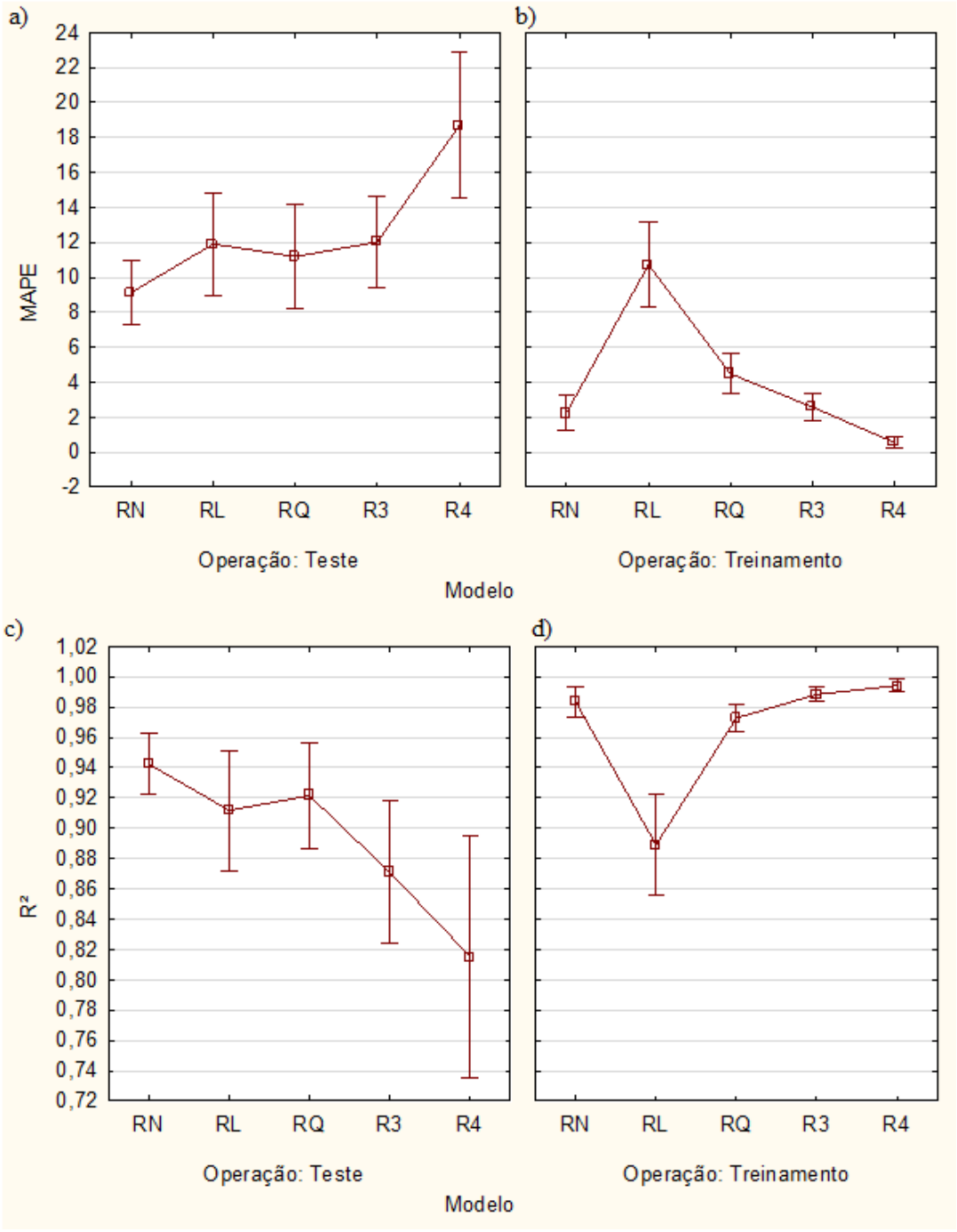
Figura 24 – Análise de variância dos indicadores de performance para força. a) MAPE x Modelo para conjunto de teste b) MAPE x Modelo para conjunto de treinamento c) R^2 x Modelo para conjunto de teste d) R^2 x Modelo para conjunto de treinamento



Fonte: Autor (2020)

A Figura 25 contém os valores das médias e intervalos de confiança individuais e observa-se, a partir da regressão quadrática, a tendência de aumento da dispersão dos valores dos indicadores com o aumento do grau quando se faz a análise com os dados de teste, enquanto a dispersão diminui quando os dados de treinamento são levados em consideração. Tal fato pode ser um indicativo para o sobre-ajuste, sendo mais evidente na regressão de 4° grau.

Figura 25 – Média e intervalos de confiança individuais dos indicadores de performance para força. a) MAPE x Modelo para conjunto de teste b) MAPE x Modelo para conjunto de treinamento c) R^2 x Modelo para conjunto de teste d) R^2 x Modelo para conjunto de treinamento

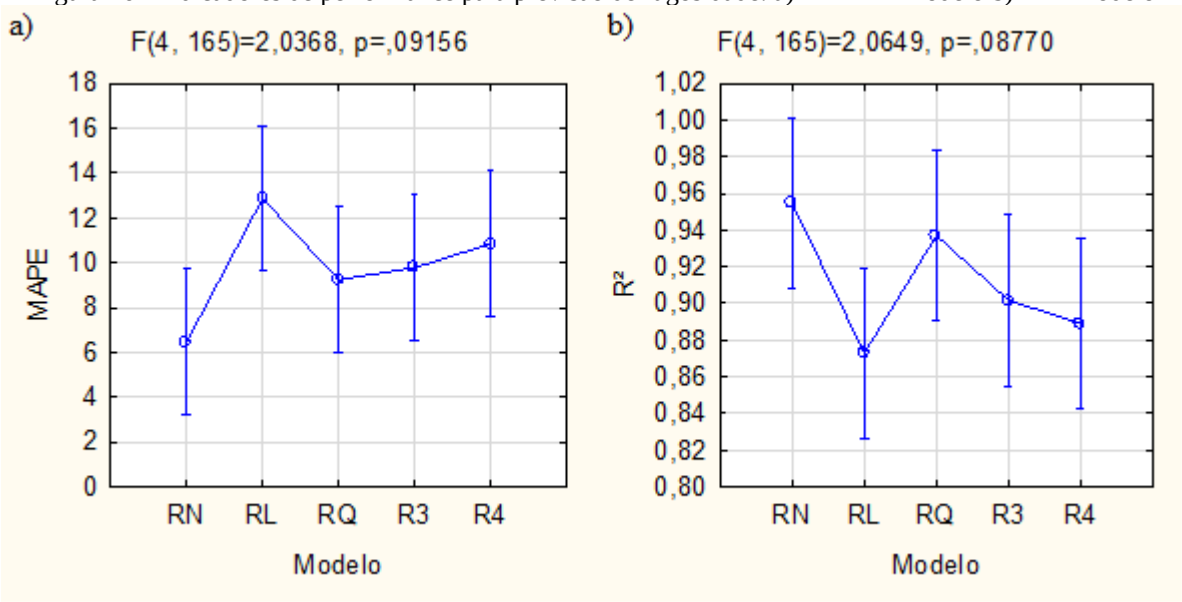


Fonte: Autor (2020)

4.2 Resultados para rugosidade

Para a rugosidade, segundo a Figura 26, é possível observar uma tendência de que a elevação do grau da regressão provoca perca de precisão a partir da regressão quadrática.

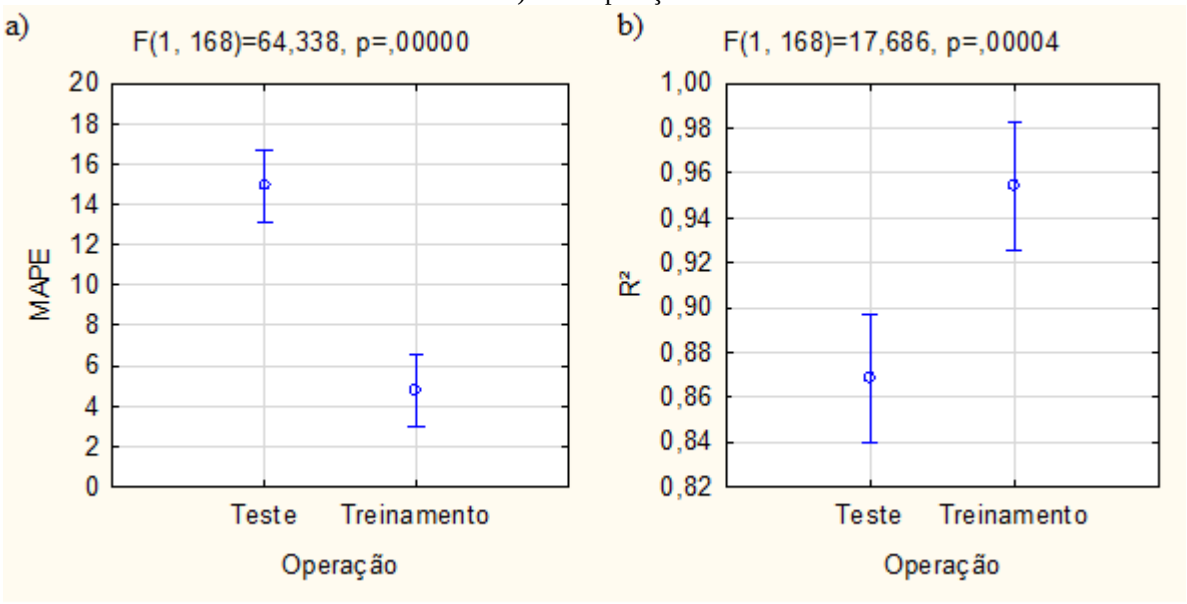
Figura 26 - Indicadores de performance para previsão de rugosidade. a) MAPE x Modelo b) R^2 x Modelo



Fonte: Autor (2020)

Em relação aos conjuntos de dados, a Figura 27 indica que os modelos apresentam melhores performances para dados de treinamento, de modo igual à análise dos gráficos da Figura 23 para a força.

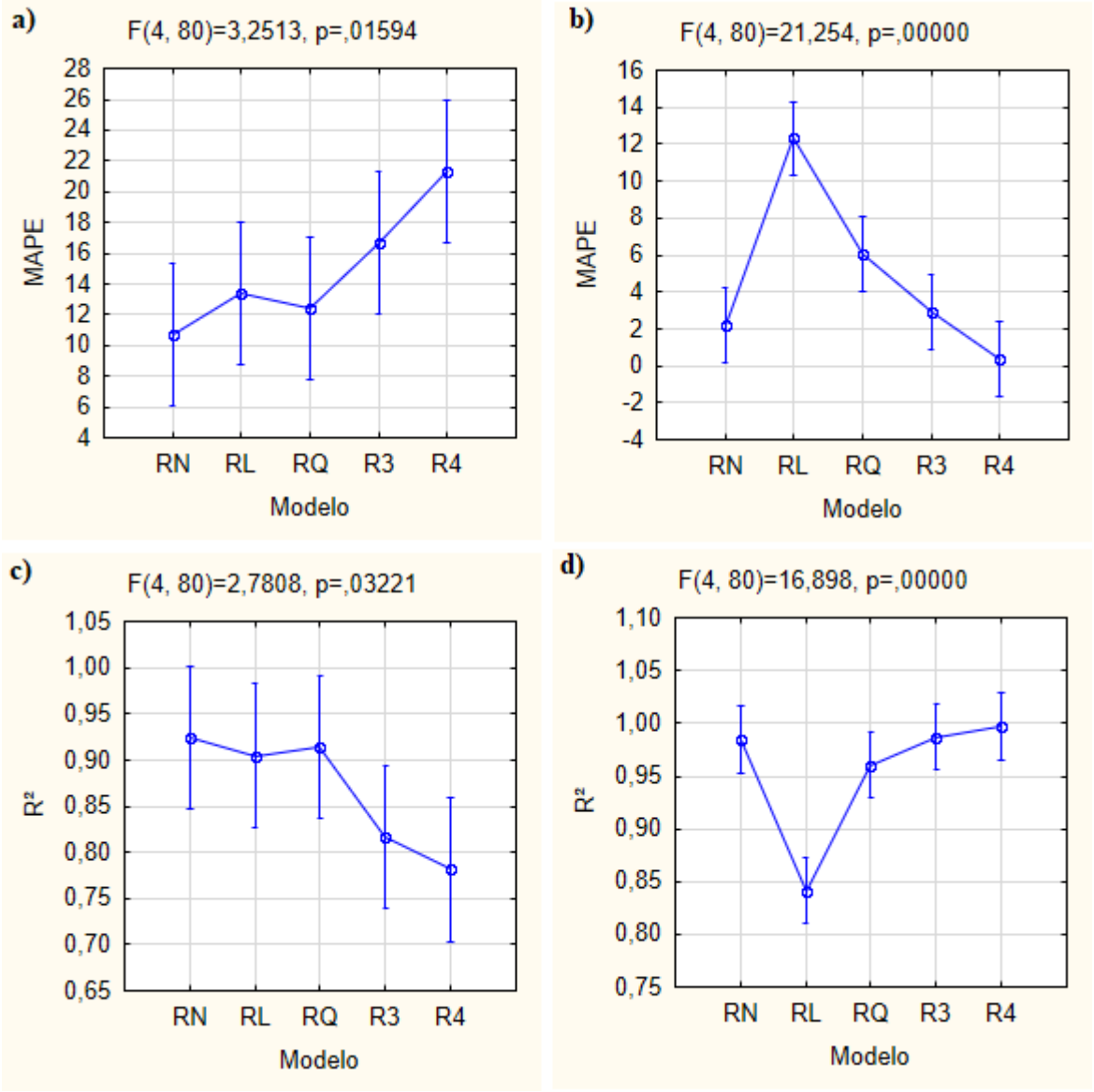
Figura 27 - Indicadores de performance para previsão de rugosidade. a) MAPE x Operação
b) R^2 x Operação



Fonte: Autor (2020)

Na análise dos indicadores para cada método em relação aos conjuntos de dados separadamente, demonstrado pela Figura 28, é possível observar que a regressão linear apresentou os piores indicadores para o conjunto de treinamento devido à baixa complexidade do modelo e também existe a tendência de sobre-ajuste aos dados de treinamento com a elevação do grau da regressão, a partir da regressão quadrática.

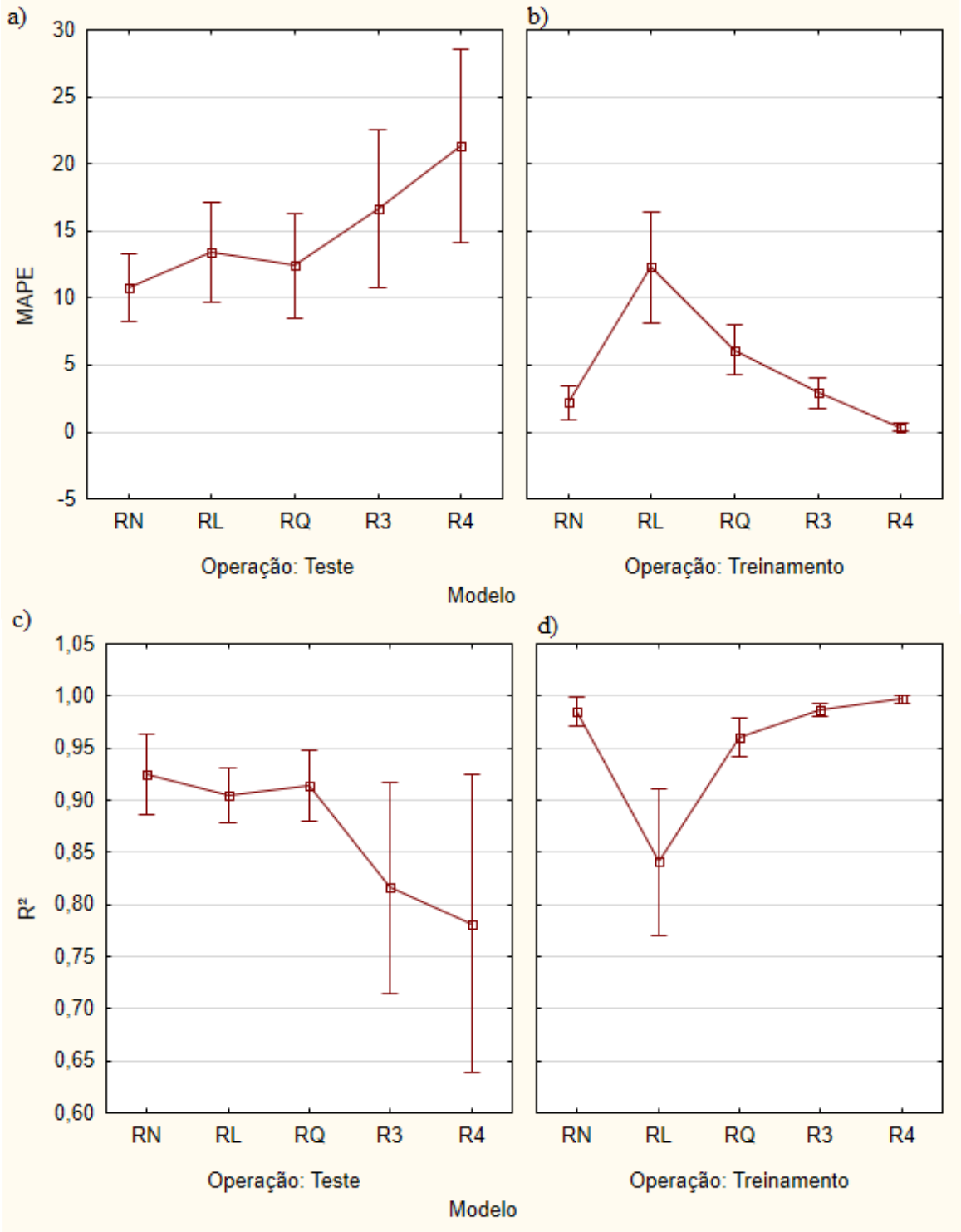
Figura 28 - Indicadores de performance para rugosidade. a) MAPE x Modelo para conjunto de teste b) MAPE x Modelo para conjunto de treinamento c) R^2 x Modelo para conjunto de teste d) R^2 x Modelo para conjunto de treinamento



Fonte: Autor (2020)

Através da análise dos intervalos de confiança individuais da Figura 29, observa-se a tendência de aumento da dispersão dos indicadores de acurácia em relação aos dados de teste com o aumento da regressão, enquanto há a diminuição da dispersão em relação ao conjunto de treinamento. Percebe-se esse fato principalmente na regressão de 4° grau e pode ser um indicativo de sobre-ajuste.

Figura 29 - Média e intervalos de confiança individuais dos indicadores de performance para rugosidade. a) MAPE x Modelo para conjunto de teste b) MAPE x Modelo para conjunto de treinamento c) R^2 x Modelo para conjunto de teste d) R^2 x Modelo para conjunto de treinamento



Fonte: Autor (2020)

5 CONSIDERAÇÕES FINAIS

A partir deste trabalho foi possível a criação de um programa com interface gráfica que aplica os modelos de rede neurais artificiais e de regressão em dados de força e rugosidade extraídos no torneamento de diversos materiais. O programa permite:

- Cadastramento de estudos que contêm dados experimentais de força e rugosidade durante o torneamento;
- Busca dos melhores hiperparâmetros para as redes neurais artificiais;
- Utilização dos modelos matemáticos para realizar previsões;
- A comparação de acurácia destes modelos através de métricas e de gráficos de dispersão.

Foram cadastrados ao todo 34 componentes de força e 17 tipos de rugosidades de diversos estudos que possuem dados de medição de força e rugosidade para o processo de torneamento.

6 CONCLUSÃO

Através da comparação dos métodos, com a utilização dos indicadores MAPE e R^2, pode-se concluir que:

- Os modelos apresentaram melhores indicadores de acurácia relacionados aos dados de treinamento;
- A regressão linear apresentou o pior desempenho em relação aos dados de treinamento para as componentes de força e rugosidade, indicando um sub-ajuste;
- Tendência de sobre-ajuste aos dados de treinamento com a elevação do grau da regressão a partir da regressão quadrática;
- A regressão de 4° grau apresentou o pior MAPE para força entre os modelos em relação ao conjunto de teste, no qual pode indicar um sobre-ajuste aos dados de treinamento.

7 SUGESTÃO PARA TRABALHOS FUTUROS

- Aplicação dos modelos matemáticos para a previsão da temperatura durante o torneamento;
- Aplicação dos modelos matemáticos para previsão de forças e rugosidade em outros processos de usinagem como o fresamento;
- Utilização de outros dados de entrada como a vibração, raio da ponta e desgaste da ferramenta para a previsão das forças e rugosidade durante o torneamento.

REFERÊNCIAS

AOUICI, H. et al. Experimental investigation of cutting parameters influence on surface roughness and cutting forces in hard turning of X38CrMoV5-1 with CBN tool. **Sadhana - Academy Proceedings in Engineering Sciences**, v. 38, n. 3, p. 429–445, 2013.

ASILTÜRK, I.; ÇUNKAŞ, M. Modeling and prediction of surface roughness in turning operations using artificial neural network and multiple regression method. **Expert Systems with Applications**, v. 38, n. 5, p. 5826–5832, 2011.

ASILTÜRK, I.; NEŞELI, S. Multi response optimisation of CNC turning parameters via Taguchi method-based response surface analysis. **Measurement: Journal of the International Measurement Confederation**, v. 45, n. 4, p. 785–794, 2012.

BARTARYA, G.; CHOUDHURY, S. K. Effect of cutting parameters on cutting force and surface roughness during finish hard turning aisi52100 grade steel. **Procedia CIRP**, v. 1, n. 1, p. 651–656, 2012.

BIASIBETTI, G. R. DOS S. et al. Análise da rugosidade superficial de barras de aço SAE 1045 após torneamento. **Matéria (Rio de Janeiro)**, v. 24, n. 1, 2019.

BORSOS, B. et al. Two-dimensional finite element analysis of turning processes. **Periodica Polytechnica Mechanical Engineering**, v. 61, n. 1, p. 44–54, 2017.

BRAGA, A. DE P. B.; CARVALHO, A. P. DE L. E. DE; LUDERMIR, T. B. **Redes Neurais Artificiais: Teoria e Aplicações**. Rio de Janeiro, RJ: LTC - LIVROS TÉCNICOS E CIENTÍFICOS EDITORA S.A, 2000.

CAKIR, M. C.; ENSARIOGLU, C.; DEMIRAYAK, I. Mathematical modeling of surface roughness for evaluating the effects of cutting parameters and coating material. **Journal of Materials Processing Technology**, v. 209, n. 1, p. 102–109, 2009.

CHINCHANIKAR, S.; CHOUDHURY, S. K. Effect of work material hardness and cutting parameters on performance of coated carbide tool when turning hardened steel: An optimization approach. **Measurement: Journal of the International Measurement Confederation**, v. 46, n. 4, p. 1572–1584, 2013.

DESHPANDE, Y.; ANDHARE, A.; SAHU, N. K. Estimation of surface roughness using cutting parameters, force, sound, and vibration in turning of Inconel 718. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, v. 39, n. 12, p. 5087–5096, 2017.

DINIZ, A. E.; MARCONDES, F. C.; COPPINI, N. L. **07 - Tecnologia de Usinagem dos Materiais**São Paulomm editora, , 2013.

FAHRMEIR, L. et al. **Model, Methods and Applications**. [s.l.] Springer, 2013.

FERRARESI, D. **Fundamentos da Usinagem dos Metais**. 1. ed. São Paulo: Edgard Blucher LTDA, 1970.

FNIDES, B. et al. Application of response surface methodology for determining cutting force model in turning hardened AISI H11 hot work tool steel. **Sadhana - Academy Proceedings in Engineering Sciences**, v. 36, n. 1, p. 109–123, 2011.

GALANIS, N. I.; MANOLAKOS, D. E. Surface roughness prediction in turning of femoral head. **International Journal of Advanced Manufacturing Technology**, v. 51, n. 1–4, p. 79–86, 2010.

GERON, A. **Hands-On Machine Learing With Scikit-Learn & Tensor Flow**. [s.l.] O'Reilly Media, 2017.

HANIEF, M.; WANI, M. F.; CHAROO, M. S. Modeling and prediction of cutting forces during the turning of red brass (C23000) using ANN and regression analysis. **Engineering Science and Technology, an International Journal**, v. 20, n. 3, p. 1220–1226, 2017.

HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. Ontario: Pearson, 2008. v. 127

HOFFMANN, R. **Uma Introdução à Econometria**. 4. ed. Piracicaba: HUCITEC, 2016.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. **Automated Machine Learning**. 1. ed. [s.l.] Springer, 2019. v. 498

KEBLOUTI, O. et al. Effects of coating material and cutting parameters on the surface roughness and cutting forces in dry turning of AISI 52100 steel. **Structural Engineering and Mechanics**, v. 61, n. 4, p. 519–526, 2017.

LALWANI, D. I.; MEHTA, N. K.; JAIN, P. K. Experimental investigations of cutting parameters influence on cutting forces and surface roughness in finish hard turning of MDN250 steel. **Journal of Materials Processing Technology**, v. 206, n. 1–3, p. 167–179, 2008.

LAOUISSI, A. et al. Investigation, modeling, and optimization of cutting parameters in turning of gray cast iron using coated and uncoated silicon nitride ceramic tools. Based on ANN, RSM, and GA optimization. **International Journal of Advanced Manufacturing Technology**, v. 101, n. 1–4, p. 523–548, 2019.

LIMA, J. G. et al. Hard turning: AISI 4340 high strength low alloy steel and AISI D2 cold work tool steel. **Journal of Materials Processing Technology**, v. 169, n. 3, p. 388–395, 2005.

LIN, W. S.; LEE, B. Y.; WU, C. L. Modeling the surface roughness and cutting force for turning. **Journal of Materials Processing Technology**, v. 108, n. 3, p. 286–293, 2001.

MACHADO, A. R. et al. **Teoria da Usinagem dos Materiais**. 1. ed. São Paulo: Blucher, 2009.

MASSART, D. L. et al. **Handbook of Chemometrics and Qualimetrics, Volume 20, Part 1**. [s.l: s.n.].

MATA, F. et al. Multiple regression prediction model for cutting forces in turning carbon-reinforced PEEK CF30. **Advances in Materials Science and Engineering**, v. 2010, 2010.

MEDDOUR, I. et al. Investigation and modeling of cutting forces and surface roughness when hard turning of AISI 52100 steel with mixed ceramic tool: cutting conditions optimization. **International Journal of Advanced Manufacturing Technology**, v. 77, n. 5–8, p. 1387–1399, 2015.

NWANKPA, C. et al. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. p. 1–20, 2018.

RAMANUJAM, R.; MUTHUKRISHNAN, N.; RAJU, R. Optimization of cutting parameters for turning Al-SiC(10p) MMC using ANOVA and Grey relational analysis. **International Journal of Precision Engineering and Manufacturing**, v. 12, n. 4, p. 651–656, 2011.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning**. 2. ed. [s.l.] Packt, 2017.

SHANKAR, S. et al. Prediction of cutting force in turning process: An experimental and fuzzy approach. **Journal of Intelligent and Fuzzy Systems**, v. 28, n. 4, p. 1785–1793, 2015.

SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas. **São Paulo: Artliber**, 2010.

SUHAIL. Optimization of Cutting Parameters Based on Surface Roughness and Assistance of Workpiece Surface Temperature in Turning Process. **American Journal of Engineering and Applied Sciences**, v. 3, n. 1, p. 102–108, 2010.

THANGARASU, S. K. et al. Prediction of Cutting Force in Turning Process-an Experimental Approach. **IOP Conference Series: Materials Science and Engineering**, v. 310, n. 1, 2018.

WILLIAM D. CALLISTER, J.; RETHWISCH, D. G. Phase Transformations: Development of Microstructure and Alternation of Mechanical Properties. **Materias Science and Engineering: An Introduction**, p. 356–407, 2013.

APÊNDICE A – TABELA DE RESULTADOS (FORÇA)

(continua)

Número	Rede neural artificial				Regressão linear				Regressão quadrática				Regressão 3				Regressão 4			
	Teste		Treinamento		Teste		Treinamento		Teste		Treinamento		Teste		Treinamento		Teste		Treinamento	
	MAPE	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**
1	2.94	1	0.31	1	23.16	0.89	12.07	0.95	6.12	1	2.34	1	2.48	1	0.42	1	22.33	0.95	0	1
2	0.85	1	0.16	1	0.49	1	0.34	1	0.31	1	0.24	1	0.73	1	0.17	1	3.5	0.99	0	1
3	7.54	0.99	0.17	1	36.13	0.87	22.16	0.9	6.93	1	5.01	0.99	6.42	0.99	2.5	1	14.33	0.99	0	1
4	10.79	0.93	3.44	0.98	11.28	0.95	5.26	0.97	10.74	0.93	3.46	0.98	10.79	0.93	3.44	0.98	10.79	0.93	3.44	0.98
5	3.34	1	1.33	1	5.89	0.99	9.06	0.97	8.27	0.99	4.01	0.99	4.48	0.99	2.51	1	20.55	0.89	0	1
6	7.16	0.99	1.92	1	21.25	0.82	19.36	0.86	19.51	0.83	9.64	0.95	30.2	0.81	4.96	0.99	5.02	0.99	0	1
7	4.6	0.98	1.49	0.99	5.81	1	5.81	0.9	4.61	1	1.58	0.99	3.55	1	1.49	0.99	7.32	0.98	1.49	0.99
8	12.01	0.92	1.05	0.98	9.08	0.95	3.51	0.93	5.4	0.98	2.3	0.96	13.3	0.95	1.05	0.98	11.7	0.86	1.05	0.98
9	13.09	0.91	0	1	18.81	0.82	15.16	0.77	13.28	0.86	6.08	0.96	13.2	0.92	4.22	0.98	15.42	0.82	0	1
10	6.44	0.95	8.95	0.96	9.63	0.94	12.24	0.93	9.76	0.91	10.15	0.96	17.69	0.76	5.89	0.99	24.4	0.68	0	1
11	11.87	0.89	1.78	1	9.29	0.93	9.59	0.96	9.73	0.92	2.27	1	13.34	0.89	1.51	1	30.88	0.78	0	1
12	3.97	0.99	4.49	0.95	4.28	0.99	3.76	0.96	5.05	0.99	3.1	0.97	5.22	0.99	3.02	0.97	5.22	0.99	3.02	0.97
13	7.6	0.92	0.23	1	4.42	0.97	3.93	0.98	2.06	0.99	2.08	0.99	14.55	0.77	1.45	1	9.79	0.9	0	1
14	10.06	0.95	3.38	0.99	16.75	0.85	14.43	0.9	9.85	0.99	5.56	0.98	17.65	0.87	0.59	1	12	0.95	0	1
15	7.57	0.98	0.02	1	24.49	0.79	22.41	0.81	37.27	0.65	14.92	0.89	26.69	0.85	8.72	0.97	4.47	1	0	1
16	5.43	0.99	2.04	0.98	4.49	0.98	5.16	0.89	3.19	0.99	2.15	0.98	4.95	0.99	1.28	0.98	10.58	0.91	1.28	0.98
17	2.68	0.98	0.84	0.98	6.21	0.99	4.57	0.89	4.57	0.97	2.43	0.96	8.89	0.78	0.84	0.98	21.14	0.67	0.84	0.98
18	7.69	0.72	2.46	0.98	6.61	0.87	9.12	0.76	6.41	0.9	3.27	0.97	13.6	0.6	2.1	0.98	14.6	0.18	0	1
19	10.5	0.96	6.41	0.93	7.89	0.88	9.81	0.83	10.51	0.58	6.98	0.94	13.85	0.51	5.33	0.97	46.2	0.4	0	1
20	6.24	0.84	0.26	1	11.95	0.38	8.29	0.77	8.9	0.74	3.74	0.96	7.72	0.78	1.13	0.99	29.33	0.23	0	1
21	7.43	0.99	2.58	0.98	11.69	0.97	7.53	0.94	6.84	0.99	2.74	0.98	7.42	0.99	2.53	0.98	7.42	0.99	2.53	0.98
22	9.04	0.93	1.82	1	9.3	0.96	6.81	0.97	8.23	0.97	3.24	0.99	11.31	0.9	1.49	1	9.49	0.98	0	1
23	26.86	0.89	0.03	1	33.47	0.98	32.02	0.84	19.06	0.96	9.96	0.97	27.23	0.51	4.94	1	35.6	0.69	0	1

(continua)

24	14.75	0.88	7.69	0.93	14.45	0.88	7.82	0.94	11.84	0.89	4.22	0.97	11.64	0.95	2.92	0.99	19.05	0.72	0	1
25	14.71	0.98	1.94	0.99	14.2	0.9	5.33	0.94	14.32	0.98	1.91	0.99	21.21	0.76	0.8	0.99	39.91	0.23	0.8	0.99
26	5.48	0.94	0.91	1	7.42	0.95	4.27	0.96	6.78	0.96	1.27	1	15.88	0.71	0.67	1	14.67	0.74	0.67	1
27	7.93	0.94	0.11	1	7.29	0.97	9.19	0.84	8.26	0.92	6.34	0.92	10.36	0.94	1.9	0.99	6.49	0.94	0	1
28	11.6	0.97	12.11	0.87	15.46	0.93	12.24	0.89	5.41	0.99	9.23	0.93	12.02	0.87	7.74	0.96	32.37	0.9	0	1
29	9.8	0.93	0.43	1	5.01	1	17.15	0.68	28.32	0.85	1.18	1	8.75	0.97	0.42	1	35.71	0.97	0.42	1
30	16.5	0.98	3.23	0.95	7.96	0.99	20.92	0.56	27.08	0.84	4.41	0.95	6.22	0.97	3.23	0.95	37.56	0.97	3.23	0.95
31	12.23	0.96	1.51	1	7.55	0.98	14.88	0.9	26.96	0.95	8.44	0.97	23.49	0.93	4.23	0.99	28.56	0.9	0	1
32	17.55	0.84	2.67	1	23	0.8	15.22	0.94	20.26	0.86	3.64	0.99	16.51	0.84	2.13	1	25.05	0.86	0	1
33	13.31	0.96	0.53	1	8.13	0.98	13.83	0.96	13.64	0.98	3.92	1	5.35	0.99	1.58	1	21.79	0.89	0	1
34	0.94	0.96	0	1	1.93	0.85	1.22	0.94	0.92	0.97	0.51	0.99	1.9	0.9	0.3	1	2.04	0.85	0	1

Fonte: Autor (2020)

APÊNDICE B – TABELA DE RESULTADOS (RUGOSIDADE)

Número	Rede neural artificial				Regressão linear				Regressão quadrática				Regressão 3				Regressão 4			
	Teste		Treinamento		Teste		Treinamento		Teste		Treinamento		Teste		Treinamento		Teste		Treinamento	
	MAPE	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**	**MAPE**	**R²**
1	16.61	0.99	0.15	1	36.02	0.91	26.1	0.95	20.83	0.99	7.08	1	3.58	1	1.33	1	37.05	0.96	0.15	1
2	17.77	0.87	1.06	1	13.61	0.92	29.21	0.51	27.69	0.86	15.17	0.87	23.45	0.78	6.73	0.97	25.84	0.9	0	1
3	8.86	0.84	0.56	0.99	5.92	0.99	3.63	0.74	7.41	0.84	0.68	0.98	7.57	0.84	0.56	0.99	11.3	0.1	0.56	0.99
4	3.2	0.98	0.79	0.97	4.15	0.95	3.45	0.81	3.91	0.96	1.26	0.97	3.19	0.91	0.79	0.97	5.9	0.75	0.79	0.97
5	7.83	0.99	0.82	1	12.19	0.91	11.88	0.91	8.3	0.94	8.12	0.95	31.06	0.62	6.83	0.97	29.89	0.97	0	1
6	17.2	0.9	0	1	15.99	0.88	11.31	0.91	12.62	0.88	8.11	0.97	14.02	0.93	2.22	1	13.55	0.91	0	1
7	17.42	0.94	0.65	1	18.97	0.87	21.84	0.65	8.55	0.99	3.68	0.99	12.7	0.97	0.65	1	22.14	0.91	0.65	1
8	8.9	1	2.33	0.99	15.57	0.91	18.81	0.74	21.76	0.89	5.2	0.98	12.53	0.98	2.33	0.99	30.32	0.79	2.33	0.99
9	8.15	0.97	3.82	0.99	13.12	0.89	13.23	0.92	8.28	0.97	4.28	0.99	5.58	0.98	0.88	1	22.23	0.88	0	1
10	6.93	0.93	2.81	0.98	13.63	0.91	6.69	0.93	8.49	0.96	5.17	0.96	17.21	0.59	3.49	0.98	24.9	0.79	0	1
11	4.54	1	3.46	1	8.78	0.98	8.11	0.98	4.95	0.99	6.1	0.99	16.34	0.93	3.4	1	17.88	0.98	0	1
12	17.26	0.83	4.64	0.98	9.66	0.96	11.68	0.87	11.27	0.89	10.4	0.92	39.15	0.53	4.72	0.98	20.19	0.78	0	1
13	6.66	0.98	0.19	1	12.2	0.87	6.67	0.95	8.27	0.95	2.78	0.99	10.57	0.97	0.72	1	5.54	1	0.19	1
14	7.67	0.94	0.22	1	8.05	0.94	6.31	0.95	7.21	0.95	4.83	0.97	12.43	0.84	2.81	0.99	9.74	0.84	0.22	1
15	7.51	0.73	0.87	1	7.38	0.79	4.67	0.93	6.06	0.86	4.37	0.94	8.95	0.95	1.62	0.99	6.6	0.86	0.87	1
16	15.07	0.97	6.55	0.94	14.8	0.88	19.57	0.64	21.56	0.86	10.34	0.9	40.48	0.31	6.18	0.97	61.98	0.04	0.13	1
17	11.47	0.86	8.07	0.9	18.41	0.82	6.04	0.91	23.5	0.76	6.04	0.95	24.47	0.75	4.06	0.98	17.88	0.82	0	1

Fonte: Autor (2020)

APÊNDICE C– ALGORITMO

Segue abaixo o código para o programa criado separado em: aplicação dos modelos, busca dos hiperparâmetros, aplicação das regressões, interface gráfica, banco de dados e interação com o usuário.

Apêndice C.1 – Aplicação dos modelos

```
from Bancodedadosreg import *
import warnings
warnings.filterwarnings('ignore')
import json
from docx import Document,shared
from numpy.random import seed
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
import numpy as np
from sklearn import metrics
val_loss=[]
import joblib
from scipy.stats import uniform, truncnorm, randint
from sklearn.model_selection import RandomizedSearchCV
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline

initDB()

def graph_scatter(pred,real,title,diretorio,saida,label):

    fig,ax =plt.subplots(dpi=200)
    ax.scatter(real,pred,c='r',label=label)
    ax.legend()
    ax.axis('square')
    menor=min([ax.get_xlim()[0],ax.get_ylim()[0]])
    maior=max([ax.get_xlim()[1],ax.get_ylim()[1]])
    ax.set_xlim(menor,maior)
```

```
    ax.set_ylim(menor,maior)
    ax.plot([menor,maior], [menor,maior],c='black',linewidth=0.65)
    ax.title.set_text(title)
    ax.set(xlabel="Valor real ({})".format(saida),ylabel="Previsão ({})".format(saida))

    plt.savefig(diretorio,bbox_inches='tight')
    plt.clf()
    plt.close

def norm_x(x,train_stats_x):
  return (x - train_stats_x['mean'].values) / (train_stats_x['std'].values)

def norm_rev_x(x,train_stats_x):
  return ((x* (train_stats_x['std'].values))+train_stats_x['mean'].values)

def norm_y(x,train_stats_y):
  return (x - train_stats_y['mean']) / (train_stats_y['std'])

def norm_rev_y(x,train_stats_y):
  return ((x* (train_stats_y['std']))+train_stats_y['mean'])

class modelo():
    def __init__(self,referencia,tipo):
        initDB()
        self.referencia=referencia
        self.tipo=tipo
        self.name=referencia+'_'+tipo
        a=view_one(self.referencia,self.tipo)
        self.grandeza=a[0][1]
        self.ferramenta=a[0][3]
        self.material=a[0][4]
        self.numero=a[0][5]
        self.observacoes=a[0][6]
        self.u_velocidade=a[0][7]
        self.u_avanco=a[0][8]
        self.u_profundidade=a[0][9]
        self.u_saida=a[0][10]

        df = pd.read_csv('Arquivos\Dados/'+self.name+'_dados.csv',sep=';')
        self.df = df.sort_values(by=self.grandeza)
```

```
    x=self.df
    y=x.pop(self.grandeza)

    train_stats_x = x.describe()
    self.train_stats_x = train_stats_x.transpose()
    train_stats_y = y.describe()
    self.train_stats_y = train_stats_y.transpose()

    x=norm_x(x,self.train_stats_x).values
    y=norm_y(y,self.train_stats_y).values

    self.x_train, self.x_test, self.y_train, self.y_test = train_test_split(x, y, test_size=0.2,random_state=0)

    self.y_train_r=norm_rev_y(self.y_train,self.train_stats_y)
    self.y_test_r=norm_rev_y(self.y_test,self.train_stats_y)

    self.treino_x=l=pd.DataFrame(self.x_train)
    self.treino_x=norm_rev_x(self.treino_x,self.train_stats_x)
    self.treino_x.columns=['n','f','a']

    self.treino_y=l=pd.DataFrame(self.y_train)
    self.treino_y=norm_rev_y(self.treino_y,self.train_stats_y)
    self.treino_y.columns=[self.grandeza]

    self.treino=np.round(self.treino_y.join(self.treino_x),2)

    self.teste_x=l=pd.DataFrame(self.x_test)
    self.teste_x=norm_rev_x(self.teste_x,self.train_stats_x)
    self.teste_x.columns=['n','f','a']

    self.teste_y=l=pd.DataFrame(self.y_test)
    self.teste_y=norm_rev_y(self.teste_y,self.train_stats_y)
    self.teste_y.columns=[self.grandeza]

    self.teste=np.round(self.teste_y.join(self.teste_x),2)

def criar_modelos(self,cv,ite):

    def tabela(df,doc):
        t = doc.add_table(df.shape[0]+1, df.shape[1])
```

```
    t.style = 'Medium List 1 Accent 1'

    for j in range(df.shape[-1]):
        t.cell(0,j).text = df.columns[j]

    for i in range(df.shape[0]):
        for j in range(df.shape[-1]):
            t.cell(i+1,j).text = str(df.values[i,j])

    for row in t.rows:
        for cell in row.cells:
            paragraphs = cell.paragraphs
            paragraph = paragraphs[0]
            run_obj = paragraph.runs
            run = run_obj[0]
            font = run.font
            font.size =  shared.Pt(11)

document = Document()
style = document.styles['Normal']
font = style.font
font.name = 'Times New Roman'
font.size = shared.Pt(12)

a=document.add_heading('Informações do estudo')
a.alignment=1
document.add_paragraph('Referência: {}'.format(self.referencia))
document.add_paragraph('Grandeza: {}'.format(self.grandeza))
document.add_paragraph('Tipo: {}'.format(self.tipo))
document.add_paragraph('Material: {}'.format(self.material))
document.add_paragraph('Ferramenta: {}'.format(self.ferramenta))
document.add_paragraph('Número de experimentos: {}'.format(self.numero))
document.add_paragraph('Observações:\n{}'.format(self.observacoes))
document.add_heading('Unidades')
document.add_paragraph('Velocidade: {}'.format(self.u_velocidade))
document.add_paragraph('Avanço: {}'.format(self.u_avanco))
document.add_paragraph('Profundidade de corte: {}'.format(self.u_profundidade))
document.add_paragraph('{}: {}'.format(self.grandeza,self.u_saida))
document.add_heading('Dados de teste')
tabela(self.teste,document)
```

```
document.add_heading('Dados de treino')
tabela(self.treino,document)
document.add_page_break()

self.dici_erros=[]
self.data_train_erro={}
self.data_test_erro={}
def erro(y_test_r,pred_test,y_train_r,pred_train,diretorio,name):

    erro_test=np.array([])
    erro_train=np.array([])

    for i in range(len(y_train_r)):
        erro_train=np.append(erro_train,abs(pred_train[i]-y_train_r[i])*100/y_train_r[i])

    for i in range(len(y_test_r)):
        erro_test=np.append(erro_test,abs(pred_test[i]-y_test_r[i])*100/y_test_r[i])



    erro_dic_train={diretorio+' Previsto': pred_train, diretorio+' Erro (%)': erro_train}
    erro_dic_test={diretorio+' Previsto': pred_test, diretorio+' Erro (%)': erro_test}

    self.data_train_erro.update(erro_dic_train)
    self.data_test_erro.update(erro_dic_test)

    errorel_test=np.round(sum(erro_test)/len(erro_test),2)
    errorel_train=np.round(sum(erro_train)/len(erro_train),2)

    cor_train=np.round(np.corrcoef(y_train_r,pred_train)[1,0],2)
    cor_test=np.round(np.corrcoef(y_test_r,pred_test)[1,0],2)
    det_test = np.round(r2_score(y_test_r,pred_test),2)
    det_train = np.round(r2_score(y_train_r,pred_train),2)
    mse_test = np.round(metrics.mean_squared_error(pred_test,y_test_r),2)
    mse_train = np.round(metrics.mean_squared_error(pred_train,y_train_r),2)
    rmse_test = np.round(np.sqrt(mse_test),2)
    rmse_train = np.round(np.sqrt(mse_train),2)

    self.dici_erros={'errorel_test': errorel_test, 'errorel_train': errorel_train,
            'cor_test': cor_test, 'cor_train': cor_train,
            'det_test': det_test, 'det_train': det_train,
```

```
            'mse_test': mse_test , 'mse_train': mse_train,
            'rmse_test': rmse_test, 'rmse_train': rmse_train}

    with open('Arquivos/'+diretorio+'/Erros/'+name+'_erros.json', 'w') as json_file:
        json.dump(self.dici_erros, json_file, indent=4)

fig,ax =plt.subplots(dpi=200)
fig2,ax2 =plt.subplots(dpi=200)
def graficos_ind(pred_test,y_test_r,pred_train,y_train_r,u_saida, diretorio,name,label_test,label_train):
    graph_scatter(pred_test,y_test_r,'Previsão (dados teste) - '+diretorio,
                'Arquivos/'+diretorio+'/Gráficos/'+name+'_teste.png',u_saida,label_test)
    graph_scatter(pred_train,y_train_r,'Previsão (dados treino) - '+diretorio,
                'Arquivos/'+diretorio+'/Gráficos/'+name+'_treino.png',u_saida,label_train)
```

Apêndice C.2 – Busca dos hiperparâmetros e aplicação das RNAs

```
import tensorflow as tf
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense
tf.compat.v1.set_random_seed(2)
seed(1)
from keras.wrappers.scikit_learn import KerasRegressor

def create_model(n,ln,fun,layer2,epochs):

    initializer = tf.compat.v1.keras.initializers.glorot_uniform(seed=0)
    model = Sequential()
    model.add(Dense(int(n), input_dim=self.x_train.shape[1], activation=fun,
                use_bias=True,kernel_initializer=initializer))
    if layer2==True:
        model.add(Dense(n, activation=fun,use_bias=True,kernel_initializer=initializer)) # Hidden 2
    model.add(Dense(1,kernel_initializer=initializer))
    optimizer = tf.keras.optimizers.Adam(learning_rate=ln)
    model.compile(loss='mean_squared_error', optimizer=optimizer,metrics=['mse'])
    return model

ln = [0.1,0.01,0.001, 0.0001]
n=randint(1,100)
layer2=[False,True]
```

```
        epochs=randint(1,1000)
        fun=['tanh','relu']

        param_grid = dict(n=n,ln=ln,fun=fun,layer2=layer2,epochs=epochs)

        model = KerasRegressor(build_fn=create_model, verbose=0)

        random = RandomizedSearchCV(estimator=model, param_distributions=param_grid, n_jobs=1,
cv=cv,n_iter=ite,verbose=2)

        random_result =random.fit(self.x_train, self.y_train)

        arquivo=open('Arquivos/RN/Iterações/'+self.name+'_iteraçoes.txt','w')

        arquivo.write("Melhores parâmetros: %f com %s" % (random_result.best_score_,
random_result.best_params_))
        arquivo.write('\n\n')

        means = random_result.cv_results_['mean_test_score']
        stds = random_result.cv_results_['std_test_score']
        params = random_result.cv_results_['params']

        iteracao={'Média': [],'Desvio': [],'n': [], 'ln': [],'2° camada': [], 'Função': [],'Épocas': []}

        for mean, stdev, param in zip(means, stds, params):

            iteracao['Média'].append(round(mean,4))
            iteracao['Desvio'].append(round(stdev,4))
            iteracao['n'].append(param['n'])
            iteracao['ln'].append(param['ln'])
            iteracao['2° camada'].append(param['layer2'])
            iteracao['Função'].append(param['fun'])
            iteracao['Épocas'].append(param['epochs'])

        self.iteracao=pd.DataFrame.from_dict(iteracao)
        arquivo.write(self.iteracao.to_string())
        arquivo.close()

        with open('Arquivos/RN/Parâmetros/'+self.name+'_parametros.json', 'w') as json_file:
            json.dump(random_result.best_params_, json_file, indent=4)
```

```
self.model=create_model(**random_result.best_params_)
history=self.model.fit(self.x_train,self.y_train,
            validation_data=(self.x_test,self.y_test),
            verbose=0,epochs=random_result.best_params_['epochs'])
history
self.model.save('Arquivos/RN/Modelos/'+self.name+'_modelo.mdl')

pred_train0=self.model.predict(self.x_train)
pred_train0_r=norm_rev_y(pred_train0,self.train_stats_y)
pred_train=np.array([])
for item in pred_train0_r:
   pred_train=np.append(pred_train,round(float(item),2))

pred_test0 = self.model.predict(self.x_test)
pred_test0_r=norm_rev_y(pred_test0,self.train_stats_y)
pred_test=np.array([])
for item in pred_test0_r:
   pred_test=np.append(pred_test,round(float(item),2))

det_test = np.round(r2_score(self.y_test_r,pred_test),2)
det_train = np.round(r2_score(self.y_train_r,pred_train),2)

erro(self.y_test_r,pred_test,self.y_train_r,pred_train,'RN',self.name)
graficos_ind(pred_test,self.y_test_r,pred_train,self.y_train_r,
        self.u_saida, 'RN',self.name,'RN $R^2$={:.2f}'.format(det_test),
        'RN $R^2$={:.2f}'.format(det_train))

det_test = np.round(r2_score(self.y_test_r,pred_test),2)
det_train = np.round(r2_score(self.y_train_r,pred_train),2)

ax.scatter(self.y_test_r,pred_test,label='RN $R^2$={:.2f}'.format(det_test),marker='s')
ax2.scatter(self.y_train_r,pred_train,label='RN $R^2$={:.2f}'.format(det_train),marker='s')

arquivo2=open('Arquivos/RN/Pesos/'+self.name+'_pesos.txt','w')
k=self.model.get_weights()

arquivo2.write('Pesos - camada oculta 1')
arquivo2.write('\n')
arquivo2.write(str(k[0]))
```

```
arquivo2.write('\n')
arquivo2.write('Bias - camada oculta')
arquivo2.write('\n')
arquivo2.write(str(k[1]))
arquivo2.write('\n')
if random_result.best_params_['layer2']==False:
    arquivo2.write('Pesos - camada saída')
    arquivo2.write('\n')
    arquivo2.write(str(k[2].transpose()))
    arquivo2.write('\n')

elif random_result.best_params_['layer2']==True:
    arquivo2.write('Pesos - camada oculta 2')
    arquivo2.write('\n')
    arquivo2.write(str(k[2]))
    arquivo2.write('\n')
    arquivo2.write('Bias - camada oculta 2')
    arquivo2.write('\n')
    arquivo2.write(str(k[3]))
    arquivo2.write('\n')
    arquivo2.write('Pesos - camada saída')
    arquivo2.write('\n')
    arquivo2.write(str(k[4].transpose()))
    arquivo2.write('\n')

arquivo2.close()

c=document.add_heading('RN')
c.alignment=1

document.add_paragraph('Número de neurônios: {}'.format(int(random_result.best_params_['n'])))
document.add_paragraph('Taxa de aprendizado: {:e}'.format(random_result.best_params_['ln']))
document.add_paragraph('Número de épocas: {}'.format(int(random_result.best_params_['epochs'])))
document.add_paragraph('2° camada: {}'.format(random_result.best_params_['layer2']))
document.add_paragraph('Função de ativação: {}'.format(random_result.best_params_['fun']))

a=document.add_heading('Erros')
a.alignment=1
p=document.add_paragraph()
p.add_run('Dados de teste').bold = True
```

```
        document.add_paragraph('Erro relativo médio: {}'.format(self.dici_erros['errorel_test']),style='List Bullet')
        document.add_paragraph('Coeficiente de correlação: {}'.format(self.dici_erros['cor_test']),style='List
Bullet')
        document.add_paragraph('Coeficiente de determinação: {}'.format(self.dici_erros['det_test']),style='List
Bullet')
        document.add_paragraph('MSE: {}'.format(self.dici_erros['mse_test']),style='List Bullet')
        document.add_paragraph('RMSE: {}'.format(self.dici_erros['rmse_test']),style='List Bullet')
        a=document.add_picture('Arquivos/RN/Gráficos/'+self.name+'_teste.png')
        a.alignment=1

        p=document.add_paragraph()
        p.add_run('Dados de treino').bold = True
        document.add_paragraph('Erro relativo médio: {}'.format(self.dici_erros['errorel_train']),style='List Bullet')
        document.add_paragraph('Coeficiente de correlação: {}'.format(self.dici_erros['cor_train']),style='List
Bullet')
        document.add_paragraph('Coeficiente de determinação: {}'.format(self.dici_erros['det_train']),style='List
Bullet')
        document.add_paragraph('MSE: {}'.format(self.dici_erros['mse_train']),style='List Bullet')
        document.add_paragraph('RMSE: {}'.format(self.dici_erros['rmse_train']),style='List Bullet')
        a=document.add_picture('Arquivos/RN/Gráficos/'+self.name+'_treino.png')
        a.alignment=1

        a=document.add_heading('Pesos')
        a.alignment=1

        k=self.model.get_weights()
        document.add_paragraph('Pesos - camada oculta 1')
        document.add_paragraph(str(k[0]))
        document.add_paragraph('Bias - camada oculta')
        document.add_paragraph(str(k[1]))
        if random_result.best_params_['layer2']==False:
            document.add_paragraph('Pesos - camada saída')
            document.add_paragraph(str(k[2].transpose()))

        elif random_result.best_params_['layer2']==True:
            document.add_paragraph('Pesos - camada oculta 2')
            document.add_paragraph(str(k[2]))
            document.add_paragraph('Bias - camada oculta 2')
            document.add_paragraph(str(k[3]))
            document.add_paragraph('Pesos - camada saída')
```

```
    document.add_paragraph(str(k[4].transpose()))


a=document.add_heading('Iterações')
a.alignment=1

tabela(self.iteracao,document)
document.add_page_break()
```

Apêndice C.3 – Aplicação das regressões

```
degree=0
for reg in ['RL','RP2','RP3','RP4']:
    degree+=1
    self.par=PolynomialFeatures(degree)
    self.line=LinearRegression()

    self.poly= make_pipeline(self.par, self.line,verbose=1)
    self.poly.fit(self.x_train,self.y_train)

    joblib.dump(self.poly, 'Arquivos/'+reg+'/Modelos/'+self.name+'_modelo.mdl')

    arquivo3=open('Arquivos/'+reg+'/Coeficientes/'+self.name+'_coeficientes.txt','w')
    arquivo3.write(str(self.line.coef_))
    arquivo3.close()

    pred_train0=self.poly.predict(self.x_train)
    pred_train0_r=norm_rev_y(pred_train0,self.train_stats_y)
    pred_train=np.array([])
    for item in pred_train0_r:
        pred_train=np.append(pred_train,round(float(item),2))

    pred_test0 = self.poly.predict(self.x_test)
    pred_test0_r=norm_rev_y(pred_test0,self.train_stats_y)
    pred_test=np.array([])
    for item in pred_test0_r:
        pred_test=np.append(pred_test,round(float(item),2))

    poui=(self.poly.score(self.x_test,self.y_test))
```

```
        det_test = np.round(r2_score(self.y_test_r,pred_test),2)
        det_train = np.round(r2_score(self.y_train_r,pred_train),2)

        erro(self.y_test_r,pred_test,self.y_train_r,pred_train,reg,self.name)
        graficos_ind(pred_test,self.y_test_r,pred_train,
                self.y_train_r,self.u_saida,reg ,self.name,
                reg+' $R^2$={:.2f}'.format(det_test),reg+' $R^2$={:.2f}'.format(det_train))

        ax.scatter(self.y_test_r,pred_test,label=reg+' $R^2$={:.2f}'.format(det_test))
        ax2.scatter(self.y_train_r,pred_train,label=reg+' $R^2$={:.2f}'.format(det_train))

        c=document.add_heading(reg)
        c.alignment=1

        c=document.add_heading('Coeficientes')
        c.alignment=1
        document.add_paragraph(str(self.line.coef_))

        a=document.add_heading('Erros')
        a.alignment=1
        p=document.add_paragraph()
        p.add_run('Dados de teste').bold = True
        document.add_paragraph('Erro relativo médio: {}'.format(self.dici_erros['errorel_test']),style='List
Bullet')
        document.add_paragraph('Coeficiente de correlação: {}'.format(self.dici_erros['cor_test']),style='List
Bullet')
        document.add_paragraph('Coeficiente de determinação: {}'.format(self.dici_erros['det_test']),style='List
Bullet')
        document.add_paragraph('MSE: {}'.format(self.dici_erros['mse_test']),style='List Bullet')
        document.add_paragraph('RMSE: {}'.format(self.dici_erros['rmse_test']),style='List Bullet')
        a=document.add_picture('Arquivos/'+reg+'/Gráficos/'+self.name+'_teste.png')
        a.alignment=1

        p=document.add_paragraph()
        p.add_run('Dados de treino').bold = True
        document.add_paragraph('Erro relativo médio: {}'.format(self.dici_erros['errorel_train']),style='List
Bullet')
        document.add_paragraph('Coeficiente de correlação: {}'.format(self.dici_erros['cor_train']),style='List
Bullet')
```

```
        document.add_paragraph('Coeficiente de determinação: {}'.format(self.dici_erros['det_train']),style='List
Bullet')
        document.add_paragraph('MSE: {}'.format(self.dici_erros['mse_train']),style='List Bullet')
        document.add_paragraph('RMSE: {}'.format(self.dici_erros['rmse_train']),style='List Bullet')
        a=document.add_picture('Arquivos/'+reg+'/Gráficos/'+self.name+'_treino.png')
        a.alignment=1
        document.add_page_break()

    ax.legend()
    menor=min([ax.get_xlim()[0],ax.get_ylim()[0]])
    maior=max([ax.get_xlim()[1],ax.get_ylim()[1]])
    ax.set_xlim(menor,maior)
    ax.set_ylim(menor,maior)
    ax.plot([menor,maior], [menor,maior],c='black',linewidth=0.65)
    ax.title.set_text('Modelos de previsão (dados de teste)')
    ax.set(xlabel="Valor real ({})".format(self.u_saida),ylabel="Previsão ({})".format(self.u_saida))
    fig.savefig('Arquivos/Gráficos/'+self.name+'_teste.png',bbox_inches='tight')

    ax2.legend()
    menor=min([ax2.get_xlim()[0],ax2.get_ylim()[0]])
    maior=max([ax2.get_xlim()[1],ax2.get_ylim()[1]])
    ax2.set_xlim(menor,maior)
    ax2.set_ylim(menor,maior)
    ax2.plot([menor,maior], [menor,maior],c='black',linewidth=0.65)
    ax2.title.set_text('Modelos de previsão (dados de treino)')
    ax2.set(xlabel="Valor real ({})".format(self.u_saida),ylabel="Previsão ({})".format(self.u_saida))
    fig2.savefig('Arquivos/Gráficos/'+self.name+'_treino.png',bbox_inches='tight')

    plt.close

    data_train={'Valor real':self.y_train_r}
    data_train.update(self.data_train_erro)
    data_test={'Valor real':self.y_test_r}
    data_test.update(self.data_test_erro)
    self.data_train=np.round(pd.DataFrame(data=data_train),2)
    self.data_test=np.round(pd.DataFrame(data=data_test),2)

    a=document.add_heading('Geral')
    a.alignment=1
    a=document.add_picture('Arquivos/Gráficos/'+self.name+'_teste.png')
```

```
    a.alignment=1
    a=document.add_picture('Arquivos/Gráficos/'+self.name+'_treino.png')
    a.alignment=1
    document.add_page_break()
    p=document.add_paragraph()
    p.add_run('Dados de teste').bold = True
    tabela(self.data_test,document)
    document.add_paragraph()
    p=document.add_paragraph()
    p.add_run('Dados de treino').bold = True
    tabela(self.data_train,document)

    document.save('Arquivos/Relatórios/'+self.name+'_relatorio.docx')

def variaveis_exibir(self):

    self.errorel_test={}
    self.errorel_train={}
    self.cor_test={}
    self.cor_train={}
    self.det_test={}
    self.det_train={}
    self.mse_test={}
    self.mse_train={}
    self.rmse_test={}
    self.rmse_train={}


    with open('Arquivos/RN/Parâmetros/'+self.name+'_parametros.json', 'r') as json_file:
        parametros = json.load(json_file)
    self.neuronios=parametros['n']
    self.ln=parametros['ln']
    self.epocas=parametros['epochs']
    self.funcao=parametros['fun']
    self.layer=parametros['layer2']
    if self.layer==True:
        self.layer=2
    elif self.layer==False:
        self.layer=1
```

```
    for reg in ['RL','RP2','RP3','RP4','RN']:

        with open('Arquivos/'+reg+'/Erros/'+self.name+'_erros.json', 'r') as json_file:
            rn_erros = json.load(json_file)
        self.errorel_test[reg]=rn_erros['errorel_test']
        self.errorel_train[reg]=rn_erros['errorel_train']
        self.cor_test[reg]=rn_erros['cor_test']
        self.cor_train[reg]=rn_erros['cor_train']
        self.det_test[reg]=rn_erros['det_test']
        self.det_train[reg]=rn_erros['det_train']
        self.mse_test[reg]=rn_erros['mse_test']
        self.mse_train[reg]=rn_erros['mse_train']
        self.rmse_test[reg]=rn_erros['rmse_test']
        self.rmse_train[reg]=rn_erros['rmse_train']

def carregar(self):
    from tensorflow.keras.models import load_model

    self.model={}
    self.model['RN']=load_model('Arquivos/RN/Modelos/'+self.name+'_modelo.mdl')

    for reg in ['RL','RP2','RP3','RP4']:
        self.model[reg]=joblib.load('Arquivos/'+reg+'/Modelos/'+self.name+'_modelo.mdl')

def pred(self,v,f,a):
    self.pred={}
    r=[[v,f,a]]
    for reg in ['RL','RP2','RP3','RP4','RN']:
        self.r=norm_x(r,self.train_stats_x)
        self.pred[reg]=self.model[reg].predict(self.r)
        self.pred[reg]=norm_rev_y(self.pred[reg],self.train_stats_y)
        self.pred[reg]=np.round(float(self.pred[reg]),2)
    return(self.pred)
```

Apêndice C.4 – Interface gráfica

```
import tkinter as tk
```

```
import tkinter.ttk as ttk

fonte=('Segoe UI',10,'bold')

class SampleApp(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        container = ttk.Frame(self)
        container.pack(side="right", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}
        for F in (Pagina1,Pagina2,Pagina3,Pagina4,Pagina5,Pagina6):
            page_name = F.__name__
            frame = F(parent=container, controller=self)
            self.frames[page_name] = frame
            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame("Pagina1")

    def show_frame(self, page_name):
        frame = self.frames[page_name]
        frame.tkraise()

class Pagina1(ttk.Frame):

    def __init__(self, parent, controller):
        ttk.Frame.__init__(self, parent)

        self.controller = controller

        self.button_1 = ttk.Button(self,text='Abrir')
        self.button_1.grid(column='0', row='4',pady=10)

        self.button_2 = ttk.Button(self,text='Novo')
        self.button_2.grid(column='1', row='4',pady=10)
```

```
        self.button_3 = ttk.Button(self,text='Editar')
        self.button_3.grid(column='2', row='4',pady=10)

        self.button_4 = ttk.Button(self,text='Deletar')
        self.button_4.grid(column='3', row='4',pady=10)

        self.tree = ttk.Treeview(self,height='30')
        self.tree["columns"]=("one","two")
        self.tree.column('#0', width=300 )
        self.tree.heading("#0", text="Referência")
        self.tree.column("one", width=250 )
        self.tree.column("two", width=250)
        self.tree.heading("one", text="Material")
        self.tree.heading("two", text="Ferramenta")
        self.tree.grid(column='0', columnspan='4', row='3',padx=10,pady=10)

        self.grid_columnconfigure(0, weight=1)
        self.grid_columnconfigure(1, weight=1)
        self.grid_columnconfigure(2, weight=1)
        self.grid_columnconfigure(3, weight=1)

class Pagina2(ttk.Frame):

    def __init__(self, parent, controller):
        ttk.Frame.__init__(self, parent)
        self.controller = controller

        self.label_10 = ttk.Label(self,text='Grandeza:')
        self.label_10.grid(padx='10', pady='10', row='1', sticky='e')

        self.text_11 = tk.Text(self, height='1',  width='50')
        self.text_11.grid(column='1', padx='10', pady='10', row='1')

        self.label_40 = ttk.Label(self,text='Material:')
        self.label_40.grid(column='0', padx='10', pady='10', row='4', sticky='e')

        self.text_41 = tk.Text(self,height='1', width='50')
        self.text_41.grid(column='1', padx='10', pady='10', row='3')

        self.label_30 = ttk.Label(self,text='Ferramenta de corte:')
```

```
self.label_30.grid(column='0', padx='10', pady='10', row='3', sticky='e')

self.text_31 = tk.Text(self,height='1', state='normal', width='50')
self.text_31.grid(column='1', padx='10', pady='10', row='4')

self.label_50 = ttk.Label(self,text='N° de experimentos:')
self.label_50.grid(column='0', padx='10', pady='10', row='5', sticky='e')

self.text_51 = tk.Text(self,height='1', width='50')
self.text_51.grid(column='1', padx='10', pady='10', row='5')

self.label_60 = ttk.Label(self,text='Observações:')
self.label_60.grid(column='0', padx='10', pady='10', row='6', sticky='ne')

self.text_61 = tk.Text(self,height='10', width='50')
self.text_61.grid(column='1', padx='10', pady='10', row='6')

self.label_20 = ttk.Label(self,text='Referência:')
self.label_20.grid(column='0', padx='10', pady='10', row='2', sticky='ne')

self.text_21 = tk.Text(self,height='1', undo='false', width='50')
self.text_21.grid(column='1', padx='10', pady='10', row='2')

self.separator_1 = ttk.Separator(self,orient='vertical')
self.separator_1.grid(column='2', row='0', rowspan='11',sticky='ns',padx='15', pady='10')

self.label_03 = ttk.Label(self,text='Estimativa',font=fonte)
self.label_03.grid(column='3', padx='10', pady='10', row='0',columnspan='2')

self.label_00 = ttk.Label(self,text='Informações do estudo',font=fonte)
self.label_00.grid(column='0', columnspan='2', padx='10', pady='10', row='0')

self.label_13 = ttk.Label(self,text='Dados de entrada:')
self.label_13.grid(column='3', padx='10', pady='10', row='1')

self.label_23 = ttk.Label(self,text='Velocidade:')
self.label_23.grid(column='3', padx='10', pady='10', row='2', sticky='e')

self.label_33 = ttk.Label(self,text='Avanço:')
self.label_33.grid(column='3', padx='10', pady='10', row='3', sticky='e')
```

```
self.label_43 = ttk.Label(self,text='Profundidade de corte:')
self.label_43.grid(column='3', padx='10', pady='10', row='4', sticky='e')

self.entry_v = tk.Entry(self,width='15')
self.entry_v.grid(column='4', padx='10', pady='10', row='2')

self.u_v = ttk.Label(self,text='')
self.u_v.grid(column='5', padx='10', pady='10', row='2', sticky='nw')

self.entry_f = tk.Entry(self,width='15')
self.entry_f.grid(column='4', padx='10', pady='10', row='3')

self.u_f = ttk.Label(self,text='')
self.u_f.grid(column='5', padx='10', pady='10', row='3', sticky='nw')

self.entry_a = tk.Entry(self,width='15')
self.entry_a.grid(column='4', padx='10', pady='10', row='4')

self.u_a = ttk.Label(self,text='')
self.u_a.grid(column='5', padx='10', pady='10', row='4', sticky='nw')

self.label_63 = ttk.Label(self,text='Saída:')
self.label_63.grid(column='3', padx='10', pady='10', row='6', sticky='ne')

self.text_64 = tk.Text(self,height='1', width='11')
self.text_64.grid(column='4', padx='10', pady='10', row='6', sticky='ne')

self.u_s = ttk.Label(self,text='')
self.u_s.grid(column='5', padx='10', pady='10', row='6', sticky='nw')

self.label_70=ttk.Label(self,text='Informações da rede neural' ,font=fonte)
self.label_70.grid(column='0',row='7',padx='10', pady='10',columnspan='2')

self.frame_inf=ttk.Frame(self)
self.frame_inf.grid(column='0',row='8',columnspan='2')

self.label_80=ttk.Label(self.frame_inf,text='Número de neurônios:')
self.label_80.grid(column='0',row='0',padx='10', pady='10', sticky='ne')
```

```
self.text_81 = tk.Text(self.frame_inf,height='1', width='10')
self.text_81.grid(column='1', padx='10', pady='10', row='0', sticky='nw')

self.label_90=ttk.Label(self.frame_inf,text='Taxa de aprendizado:')
self.label_90.grid(column='0',row='1',padx='10', pady='10', sticky='ne')

self.text_91 = tk.Text(self.frame_inf,height='1', width='10')
self.text_91.grid(column='1', padx='10', pady='10', row='1', sticky='nw')

self.label_100=ttk.Label(self.frame_inf,text='Épocas:')
self.label_100.grid(column='0',row='2',padx='10', pady='10', sticky='ne')

self.text_101 = tk.Text(self.frame_inf,height='1', width='10')
self.text_101.grid(column='1', padx='10', pady='10', row='2', sticky='nw')

self.label_2cam=ttk.Label(self.frame_inf,text='N° de camadas:')
self.label_2cam.grid(column='2',row='0',padx='10', pady='10', sticky='ne')

self.text_2cam = tk.Text(self.frame_inf,height='1', width='10')
self.text_2cam.grid(column='3', padx='10', pady='10', row='0', sticky='nw')

self.label_fun=ttk.Label(self.frame_inf,text='Função de ativação:')
self.label_fun.grid(column='2',row='1',padx='10', pady='10', sticky='ne')

self.text_fun = tk.Text(self.frame_inf,height='1', width='10')
self.text_fun.grid(column='3', padx='10', pady='10', row='1', sticky='nw')



self.button_1 = ttk.Button(self,text='Previsão')
self.button_1.grid(column='3', columnspan='2', padx='10', pady='10', row='5')

self.frame_button=ttk.Frame(self)
self.frame_button.grid(column='0', row='11',columnspan='3',sticky='e', padx='10', pady='10')

self.button_2 = ttk.Button(self.frame_button,text='Dados experimentais', width='20')
self.button_2.grid(column='0', row='0',padx='10', pady='10')

self.button_3 = ttk.Button(self.frame_button,text='Dados do modelo', width='20')
self.button_3.grid(column='1', row='0',padx='10', pady='10')
```

```
        self.button_4 = ttk.Button(self,text='Voltar')
        self.button_4.place(relx=0.9,rely=0.95)

        self.button_5=ttk.Button(self.frame_button,text='Abrir artigo',width='20')
        self.button_5.grid(column='2', row='0',padx='10', pady='10')

        self.button_6=ttk.Button(self.frame_button,text='Iterações',width='20')
        self.button_6.grid(column='0', row='1',padx='10', pady='10')

        self.button_7=ttk.Button(self.frame_button,text='Pesos',width='20')
        self.button_7.grid(column='1', row='1',padx='10', pady='10')

        self.button_8=ttk.Button(self.frame_button,text='Relatório',width='20')
        self.button_8.grid(column='2', row='1',padx='10', pady='10')


class Pagina3(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        self.label_10 = ttk.Label(self,text='Erro relativo médio:')
        self.label_10.grid(padx='10', pady='10', row='1', sticky='e')

        self.text_11 = tk.Text(self,height='1', width='10')
        self.text_11.grid(column='1', padx='10', pady='10', row='1')

        self.label_40 = ttk.Label(self,text='MSE:')
        self.label_40.grid(column='0', padx='10', pady='10', row='4', sticky='e')

        self.text_31 = tk.Text(self,height='1', width='10')
        self.text_31.grid(column='1', padx='10', pady='10', row='3')

        self.label_30 = ttk.Label(self,text='Coeficiente de determinaçao:')
        self.label_30.grid(column='0', padx='10', pady='10', row='3', sticky='e')

        self.text_41 = tk.Text(self,height='1', width='10')
        self.text_41.grid(column='1', padx='10', pady='10', row='4')
```

```
self.label_50 = ttk.Label(self,text='RMSE:')
self.label_50.grid(column='0', padx='10', pady='10', row='5', sticky='e')

self.text_51 = tk.Text(self,height='1', width='10')
self.text_51.grid(column='1', padx='10', pady='10', row='5')

self.label_20 = ttk.Label(self,text='Coeficiente de correlação:')
self.label_20.grid(column='0', padx='10', pady='10', row='2', sticky='ne')

self.text_21= tk.Text(self,height='1', width='10')
self.text_21.grid(column='1', padx='10', pady='10', row='2')

self.separator = ttk.Separator(self,orient='vertical')
self.separator.grid(column='2', row='0', rowspan='6',padx='15', pady='10', sticky='ns')

self.label_03 = ttk.Label(self,text='Dados de treino',font=fonte)
self.label_03.grid(column='3', columnspan='2', padx='10', pady='10', row='0')

self.label_00 = ttk.Label(self,text='Dados de teste',font=fonte)
self.label_00.grid(column='0', columnspan='2', padx='10', pady='10', row='0')

self.label_13 = ttk.Label(self,text='Erro relativo médio:')
self.label_13.grid(padx='10', pady='10', row='1', sticky='e',column='3')

self.text_14 = tk.Text(self,height='1', width='10')
self.text_14.grid(column='4', padx='10', pady='10', row='1')

self.label_43 = ttk.Label(self,text='MSE:')
self.label_43.grid(column='3', padx='10', pady='10', row='4', sticky='e')

self.text_34 = tk.Text(self,height='1', width='10')
self.text_34.grid(column='4', padx='10', pady='10', row='3')

self.label_33 = ttk.Label(self,text='Coeficiente de determinaçao:')
self.label_33.grid(column='3', padx='10', pady='10', row='3', sticky='e')

self.text_44 = tk.Text(self,height='1', width='10')
self.text_44.grid(column='4', padx='10', pady='10', row='4')

self.label_53 = ttk.Label(self,text='RMSE:')
```

```
        self.label_53.grid(column='3', padx='10', pady='10', row='5', sticky='e')

        self.text_54 = tk.Text(self,height='1', width='10')
        self.text_54.grid(column='4', padx='10', pady='10', row='5')

        self.label_23 = ttk.Label(self,text='Coeficiente de correlação:')
        self.label_23.grid(column='3', padx='10', pady='10', row='2', sticky='ne')

        self.text_24 = tk.Text(self,height='1', width='10')
        self.text_24.grid(column='4', padx='10', pady='10', row='2')

        self.button_4 = ttk.Button(self,text='Voltar')
        self.button_4.place(relx=0.9,rely=0.95)

class Pagina4(ttk.Frame):

    def __init__(self, parent, controller):
        ttk.Frame.__init__(self, parent)
        self.controller = controller

        self.frame_ed=ttk.Frame(self)
        self.frame_ed.place(anchor='c',relx=0.5,rely=0.4)

        self.label_00=ttk.Label(self.frame_ed,text='Edição de modelo',font=fonte,anchor='c')
        self.label_00.grid(padx='10', pady='10', row='0', sticky='nesw',column='0',columnspan='2')

        self.label_10 = ttk.Label(self.frame_ed,text='Grandeza:')
        self.label_10.grid(padx='10', pady='10', row='1', sticky='e')

        self.text_11 = tk.Text(self.frame_ed, height='1',  width='50')
        self.text_11.grid(column='1', padx='10', pady='10', row='1')

        self.label_40 = ttk.Label(self.frame_ed,text='Material:')
        self.label_40.grid(column='0', padx='10', pady='10', row='4', sticky='e')

        self.text_31 = tk.Text(self.frame_ed,height='1', width='50')
        self.text_31.grid(column='1', padx='10', pady='10', row='3')

        self.label_30 = ttk.Label(self.frame_ed,text='Ferramenta de corte:')
        self.label_30.grid(column='0', padx='10', pady='10', row='3', sticky='e')
```

```
        self.text_41 = tk.Text(self.frame_ed,height='1', state='normal', width='50')
        self.text_41.grid(column='1', padx='10', pady='10', row='4')

        self.label_50 = ttk.Label(self.frame_ed,text='N° de experimentos:')
        self.label_50.grid(column='0', padx='10', pady='10', row='5', sticky='e')

        self.text_51 = tk.Text(self.frame_ed,height='1', width='50')
        self.text_51.grid(column='1', padx='10', pady='10', row='5')

        self.label_60 = ttk.Label(self.frame_ed,text='Observações:')
        self.label_60.grid(column='0', padx='10', pady='10', row='6', sticky='ne')

        self.text_61 = tk.Text(self.frame_ed,height='10', width='50')
        self.text_61.grid(column='1', padx='10', pady='10', row='6')

        self.label_20 = ttk.Label(self.frame_ed,text='Referência:')
        self.label_20.grid(column='0', padx='10', pady='10', row='2', sticky='ne')

        self.text_21 = tk.Text(self.frame_ed,height='1', undo='false', width='50')
        self.text_21.grid(column='1', padx='10', pady='10', row='2')

        self.button_1 = ttk.Button(self,text='Atualizar')
        self.button_1.place(relx=0.5,rely=0.75,anchor='c')

        self.button_4 = ttk.Button(self,text='Voltar')
        self.button_4.place(relx=0.9,rely=0.95)

class Pagina5(ttk.Frame):

    def __init__(self, parent, controller):
        ttk.Frame.__init__(self, parent)
        self.controller = controller

        self.label_00=ttk.Label(self,text='Criação de modelo',font=fonte,anchor='c')
        self.label_00.grid(padx='10', pady='10',columnspan='4')

        self.frame_but=ttk.Frame(self)
        self.frame_but.grid(column='1',row='1',sticky='w',padx='10', pady='10')
```

```
self.label_11=ttk.Label(self,text='Grandeza:')
self.label_11.grid(row='1',padx='10', pady='10',sticky='e')

self.tipo = tk.StringVar()
self.tipo.set('Força')
self.radiobutton_força=ttk.Radiobutton(self.frame_but,text="Força",
                        variable=self.tipo, value='Força')
self.radiobutton_rugosidade=ttk.Radiobutton(self.frame_but,text="Rugosidade",
                        variable=self.tipo, value='Rugosidade')
self.radiobutton_força.grid(sticky='w',row='0',column='0', padx='10')
self.radiobutton_rugosidade.grid(sticky='w',row='0',column='1', padx='10')

self.label_40 = ttk.Label(self,text='Material:')
self.label_40.grid(column='0', padx='10', pady='10', row='4', sticky='e')

self.text_31 = tk.Text(self,height='1', width='50')
self.text_31.grid(column='1', padx='10', pady='10', row='3')

self.label_30 = ttk.Label(self,text='Ferramenta de corte:')
self.label_30.grid(column='0', padx='10', pady='10', row='3', sticky='e')

self.text_41 = tk.Text(self,height='1', state='normal', width='50')
self.text_41.grid(column='1', padx='10', pady='10', row='4')

self.label_50 = ttk.Label(self,text='N° de experimentos:')
self.label_50.grid(column='0', padx='10', pady='10', row='5', sticky='e')

self.text_51 = tk.Text(self,height='1', width='50')
self.text_51.grid(column='1', padx='10', pady='10', row='5')

self.label_60 = ttk.Label(self,text='Observações:')
self.label_60.grid(column='0', padx='10', pady='10', row='6', sticky='ne')

self.text_61 = tk.Text(self,height='10', width='50')
self.text_61.grid(column='1', padx='10', pady='10', row='6')

self.label_20 = ttk.Label(self,text='Referência:')
self.label_20.grid(column='0', padx='10', pady='10', row='2', sticky='ne')

self.text_21 = tk.Text(self,height='1', state='normal', width='50')
```

```
        self.text_21.grid(column='1', padx='10', pady='10', row='2')

        self.label_02= tk.Label(self, text= 'Unidades:')
        self.label_02.grid(column='2', padx='10', pady='10', row='1', sticky='nesw')

        self.label_12 = ttk.Label(self,text='Velocidade:')
        self.label_12.grid(column='2', padx='10', pady='10', row='2', sticky='ne')

        self.text_13 = tk.Text(self,height='1', state='normal', width='15')
        self.text_13.grid(column='3', padx='10', pady='10', row='2')

        self.label_22 = ttk.Label(self,text='Avanço:')
        self.label_22.grid(column='2', padx='10', pady='10', row='3', sticky='ne')

        self.text_23 = tk.Text(self,height='1', state='normal', width='15')
        self.text_23.grid(column='3', padx='10', pady='10', row='3')

        self.label_32 = ttk.Label(self,text='Profundidade de corte:')
        self.label_32.grid(column='2', padx='10', pady='10', row='4', sticky='ne')

        self.text_33 = tk.Text(self,height='1', state='normal', width='15')
        self.text_33.grid(column='3', padx='10', pady='10', row='4')

        self.label_42 = ttk.Label(self,text='Força / Rugosidade:')
        self.label_42.grid(column='2', padx='10', pady='10', row='5', sticky='ne')

        self.text_43 = tk.Text(self,height='1', state='normal', width='15')
        self.text_43.grid(column='3', padx='10', pady='10', row='5')

        self.button_1 = ttk.Button(self,text='Anexar artigo',width='25')
        self.button_1.place(relx=0.1,rely=0.65)

        self.button_2=ttk.Button(self,text='Inserir dados experimentais',state='disabled',width='25')
        self.button_2.place(relx=0.35,rely=0.65)

        self.button_4 = ttk.Button(self,text='Voltar')
        self.button_4.place(relx=0.9,rely=0.95)


class Pagina6(ttk.Frame):
```

```
    def __init__(self, parent, controller):
        ttk.Frame.__init__(self, parent)
        self.controller = controller

        self.label_1=tk.Label(self,text='Dados de teste',font=fonte)
        self.label_1.grid(row='0',column='0', padx='10', pady='10')

        self.label_2=tk.Label(self,text='Dados de treino',font=fonte)
        self.label_2.grid(row='0',column='1', padx='10', pady='10')

        self.frame_1=tk.Frame(self)
        self.frame_1.grid(row='1',column='0', padx='10', pady='10')

        self.frame_2=tk.Frame(self)
        self.frame_2.grid(row='1',column='1', padx='10', pady='10')

        self.grid_columnconfigure(0, weight=1)
        self.grid_columnconfigure(1, weight=1)

        self.button=ttk.Button(self,text='Voltar')
        self.button.place(relx=0.9,rely=0.95)


class SampleApp2(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        container = ttk.Frame(self)
        container.pack(side="top", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}
        for F in (pagina,pagina2,pagina3):
            page_name = F.__name__
            frame = F(parent=container, controller=self)
            self.frames[page_name] = frame
            frame.grid(row=0, column=0, sticky="nsew")
```

```
        self.show_frame("pagina")

    def show_frame(self, page_name):
        frame = self.frames[page_name]
        frame.tkraise()

class pagina(ttk.Frame):
    def __init__(self, parent, controller):
        ttk.Frame.__init__(self, parent)
        self.controller = controller

        self.label_0=ttk.Label(self,text='Validação cruzada:')
        self.label_0.grid(column='0',row='0',padx='10', pady='10',  sticky='e')

        self.entry_0=ttk.Entry(self)
        self.entry_0.grid(column='1',row='0')

        self.label_1=ttk.Label(self,text='Número de iterações:')
        self.label_1.grid(column='0',row='1',padx='10', pady='10',  sticky='e')

        self.entry=ttk.Entry(self)
        self.entry.grid(column='1',row='1')

        self.label=ttk.Label(self,text='Após salvar os dados experimentais no excel aperte Ok para gerar o modelo
de rede.')
        self.label.grid(column='0',row='2',columnspan=2,padx='10', pady='10')

        self.button=ttk.Button(self,text='Ok')
        self.button.grid(column='0',row='3',columnspan=2,padx='10', pady='10')

class pagina2(ttk.Frame):
    def __init__(self, parent, controller):
        ttk.Frame.__init__(self, parent)
        self.controller = controller

        self.label=ttk.Label(self,text='Criando rede neural')
        self.label.place(relx=0.5, rely=0.3, anchor='center')
        self.bar=ttk.Progressbar(self, orient = 'horizontal',
            length = 100, mode = 'indeterminate')
```

```
        self.bar.place(relx=0.5, rely=0.7, anchor='center')


class pagina3(ttk.Frame):
    def __init__(self, parent, controller):
        ttk.Frame.__init__(self, parent)
        self.controller = controller

        self.label=ttk.Label(self,text='Modelo criado')
        self.label.place(relx=0.5, rely=0.3, anchor='center')
        self.button=ttk.Button(self,text='Fechar')
        self.button.place(relx=0.5, rely=0.7, anchor='center')
```

Apêndice C.5 – Banco de dados para os estudos

```
import sqlite3 as sql

class TransactionObject():
    database    = "Arquivos/dados_estudo.db"
    conn        = None
    cur         = None
    connected   = False

    def connect(self):
        TransactionObject.conn      = sql.connect(TransactionObject.database)
        TransactionObject.cur       = TransactionObject.conn.cursor()
        TransactionObject.connected = True

    def disconnect(self):
        TransactionObject.conn.close()
        TransactionObject.connected = False

    def execute(self, sql, parms = None):
        if TransactionObject.connected:
            if parms == None:
                TransactionObject.cur.execute(sql)
            else:
                TransactionObject.cur.execute(sql, parms)
            return True
        else:
```

```
            return False

    def fetchall(self):
        return TransactionObject.cur.fetchall()

    def persist(self):
        if TransactionObject.connected:
            TransactionObject.conn.commit()
            return True
        else:
            return False

def initDB():
    trans = TransactionObject()
    trans.connect()
    trans.execute("""
        CREATE TABLE IF NOT EXISTS dados_estudo (
        referencia TEXT NOT NULL,
        grandeza TEXT NOT NULL,
        tipo TEXT NOT NULL,
        ferramenta TEXT NOT NULL,
        material TEXT NOT NULL,
        numero INTEGER,
        observaçoes TEXT NOT NULL,
        uvelocidade TEXT,
        uavanco TEXT,
        uprofundidade TEXT,
        usaida TEXT
        );
        """)

    trans.persist()
    trans.disconnect()

def
insert(referencia,grandeza,tipo,ferramenta,material,numero,observaçoes,u_velocidade,u_avanco,u_profundidade,
u_saida):
    trans = TransactionObject()
    trans.connect()
```

```
    trans.execute("INSERT INTO dados_estudo VALUES(?,?,?,?,?,?,?,?,?,?,?)",
(referencia,grandeza,tipo,ferramenta,material,numero,observaçoes,u_velocidade,u_avanco,u_profundidade,u_sai
da))
    trans.persist()
    trans.disconnect()


def view_força():
    trans = TransactionObject()
    trans.connect()
    trans.execute("SELECT * FROM dados_estudo WHERE grandeza=?",('Força',))
    rows = trans.fetchall()
    trans.disconnect()
    return rows

def view_rugosidade():
    trans = TransactionObject()
    trans.connect()
    trans.execute("SELECT * FROM dados_estudo WHERE grandeza=?",('Rugosidade',))
    rows = trans.fetchall()
    trans.disconnect()
    return rows

def delete(name,tipo):
    trans = TransactionObject()
    trans.connect()
    trans.execute("DELETE FROM dados_estudo WHERE referencia = ? AND tipo=?", (name,tipo))
    trans.persist()
    trans.disconnect()

def update(referencia_new,tipo,ferramenta,material,numero,observaçoes,referencia_old,tipo_old):
    trans = TransactionObject()
    trans.connect()
    trans.execute("""
                UPDATE dados_estudo SET
                referencia =?,
                tipo=?,
                ferramenta=?,
                material=?,
                numero=?,
```

```
            observaçoes=?
            WHERE referencia = ? AND
tipo=?""",(referencia_new,tipo,ferramenta,material,numero,observaçoes,referencia_old,tipo_old))
    trans.persist()
    trans.disconnect()

def view_one(name,tipo):
    trans = TransactionObject()
    trans.connect()
    trans.execute("SELECT * FROM dados_estudo WHERE referencia=? AND tipo=?", (name,tipo))
    rows = trans.fetchall()
    trans.disconnect()
    return rows
```

Apêndice C.6 – Interação com usuário

```
from Interfacereg import *
from Modeloclasseranreg import *
from Bancodedadosreg import *
import os
import matplotlib.lines as mlines
import threading
from tkinter import filedialog
from pandastable import Table,config
import pandas as pd
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import shutil
from PIL import ImageTk,Image
import subprocess

initDB()

def atualizar_listbox():
    p_inicial.tree.delete(*p_inicial.tree.get_children())
    id2 = p_inicial.tree.insert("", 1,  text="Força")
    id3 = p_inicial.tree.insert("", 1,  text="Rugosidade")
    for r in view_força():
        p_inicial.tree.insert(id2, "end",  text=r[0], values=(r[2],r[4],r[3]))
    for r in view_rugosidade():
        p_inicial.tree.insert(id3, "end",iid=(r[0],r[2]),  text=r[0], values=(r[2],r[4],r[3]))
```

```
def selecionar():
    selected=p_inicial.tree.item(p_inicial.tree.focus())
    referencia=selected['text']
    tipo=selected['values'][0]

    return(referencia,tipo)


def dados_modelo():

    model=modelo(selecionar()[0],selecionar()[1])
    model.variaveis_exibir()
    pm=p_modelos

    pm.t_grandeza.delete("1.0","end-1c")
    pm.t_tipo.delete("1.0","end-1c")
    pm.t_referencia.delete("1.0","end-1c")
    pm.t_material.delete("1.0","end-1c")
    pm.t_ferramenta.delete("1.0","end-1c")
    pm.t_numero.delete("1.0","end-1c")
    pm.t_observacoes.delete("1.0","end-1c")

    pm.t_grandeza.insert('end',model.grandeza)
    pm.t_tipo.insert('end',model.tipo)
    pm.t_referencia.insert('end',model.referencia)
    pm.t_material.insert('end',model.material)
    pm.t_ferramenta.insert('end',model.ferramenta)
    pm.t_numero.insert('end',model.numero)
    pm.t_observacoes.insert('end',model.observacoes)

    imageg1=[]
    graficog1=[]
    pm.graficog1.destroy()
    imageg1=Image.open('Arquivos\Gráficos/'+model.name+'_teste.png')
    imageg1 = imageg1.resize((320, 270), Image.ANTIALIAS)
    graficog1 = ImageTk.PhotoImage(imageg1)

    pm.graficog1 = tk.Label(pm.f_graficos, image=graficog1)
    pm.graficog1.image=graficog1
```

```
pm.graficog1.grid(padx='5', pady='5')

imageg2=[]
graficog2=[]
pm.graficog2.destroy()
imageg2=Image.open('Arquivos\Gráficos/'+model.name+'_treino.png')
imageg2 = imageg2.resize((320, 270), Image.ANTIALIAS)
graficog2 = ImageTk.PhotoImage(imageg2)

pm.graficog2 = tk.Label(pm.f_graficos, image=graficog2)
pm.graficog2.image=graficog2
pm.graficog2.grid(padx='5', pady='5')

options = config.load_options()
options = {'font': 'Times New Roman',
        'fontsize': 12, 'cellwidth': 100}

pt = Table(pm.f_tab1, dataframe=model.teste,width=400, height=500)
config.apply_options(options, pt)
pt.show()

pt2=Table(pm.f_tab2, dataframe=model.treino,width=400, height=500)
config.apply_options(options, pt2)
pt2.show()


for reg in ['RL','RP2','RP3','RP4','RN']:

   pm.e_velocidade[reg].delete(0,"end")
   pm.e_avanco[reg].delete(0,"end")
   pm.e_profundidade[reg].delete(0,"end")

   pm.t_saida[reg].delete("1.0","end-1c")

   pm.t_erm_test[reg].delete("1.0","end-1c")
   pm.t_cd_test[reg].delete("1.0","end-1c")
   pm.t_cc_test[reg].delete("1.0","end-1c")
   pm.t_mse_test[reg].delete("1.0","end-1c")
   pm.t_rmse_test[reg].delete("1.0","end-1c")
```

```
pm.t_erm_test[reg].insert('end',model.errorel_test[reg])
pm.t_cd_test[reg].insert('end',model.det_test[reg])
pm.t_cc_test[reg].insert('end',model.cor_test[reg])
pm.t_mse_test[reg].insert('end',model.mse_test[reg])
pm.t_rmse_test[reg].insert('end',model.rmse_test[reg])

pm.t_erm_train[reg].delete("1.0","end-1c")
pm.t_cd_train[reg].delete("1.0","end-1c")
pm.t_cc_train[reg].delete("1.0","end-1c")
pm.t_mse_train[reg].delete("1.0","end-1c")
pm.t_rmse_train[reg].delete("1.0","end-1c")

pm.t_erm_train[reg].insert('end',model.errorel_train[reg])
pm.t_cd_train[reg].insert('end',model.det_train[reg])
pm.t_cc_train[reg].insert('end',model.cor_train[reg])
pm.t_mse_train[reg].insert('end',model.mse_train[reg])
pm.t_rmse_train[reg].insert('end',model.rmse_train[reg])

pm.l_u_velocidade[reg].configure(text=model.u_velocidade)
pm.l_u_avanco[reg].configure(text=model.u_avanco)
pm.l_u_profundidade[reg].configure(text=model.u_profundidade)
pm.l_saida[reg].configure(text=model.grandeza+':')
pm.l_u_saida[reg].configure(text=model.u_saida)


image=Image.open( 'Arquivos/'+reg+'/Gráficos/'+model.name+'_teste.png')
image = image.resize((250, 250), Image.ANTIALIAS)
grafico1 = ImageTk.PhotoImage(image)

pm.grafico1[reg] = tk.Label(pm.f_erro[reg], image=grafico1)
pm.grafico1[reg].image=grafico1
pm.grafico1[reg].grid(padx='10', pady='10',row='1',column='0',rowspan='7')

image2=Image.open( 'Arquivos/'+reg+'/Gráficos/'+model.name+'_treino.png')
image2 = image2.resize((250, 250), Image.ANTIALIAS)
grafico2 = ImageTk.PhotoImage(image2)

pm.grafico2[reg] = tk.Label(pm.f_erro[reg], image=grafico2)
pm.grafico2[reg].image=grafico2
pm.grafico2[reg].grid(padx='10', pady='10',row='8',column='0',rowspan='7')
```

```
def previsao(reg):
    pm=p_modelos
    pm.t_saida[reg].delete("1.0","end-1c")

    model=modelo(selecionar()[0],selecionar()[1])
    model.carregar()

    f=float(pm.e_avanco[reg].get())
    a=float(pm.e_profundidade[reg].get())
    v=float(pm.e_velocidade[reg].get())
    força=round(model.pred(v,f,a)[reg],4)

    pm.t_saida[reg].insert('end',força)

    pm.e_avanco[reg].delete(0,"end")
    pm.e_profundidade[reg].delete(0,"end")
    pm.e_velocidade[reg].delete(0,"end")

def criar():
    pn=p_novo


    referencia=pn.t_referencia.get("1.0","end-1c")
    grandeza=pn.grandeza_r.get()
    tipo=pn.t_tipo.get("1.0","end-1c")
    material=pn.t_material.get("1.0","end-1c")
    ferramenta=pn.t_ferramenta.get("1.0","end-1c")
    numero=pn.t_numero.get("1.0","end-1c")
    observacoes=pn.t_observacoes.get("1.0","end-1c")
    u_velocidade=pn.t_u_velocidade.get("1.0","end-1c")
    u_avanco=pn.t_u_avanco.get("1.0","end-1c")
    u_profundidade=pn.t_u_profundidade.get("1.0","end-1c")
    u_saida=pn.t_u_saida.get("1.0","end-1c")

    insert(referencia,grandeza,tipo,ferramenta,material,numero,observacoes,u_velocidade,
        u_avanco,u_profundidade,u_saida)
    data = {grandeza: [], 'n': [],'f': [],'a': []}
    df=pd.DataFrame(data)
    df.to_csv('Arquivos\Dados/'+referencia+'_'+tipo+'_dados.csv',
```

```
                index=False,sep=";")
    os.startfile('Arquivos\Dados/'+referencia+'_'+tipo+'_dados.csv')

    atualizar_listbox()

def exibir_editar():
    model=modelo(selecionar()[0],selecionar()[1])
    pe=p_editar

    pe.t_grandeza.delete("1.0","end-1c")
    pe.t_tipo.delete("1.0","end-1c")
    pe.t_referencia.delete("1.0","end-1c")
    pe.t_material.delete("1.0","end-1c")
    pe.t_ferramenta.delete("1.0","end-1c")
    pe.t_numero.delete("1.0","end-1c")
    pe.t_observacoes.delete("1.0","end-1c")
    pe.t_u_velocidade.delete("1.0","end-1c")
    pe.t_u_avanco.delete("1.0","end-1c")
    pe.t_u_profundidade.delete("1.0","end-1c")
    pe.t_u_saida.delete("1.0","end-1c")

    pe.t_grandeza.insert('end',model.grandeza)
    pe.t_tipo.insert('end',model.tipo)
    pe.t_referencia.insert('end',model.referencia)
    pe.t_material.insert('end',model.material)
    pe.t_ferramenta.insert('end',model.ferramenta)
    pe.t_numero.insert('end',model.numero)
    pe.t_observacoes.insert('end',model.observacoes)
    pe.t_u_velocidade.insert('end',model.u_velocidade)
    pe.t_u_avanco.insert('end',model.u_avanco)
    pe.t_u_profundidade.insert('end',model.u_profundidade)
    pe.t_u_saida.insert('end',model.u_saida)

    pe.l_u_saida.configure(text=model.grandeza+':')

def editar():
    referencia=pe.t_referencia.get("1.0","end-1c")
    grandeza=pe.t_grandeza.get("1.0","end-1c")
    tipo=pe.t_tipo.get("1.0","end-1c")
    material=pe.t_material.get("1.0","end-1c")
```

```
    ferramenta=pe.t_ferramenta.get("1.0","end-1c")
    numero=pe.t_numero.get("1.0","end-1c")
    observacoes=pe.t_observacoes.get("1.0","end-1c")
    u_velocidade=pe.t_u_velocidade.get("1.0","end-1c")
    u_avanco=pe.t_u_avanco.get("1.0","end-1c")
    u_profundidade=pe.t_u_profundidade.get("1.0","end-1c")
    u_saida=pe.t_u_saida.get("1.0","end-1c")

    os.rename(r'Arquivos\Dados/'+selecionar()[0]+'_'+selecionar()[1]+'_dados.csv',
            r'Arquivos\Dados/'+referencia+'_'+tipo+'_dados.csv')

    os.rename(r'Arquivos\Artigos/'+selecionar()[0]+'_'+selecionar()[1]+'_artigo.pdf',
            r'Arquivos\Artigos/'+referencia+'_'+tipo+'_artigo.pdf')

    update(referencia,tipo,ferramenta,material,numero,observacoes,selecionar()[0],selecionar()[1])

    atualizar_listbox()

    os.startfile('Arquivos\Dados/'+referencia+'_'+tipo+'_dados.csv')

def popup(modo):
    global app2, pi,pc,pp
    app2=SampleApp2()
    pi=app2.frames[popup_inserir.__name__]
    pc=app2.frames[popup_criando.__name__]
    pp=app2.frames[popup_pronto.__name__]

    if modo=='edit':
        pi.b_ok.configure(command=lambda: [funçao(1),])
    elif modo=='new':
        pi.b_ok.configure(command=lambda: [funçao(2),])
    else:
        print('Erro ')

    app2.mainloop()

def funçao(modo):
    pi.controller.show_frame('popup_criando')
    pc.bar.start()
```

```
    if modo==1:
        referencia=pe.t_referencia.get("1.0","end-1c")
        tipo=pe.t_tipo.get("1.0","end-1c")
        pe.t_grandeza.delete("1.0","end-1c")
        pe.t_tipo.delete("1.0","end-1c")
        pe.t_referencia.delete("1.0","end-1c")
        pe.t_material.delete("1.0","end-1c")
        pe.t_ferramenta.delete("1.0","end-1c")
        pe.t_numero.delete("1.0","end-1c")
        pe.t_observacoes.delete("1.0","end-1c")
        pe.t_u_velocidade.delete("1.0","end-1c")
        pe.t_u_avanco.delete("1.0","end-1c")
        pe.t_u_profundidade.delete("1.0","end-1c")
        pe.t_u_saida.delete("1.0","end-1c")
    elif modo==2:
        referencia=pn.t_referencia.get("1.0","end-1c")
        tipo=pn.t_tipo.get("1.0","end-1c")

        pn.t_tipo.delete("1.0","end-1c")
        pn.t_referencia.delete("1.0","end-1c")
        pn.t_material.delete("1.0","end-1c")
        pn.t_ferramenta.delete("1.0","end-1c")
        pn.t_numero.delete("1.0","end-1c")
        pn.t_observacoes.delete("1.0","end-1c")
        pn.t_u_velocidade.delete("1.0","end-1c")
        pn.t_u_avanco.delete("1.0","end-1c")
        pn.t_u_profundidade.delete("1.0","end-1c")
        pn.t_u_saida.delete("1.0","end-1c")
    else:
        print('erro')

    t=threading.Thread(target=criando_modelos,args=(referencia,tipo)).start()

def criando_modelos(referencia,tipo):
    model=modelo(referencia,tipo)
    cv=int(pi.e_validacao.get())
    ite=int(pi.e_iteracoes.get())
    pi.e_validacao.delete(0,"end")
    pi.e_iteracoes.delete(0,"end")
    model.criar_modelos(cv,ite)
```

```
                pc.bar.stop()
                pc.controller.show_frame('popup_pronto')
                p_novo.b_dados_experimentais.configure(state='disabled'),
                p_novo.controller.show_frame("p_inicial")
                pp.b_fechar.configure(command=lambda: app2.destroy())


            def anexar_artigo():
                referencia=pn.t_referencia.get("1.0","end-1c")
                tipo=pn.t_tipo.get("1.0","end-1c")
                source =filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (("pdf files","*.pdf"),("all files","*.*")))

                destination ='Arquivos/Artigos/'+referencia+'_'+tipo+'_artigo.pdf'
                dest = shutil.copy(source, destination)
                p_novo.b_dados_experimentais.configure(state='normal')

            def deletar():
                delete(selecionar()[0],selecionar()[1])
                v=[]
                v.append(lambda:
os.remove('Arquivos\Dados/'+selecionar()[0]+'_'+selecionar()[1]+'_dados.csv'))
                v.append(lambda:
os.remove('Arquivos\Artigos/'+selecionar()[0]+'_'+selecionar()[1]+'_artigo.pdf'))
                v.append(lambda:
os.remove('Arquivos\Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_teste.png'))
                v.append(lambda:
os.remove('Arquivos\Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_treino.png'))
                v.append(lambda:
os.remove('Arquivos\Relatórios/'+selecionar()[0]+'_'+selecionar()[1]+'_relatorio.docx'))


                v.append(lambda:
os.remove('Arquivos\RN/Erros/'+selecionar()[0]+'_'+selecionar()[1]+'_erros.json'))
                v.append(lambda:
os.remove('Arquivos\RN/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_teste.png'))
                v.append(lambda:
os.remove('Arquivos\RN/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_treino.png'))
```

```
                v.append(lambda:
os.remove('Arquivos\RN/Modelos/'+selecionar()[0]+'_'+selecionar()[1]+'_modelo.mdl'))

                v.append(lambda:
os.remove('Arquivos\RL/Erros/'+selecionar()[0]+'_'+selecionar()[1]+'_erros.json'))
                v.append(lambda:
os.remove('Arquivos\RL/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_teste.png'))
                v.append(lambda:
os.remove('Arquivos\RL/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_treino.png'))
                v.append(lambda:
os.remove('Arquivos\RL/Modelos/'+selecionar()[0]+'_'+selecionar()[1]+'_modelo.mdl'))

                v.append(lambda:
os.remove('Arquivos\RP2/Erros/'+selecionar()[0]+'_'+selecionar()[1]+'_erros.json'))
                v.append(lambda:
os.remove('Arquivos\RP2/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_teste.png'))
                v.append(lambda:
os.remove('Arquivos\RP2/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_treino.png'))
                v.append(lambda:
os.remove('Arquivos\RP2/Modelos/'+selecionar()[0]+'_'+selecionar()[1]+'_modelo.mdl'))

                v.append(lambda:
os.remove('Arquivos\RP3/Erros/'+selecionar()[0]+'_'+selecionar()[1]+'_erros.json'))
                v.append(lambda:
os.remove('Arquivos\RP3/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_teste.png'))
                v.append(lambda:
os.remove('Arquivos\RP3/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_treino.png'))
                v.append(lambda:
os.remove('Arquivos\RP3/Modelos/'+selecionar()[0]+'_'+selecionar()[1]+'_modelo.mdl'))

                v.append(lambda:
os.remove('Arquivos\RP4/Erros/'+selecionar()[0]+'_'+selecionar()[1]+'_erros.json'))
                v.append(lambda:
os.remove('Arquivos\RP4/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_teste.png'))
                v.append(lambda:
os.remove('Arquivos\RP4/Gráficos/'+selecionar()[0]+'_'+selecionar()[1]+'_treino.png'))
                v.append(lambda:
os.remove('Arquivos\RP4/Modelos/'+selecionar()[0]+'_'+selecionar()[1]+'_modelo.mdl'))
```

```
                v.append(lambda:
os.remove('Arquivos\RN/Pesos/'+selecionar()[0]+'_'+selecionar()[1]+'_pesos.txt'))
                v.append(lambda:
os.remove('Arquivos\RN/Parâmetros/'+selecionar()[0]+'_'+selecionar()[1]+'_parametros.json'))

                v.append(lambda:
os.remove('Arquivos\RL/Coeficientes/'+selecionar()[0]+'_'+selecionar()[1]+'_coeficientes.txt'))

                v.append(lambda:
os.remove('Arquivos\RP2/Coeficientes/'+selecionar()[0]+'_'+selecionar()[1]+'_coeficientes.txt'))

                v.append(lambda:
os.remove('Arquivos\RP3/Coeficientes/'+selecionar()[0]+'_'+selecionar()[1]+'_coeficientes.txt'))

                v.append(lambda:
os.remove('Arquivos\RP4/Coeficientes/'+selecionar()[0]+'_'+selecionar()[1]+'_coeficientes.txt'))

                for func in v:
                    try:
                        func()
                    except:
                        print("Arquivo nao encontrado")

                atualizar_listbox()


            def abrir_artigo():
                os.startfile('Arquivos\Artigos/'+selecionar()[0]+'_'+selecionar()[1]+'_artigo.pdf')


            def abrir_relatorio():
                os.startfile(r'Arquivos\Relatórios/'+selecionar()[0]+'_'+selecionar()[1]+'_relatorio.docx')

            def abrir_arquivos():
                os.startfile(r'Arquivos')

            def abrir_iteracoes():
                os.startfile(r'Arquivos\RN/Iterações/'+selecionar()[0]+'_'+selecionar()[1]+'_iteraçoes.txt')

            def abrir_pesos():
```

```
            os.startfile(r'Arquivos\RN/Pesos/'+selecionar()[0]+'_'+selecionar()[1]+'_pesos.txt')

        def abrir_parametros():

subprocess.call(['notepad.exe',r'Arquivos\RN/Parâmetros/'+selecionar()[0]+'_'+selecionar()[1]+'_parametros.json
'])

        def abrir_coeficientes(reg):

os.startfile(r"Arquivos\\"+reg+'/Coeficientes/'+selecionar()[0]+'_'+selecionar()[1]+'_coeficientes.txt')

        if __name__ == "__main__":

            initDB()
            app = SampleApp()

            menubar = tk.Menu(app)

            filemenu = tk.Menu(menubar, tearoff=0)
            filemenu.add_command(label="Open")
            filemenu.add_command(label="Save")
            filemenu.add_command(label="Exit")

            menubar.add_cascade(label="File", menu=filemenu)

            app.config(menu=menubar)

            app.title("Rede neural no torneamento")
            app.geometry('1000x650+50+0')
            p_inicial=app.frames[p_inicial.__name__]
            p_novo=app.frames[p_novo.__name__]
            pn=p_novo
            p_editar=app.frames[p_editar.__name__]
            pe=p_editar
            p_modelos=app.frames[p_modelos.__name__]
            pm=p_modelos

            atualizar_listbox()
```

```
                p_inicial.button_abrir.configure(command=lambda:
[p_inicial.controller.show_frame("p_modelos"),
                                                 dados_modelo()])
                p_inicial.button_novo.configure(command=lambda:
p_inicial.controller.show_frame("p_novo"))

                p_inicial.button_editar.configure(command=lambda:
[p_inicial.controller.show_frame("p_editar"),
                                                  exibir_editar()])
                p_inicial.button_deletar.configure(command=lambda:deletar())

                p_novo.b_anexar_artigo.configure(command=lambda: anexar_artigo())
                p_novo.b_dados_experimentais.configure(command=lambda: [criar(),popup('new')])
                p_novo.b_voltar.configure(command=lambda:p_novo.controller.show_frame("p_inicial"))

                p_editar.b_atualizar.configure(command=lambda:[editar(),
                                              popup('edit')])
                p_editar.b_voltar.configure(command=lambda:p_novo.controller.show_frame("p_inicial"))

                p_modelos.b_previsao['RN'].configure(command=lambda: previsao('RN'))
                p_modelos.b_iteracoes.configure(command=lambda: abrir_iteracoes())
                p_modelos.b_pesos.configure(command=lambda: abrir_pesos())
                p_modelos.b_parametros.configure(command=lambda: abrir_parametros())
                p_modelos.b_previsao['RL'].configure(command=lambda: previsao('RL'))
                p_modelos.b_coeficientes['RL'].configure(command=lambda: abrir_coeficientes('RL'))
                p_modelos.b_previsao['RP2'].configure(command=lambda: previsao('RP2'))
                p_modelos.b_coeficientes['RP2'].configure(command=lambda: abrir_coeficientes('RP2'))
                p_modelos.b_previsao['RP3'].configure(command=lambda: previsao('RP3'))
                p_modelos.b_coeficientes['RP3'].configure(command=lambda: abrir_coeficientes('RP3'))
                p_modelos.b_previsao['RP4'].configure(command=lambda: previsao('RP4'))
                p_modelos.b_coeficientes['RP4'].configure(command=lambda: abrir_coeficientes('RP4'))
                p_modelos.b_artigo.configure(command=lambda:abrir_artigo())
                p_modelos.b_relatorio.configure(command=lambda:abrir_relatorio())
                p_modelos.b_arquivos.configure(command=lambda:abrir_arquivos())

p_modelos.b_voltar.configure(command=lambda:p_modelos.controller.show_frame("p_inicial"))
                app.mainloop()
```