UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Faculdade de Engenharia e Ciências de Guaratinguetá

JOÃO OTÁVIO BELIZÁRIO TONHÃO

Sistema Em Python Para Monitoramento De Processos De Laminação Utilizando Machine Learning

Guaratinguetá
2023

João Otávio Belizário Tonhão

Sistema Em Python Para Monitoramento De Processos De Laminação Utilizando Machine Learning

Dissertação apresentada à Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, para obtenção do título de Mestre em Engenharia, área de Produção.

Orientadora: Profa. Dra. Paloma Maria Silva Rocha Rizol

Guaratinguetá
2023

Tonhão, João Otávio Belizário
T665s Sistema em Python para monitoramento de processos de laminação utilizando machine learning / João Otávio Belizário Tonhão - Guaratinguetá, 2023.
95 f : il.
Bibliografia: f. 92-95

Dissertação (Mestrado) – Universidade Estadual Paulista, Faculdade de Engenharia e Ciências de Guaratinguetá, 2023.

Orientadora: Profª. Drª. Paloma Maria Silva R. Rizol

1. Python (Linguagem de programação de computador).
2. Processo decisório. 3. Controle de processo. I. Título.

CDU 658.511.3(043)

Luciana Máximo
Bibliotecária/CRB-8 3595

IMPACTO POTENCIAL DESTA PESQUISA

Esse trabalho possui um impacto estratégico para a sustentabilidade do setor industrial de laminados no Brasil diante da concorrência externa, uma vez que a otimização de processos e aumentando a eficiência, além de garantir o volume de produção, impacta diretamente em redução de custos de produção, capacidade de produção, emissão de poluentes, consumo de energia, consumo de matéria prima, entre outros.

POTENTIAL IMPACT OF THIS RESEARCH

This work has a strategic impact for the sustainability of the rolling products industrial sector in Brazil in the face of external competition, since optimizing processes and increase material recovery, besides to guarantee the production capacity, has a direct impact on reducing production costs, emissions pollutants, energy consumption, raw material consumption, among others.

unesp

UNIVERSIDADE ESTADUAL PAULISTA
CAMPUS DE GUARATINGUETÁ

JOÃO OTÁVIO BELIZÁRIO TONHÃO

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO TÍTULO DE "MESTRE EM ENGENHARIA"

PROGRAMA: ENGENHARIA
CURSO: MESTRADO

APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO

gov.br Documento assinado digitalmente
MANOEL CLEBER DE SAMPAIO ALVES
Data: 02/01/2024 09:30:16-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Manoel Cleber de Sampaio Alves
Coordenador

BANCA EXAMINADORA:

gov.br Documento assinado digitalmente
PALOMA MARIA SILVA ROCHA RIZOL
Data: 04/12/2023 11:17:48-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. PALOMA MARIA SILVA ROCHA RIZOL
Orientador - UNESP

gov.br Documento assinado digitalmente
MARCELA APARECIDA GUERREIRO MACHADO DI
Data: 11/12/2023 07:53:49-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. MARCELA APARECIDA GUERREIRO MACHADO DE FREITAS
UNESP

gov.br Documento assinado digitalmente
FABIO RODOLFO MIGUEL BATISTA
Data: 08/12/2023 13:38:02-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. FABIO RODOLFO MIGUEL BATISTA
USP

NOVEMBRO de 2023

DADOS CURRICULARES

JOÃO OTÁVIO BELIZÁRIO TONHÃO

NASCIMENTO 03.02.1991 – Andradas / MG

FILIAÇÃO João Batista Tonhão
Maria Aparecida Belizário Tonhão

2010/2014 Curso de Graduação
Engenharia de Controle e Automação – Universidade Federal de Ouro Preto - UFOP

Dedico este trabalho de modo especial, à minha família e esposa que sempre me apoiou e me incentivou para que essa conquista fosse alcançada.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me dar condições para buscar a realização dos meus objetivos;

a minha esposa Paula pelo companheirismo e porto seguro;

aos meus Pais pelo suporte, exemplo e orientação;

e a professora Paloma Rizol pelo apoio e orientação ao longo deste trabalho.

RESUMO

Com o pleno avanço da Industria 4.0 nos mais variados setores industriais, desponta como uma das principais ferramentas a utilização de técnicas de *Machine Learning*. Á medida que os processos ficam mais rápidos e assertivos, aumenta a necessidade de maiores aquisições de dados para monitoramento, classificação e aperfeiçoamento do produto. Este alto volume de dados torna inviável a aplicação de análises manuais por seres humanos e destaca-se assim algoritmos com técnicas capazes de trabalhar com diferentes bases de dados simultaneamente. As técnicas de *Machine Learning*, como Árvore de Decisão, são altamente recomendáveis para sistemas de classificação de produto. É possível, por meio de critérios pré-estabelecidos ou por regressão, analisar conjuntos de variáveis pré-definidas de acordo com padrões desejáveis e auxiliar na tomada de decisão por meio das suas saídas. Neste contexto, a proposta deste trabalho é desenvolver um algoritmo em Python utilizando a técnica de árvore de decisão para o controle de qualidade no processo de laminação, comparando sua eficiência frente aos sistemas utilizados atualmente para este mercado. A utilização da linguagem Python permitiu a manipulação de grande quantidade de dados por meio de ferramentas estatísticas, geração de gráficos, simulação e supervisório. Com isso, foi criado um sistema de monitoramento de processos de laminação baseado na técnica de *Machine Learning* Árvore de Decisão Classificatória, possibilitando o cálculo das regiões a serem descartadas no processo e o correto direcionamento do produto de acordo com as especificações dos clientes, optando por seguir o processo original, redirecionar para outro cliente ou sucatear o produto caso não haja a possibilidade de aproveitamento. O modelo apresentou uma acurácia de 83,33% e profundidade de 9 com ccp_alpha de 0,012. Resultando em um modelo otimizado com precisão de 83,58%, recall de 83,33% e f-score de 82,07%, onde concluiu-se ser possível construir um sistema eficiente para aplicação na indústria usando software de código aberto com baixo custo de implementação e manutenção, incluindo uma alta flexibilidade para fazer adaptações para cada demanda de processo.

PALAVRAS-CHAVE: Aprendizado de Máquina; Tomada de decisão; Python; Laminação; Controle de Processos.

ABSTRACT

With the full advancement of Industry 4.0 in the most varied industrial sectors, the use of Machine Learning techniques emerges as one of the main tools. As processes become faster and more assertive, the need for greater data acquisition for monitoring, classification and product improvement increases. This high volume of data makes the application of manual analyzes by humans unfeasible and thus highlights algorithms with techniques capable of working with different databases simultaneously. Machine Learning techniques, such as Decision Tree, are highly recommended for product classification systems. It is possible, through pre-established criteria or regression, to analyze sets of pre-defined variables according to desirable standards and assist in decision-making through their outputs. In this context, the purpose of this work is to develop an algorithm in Python using the decision tree technique for quality control in the lamination process, comparing its efficiency compared to the systems currently used for this market. The use of the Python language allowed the manipulation of large amounts of data through statistical tools, graph generation, simulation and supervision. With this, a lamination process monitoring system was created based on the Machine Learning Classification Decision Tree technique, enabling the calculation of the regions to be discarded in the process and the correct direction of the product according to customer specifications, opting for follow the original process, redirect to another customer or scrap the product if it cannot be used. The system presented an accuracy of 83.3%, where it was concluded that it was possible to build an efficient system for application in industry using open-source software with low implementation and maintenance costs, including high flexibility to make adaptations to each process demand.

KEYWORDS: Machine learning; Python; decision tree; rolling; quality control.

LISTA DE FIGURAS

Figura 1 - Classificação da pesquisa científica 20
Figura 2 - Número de publicações ao longo dos últimos anos com as palavras-chave *quality control* e *machine learning* 23
Figura 3 - Publicações por Área de Atuação 23
Figura 4 - Países Coautores 24
Figura 5 - Coocorrência de palavras-chave 25
Figura 6 - Processo de vazamento de placas 32
Figura 7 - Processo de lingotamento 33
Figura 8 - Diagrama de laminador "4-High" 33
Figura 9 - Fluxo de dados 35
Figura 10 - Fluxograma do software 36
Figura 11 - Fluxograma da pesquisa 37
Figura 12 - Demonstração IbaAnalyzer 40
Figura 13 - Demonstração IbadatCoordinator 40
Figura 14 - Demonstração de aplicação do PostgreSQL 41
Figura 15 - Interface Software Pycharm 42
Figura 16 - Exemplo de análise de descarte por espessura 43
Figura 17 - Exemplo de análise de descarte por temperatura 43
Figura 18 - início da gravação do arquivo 44
Figura 19 - Representação do evento StopTrigger 45
Figura 20 - tarefa de conversão - ibadatCoordinator 46
Figura 21 - Geração da fila de processamento 48
Figura 22 - Nomenclatura dos arquivos de produção 49
Figura 23 - Dataframe Principal 51
Figura 24 - Representação do descarte inicial por espessura 52
Figura 25 - Selecionando a região de descarte inicial 53
Figura 26 - Seleção dos valores de espessura fora dos limites 54
Figura 27 - Representação gráfica dos valores para descarte inicial de temperatura 56
Figura 28 - Filtrando a região de descarte inicial 57
Figura 29 - Seleção dos valores de temperatura fora dos limites 58
Figura 30 - Representação do descarte final por espessura 59
Figura 31 - Gráfico com a região de descarte final por espessura 60

LISTA DE FIGURAS

Figura 32 - Seleção dos valores de espessura fora dos limites 61
Figura 33 - Filtrando a região de descarte final por temperatura 63
Figura 34 - Seleção dos valores de temperatura final fora dos limites 64
Figura 35 - Trecho da tabela clientes 67
Figura 36 - Trecho da tabela resultado 69
Figura 37 - Dataframe de features do modelo originado da tabela "resultado" 74
Figura 38 - Estrutura da função train_test_split 75
Figura 39 - Visão gráfica da árvore de decisão inicial 79
Figura 40 - Visão gráfica da árvore após Pruning 84
Figura 41 - Aspectos de profundidade e nós da árvore de acordó com os valores de ccp_alpha 86
Figura 42 - Alfa vs acurácia 87
Figura 43 - Alfa vs acurácia (zoom) 87
Figura 44 - Ranking de importância das features 89
Figura 45 - Tela principal do supervisório 90

LISTA DE QUADROS

Quadro 1 - Código para listagem dos arquivos do diretório padrão 47
Quadro 2 - Código para identificação e classificação dos arquivos de produção 50
Quadro 3 - Armazena o nome dos arquivos e faz a leitura dos dados 51
Quadro 4 - Código para a selecão da região de descarte inicial 53
Quadro 5 - Código para identificação do comprimento do descarte inicial por espessura .. 55
Quadro 6 - Definição dos limites e criação de *dataframe* para cálculo do descarte inicial por temperatura 55
Quadro 7 – Código para a seleção região para descarte inicial por temperatura 56
Quadro 8 - Código para seleção dos valores de erro de temperatura fora dos limites especificados 57
Quadro 9 - Código para identificação do comprimento do descarte inicial por temperatura 58
Quadro 10 - Código para determinar a região de descarte final por espessura 59
Quadro 11 - Código para seleção dos valores fora dos limites especificados para a região de descarte final por espessura 61
Quadro 12 - Código para determinar o comprimento do descarte final por espessura 62
Quadro 13 - Código para determinar a região de descarte final por temperatura 62
Quadro 14 - Código para seleção dos valores fora dos limites especificados para a região de descarte final por temperatura 63
Quadro 15 - Código para determinar o comprimento do descarte final por temperatura 64
Quadro 16 - Código para cálculo do comprimento final eliminando as regiões de descarte 65
Quadro 17 - Script para a criação do banco de dados 66
Quadro 18 - Script para a criação da tabela clientes 66
Quadro 19 - Script para a criação da tabela resultado 68
Quadro 20 - Código para gerenciamento de arquivos processados 70
Quadro 21 - Código para a criação da conexão com o banco de dados 71
Quadro 22 - Código para a inserção de dados na tabela resultado 72
Quadro 23 - Código para criação dos conjuntos de dados de teste e treinamento 76
Quadro 24 - Código para a criação da árvore de decisão inicial 77
Quadro 25 - Código para a verificação de acurácia do modelo inicial 78
Quadro 26 - Código para a plotagem gráfica da árvore utilizando graphviz 79
Quadro 27 - Trecho do código para testes de profundidade com o critério Gini 80

Quadro 28 - Trecho do código para testes de profundidade com o critério Entropia..... 82
Quadro 29 – Códigos para a construção do modelo com os parâmetros otimizados e plotagem gráfica 84

LISTA DE TABELAS

Tabela 1 - Publicações em Scopus utilizando as palavras-chave *Machine Learning, Decision Making, Rolling, Decision Tree* e *Quality Control* 22
Tabela 2 - Pesquisa Aprofundada (Scopus) 26
Tabela 3 - Resultados de acurácia para o critério Gini 81
Tabela 4 - Resultados de acurácia para o critério Entropia 82
Tabela 5 - Comparativo de acurácia para os critérios Gini e Entropia 83

SUMÁRIO

1 INTRODUÇÃO 16

1.1 QUESTÃO DA PESQUISA E OBJETIVOS 18

1.1.1 Questão da pesquisa 18

1.1.2 Objetivo Geral 18

1.1.3 Objetivos Específicos 18

1.2 CLASSIFICAÇÃO DA PESQUISA CIENTÍFICA 19

1.3 DELIMITAÇÃO DA PESQUISA 20

1.4 ESTRUTURA DA TESE 20

2 DESENVOLVIMENTO 22

2.1 REVISÃO DE LITERATURA 22

2.1.1 Justificativa e oportunidades 22

2.2 REFERENCIAL TEÓRICO 26

2.2.1 Redes neurais artificiais 27

2.2.2 Naive Bayes 27

2.2.3 Regressão linear 28

2.2.4 Máquinas De Vetores De Suporte (SVM) 28

2.2.5 K-Vizinhos mais próximos 28

2.2.6 Árvores de decisão 29

2.2.7 Métricas de avaliação 30

2.2.8 O Processo de laminação 31

3 MATERIAIS E MÉTODOS 34

3.1 SISTEMÁTICA DO TRABALHO 34

3.2 FERRAMENTAS UTILIZADAS 39

3.2.1 Softwares iba 39

3.2.2 Arquivos com extensão “parquet” 40

3.2.3 Banco de dados PostgreeSQL 41

3.2.4 Linguagem Python 41

3.2.5 Software Pycharm para programação em Python 42

3.3 TRATATIVA DE DADOS 42

SUMÁRIO

3.3.1 **Parâmetros para monitoramento e controle de qualidade** 42
3.3.2 **Coleta de dados** 44
3.3.3 **Conversão de arquivos .dat para .parquet** 46
3.3.4 **Gerenciamento dos arquivos .parquet** 47
3.3.5 **Identificação do produto** 49
3.3.6 **Cálculo das regiões de descarte** 50
3.3.7 **Cálculo do comprimento final após descartes** 65
3.3.8 **Armazenamento de dados** 65
3.3.9 **Gestão dos arquivos processados** 69
3.3.10 **Ingestão e consulta de dados** 70
3.4 CONSTRUÇÃO DO MODELO 72
3.4.1 **Filtragem de dados** 73
3.4.2 **Separação de dados (treinamento e teste)** 74
3.4.3 **Construindo a árvore de decisão classificatória** 76
4 **RESULTADOS E DISCUSSÕES** 79
4.1 OTIMIZAÇÃO DO MODELO (PRUNING) 79
4.2 MÉTRICAS DE VALIDAÇÃO 88
4.3 SUPERVISÓRIO ITERATIVO PARA O USUÁRIO FINAL 89
5 **CONCLUSÃO** 90
REFERÊNCIAS 92

1 INTRODUÇÃO

O desenvolvimento da indústria 4.0 tem se apresentado substancialmente presente nas indústrias de transformação. Ou seja, os processos dessa nova revolução estão associados as denominadas fábricas inteligentes, nas quais fazem uso da tecnologia para otimizar suas atividades e garantir resultados precisos em comparação com as empresas que não estão inseridas no modelo da indústria 4.0 (STOCK; SELIGER, 2016).

Que tem como um dos pilares a coleta, armazenamento e processamento de grande quantidade de dados simultaneamente no intuito de utilizá-los para a criação de algoritmos e modelos que possam predizer resultados de um determinado processo, encontrar correlações entre variáveis críticas para o processo, criar modelos compatíveis com o funcionamento de equipamentos reais e simular ajustes sem qualquer intervenção real, analisar conjuntos de variáveis como largura, espessura, composição química e comprimento final após remoção das regiões de transiente e classifica-las de acordo com padrões pré-estabelecidos pelos clientes ou para a próxima etapa do processo.

O processo de laminação tem várias particularidades de acordo com o tipo de produto produzido e sua aplicação, sendo necessária a utilização de equipamentos complexos (laminadores) que exigem o emprego dos mais diversos tipos de instrumentos, Controladores Lógico Programáveis (CLPs) e sistemas de aquisição de dados para garantir a qualidade do material ao longo do comprimento. Porém, qualquer desvio no processo pode causar defeitos graves no material, necessitando também de um sistema eficaz para análise dos KPOs (*Key Process Outputs*) de modo a classificar os possíveis defeitos e fazer o direcionamento para a tratativa adequada (descarte da região do defeito, direcionamento para outro tipo de produto (reclassificação) ou até mesmo a sucata total da bobina.

Atualmente, é comum no mercado o envolvimento de especialistas para a análise dos arquivos de produção e tomadas de decisão para o material. Além da necessidade de profissionais com vasta experiência para garantir a análise e tratativa correta, este processo demanda muito tempo da equipe, sobrecarregando a rotina e impedindo-os de executar outras ações estratégicas para o negócio e muitas vezes gerando classificações fora do padrão devido a erros de interpretação ou informações divergentes entre os analistas. Esse fato leva a impactos negativos em todo o fluxo da cadeia, prejudicando a entrega aos clientes e aumentando os custos de produção devido a necessidade de produzir novamente o item que foi sucateado.

Conforme caraterísticas do produto se torna mais complexo, a configuração até um limite apropriado, como um controle de espessura, para identificar variação desconhecida dentro da tolerância torna-se difícil tarefa. Esta é uma das razões pelas quais a indústria ainda usa inspetores humanos para realizar tarefas implícitas de inspeção para detectar defeitos não mapeados previamente como acabamentos ruins, amassados, orientações erradas, e muitas outras falhas mal definidas. Contudo, em relação aos inspetores humanos, muitos sistemas de inspeção automatizados possuem melhor desempenho em termos de precisão, consistência e velocidade de inspeção sem a necessidade de definir um buffer para inspeção, além de reduzir os custos trabalhistas (YACOB; SEMERE; NORDGREN, 2019).

Contando com procedimentos sólidos e especificações detalhadas do material, surge a oportunidade de criação de um sistema capaz de analisar todo o material produzido de forma rápida, consistente e eficaz. Aliviando a carga dos profissionais envolvidos e garantindo análises padronizadas, eliminando possíveis divergências de interpretação entre diferentes profissionais.

Nos últimos anos, surgiram alguns novos métodos de aprendizagem de máquinas, como *Support Vector Machine* (SVM), *Classification* e Árvore de Regressão (CART), Árvore de Regressão de Bagging (BRT), Redução Mínima Absoluta e Seleção Operador (LASSO - *Least Absolute Shrinkage and Selection Operator)* e Regressão Gaussiana do Processo (GPR). Para a pesquisa no campo de laminação, percebeu-se que métodos de aprendizado de máquina podem ser utilizados para auxiliar os funcionários em tomadas de decisão referentes a classificação de produtos após o término do processamento, set up de equipamento de acordo com o material a ser produzido, manutenção preventiva, entre outros (XU, FENG e YAN, 2020).

Como ferramenta para a implementação de códigos utilizando técnicas de *machine learning*, tem-se a linguagem de programação Python, orientada a objetos, que ganhou popularidade devido ao fácil acesso a muitos pacotes de desenvolvimento, como Numpy, OpenCV, TensorFlow e PyQT. Python também inclui módulos bem conhecidos para aprendizado de máquina e computação científica (KIM, 2022). Além disso, com uma comunidade muito ativa, Python se tornou a linguagem de programação preferida para desenvolvimento de projetos nos últimos anos, ocupando o primeiro lugar no ranking interativo anual do IEEE Spectrum (CASS, 2021).

Objetivando a utilização de técnicas e ferramentas utilizadas atualmente na indústria 4.0, este trabalho consiste no desenvolvimento e implementação de uma ferramenta de análise para controle de qualidade de matérias laminados e bobinados, utilizando arquivos

de produtos contendo sinais do material e equipamento convertido em formato de texto (.parquet) para otimizar o processo de leitura, tratativa e processamento dos dados para a criação, treinamento e execução de um modelo de aprendizado de máquinas no software Python utilizando árvore de decisão.

Com este tipo de suporte automatizado para a tomada de decisão, a carga de trabalho e a carga cognitiva de os trabalhadores podem ser diminuídas e eles podem se concentrar na melhoria do processo quando ocorre um alarme de aumento do risco de falha durante a laminação, por exemplo. Os modelos de previsão estatística, tais como utilizando séries temporais, são especialmente adequados para a tomada de decisões; com sua capacidade de prever o resultado futuro, o tempo de reação após um alarme é maior (TAMMINEN et al, 2018).

1.1 QUESTÃO DA PESQUISA E OBJETIVOS

1.1.1 Questão da pesquisa

Quão eficiente seria o monitoramento do processo de laminação utilizando *machine learning*?

1.1.2 Objetivo Geral

Desenvolver um sistema supervisionado de monitoramento de um processo de laminação utilizando árvore de decisão classificatória para otimização da disposição do produto.

1.1.3 Objetivos Específicos

- Identificar dados dos recém-nascidos da cidade de São José dos Campos-SP, nos anos de 2016 a 2018
- Identificar relatórios diários dos poluentes do ar, PM_{10} e NO_2, para o mesmo período
- Desenvolver um modelo baseado na lógica *fuzzy e* neuro-*fuzzy*, relacionando exposição das gestantes aos poluentes do ar com parto prematuro
- Realizar a comparação entre o modelo *fuzzy* e o sistema neuro-*fuzzy ANFIS* utilizados
- Estimar os parâmetros do processo;

- Desenvolver um algoritmo em Python utilizando técnica de *Machine Learning* (para monitorar o processo;
- Criar um sistema para geração automática e padronizada de Relatórios de Não Conformidade (RNC) para diferentes classes de material utilizando *Machine Learning* para tomada de decisão;

1.2 CLASSIFICAÇÃO DA PESQUISA CIENTÍFICA

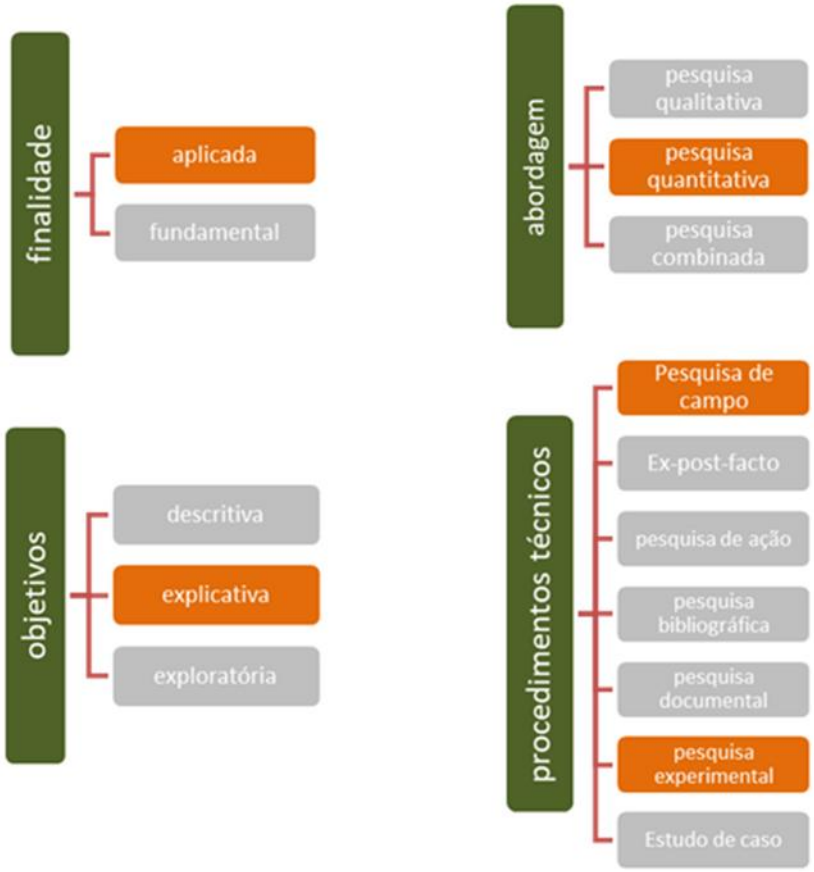
Conforme apresentado na Figura 6, trata-se de uma pesquisa aplicada, com objetivo explicativo, com abordagem quantitativa, de caráter explicativa, utilizando o procedimento experimental e pesquisa de campo.

Quanto à natureza é classificada como pesquisa aplicada, pois pretende gerar resultados para aplicação prática dirigida à solução de problemas específicos, neste caso relacionado ao controle de qualidade de materiais laminados. (LAKATOS; MARCONI, 2010).

Quanto à forma de abordagem, caracteriza-se como quantitativa devido às variáveis de decisão envolvidas no método proposto, com classificação através de registros de análises. (LAKATOS; MARCONI, 2010).

Quanto aos objetivos caracteriza-se como pesquisa explicativa pois visa identificar os fatores que determinam fenômenos e explica o porquê das coisas. Envolve levantamento de dados de processo, especificação de clientes, categoria de material e variáveis críticas de processo que determinam os fenômenos de defeito ja descritos e detalhados. Quanto aos procedimentos técnicos utiliza-se a pesquisa experimental com base no levantamento das variáveis que seriam capazes de influenciar na qualidade do produto, e de campo por conta com a experiencia de profissionais qualificados para confrontar os dados obtidos pelo modelo com as técnicas de análise de qualidade utilizadas atualmente (GIL, 2010).

Figura 1 - Classificação da pesquisa científica



Fonte: Adaptado de Kothari (2013).

1.3 DELIMITAÇÃO DA PESQUISA

A pesquisa está delimitada ao processo de laminação. Para o estudo e desenvolvimento foi utilizado um conjunto de 470 arquivos de produção de bobinas de material laminado no ano de 2022.

Os arquivos contendo os dados coletados do processo foram convertidos para uma extensão especifica (.parquet) onde os sinais foram ser importados, tratados e utilizados como base para a criação de um modelo de arvore de decisão utilizando a ferramenta Python.

1.4 ESTRUTURA DA TESE

Este trabalho está organizado em 5 capítulos, o primeiro contém a introdução sobre o tema, assim como objetivo geral, específicos e oportunidades encontradas na pesquisa bibliométrica.

O segundo capítulo, Fundamentação Teórica, aborda os conceitos de *machine learning*, assim como as técnicas utilizadas, métricas de validação do modelo e o processo utilizado como base para este estudo.

O terceiro capítulo, Ferramentas Utilizadas, aborda detalhes sobre as ferramentas utilizadas para o desenvolvimento do projeto, assim como o passo a passo executados ao longo do desenvolvimento do código e funcionalidades de cada função.

No quarto capítulo, são apresentados os resultados obtidos durante a criação e após otimização do modelo, com detalhamento de métricas de validação do mesmo e comparações com os métodos de classificação convencionais já utilizados neste processo.

Por fim, o último capítulo expõe as conclusões alcançadas e contribuições deste estudo para a sociedade.

2 DESENVOLVIMENTO

O capítulo de Desenvolvimento apresenta a revisão na literatura sobre o tema, assim como o referencial teórico para todos os tópicos utilizados para a construção desse trabalho.

2.1 REVISÃO DE LITERATURA

2.1.1 Justificativa e oportunidades

Um levantamento realizado nas bases de dados Scopus e Web of Science mostra um aumento crescente no número de publicações envolvendo *Machine Learning* (Aprendizado de Máquina), *Decision Making* (Tomada de Decisão) e *Process Control* (Controle de Processos), tendência essa diretamente correlacionada com o avanço tecnológico na área computacional, especialmente em componentes eletrônicos como processadores e memória, possibilitando atingir o alto poder de processamento suficiente para viabilizar a utilização destes métodos.

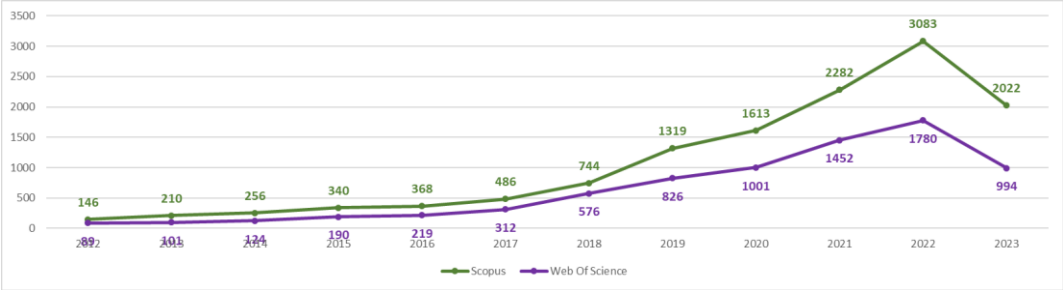
Tabela 1 - Publicações em Scopus utilizando as palavras-chave *Machine Learning*, *Decision Making*, *Rolling*, *Decision Tree* e *Quality Control*

PESQUISA	TEMPO ESTIPULADO	TOTAL DE PUBLICAÇÕES
"Machine Learning" AND *"Decision Making"*	2014 - 2023	19808
"Machine Learning" AND "Decision Tree"	2014 - 2023	38352
"Machine Learning" AND "Rolling"	2014 - 2023	1330
"Machine Learning" AND "Quality Control"	2014 - 2023	9610
"Machine Learning" AND "Decision Making" AND "Quality Control"	2012 - 2023	618

Fonte: SCOPUS (2023).

Na Figura 2 é apresentada a tendência do número de publicações relacionadas a *Machine Learning* e *Quality Control* ao longo dos últimos anos. Houxve uma queda em 2023 porque o ano ainda não terminou

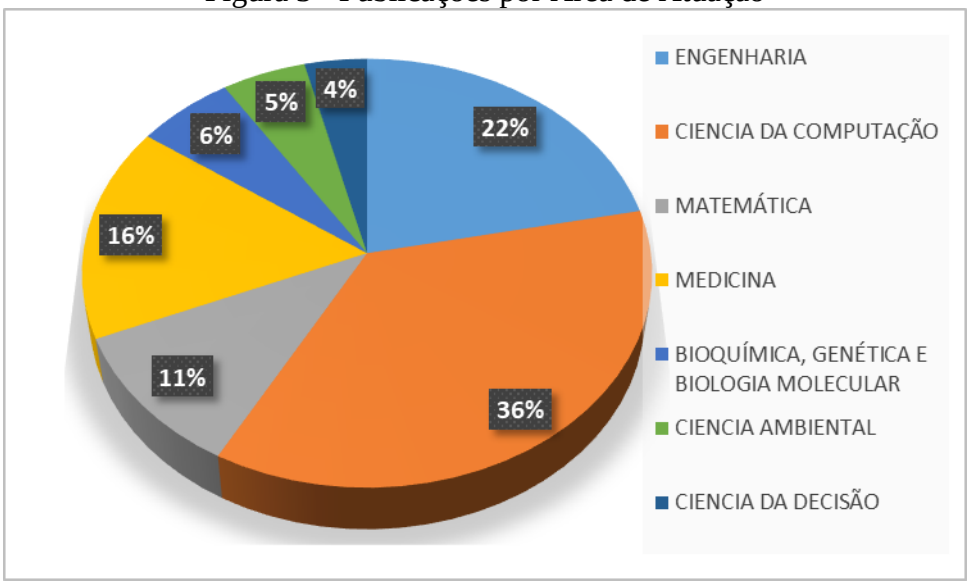
Figura 2 - Número de publicações ao longo dos últimos anos com as palavras-chave *quality control* e *machine learning*



Fonte: Elaborado pelo autor.

Detalhando os dados acima, a Figura 3 apresenta as áreas de aplicação relacionadas as técnicas de *Machine Learning* para tomada de decisão e controle de processos.

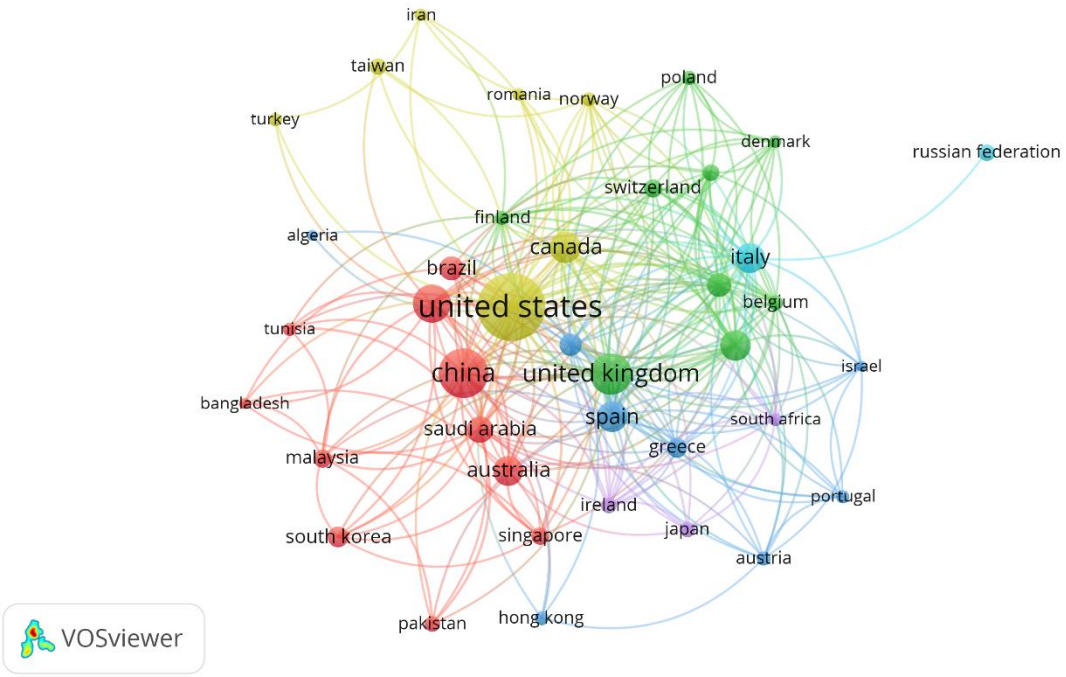
Figura 3 - Publicações por Área de Atuação



Fonte: Web of Science (2023).

O software VoSviewer foi utilizado para detectar os países dos coautores e analisar as suas interações através de dados gerais com as palavras-chave exibidas na Tabela 1. Foram mostrados 40 países, 6 clusters e 278 links, conforme apresentado na Figura 4. Cada país representa um nó da rede sendo os links entre países a relação entre os autores responsáveis pelo desenvolvimento do conteúdo. As cores dos links separam as maiores correlações em desenvolvimentos conjuntos entre os países.

Figura 4 - Países Coautores



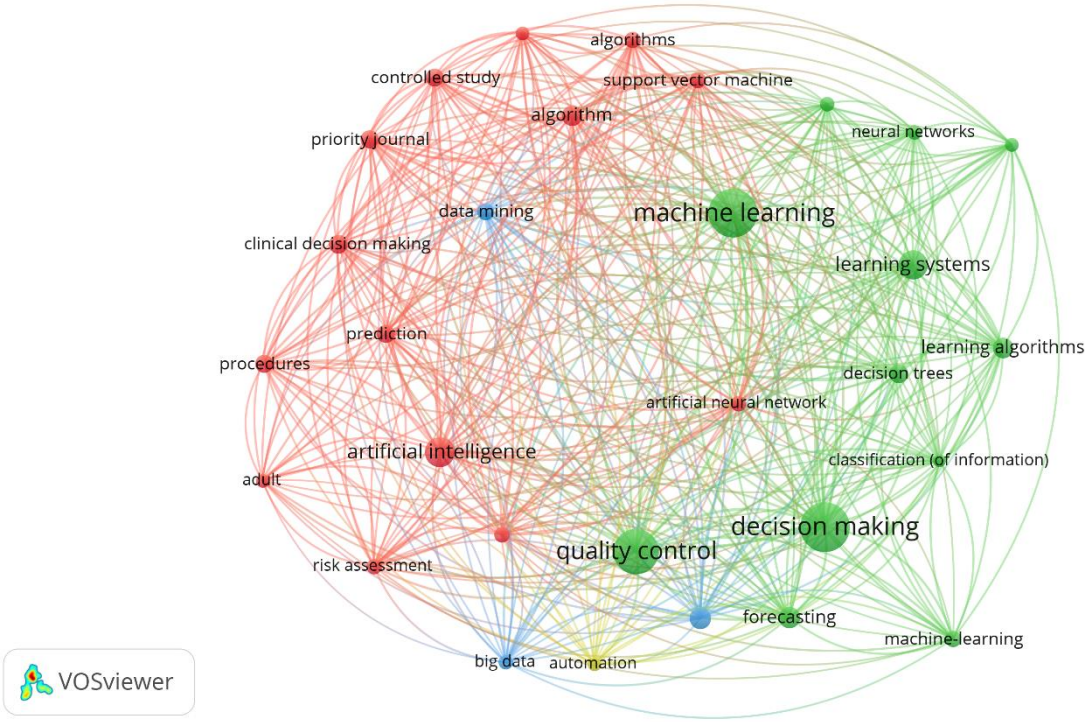
Fonte: Elaborado pelo autor.

Os três principais países com maior impacto são Estados Unidos, China e Reino Unido respectivamente. O tamanho do nó representa o montante de trabalhos publicados de acordo com a nacionalidade dos autores e co-autores. Assim, quanto maior o nó, maior capacidade de colaboração e impacto no país.

A segunda análise realizada no VOSviewer foi dos estudos relacionados as palavras chaves *machine learning*, *quality control* e *decision making* foi feita uma filtragem de dados baseado no conhecimento prévio do setor no intuito de dar ênfase as variáveis relacionadas a este estudo. Para este estudo, foram consideradas as variáveis com ao menos 25 citações na literatura.

No software, foi gerada uma VOSviewer rede contendo 4 clusters principais e 418 links entre as variáveis (Figura 5), onde a proximidade entre as mesmas expõe o nível de correlação entre si nos trabalhos expostos na literatura até o momento. Assim como o tamanho do nó representa a abrangência desta variável.

Figura 5 - Coocorrência de palavras-chave



Fonte: Elaborado pelo autor.

A proximidade entre os termos é maior em função da frequência com que aparecem simultaneamente nas mesmas publicações. Sendo assim, pode-se observar que os termos *quality control*, *decision making* e *machine learning* apresentam compartilhamento entre si, situando-se na região central do mapa onde se evidenciam a maior concentração das conexões. Isso demonstra a relevância dos termos e que diversas pesquisas vêm sendo desenvolvidas nessas áreas.

Por fim, foram adicionadas as palavras-chave *Rolling* (Laminação), *Sheet* (Chapa) e posteriormente *Metallurgy* (Metalurgia), onde pode-se observar que, apesar da forte tendência positiva em publicações sobre o tema, existe uma lacuna na utilização deste método para controle de processos de laminação e até mesmo na área de metalurgia de modo geral.

Tabela 2 - Pesquisa Aprofundada (Scopus)

PESQUISA	TEMPO ESTIPULADO (2012 – 2023)	
	WEB OF SCIENCE	SCOPUS
"*Machine Learning*" AND "*Decision Making*" AND "Quality *Control*" AND "*Rolling*"	2	2
"*Machine Learning*" AND "*Decision Making*" AND "Quality *Control*" AND "*Metallurgy*"	3	0
"*Machine Learning*" AND "*Decision Making*" AND "Quality *Control*" AND "*Sheet*"	1	2

Fonte: Web of *Science e* SCOPUS (2023).

2.2 REFERENCIAL TEÓRICO

Atualmente as técnicas de *Machine Learning* têm sido utilizadas em diversos tipos de aplicação em todos os setores industriais, no setor siderúrico, Feng et al (2020) implementaram e compararam diferentes técnicas de *Machine Learning* como Redes Neurais Atificiais (*RNA*), Árvore de Classificação e Regressão (*CART*), Processso de Regressão Gaussiano (*GPR*) e Árvore de Regressão tipo "Bagging" (*BRT*) para predição de variáveis críticas de laminação para auxiliar no *ajuste* de pressão de *Bending* (sistema responsável pela flexão dos cilindros de laminação) de um laminador a quente de aço ao iniciar o processo de laminação de forma a atingir os melhores resultados em termos de perfil e planicicidade do material, concluindo que é possível atingir grau de acurácia aceitável com baixo custo de processamento.

Yacob et al. (2019) utilizaram técnicas de classificação utilizando *Machine Learning* para identificação de anomalias em modelos de gêmeos digitais criados para a inspeção automática de superfícies. Foram comparados modelos criados utilizando Árvore de Decisão, *Supor Vector Machine* (*SVM*) *e K-Nearest Neighbors (kNN)*, onde o modelo de árvore de decisão atingiu acurácia acima de 97%.

Um trabalho realizado por Tamminen et al. (2020) utiliza técnicas de Machine Learning para criar um modelo estatístico de predição com o intuito de suportar na tomada de decisão para monitoramento de desvios na centralização do material ao longo de seu comprimento durante o processamento em um laminador de aço. O modelo utilizando Gradient Boosting Methods (GBM) foi criado com o intuito de predizer a centralização do material baseado em inputs como valores de setup de máquina e tipo de produto, assim como classificar cada variavel de entrada de acordo com o seu impacto no atingimento do

resultado final e encontrar correlações entre diferentes varíaveis de entrada que possam interferir significativamente na centralização do material.

Existem trabalhos promissores utilizando Machine Learning para prever processos de conformação mecânica de produtos laminados, Ruediger et al. (2020) desenvolveram um sistema utilizando Gêmeos Digitais (*Digital Twins*) para prever falhas em processos de estampagem e montagem de produtos automotivos utilizando metais laminados (Chapas). O sistema recebe todos os parâmetros e inputs necessários para simular cada *etapa* do processo e conflitar com os outputs do processo real, auxiliando a equipe responsável na identificação de possíveis desvios na matéria prima e/ou equipamentos envolvidos.

Kalathas e Papoutsidakis (2020) criaram um sistema baseado em *Machine Learning e Data Mining* (mineração de dados) que utiliza dados de entrada e saída de um almoxarifado de peças em uma empresa de transportes ferroviários e relatórios de tráfego dos equipamentos para prever as próximas ocorrências de falha nos equipamentos, sinalizando a equipe de manutenção para realização de manutenções preditivas.

2.2.1 Redes neurais artificiais

Usadas principalmente para algoritmos de deep learning, as redes neurais processam dados de treinamento imitando a interconectividade do cérebro humano por meio de camadas de nós. Cada nó é composto de entradas, pesos, um viés (ou limite) e uma saída. Se esse valor de saída exceder um determinado limite, ele "dispara" ou ativa o nó, passando dados para a camada seguinte na rede. As redes neurais aprendem essa função de mapeamento por meio do aprendizado supervisionado, ajustando-se com base na função de perda por meio do processo de descida do gradiente. Quando a função de custo é igual ou próxima de zero, podemos confiar na precisão do modelo para fornecer a resposta correta (IBM.com, 2023).

2.2.2 Naive Bayes

É uma abordagem de classificação que adota o princípio da independência condicional de classe a partir do Teorema de Bayes. Isso significa que a presença de um recurso não impacta a presença de outro na probabilidade de um determinado resultado, e

cada preditor tem um efeito igual sobre aquele resultado. Existem três tipos de classificadores Naïve Bayes: Multinomial Naïve Bayes, Bernoulli Naïve Bayes e Gaussian Naïve Bayes. Esta técnica é usada principalmente em classificação de textos, identificação de spam e sistemas de recomendações (IBM.com, 2023).

2.2.3 Regressão linear

A regressão linear é usada para identificar o relacionamento entre uma variável dependente e uma ou mais variáveis independente e normalmente é usada para fazer previsões sobre resultados futuros. Quando há apenas uma variável independente e uma variável dependente, isso é conhecido como regressão linear simples. À medida que o número de variáveis independentes aumenta, ela é chamada de regressão linear múltipla. Para cada tipo de regressão linear, busca-se traçar uma reta de melhor ajuste, que é calculada através do método dos mínimos quadrados (IBM.com, 2023).

2.2.4 Máquinas De Vetores De Suporte (SVM)

Uma máquina de vetores de suporte é um modelo de aprendizado supervisionado conhecido desenvolvido por Vladimir Vapnik, usado para classificação e regressão de dados. Dito isso, normalmente é usado para classificar problemas, criando um hiperplano em que a distância entre duas classes de pontos de dados é máxima. Este hiperplano é conhecido como o limite de decisão, separando as classes de pontos de dados (por exemplo, laranjas vs. maçãs) em cada lado do plano (IBM.com, 2023).

2.2.5 K-Vizinhos mais próximos

O K-vizinhos mais próximos, também conhecido como algoritmo KNN, é um algoritmo não paramétrico que classifica pontos de dados com base em sua proximidade e associação a outros dados disponíveis. Este algoritmo assume que pontos de dados semelhantes podem ser encontrados próximos uns dos outros. Como resultado, ele procura calcular a distância entre os pontos de dados, geralmente através da distância euclidiana e, então, atribui uma categoria com base na categoria ou média mais frequente. Sua facilidade de uso e baixo cálculo de tempo o tornam um algoritmo preferencial por cientistas de dados, mas à medida que o conjunto de dados de teste aumenta, o tempo de processamento

se prolonga, tornando-o menos atrativo para tarefas de classificação. O KNN é normalmente usado para mecanismos de recomendação e reconhecimento de imagem (IBM.com, 2023).

2.2.6 Árvores de decisão

Árvores de decisão (DTs) são um método de aprendizagem supervisionado não paramétrico usado para classificação e regressão. O objetivo é criar um modelo que preveja o valor de uma variável alvo, aprendendo regras de decisão simples inferidas a partir dos recursos dos dados. Uma árvore pode ser vista como uma aproximação constante por partes onde as árvores de decisão aprendem com os dados a aproximar uma curva senoidal com um conjunto de regras de decisão se-então-senão. Quanto mais profunda a árvore, mais complexas serão as regras de decisão e mais ajustado será o modelo (Scikit-Learn.org, 2023).

Ao baixar os casos no conjunto de teste para as DTs, a árvore de decisão (DT) é selecionada com base na precisão (LOH, 2011), (DESHPANDE, 2016). Devido ao problema de *over-fitting* (sobreajuste), se a árvore for excessivamente grande, ela terá uma taxa maior de erro de classificação ou erro, portanto, a poda é necessária para remover alguns galhos ou hubs (SHAMRAT et al, 2021).

A poda é um método em IA que diminui o tamanho das árvores de escolha evacuando partes da árvore. Além disso, diminui a dificuldade do classificador em fornecer previsões mais precisas, removendo fragmentos superajustados e dados ruidosos ou errôneos (BRAMER, 2021).

Em termos de poda do modelo, foram feitos vários testes para verificar a melhor combinação de profundidade da árvore (níveis de ramificação) e índice de gini e critério de entropia.

2.2.6.1 Gini Index

O índice de Gini é usado como uma medida estatística para calcular se uma feição é capaz de separar instâncias de diferentes classes (CHENG et al, 2021). O índice de Gini pode ser formulado por (SINGER, 2020):

$$gini_{index(X_K)} = \min_{Z}(p(Z)(1 - \sum_{j=1}^{c} p\,(C_j|Z)^2) + p(\bar{Z})(1 - \sum_{j=1}^{c} p(C_j\,|\bar{Z})(C_j|\overline{Z})^2))$$

(1)

onde Z, Z são o conjunto de instâncias em que o valor do recurso é menor ou igual ao i-ésimo valor do recurso e maior que o i-ésimo valor do recurso, respectivamente. Além disso, p(.) ep(.| .) denotam probabilidade e probabilidade condicional, respectivamente.

2.2.6.2 Entropia

O conceito mais comum é a entropia de Shannon. Em um nível conceitual, a Entropia de Shannon é simplesmente a "quantidade de informação" em uma variável. Basicamente, refere-se à quantidade de armazenamento necessária para armazenar a variável. Que consiste na fórmula:

$$H(c) = -\sum_{i=1}^{n} P(c_i) log_b P(c_i)$$

(2)

2.2.7 Métricas de avaliação

2.2.7.1 Acurácia

Refere-se ao número de positivos verdadeiros, ou seja, número de itens classificados corretamente) divido pelo número total de classificações.

$$Acurácia = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

(3)

Onde tp é o número de testes que indicou corretamente a presença de uma condição ou característica e fp é o número de indicou ausência de uma condição ou característica para o teste.

2.2.7.2 Recall

Refere-se ao número de positivos verdadeiros, ou seja, número de itens classificados corretamente) divido pelo número total de classificações que deveriam ter sido classificadas corretamente.

$$Recall = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{4}$$

Onde tp é o número de testes que indicou corretamente a presença de uma condição ou característica e fn é o número de testes classificado incorretamente.

2.2.7.3 F-Score

Calcula a performance de predição, resume-se a uma média harmônica dos índices de Precisão e *Recall*

$$F - Score = \frac{2\text{tp}}{2\text{tp} + \text{fp} + \text{fn}} \tag{5}$$

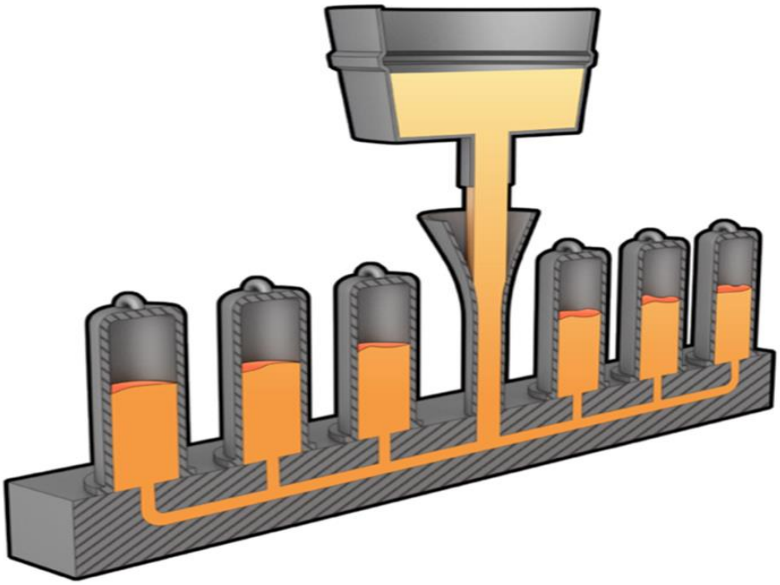
Onde tp é o número de testes que indicou corretamente a presença de uma condição ou característica e fn é o número de testes classificado incorretamente.

2.2.8 O Processo de laminação

O processo de laminação consiste basicamente na redução de espessura do material em formado de chapa. Ela é feita por conjuntos de cilindros feitos de aço forjado ou fundido que são posicionados utilizando cilindros hidráulicos e utilizam sistema de arrefecimento/lubrificação a base de emulsão (mistura de água e óleo) ou apenas óleo de laminação.

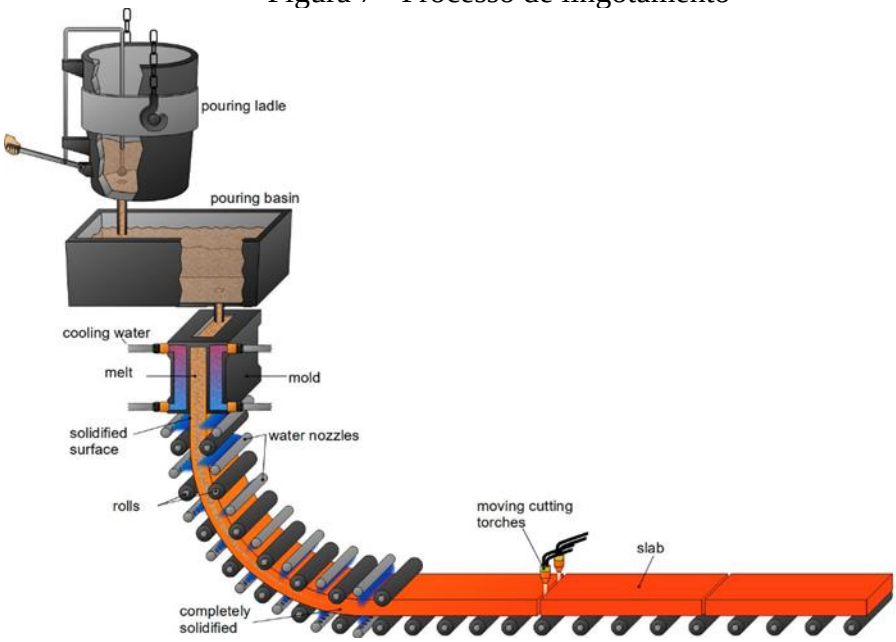
Na primeira etapa, o material fundido é envasado no formato de placa utilizando moldes de formato retangular apoiados por cilindros de elevação ou utilizando o processo de lingotamento contínuo, onde o material fundido é depositado entre um conjunto de cilindros ou esteiras posicionados a uma distância referente a espessura desejada.

Figura 6 - Processo de vazamento de placas



Fonte: MANUFACTURING GUIDE (2023).

Figura 7 - Processo de lingotamento

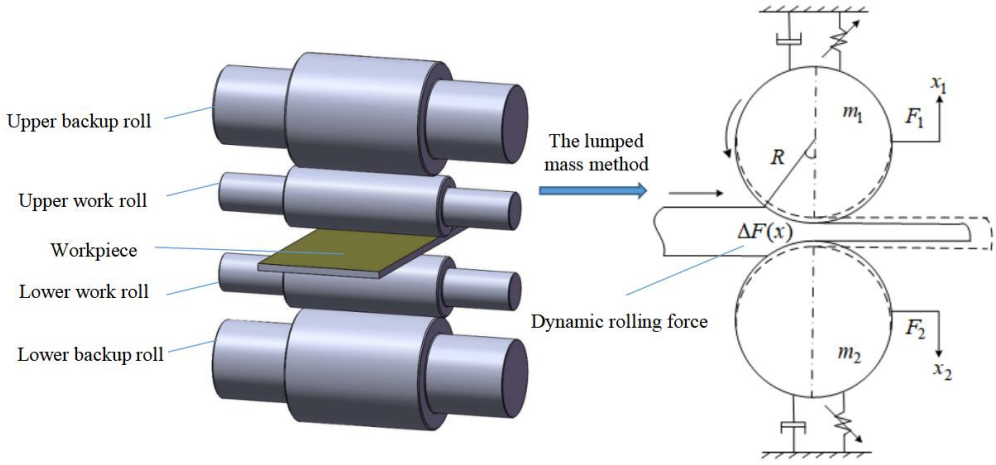


Fonte: TEC-SCIENCE (2023).

Nas etapas seguintes, o material é laminado até alcançar a espessura final desejada. A quantidade de etapas de laminação irá variar de acordo com o tipo e aplicação do produto.

O processo de laminação é realizado por equipamentos de alta precisão, com malhas de controle complexas e instrumentação delicada para ajustes rápidos e precisos de modo a manter as variáveis de interesse sob controle. De modo geral, um laminador (Figura 1) consiste basicamente em atuadores (cilindros hidráulicos e/ou parafusos acoplados a motores), cilindros de laminação e sistemas de lubrificação/arrefecimento (PENG, 2023).

Figura 8 - Diagrama de laminador "4-High"



Fonte: PENG (2023).

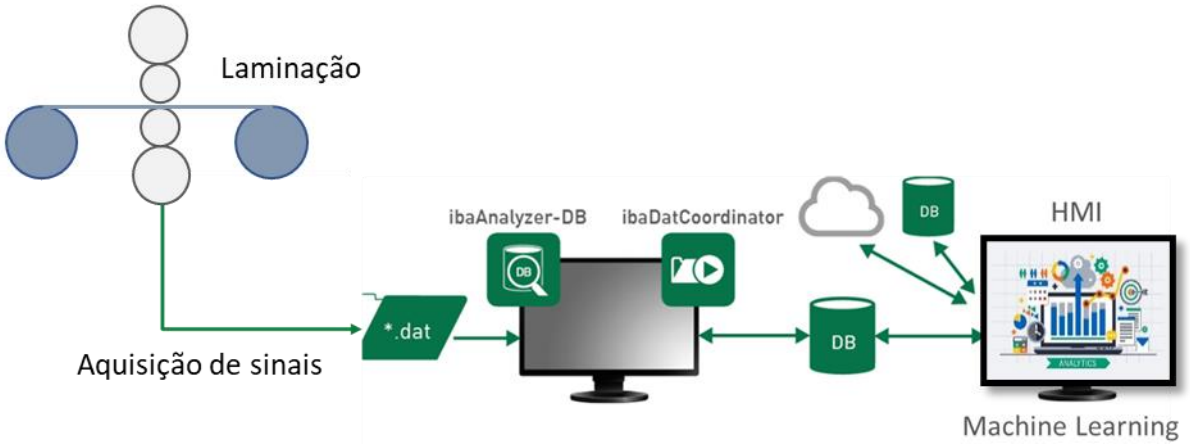
Levando em consideração os diâmetros dos cilindros de laminação (roletes que estarão em contato com o material), dilatação térmica, características do material a ser laminação e a espessura final desejada, é feito o posicionamento dos atuadores ao longo do processo de modo a manter a espessura o mais próximo possível do *target*. Em paralelo, é controlada a velocidade dos motores que giram os cilindros de laminação com o objetivo de controlar a temperatura do material ao longo do comprimento, atingindo as propriedades mecânicas necessárias.

3 MATERIAIS E MÉTODOS

3.1 SISTEMÁTICA DO TRABALHO

O conceito utilizado consiste na aquisição de sinais de campo pelo sistema ibaPDA, seguido da utilização de formulas para tratativa inicial dos sinais e construção de formulas para criação de novos indicadores, após o término da operação, os dados foram armazenados em arquivos brutos (.dat) e enviados a um software de gerenciamento de arquivos, o IbaDatCoordinator, fazendo conversão destes arquivos para a extensão .parquet (arquivo tipo texto utilizado em processamento em massa, ex.: *Big Data*) e enviando os mesmos para uma nuvem de armazenamento. Ao identificar a adição de um novo arquivo, o executável contendo o algoritmo para a análise de qualidade utilizando árvore de decisão é iniciado e faz a análise dos dados, identificando a classe/aplicação do produto e analisando as variáveis críticas utilizando os critérios pré-estabelecidos para aquele produto, retornando como resultado um relatório detalhado sobre o processo, liberando o material para seguir processo, retrabalhar possíveis regiões com defeito, reclassificar para outra aplicação e/ou sucatá-lo para impedir impactos negativos nas próximas operações internas ou até mesmo nas linhas dos clientes finais.

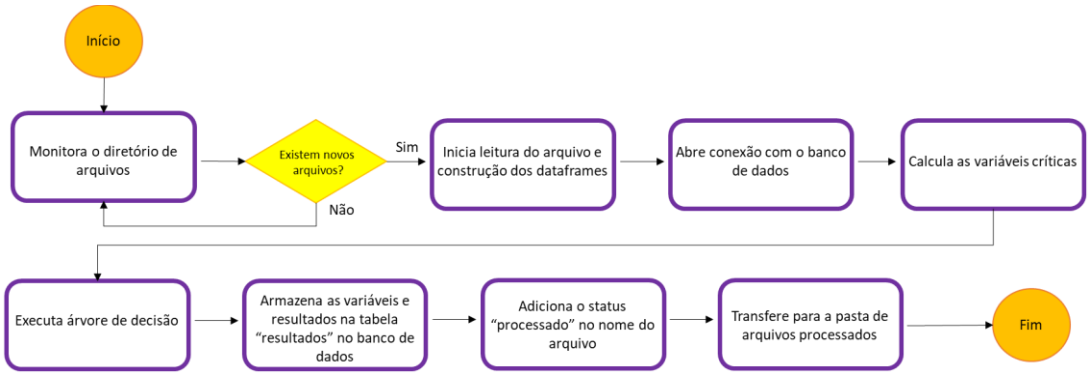
Figura 9 - Fluxo de dados



Fonte: Elaborado pelo autor.

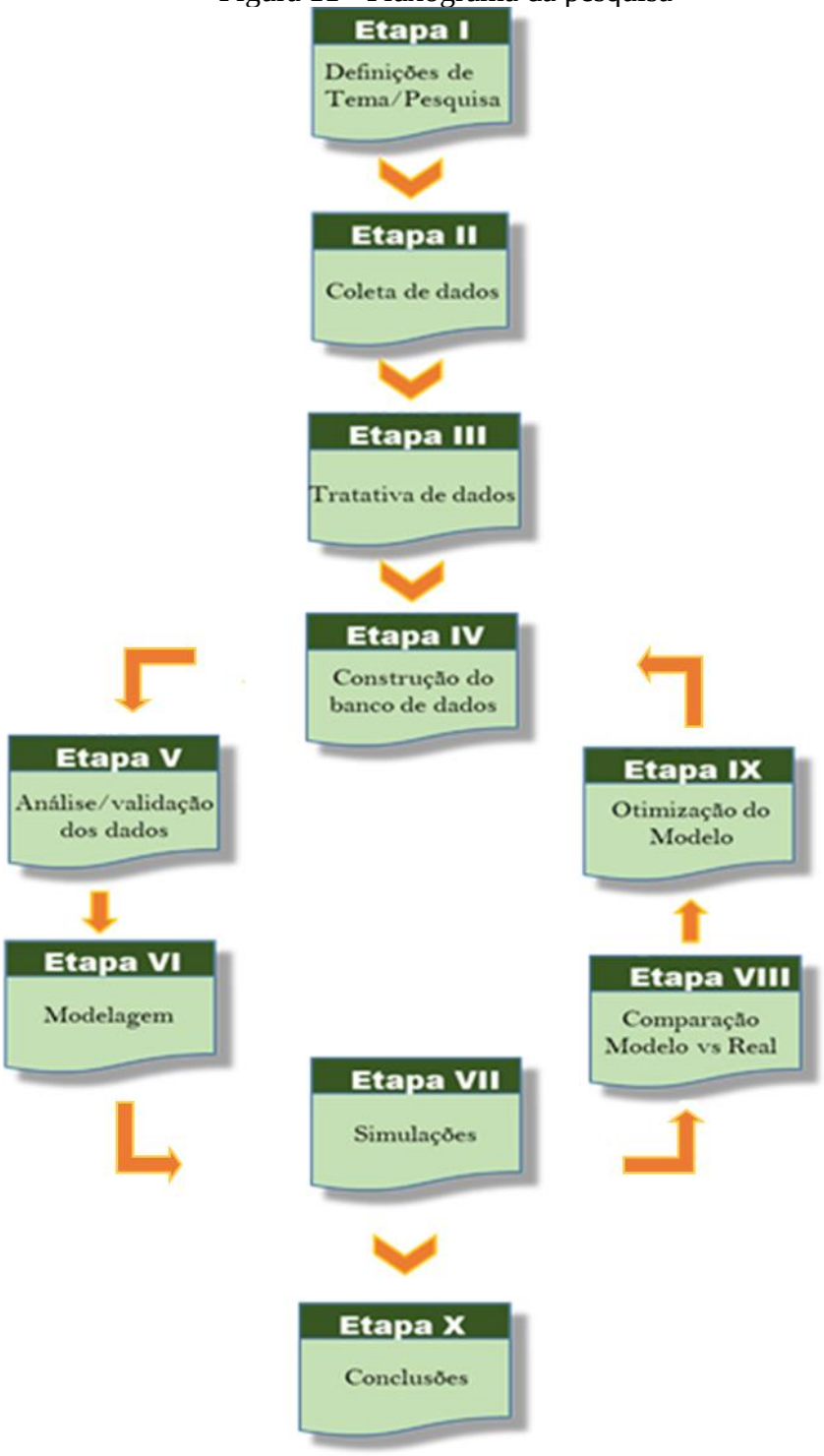
Os equipamentos evolvidos no processo de laminação (laminadores) possuem ferramentas e malhas de controle para mitigar desvios inerentes do processo de laminação, como variações de espessura decorrentes de vibração e/ou relativas a excentricidade dos cilindros de laminação com a constante troca de calor com o material laminado e ambiente. Além da análise detalhada do processo recém executado, é possível utilizar as técnicas de aprendizado de máquina (*Machine Learning*) para estabelecer padrões de laminação e características do material após o seu processamento, assim como retroalimentar o sistema de tomada de decisão utilizando dados dos processos seguintes, tornando o sistema mais assertivo através de predições de resultados baseados em históricos de defeitos e capacidade de correção dos mesmos nos próximos processos, assim como direcionar o material para uma rota de equipamentos especifica de modo a tratar da melhor forma possível o defeito, como por exemplo, direcionar um material com variações de espessura em determinada faixa de frequência para os equipamentos como melhores malhas para correção deste tipo de desvio.

Figura 10 - Fluxograma do software



Fonte: Elaborado pelo autor.

Figura 11 - Fluxograma da pesquisa



Fonte: Elaborado pelo autor.

A Figura 11 expõe a sistemática utilizada, sendo divido em nove etapas envolvendo manipulação e tratativa de dados, implementação/otimização do modelo e supervisão.

Etapa I – Definição de tema: Essa etapa buscou encontrar um tema que seja uma lacuna de recurso enfrentado atualmente na realidade cotidiana no setor de laminação.

Etapa II – Coleta de dados: Os dados em tempo real do processo em geral (equipamento/produto) serão coletados utilizando o *software* ibaPDA e armazenados em arquivos específicos com a extensão .dat para períodos específicos, neste caso, iniciando e finalizando o armazenamento nestes respectivos momentos durante o processo de laminação de cada produto.

Etapa III – Tratativa de dados: Os arquivos ".dat" situados no diretório padrão do ibaPDA serão reprocessados pelo software ibaDATCoordinator, responsável por extrair os dados convertendo-os para um novo arquivo com a extensão ".parquet" e enviando-lhes para um novo diretório sob monitoramento do sistema proposto.

Por sua vez, o sistema faz varreduras periódicas neste diretório a procura de novos arquivos para processamento. Ao encontrar um arquivo, o sistema faz a leitura de todos os dados identificando as variáveis críticas para a execução das equações para detecção dos tipos e características dos eventuais desvios encontrados como amplitude do defeito e posicionamento ao longo do processo no intuito de segregar toda a região envolvida no defeito, impedindo o envio de material fora da especificação para as próximas etapas do processo.

Etapa IV – Construção do banco de dados: O sistema é capaz de inserir os *outputs* das equações e modelos em um banco de dados pré-configurado, inicialmente, foram inseridos apenas os *outputs* das equações com detalhes dos desvios encontrados e características do material para uma posterior análise detalhadas pelos especialistas.

Etapa V – Análise/Validação dos dados: Os profissionais mais experientes responsáveis pela caracterização e disposição de material fizeram uma conferência de um conjunto determinado de resultados conflitando-os com os resultados obtidos com o sistema atual utilizando a ferramenta de análise de dados de produção ibaAnalyzer, que permite observar os traços dos sinais ao longo do tempo e execução de cada detalhe das equações no intuito de confirmar a assertividade das novas equações na determinação da amplitude e posicionamento dos defeitos. Esta equipe de profissionais também fez a inserção manual no banco de dados com as saídas esperadas pelo modelo, sendo elas: 1 - "Seguir Processo", 2 - "Reclassificar" ou 3 – "Sucatar" baseado nos resultados das equações. Este conjunto de dados foi utilizado para o treinamento do modelo.

Adicionalmente, foram feitas consultas no banco de dados a procura de dados nulos e/ou inconsistentes para identificar possíveis falhas no processamento dos dados crus.

Etapa IV – Modelagem: O conjunto de dados revisados foi utilizado para modelar a árvore de decisão com diferentes parâmetros de modo a encontrar a combinação com maior acurácia.

Etapa VIII – Comparação Modelos x Dados Reais: Após a simulação utilizando dados de treinamento, cada modelo será avaliado de acordo com sua convergência aos dados inseridos manualmente buscando encontrar o modelo com maior assertividade.

Etapa IX – Otimização: As saídas do modelo escolhido foram conflitadas com os dados do sistema atual contendo dados manuais inseridos rotineiramente pelos especialistas. Eventuais *outliers* e/ou resultados inesperados foram tratados pontualmente e reinseridos no conjunto de treinamento do modelo no intuito de otimizar os resultados.

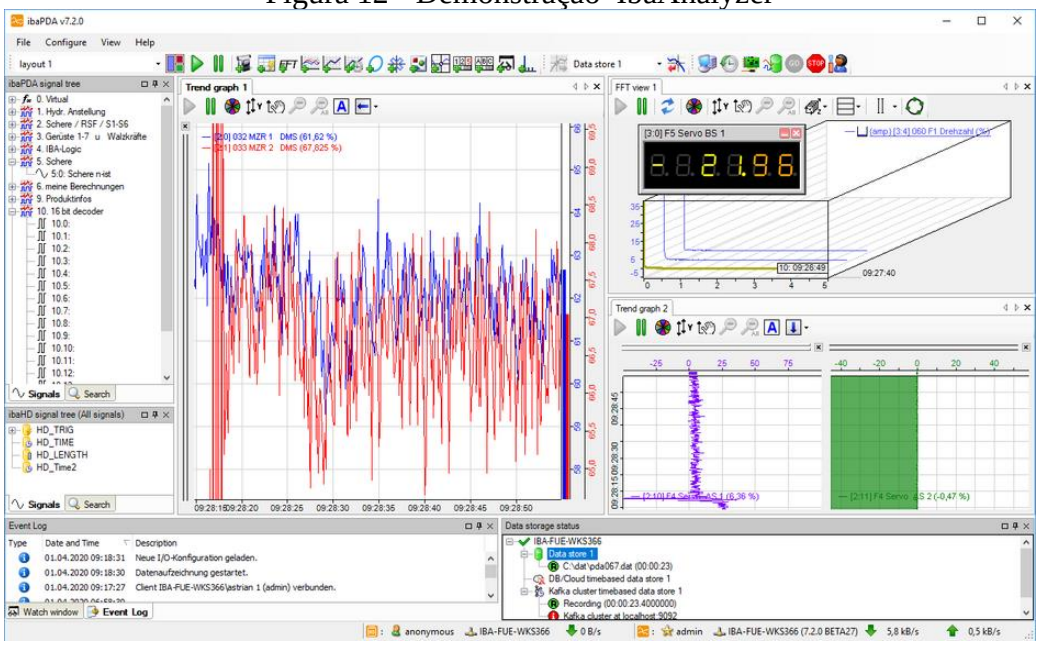
3.2 FERRAMENTAS UTILIZADAS

Neste trabalho, foram priorizadas a utilização de softwares livres de modo a buscar a melhor combinação de custo-benefício. Dentre elas, foram escolhidas as ferramentas nas quais o autor possui maior afinidade e conhecimento de manuseio e programação, tais como:

3.2.1 Softwares iba

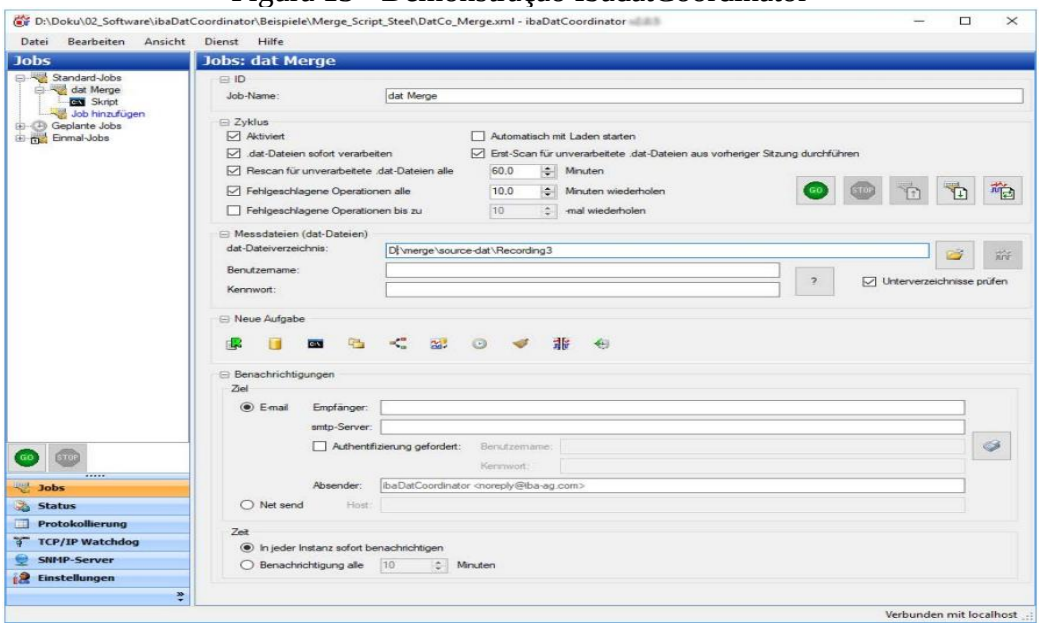
Iba é uma empresa alemã especializada em aquisição e tratamento de dados contando com mais de 200 mil sistemas ativos, 20 escritórios em diferentes países e diversas plataformas e equipamentos que possibilitam gerenciar dados desde a sua aquisição, tratamento, análises técnicas em um período específico ou até análises complexas utilizando *big data* ou *data lake*. Neste trabalho, serão utilizados os *softwares* ibaPDA para aquisição de dados, ibaAnalyzer para análise dos sinais reais e o ibaDATCoordinator para conversão dos arquivos raiz (.dat) para a extensão parquet e respectivo manuseio dos arquivos para o diretório de monitoramento para início do processamento dos dados pelo sistema proposto neste trabalho.

Figura 12 - Demonstração IbaAnalyzer



Fonte: IBA AG (2023).

Figura 13 - Demonstração IbadatCoordinator



Fonte: IBA AG (2023).

3.2.2 Arquivos com extensão “parquet”

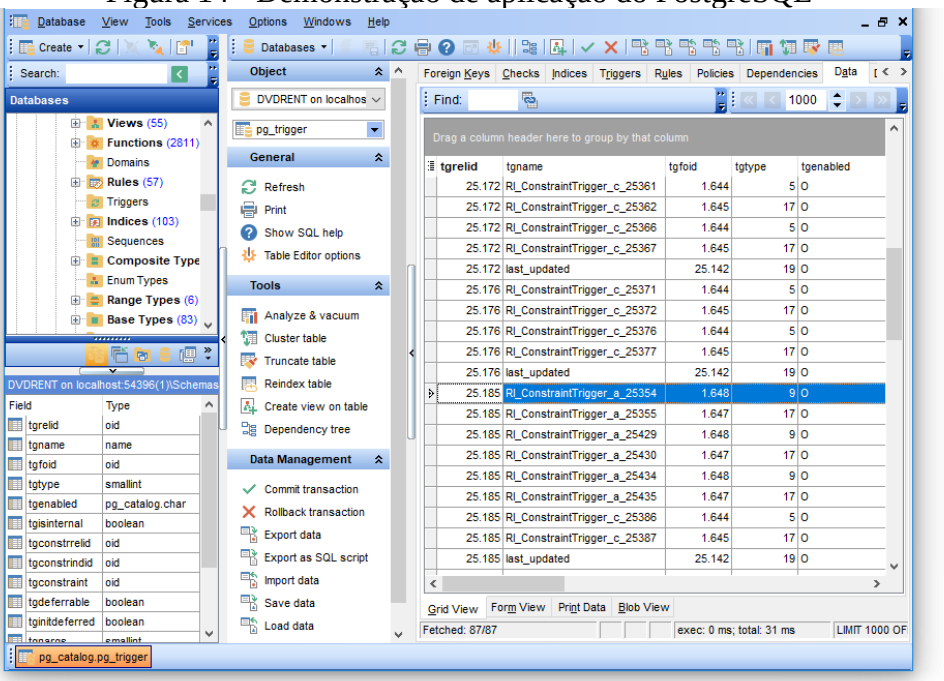
Os arquivos contendo dados de produção são armazenados na extensão Apache Parquet (.parquet), sendo este um arquivo de código fonte aberto com a disposição de dados baseada em colunas e tabulação, com suporte para esquemas de compressão e codificação

de dados, se tornando mais eficientes se comparando com os arquivos de tabulação usuais, baseado em linhas, como a extensão .csv. Arquivos com estruturas colunares se tornar mais interessantes pois possibilitam uma eficiência maior, tanto em termos de tamanho de arquivos quanto performance de consulta dos dados (White, 2015).

3.2.3 Banco de dados PostgreeSQL

O *software* livre de banco de dados utilizado para o desenvolvimento desta pesquisa é o PostgressSQL, armazenando variáveis tratadas e geradas a partir de dados de produção (.parquet) de uma empresa brasileira multinacional no ramo da siderurgia, com mais de 20 unidades em mais de 10 países e cerca de 15 mil funcionários; fabricante de produtos laminados para os setores automobilístico, farmacêutico, alimentício, aeroespacial e utilidades. Os dados envolvidos neste trabalho foram retirados do setor de laminação.

Figura 14 - Demonstração de aplicação do PostgreSQL



Fonte: adaptado de SQLMANAGER.NET (2023).

3.2.4 Linguagem Python

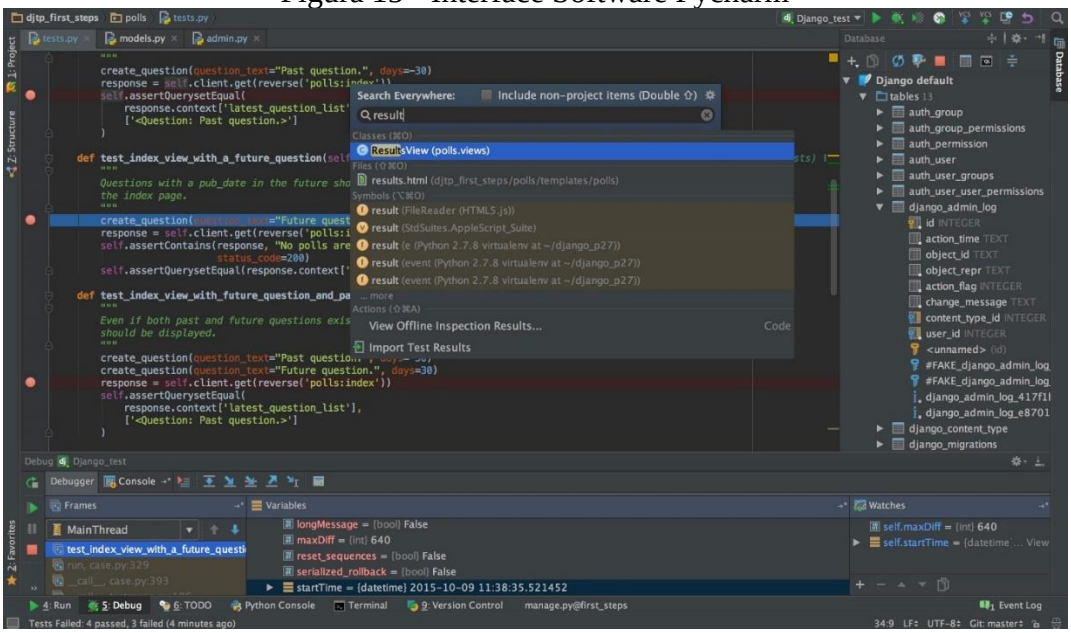
A linguagem Python foi criada por Guido van Rossum, há quinze anos, com a ajuda de dois colegas, Jack Jansen e Sjoerd Mullender, como um passatempo. O objetivo dos

criadores de Python era criar uma linguagem orientada a objetos, altamente portável e menos complexa do que Java ou C++ (SONGINI, 2005).

3.2.5 Software Pycharm para programação em Python

Desenvolvido pela empresa tcheca JetBrains, o Pycharm é um Desenvolvimento de ambiente integrado que inclui um depurador integrado e um executor de teste, um profiler Python, um terminal incorporado, integração com as principais ferramentas de banco de dados incorporadas e VCS, recursos de desenvolvimento remoto com interpretadores remotos, um terminal SSH integrado e integração com o Docker e o Vagrant (JETBRAINS, 2023).

Figura 15 - Interface Software Pycharm



Fonte: JETBRAINS (2023).

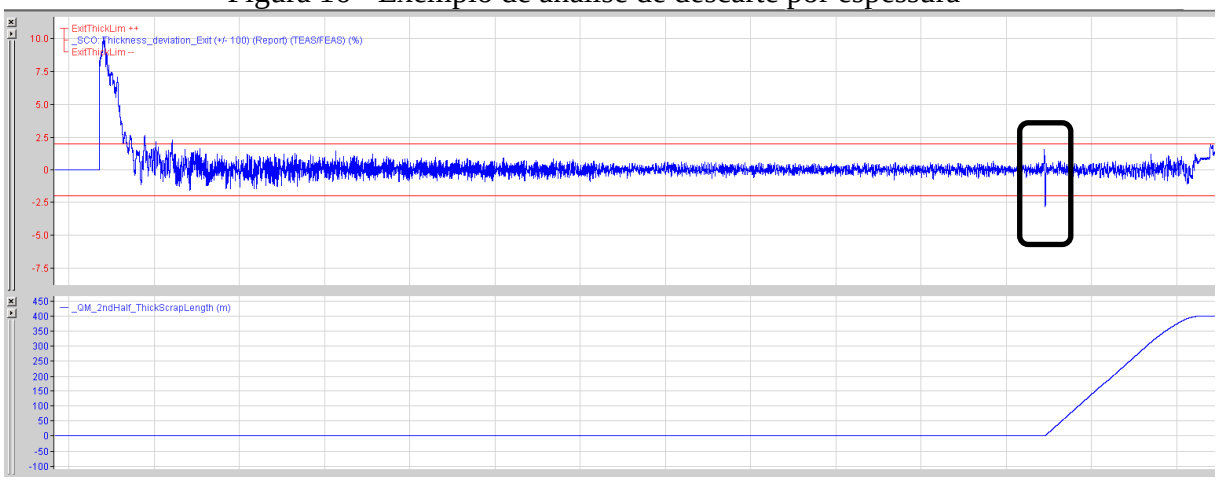
3.3 TRATATIVA DE DADOS

3.3.1 Parâmetros para monitoramento e controle de qualidade

O sistema desenvolvido neste trabalho faz a análise de desvio de espessura e temperatura ao longo de todo o comprimento da bobina, correlacionando com outras variáveis como classe de produto, largura, espessura alvo e comprimento para detalhar o

tipo de defeito e sugerir ações para o restante do processo, auxiliando na tomada de decisão dos especialistas.

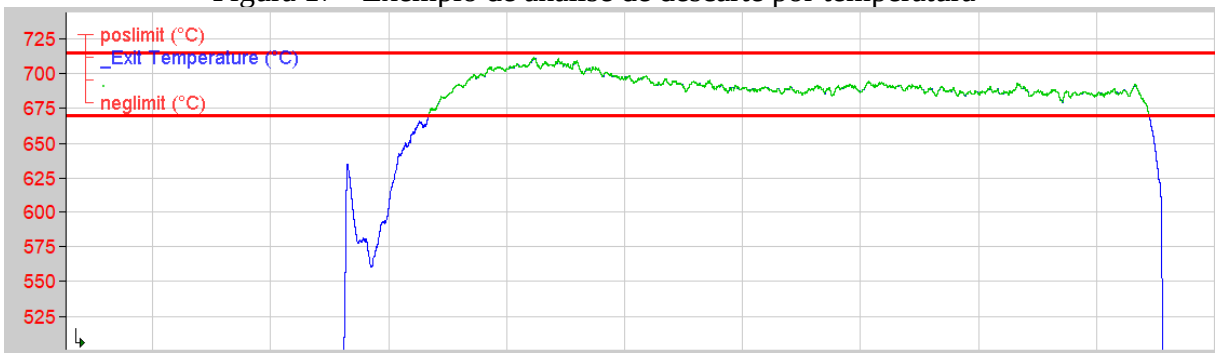
Figura 16 - Exemplo de análise de descarte por espessura



Fonte: Elaborado pelo autor.

A Figura 16 exemplifica uma bobina com um pico de espessura localizado a cerca de ¾ do final, seguido por um gráfico simbolizando qual deveria ser o comprimento do corte para a eliminação do defeito (neste exemplo, aproximadamente 400m).

Figura 17 - Exemplo de análise de descarte por temperatura



Fonte: Elaborado pelo autor.

A Figura 17 exemplifica uma avaliação de região de descarte de acordo com os limites pré-estabelecidos, onde a região detalhada em azul deverá ser descartada na próxima etapa do processo.

Devido a grande variedade de especificações e características de cada material, é necessária a criação de diversos descritivos dedicados a cada grupo, contendo uma lista de variaveis criticas a serem avaliadas, como os devidos limites de tolerância.

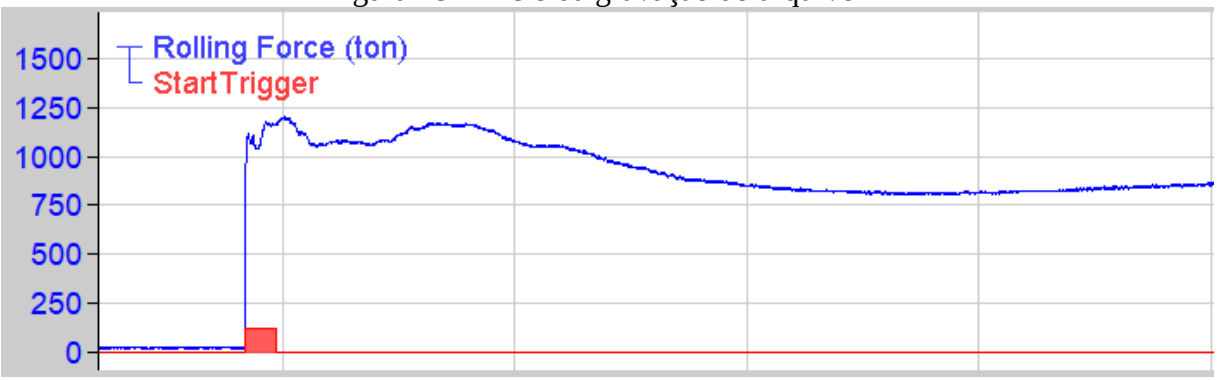
Um fator determinante para a qualidade das análises dos produtos consiste na escolha do sistema de aquisição de dados, para processos de laminação é necessário um sistema capaz de coletar sinais em altas frequências e baixos ciclos de tempo. Para este trabalho, serão utilizadas ferramentas da Iba, uma das maiores empresas no setor de aquisição de dados do mercado atual.

3.3.2 Coleta de dados

O software ibaPDA possui um recurso de gravação de arquivos utilizando *triggers* para início e final. Para este trabalho, foram criados 2 triggers:

StartTrigger – Momento em que o material a ser laminado toca a cadeira de laminação, este evento é caracterizado por um aumento repentino na força de laminação, (representada pela pressão do sistema hidráulico responsável pelo posicionamento do conjunto de cilindros de laminação) este evento simboliza a atuação dos controladores de posição aplicando uma força de reação para manter o cilindro na posição de referência pois algum objeto (neste caso o material) está exercendo uma força contrária.

Figura 18 - início da gravação do arquivo

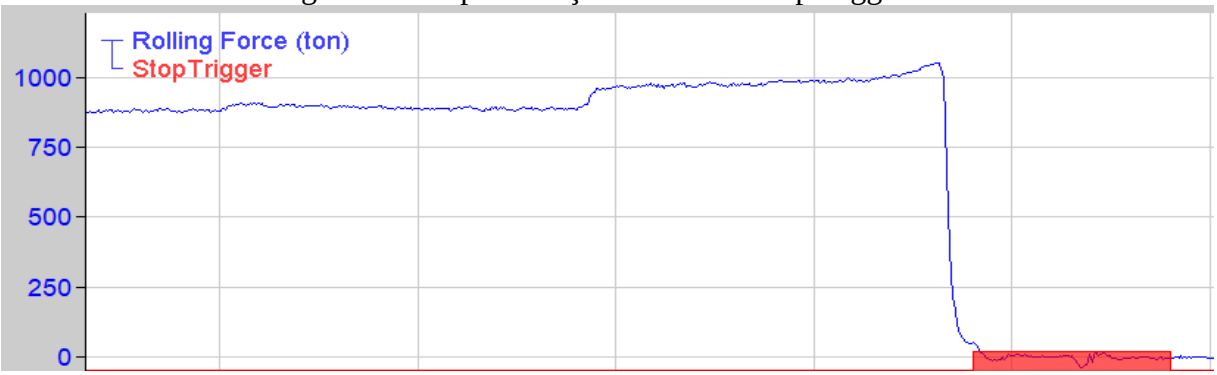


Fonte: Elaborado pelo autor.

StopTrigger – Momento em que a parte final do material em processo de laminação perde contato com a cadeira de laminação. Este evento é caracterizado pela queda repentina da força de laminação (representada pela pressão do sistema hidráulico responsável pelo

controle de posição dos cilindros de laminação) no momento em que não há mais força reativa exercida pelo material.

Figura 19 - Representação do evento StopTrigger



Fonte: Elaborado pelo autor.

No intervalo de tempo entre esses dois triggers foram coletados os sinais necessários para a análise de qualidade do material em um ciclo de 20ms, sendo eles:

- "ID" – Código de identificação do material a ser laminação;
- "Espessura_Target (µm)" – Espessura de referência do material após o processo de laminação;
- "Desvio_Espessura (µm)" – Valores de espessura real do material ao longo do comprimento logo após o processo de laminação, esses valores são adquiridos por um sistema de medição utilizando Raio-X posicionado há alguns metros da cadeira de laminação, este é capaz de inferir a espessura de acordo com o nível de radiação que passa através do material;
- "Limite_Desvio_Espessura (µm)" – Limite de desvio de espessura especificado para o tipo de material a ser laminado, essa variável será determinante no momento do cálculo da região de descarte;
- "Largura (mm)" – Largura de referência do material após o processo de laminação;
- "Comprimento (m)" – Comprimento do material ao longo do processo de laminação;
- "Liga" – Nomenclatura representando o conjunto de elementos que formam a composição química do material a ser laminado;
- "Temperatura_Target (°C)" – Temperatura de referência para o material logo após o processo de laminação;

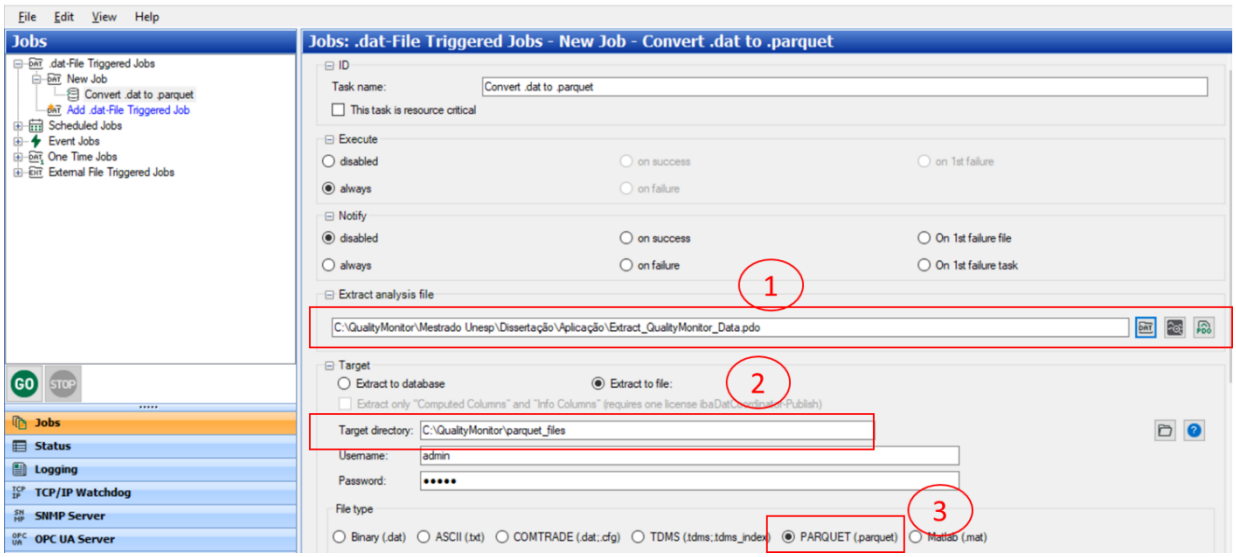
- “Desvio_Temperatura (°C)” – Temperatura do material ao longo do comprimento logo após o momento da laminação subtraído do sinal “Temperatura_Target”, os valores de temperatura real são obtidos através de um sensor infravermelho posicionado há alguns metros da cadeira de laminação.
- “Limite_Desvio_Temperatura (°C)” – Limite de desvio de temperatura especificado para o tipo de material a ser laminado, essa variável será determinante no momento do cálculo da região de descarte;

3.3.3 Conversão de arquivos .dat para .parquet

O arquivo de produção coletando os sinais citados no intervalo de tempo entre o StartTrigger e StopTrigger gerado após o término do processo de laminação é armazenado em uma pasta padrão e um formato específico utilizado pela iba, o formato .dat.

Assim que um novo arquivo é alocado na pasta padrão, o software de gerenciamento de arquivo chamado ibaDatCoordinator o identifica como novo arquivo e acessa suas informações buscando pelo campo de status, que contém as informações de processamento do arquivo. Onde, só irá processar os dados se o status estiver como “*readytoprocess*”, identificando que o arquivo está pronto para processamento e ainda não foi utilizando em nenhuma tarefa anterior, impedindo duplo processamento.

Figura 20 - tarefa de conversão - ibadatCoordinator



Fonte: Elaborado pelo autor.

Na figura 20 é possível observar os detalhes da tarefa criada para conversão dos arquivos .dat para a extensão parquet, onde pode-se observar em detalhes as etapas principais do processo.

Etapa 1 – É selecionado um arquivo padrão de análise chamado de “máscara”, este arquivo funciona como um filtro de quais sinais devem ser extraídos do arquivo original. Assim que o novo arquivo com status “*ReadyToProcess*” é encontrado, ele é executado através do ibaDatCoordinator utilizando essa máscara de filtro e seque para a etapa 2.

Etapa 2 – É estabelecido qual será o diretório padrão de envio dos arquivos gerados após a execução da tarefa.

Etapa 3 – É selecionado qual será o método de conversão do arquivo, onde pode-se observar as opções de extensão para binário, ASCII, COMTRADE, TDMS, PARQUET e Matlab. Neste trabalho, utilizaremos a extensão parquet devido a suas características de leveza, tabulação e variedade de bibliotecas dedicadas em Python.

3.3.4 Gerenciamento dos arquivos .parquet

Após a conversão dos arquivos para a extensão parquet e envio para o diretório específico. É feito o gerenciamento dos arquivos pelo próprio sistema fruto deste trabalho.

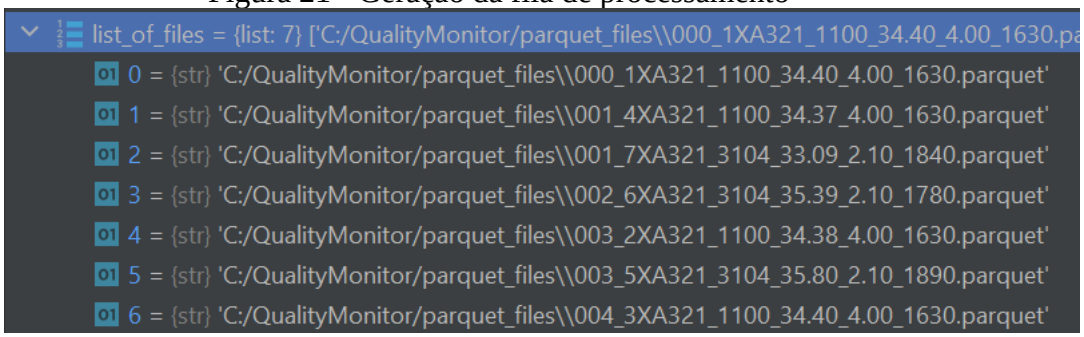
Primeiramente, é necessário identificar quais arquivos estão contidos no diretório, essa tarefa é feita através a biblioteca “glob”.

Quadro 1 - Código para listagem dos arquivos do diretório padrão

```
# Lista todos os arquivos do diretório. a expressão *.parquet simboliza todos os arquivos com extensão parquet
list_of_files = glob.glob('C:/QualityMonitor/parquet_files/*.parquet')
```

Fonte: Elaborado pelo autor.

Figura 21 - Geração da fila de processamento



Fonte: Elaborado pelo autor.

Como é possível observar, esse comando gera um vetor com o número de posições variável dependendo da quantidade de arquivos existentes. Sendo que cada uma delas contém uma *string* referente ao nome do arquivo encontrado.

Após a listagem dos arquivos, é necessário identificar a ordem de processamento, que para este trabalho foi especificada utilizando a data de modificação em ordem crescente. De modo a sempre priorizar o arquivo mais antigo.

Esse conceito foi definido levando em consideração a condição real de uma indústria, onde geralmente é utilizada a metodologia FIFO (*First In First Out*) para fluxo de material. Em resumo, esse conceito irá garantir que, em caso de um alto volume de processamento, será priorizado o material mais antigo para liberá-lo para o próximo processo, mitigando o impacto no fluxo interno e produção.

A determinação do arquivo a ser processado seguindo esse conceito é feita utilizando a função “min”.

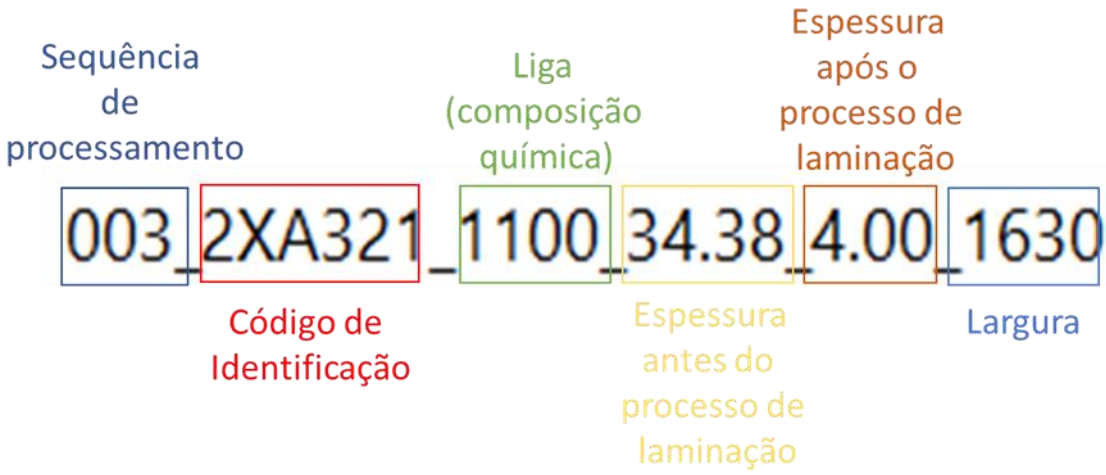
oldest_file = min(list_of_files, key=os.path.getctime) #Seleciona o arquivo com a menor data de modificação contido na lista de arquivos detectada no diretório

Ainda dentro dessa instrução, é utilizada a biblioteca “OS”, capaz de fazer operações relacionadas ao sistema operacional em execução (Python Org, 2023). Neste caso, foi utilizada a função “*path*” com a sub-rotina “*getctime*” para armazenar a data de modificação do arquivo, seguida da função “min” para definir o arquivo com a menor data de modificação, ou seja, o arquivo mais antigo contido na pasta.

3.3.5 Identificação do produto

Dentro do software de aquisição de dados de produção online da iba, o ibaPDA *(iba Process Data Acquisition)* é possível determinar a nomenclatura dos arquivos gerados utilizando os próprios sinais coletados. Utilizando este recurso, neste trabalho foi feito um cabeçalho para nomear os arquivos de produção utilizando as características principais do produto envolvido naquele ciclo do processo de laminação.

Figura 22 - Nomenclatura dos arquivos de produção



Fonte: Elaborado pelo autor.

Tendo esses dados disponíveis no nome do arquivo e considerando que os nomes dos arquivos estão em formato *string* contidos dentro do vetor "*list_of_files*" gerado anteriormente. É possível fragmentar essas strings e armazenar dados específicos eu novas variáveis para serem utilizadas em diversas funções do código. Esse trabalho é viabilizado pela biblioteca "Pandas", uma abreviação de "*Panel Data*" utilizada para manipulação de conjuntos de dados e criação dos *dataframes* amplamente utilizados na linguagem Python (MULINARI, 2023).

Utilizando o Pandas, foi feita a filtragem dos dados contidos nos nomes dos arquivos através da seleção de caracteres, armazenando informações do arquivo que serão utilizados para gerenciar os arquivos após o processamento.

No Quadro 2, pode-se ver no detalhe as instruções para segmentação e devidos resultados das mesmas.

Quadro 2 - Código para identificação e classificação dos arquivos de produção

```
#Armazena os caracteres da posição 0 até 77 do arquivo a ser processado
Complete_FileName = oldest_file[0:77]
#Exibe o conteúdo da variável Complete_FileName
print(Complete_FileName)
Resultado:
C:/QualityMonitor/parquet_files\001_7XA321_3104_33.09_2.10_1840.parquet
#Armazena os caracteres da posição 32 até 77 do arquivo a ser processado
Filename = oldest_file[32:77]
#Exibe o conteúdo da variável FileName
print(Filename)
Resultado : 001_7XA321_3104_33.09_2.10_1840.parquet
#Armazena os caracteres da posição 0 até 32 do arquivo a ser processado
Filepath = oldest_file[0:32]
#Exibe o conteúdo da variável Filepath
print(Filepath)
Resultado: C:/QualityMonitor/parquet_files\
```

Fonte: Elaborado pelo autor.

3.3.6 Cálculo das regiões de descarte

Após a identificação do arquivo, é necessário fazer a leitura dos dados referente aos sinais relacionados ao equipamento e produto durante o processo. Esses valores estão estruturados em tabelas de texto conforme a extensão *parquet* e precisam ser convertidos para uma forma mais acessível para manipulação dos dados. Essa transformação dos dados é possível também através da construção de *dataframes* com a biblioteca Pandas.

O Pandas possui uma extensão específica para a leitura de arquivos com a extensão parquet, onde é possível construir um *dataframe* de dados direcionando a função para o nome do arquivo parquet (PYDATA, 2023).

Uma vez criada a variável "*Complete_FileName*", contendo o nome do arquivo em processamento, é possível utilizá-la como referência para a função "*ready_parquet*" do Pandas.

Primeiramente, o conteúdo da variável foi replicado para uma nova variável a ser utilizada especificamente neste trecho do código.

Quadro 3 - Armazena o nome dos arquivos e faz a leitura dos dados

```
#Copia os dados da variável “Complete_FileName” para a nova variável path_parquet
path_parquet = Complete_FileName
//
#Cria o dataframe "df" utilizando a extensão "read_parquet" e os dados do arquivo representado pela variável path_parquet
df = pd.read_parquet(path_parquet)
```

Fonte: Elaborado pelo autor.

Figura 23 - Dataframe Principal

Length	ExitThickness	ExitThickness_negLimit	ExitThickness_posLimit
24	1.01311	2.06850	2.13150
25	2.12996	2.06850	2.13150
26	2.41286	2.06850	2.13150
27	2.43046	2.06850	2.13150
28	2.40095	2.06850	2.13150
29	2.39072	2.06850	2.13150
30	2.38983	2.06850	2.13150

Fonte: Elaborado pelo autor.

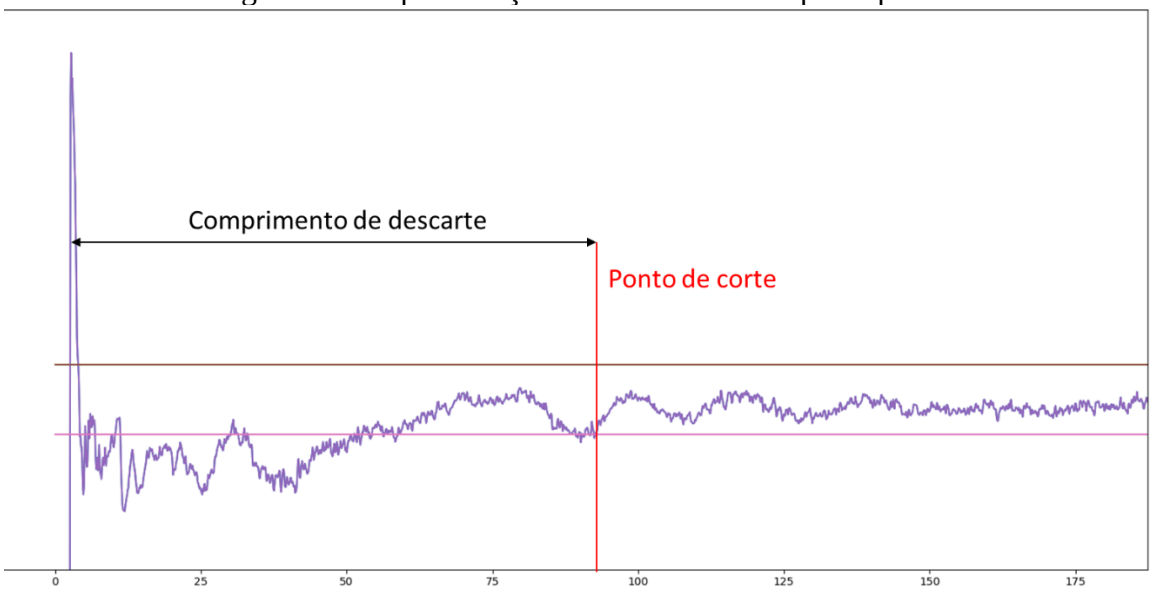
Na Figura 23, é possível ver com detalhes as características do *dataframe*, onde cada coluna representa uma variável (sinal) coletado no processo e cada linha corresponde a um ponto de coleta que, neste caso, visando simplificar a manipulação dos dados e realização dos cálculos, a coleta de dados foi parametrizada de forma que cada ponto represente 1 metro de chapa laminada.

Com o montante de dados estruturados disponíveis, é possível utilizá-los para as fórmulas de cálculo de transiente. Primeiramente, é feito o cálculo dos transientes de espessura na região inicial (aceleração do equipamento até a velocidade de referência) e final (desaceleração até a velocidade de preparação para o próximo processo, sendo essa a velocidade de engate ou zero).

3.3.6.1 Descarte inicial por espessura for a do especificado

Para o transiente inicial, é considerado o ponto de corte como sendo o último ponto onde a variável de controle se deslocou de fora para dentro dos limites, objetivando indicar que do início da laminação até essa região deverá ser descartado para garantir a qualidade do produto.

Figura 24 - Representação do descarte inicial por espessura



Fonte: Elaborado pelo autor.

A Figura 24 apresenta um gráfico de desvio de espessura gerado pela biblioteca Matplotlib, uma biblioteca feita em linguagem Python utilizada para a visualização de dados e plotagem gráfica. Seu objetivo é ser uma alternativa viável de código aberto e plataforma cruzada ao MATLAB. O MATLAB (Matrix Laboratory) é uma plataforma de programação projetada especificamente para engenheiros e cientistas de dados, para analisar e projetar sistemas e produtos (COUTINHO, 2023).

Este exemplo é uma forma visual do conceito de cálculo do descarte, onde foi determinado manualmente o ponto de corte observando o desvio de espessura vs limites e estimar o comprimento observando o valor de comprimento no eixo X do gráfico.

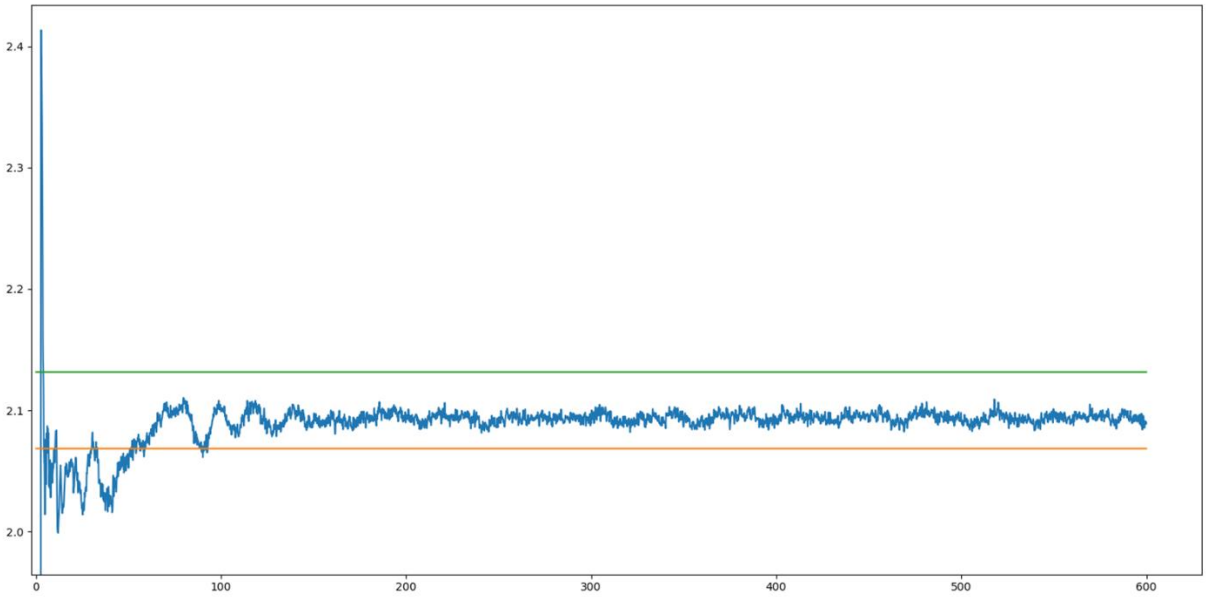
Traduzindo este conceito para o *Python*, o primeiro passo será criar um *dataframe* auxiliar contemplando apenas os sinais envolvidos no cálculo. Em seguida, é necessário filtrar o *dataframe* de modo a considerar apenas a região inicial do produto uma vez que este cálculo é para o descarte inicial.

Quadro 4 - Código para a selecão da região de descarte inicial

```
# Seleciona as colunas de espessura e limites do dataframe original
df_aux_head_Thick = df[[' ExitThickness', index, 'ExitThick_negLimit', 'ExitThick_posLimit']]
#Filtra apenas os valores onde o comprimento (index) é menor do que o limite máximo para o descarte inicial
Max_Head_Discard = 600
df_aux_head_thick = df_aux_head_thick [df_aux_head_thick [index] < Max_Head_Discard]
```

Fonte: Elaborado pelo autor.

Figura 25 - Selecionando a região de descarte inicial



Fonte: Elaborado pelo autor.

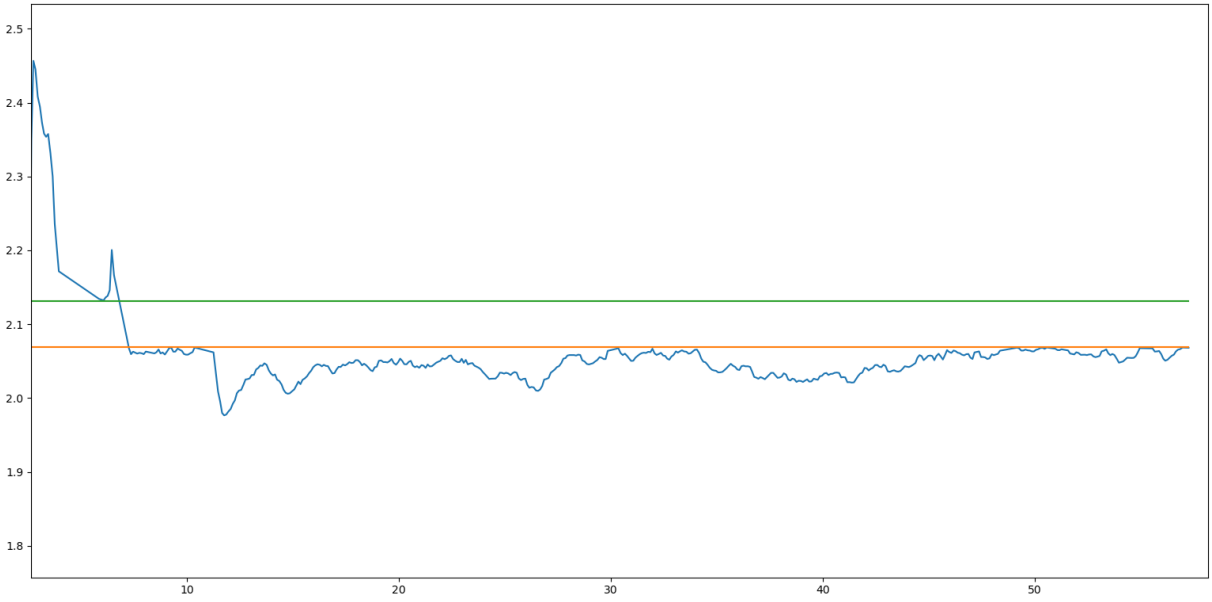
Utilizando o *dataframe* auxiliar, é feita comparação de todos os valores de espessura de saída (*ExitThickness*) com relação aos limites positivo (*ExitThick_posLimit*) e negativo (*ExitThick_negLimit*). Essa comparação visa filtrar o *dataframe* auxiliar "df_aux_head_Thick" contemplando apenas os valores que estiverem fora dos limites.

Quadro 5 - Código para a selecão da região de descarte inicial

```
# Seleciona os valores de espessura que sejam maiores do que o limite positivo OU menores do que o limite negativo
Head_ThickDiscard= df_aux_head_thick[(df_aux_head_thick['ExitThickness'] < df_aux_head_thick['ExitThick_negLimit'])
|
(df_aux_head_thick[' ExitThickness'] > df_aux_head_thick['ExitThick_posLimit'])]
```

Fonte: Elaborado pelo autor.

Figura 26 - Seleção dos valores de espessura fora dos limites



Fonte: Elaborado pelo autor.

Por fim, considerando o conceito exemplificado na figura 21, é capturado o valor de comprimento referente a última posição do vetor filtrando os valores de espessura acima do limite utilizando o recurso "iloc", onde -1 significa que será capturado o valor referente a máxima posição do vetor de comprimento (última posição com valores válidos), referente a

posição ao longo do comprimento do último ponto de medição fora dos limites estabelecidos para descarte.

Quadro 5 - Código para identificação do comprimento do descarte inicial por espessura

```
# Captura o valor da última posição da coluna “index” (comprimento) após aplicação do conceito de cálculo para a região de descarte por espessura
Head_ThickDiscard = Head_ThickDiscard[index].iloc[-1]
Resultado : 57,24
```

Fonte: Elaborado pelo autor.

3.3.6.2 Descarte inicial por temperatura fora do especificado

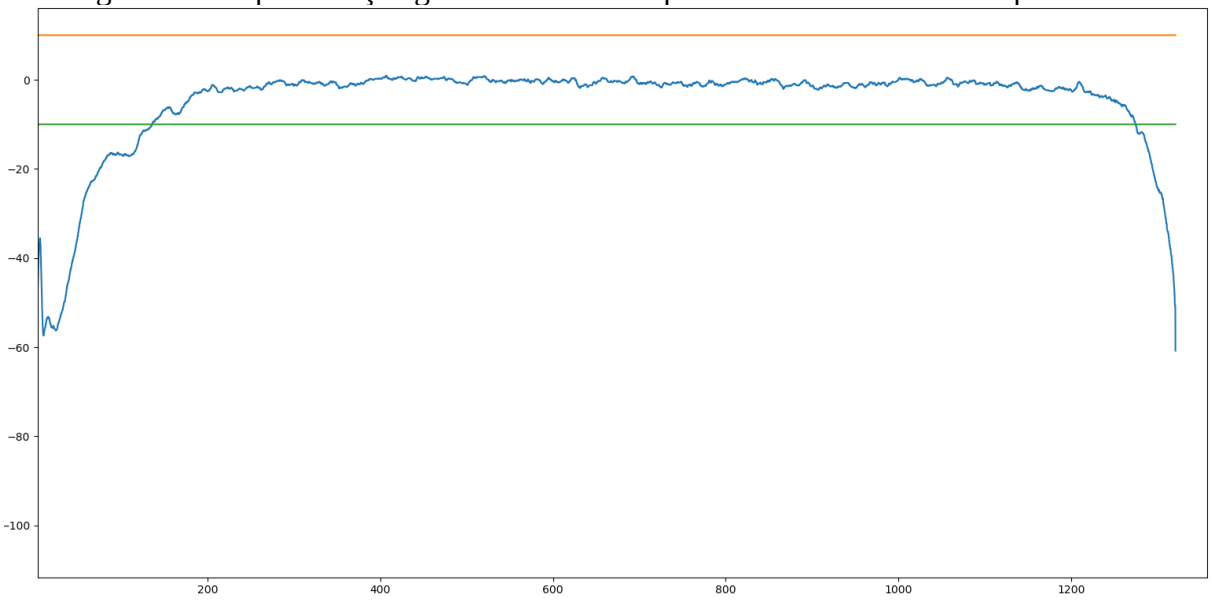
O mesmo conceito é aplicado para o cálculo da temperatura, onde inicialmente é definido o limite de desvio.

Quadro 6 - Definição dos limites e criação de *dataframe* para cálculo do descarte inicial por temperatura

```
# Limite para desvio de temperatura de 10 °C

Strip_Temperature_limit = 10.0

# Cria o dataframe “df_aux_temper” utilizando as colunas “Strip_Temperature_Error” ,“Index” e limites espeficiados do dataframe principal (df)

df_aux_temper = df[['Strip_Temperature_Error',index, 'Strip_Temperature_poslimit', 'Strip_Temperature_neglimit']]
```

Fonte: Elaborado pelo autor.

Figura 27 - Representação gráfica dos valores para descarte inicial de temperatura



Fonte: Elaborado pelo autor.

Da mesma forma como foi feito o descarte por espessura, o cálculo do descarte inicial por temperatura se inicial criando um novo *dataframe* contendo apenas as colunas a serem envolvidas para esse descarte, como se pode observar na representação gráfica da figura 27.

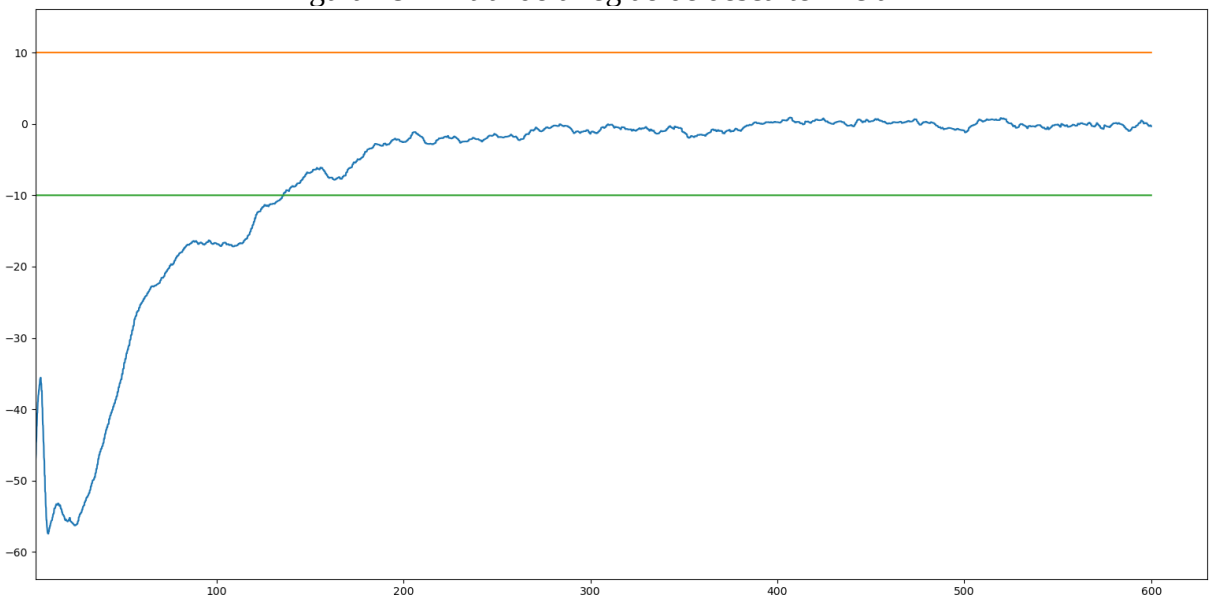
O próximo passo é remover as regiões do produto não pertencentes ao descarte inicial, utilizando novamente o parâmetro de descarte máximo “Max_Head_Discard”.

Quadro 7 – Código para a seleção região para descarte inicial por temperatura

```
#Filtra o dataframe de acordo com o comprimento máximo determinado para o descarte da região inicial

df_aux_temper = df_aux_temper[df_aux_temper[index]<Max_Head_Discard]

#Plota dataframe
plt.figure(4)
plt.plot(df_aux_temper[index], df_aux_temper['Strip_Temperature_Error'], \
df_aux_temper[index], df_aux_temper['Strip_Temperature_poslimit'],\
df_aux_temper[index], df_aux_temper['Strip_Temperature_neglimit'])
plt.plot(4)
```

Fonte: Elaborado pelo autor.

Figura 28 - Filtrando a região de descarte inicial



Fonte: Elaborado pelo autor.

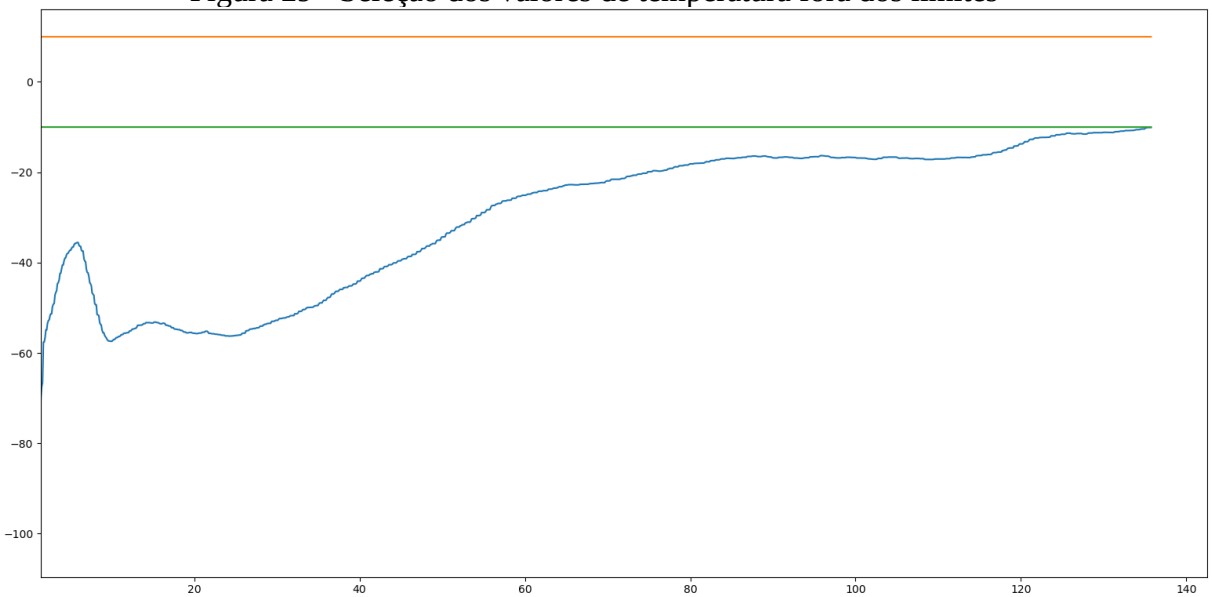
A partir deste ponto, é criado o *dataframe* “Head_TemperDiscard” contendo apenas as linhas contendo valores de desvio de temperatura acima do especificado.

Quadro 8 - Código para seleção dos valores de erro de temperatura fora dos limites especificados

```
# Seleciona os valores de temperatura que sejam maiores do que o limite positivo OU
menores do que o limite negativo

Head_TemperDiscard = df_aux_temper[(df_aux_temper['Strip_Temperature_Error'] <
df_aux_temper['Strip_Temperature_neglimit'])
 | (df_aux_temper['Strip_Temperature_Error'] >
df_aux_temper['Strip_Temperature_poslimit'])]
```

Fonte: Elaborado pelo autor.

Figura 29 - Seleção dos valores de temperatura fora dos limites



Fonte: Elaborado pelo autor.

Por fim, novamente é capturado o valor de comprimento (index) referente a última posição do vetor onde o desvio de temperatura ficou fora dos limites especificados.

Quadro 9 - Código para identificação do comprimento do descarte inicial por temperatura

Captura o valor da última posição da coluna "*index*" (comprimento) após aplicação do conceito de cálculo para a região de descarte por temperatura

Head_ThickDiscard = Head_TemperDiscard[index].iloc[-1]

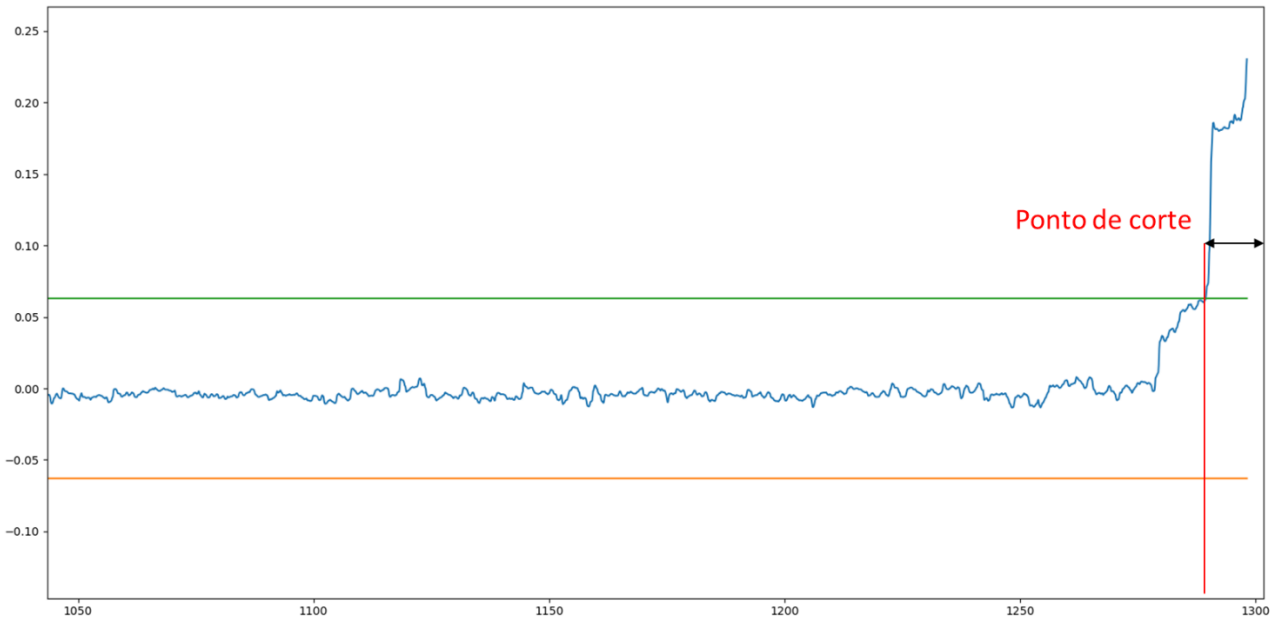
Resultado : 135,74

Fonte: Elaborado pelo autor.

3.3.6.3 Descarte final por espessura fora do especificado

Para o transiente final, é considerado o ponto de corte como sendo o primeiro ponto onde a variável de controle se deslocou de fora para dentro dos limites considerando o limite máximo de descarte desta região, considerando que para garantir a qualidade do material, todo o comprimento restante a partir do primeiro ponto de medição fora do especificado deverá ser descartado.

Figura 30 - Representação do descarte final por espessura



Fonte: Elaborado pelo autor.

A Figura 30 representa graficamente o *dataframe* criado para o descarte final, seguindo o mesmo conceito do descarte inicial onde foram filtrando apenas os valores a serem utilizados neste cálculo.

O passo seguinte consiste em separar a região de descarte final, que desta vez será de forma diferente. Para o descarte inicial basta parametrizar um valor absoluto como limite máximo tendo em vista que o ponto de partir é a posição zero (comprimento = 0), desta vez, é necessário considerar que o comprimento de cada produto pode variar e o ponto de início da verificação também sofrerá alterações.

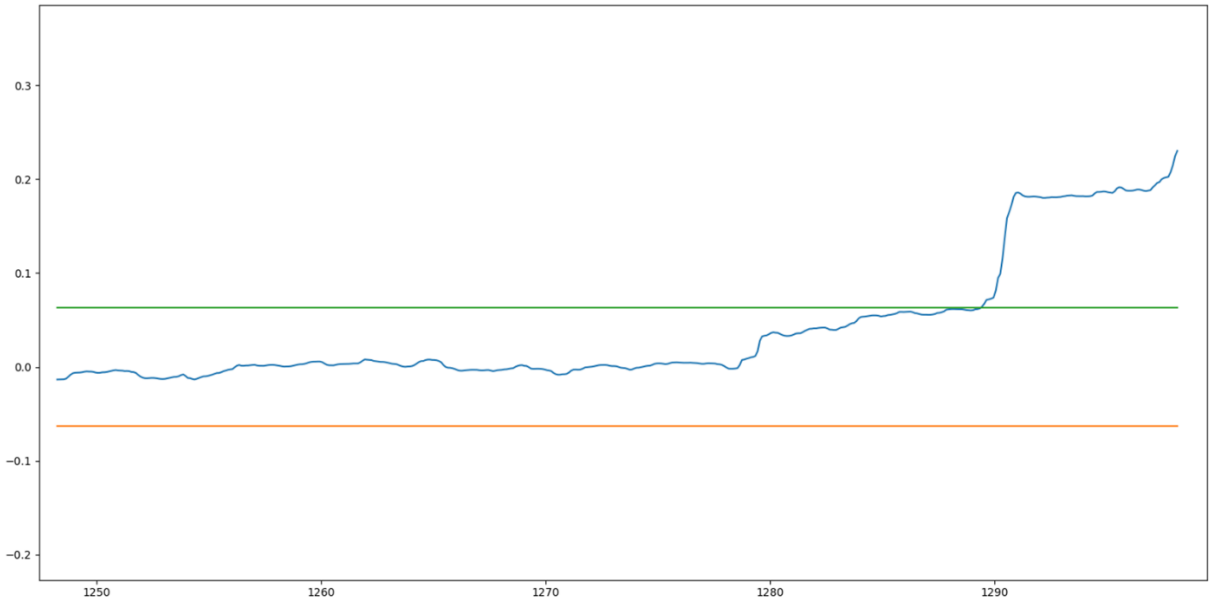
Em resumo, é utilizado um valor padrão para comprimento máximo de descarte e um cálculo de comprimento máximo da bobina. Com estes 2 dados, é possível determinar o ponto de partida para a região do descarte final.

Quadro 10 - Código para determinar a região de descarte final por espessura

```
# Cria dataframe derivado de df para cálculo do transiente de desaceleração do equipamento
df_aux_tail = df[['Exit_ThicknessDev','ExitThick_negLimit', 'ExitThick_posLimit',index]]
# Determina o comprimento máximo para o descarte da região final
Max_Tail_Discard = 50
//
# Cálculo do comprimento máximo capturando a última posição do vetor "index" (comprimento)
Max_Exit_Length = df[index].iloc[-1]
//
# Cálculo da posição de início do descarte subtraindo o valor máximo de comprimento da bobina pelo máximo comprimento de descarte e filtra o vetor a partir desta posição.
df_aux_tail = df_aux_tail[df_aux_tail[index] > (Max_Exit_Length - Max_Tail_Discard)]
```

Fonte: Elaborado pelo autor.

Figura 31 - Gráfico com a região de descarte final por espessura



Fonte: Elaborado pelo autor.

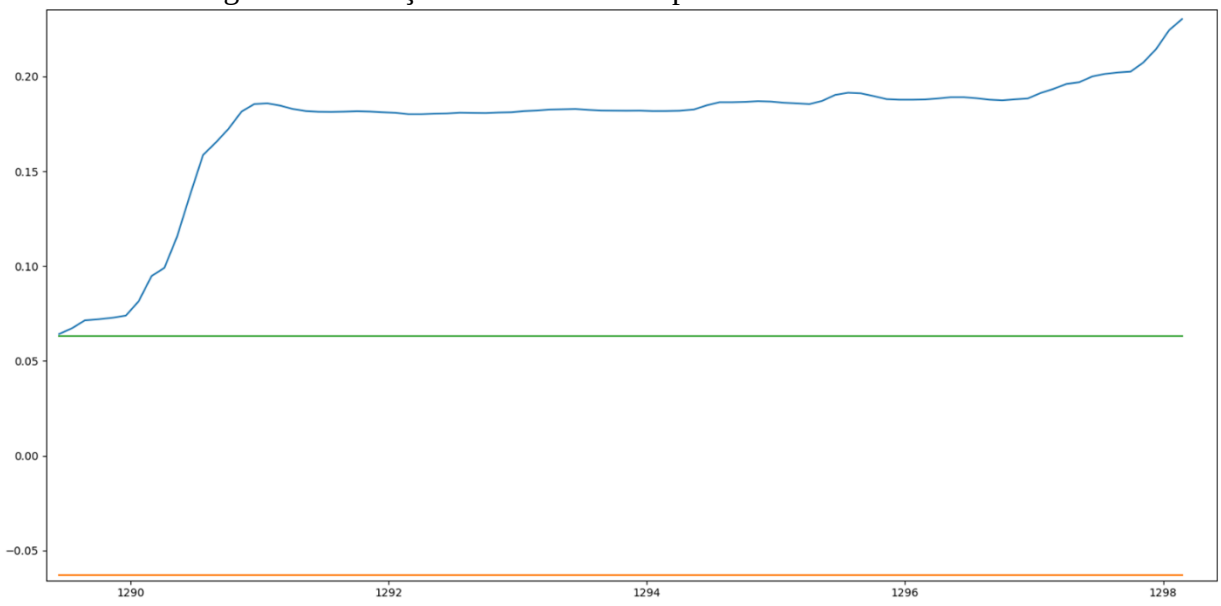
A partir deste ponto, é criado um novo *dataframe* filtrando apenas os valores desta região que estão fora dos limites especificados para o produto.

Quadro 11 - Código para seleção dos valores fora dos limites especificados para a região de descarte final por espessura

```
//
# Seleciona os valores de espessura que sejam maiores do que o limite positivo OU menores do que o limite negativo
Tail_ThickDiscard = df_aux_tail[(df_aux_tail['Exit_ThicknessDev']<df_aux_tail['ExitThick_negLimit']) | (df_aux_tail['Exit_ThicknessDev']>df_aux_tail['ExitThick_posLimit'])]
//
```

Fonte: Elaborado pelo autor.

Figura 32 - Seleção dos valores de espessura fora dos limites



Fonte: Elaborado pelo autor.

Por fim, novamente é capturado o valor de comprimento (index). Porém, para o transiente da região final é considerada a primeira posição do *dataframe* filtrado, uma vez que o descarte será feito a partir dessa posição.

Quadro 12 - Código para determinar o comprimento do descarte final por espessura

```
//
# Captura o valor da primeira posição (posição zero) da coluna "index" (comprimento) após aplicação do conceito de cálculo para a região de descarte final por espessura
Start_TailThickDiscard = Tail_ThickDiscard[index].iloc[0]
Resultado: 1289
//
#Determina o comprimento do descarte final por espessura subtraindo a posição do defeito pelo comprimento máximo da bobina
Tail_ThickDiscard = Max_Exit_Length - Start_TailThickDiscard
Resultado : 8,7
```

Fonte: Elaborado pelo autor.

3.3.6.4 Descarte final por temperatura fora do especificado

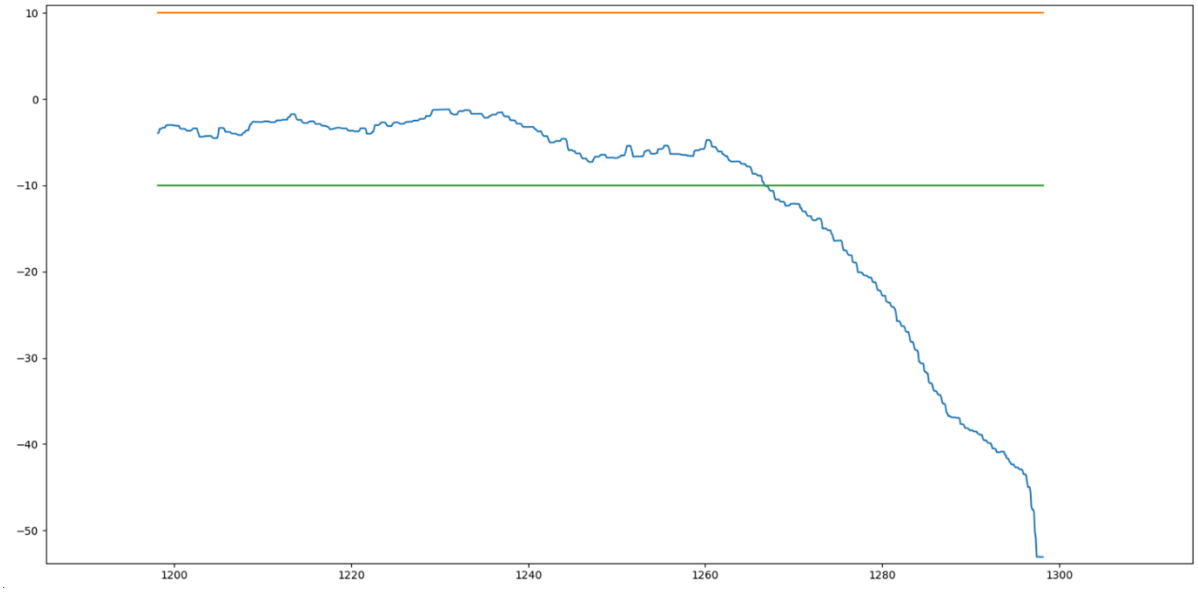
Seguindo a mesma metodologia do cálculo do transiente de espessura na região final, para calcular o transiente de temperatura é feita a criação de um novo *dataframe* derivado do *dataframe* principal filtrando apenas as colunas referentes aos sinais de temperatura e limites especificados para essa variável e filtrando a região estabelecida para este descarte.

Quadro 13 - Código para determinar a região de descarte final por temperatura

```
#Cria dataframe dedicado para o cálculo do transiente de temperatura para a região final
df_aux_temper_tail = df[['Strip_Temperature_Error',index, 'Strip_Temperature_poslimit', 'Strip_Temperature_neglimit']]
# Filtra o dataframe para a região especificada para o descarte
df_aux_temper_tail = df_aux_temper_tail[df_aux_temper_tail[index] > (Max_Exit_Length - Max_Tail_Discard)]
```

Fonte: Elaborado pelo autor.

Figura 33 - Filtrando a região de descarte final por temperatura



Fonte: Elaborado pelo autor.

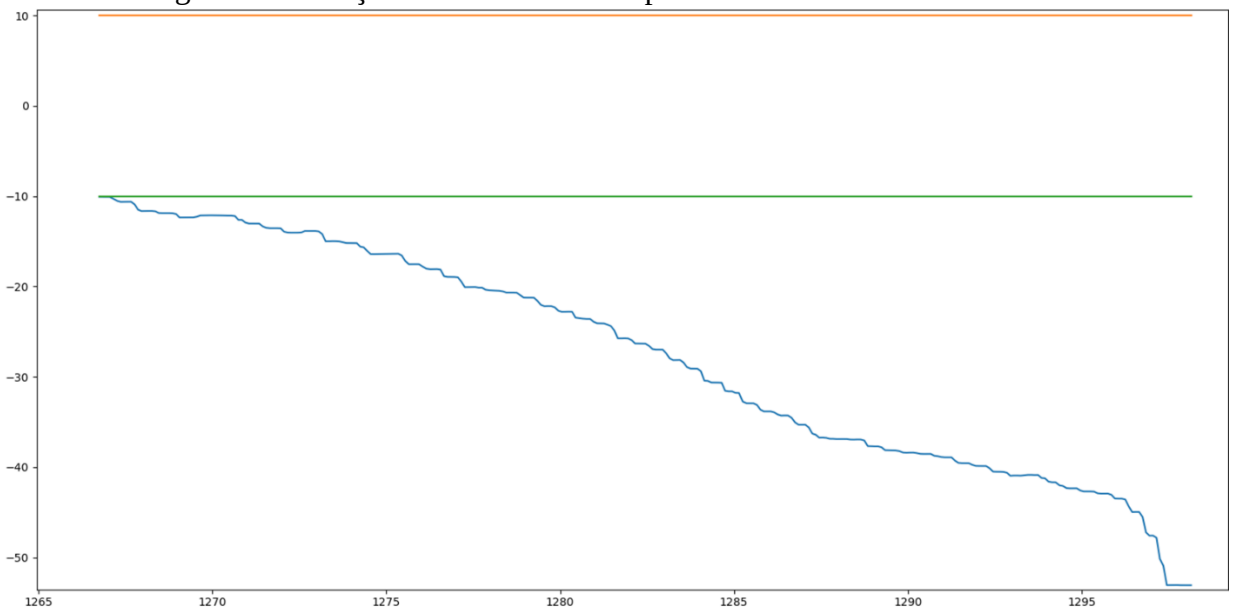
O próximo passo é criar um novo *dataframe* filtrando apenas os valores desta região que estão fora dos limites especificados para o produto.

Quadro 14 - Código para seleção dos valores fora dos limites especificados para a região de descarte final por temperatura

```
# Seleciona os valores de temperatura que sejam maiores do que o limite positivo OU menores do que o limite negativo
Tail_TempDiscard =
df_aux_temper_tail[(df_aux_temper_tail['Strip_Temperature_Error']
<df_aux_temper_tail['Strip_Temperature_neglimit']) |
(df_aux_temper_tail['Strip_Temperature_Error']>
df_aux_temper_tail['Strip_Temperature_poslimit'])]
```

Fonte: Elaborado pelo autor.

Figura 34 - Seleção dos valores de temperatura final fora dos limites



Fonte: Elaborado pelo autor.

Analogamente ao descarte final por espessura, é capturado o valor da primeira posição (posição zero) do vetor da coluna “index”, referente ao comprimento do produto.

Quadro 15 - Código para determinar o comprimento do descarte final por temperatura

```
//
#Captura o valor da primeira posição (posição zero) da coluna “index” (comprimento) após aplicação do conceito de cálculo para a região de descarte final por temperatura
Start_TailTempDiscard = Tail_TempDiscard[index].iloc[0]
Resultado : 1266
//
#Determina o comprimento do descarte final por temperatura subtraindo a posição do defeito pelo comprimento máximo da bobina
Tail_TempDiscard = Max_Exit_Length - Start_TailTempDiscard
Resultado : 31
```

Fonte: Elaborado pelo autor.

3.3.7 Cálculo do comprimento final após descartes

Após a determinação dos descartes final e inicial, é calculado o comprimento válido do produto que será destinado para a próxima etapa do processo. Essa variável é fundamental para a tomada de decisões do modelo de acordo com as especificações de cada cliente.

Quadro 16 - Código para cálculo do comprimento final eliminando as regiões de descarte

```
#Calcula o comprimento total do descarte inicial estabelecendo o maior valor entre os descartes por espessura e temperatura
Initial_Disc = max(Head_TemperDiscard,Head_ThickDiscard)
//
#Calcula o comprimento total do descarte final estabelecendo o maior valor entre os descartes por espessura e temperatura
Final_Disc = max(Tail_TemperDiscard,Tail_TemperDiscard)
//
# Calcula o comprimento válido a ser utilizado na próxima operação subtraindo a região de descarte do comprimento total laminado
Valid_Length = Max_Exit_Length - Initial_Disc - Final_Disc
```

Fonte: Elaborado pelo autor.

3.3.8 Armazenamento de dados

3.3.8.1 Construindo o banco de dados

O gerenciador de banco de dados utilizando para este trabalho é o software livre PostgressSQL, baseado na linguagem SQL. Inicialmente, é criado um banco de dados sem configuração chamado “QualityMonitor”, utilizando o script exposto no Quadro 17:

Quadro 17 - Script para a criação do banco de dados

```
-- Create Database: "QualityMonitor"

-- DROP DATABASE "QualityMonitor";

CREATE DATABASE "QualityMonitor"
  WITH OWNER = postgres
      ENCODING = 'UTF8'
      TABLESPACE = pg_default
      LC_COLLATE = 'Portuguese_Brazil.1252'
      LC_CTYPE = 'Portuguese_Brazil.1252'
      CONNECTION LIMIT = -1;
```

Fonte: Elaborado pelo autor.

Posteriormente, são criadas as tabelas “clientes” e “resultado”.

3.3.8.2 Tabela “clientes”

Na tabela clientes serão armazenadas as especificações básicas de todos os clientes, incluindo a liga, largura final, espessura target após a etapa de laminação e temperatura durante a laminação.

Essa tabela será utilizada como referência para a tomada de decisão do modelo após os cálculos de região de transiente e comprimento final. A tabela clientes é criada utilizando o script do Quadro 18:

Quadro 18 - Script para a criação da tabela clientes

```
-- Cria a tabela clientes
-- DROP TABLE clientes;
CREATE TABLE clientes
(nome text, largura numeric, espessura_target numeric,liga numeric,
comprimento_min real)
WITH ( OIDS=TRUE );
ALTER TABLE clientes
  OWNER TO postgres;
```

Fonte: Elaborado pelo autor.

Neste caso, é utilizando o comando “create table” especificando o nome da tabela transferindo como parâmetro os nomes das colunas desejadas seguido do formado, onde neste caso, foi definido o formato de texto para os nomes dos clientes e o restante como formato *numeric* (numérico). Nesta tabela, também foi ativado o recurso de OIDS (*objects indentifiers*) que serão utilizados como chaves primárias possibilitando a eventual conexão entre várias tabelas. Em seguida, são inseridos os dados dos clientes de forma manual utilizando os arquivos de referência, como por exemplo, o contrato de especificação entre a empresa de laminação e o cliente final.

Figura 35 - Trecho da tabela clientes

oid	nome text	largura numeric	espessura_target numeric	liga numeric	comprimento_min real
49489	M	1100	3.5	80791	150
49490	N	2000	10	5052	200
49491	O	1630	4	11001	500
49492	P	1530	5	5052	150
49493	Q	1530	4.2	5052	250
49494	R	1530	3.8	5052	400
49495	S	1640	4	5052	350
49496	T	1620	3	5052	150
49498	U	1600	3.5	80791	150

Fonte: Elaborado pelo autor.

3.3.8.3 Tabela “resultado”

Do mesmo modo, é criada a tabela para armazenar os resultados dos cálculos de transiente, comprimento final, especificações dos produtos processados e saída do modelo de tomada de decisão. Além de ser utilizada como histórico de produção, essa tabela também será utilizada para o treinamento contínuo do modelo.

Quadro 19 - Script para a criação da tabela resultado

```
-- Criação da tabela "resultado"
-- DROP TABLE resultado;
CREATE TABLE resultado
(
  ot numeric NOT NULL,
  espessura numeric,
  liga numeric,
  largura numeric,
  comprimento_final numeric,
  descarte_espessura numeric,
  descarte_temperatura numeric,
  resultado_final character(50),
  CONSTRAINT ot PRIMARY KEY (coilid)
)
WITH (
  OIDS=TRUE
);
ALTER TABLE resultado
  OWNER TO postgres;
```

Fonte: Elaborado pelo autor.

Para essa tabela, foram determinadas algumas características cruciais para o bom funcionamento do sistema, como por exemplo o parâmetro "*not null*" para a coluna coilid, impedindo que sejam armazenados dados nulos ou vazios no campo de identificação do lote produzido, umas vez que esses dados será utilizado como referência para a consulta dos dados a serem utilizados como PDI (*Process Data Input*) para o próximo equipamento que realizará o descarte das regiões de transiente e o ajuste de largura necessários para o cliente final.

Uma outra característica importante dessa tabela é o campo "final_result", no formato de texto com um tamanho máximo de 50 caracteres, onde serão estabelecidas as classes do modelo.

Figura 36 - Trecho da tabela resultado

oid	coilid [PK] numeric	targetthickness_mm numeric	alloy numeric	targetwidth_mm numeric	validlength_m numeric	head_thickdiscard_m numeric	head_temperdiscard_m numeric	finaldiscard_m numeric	final_result character(50)
49534	971304	3.5	80791	1100	656.649169921875	35.653877	100	100	Seguir processo para o cliente M
49533	971512	3.5	80791	1100	669.0496215820312	8.149864	100	100	Seguir processo para o cliente M
49532	971513	3.5	80791	1100	654.0503540039062	12.450636	100	100	Seguir processo para o cliente M
49531	971575	3.5	80791	1100	655.3574829101562	11.555463	100	100	Seguir processo para o cliente M
49530	971767	3.5	80791	1100	660.3526611328125	11.654041	100	100	Seguir processo para o cliente M
49516	971831	3.5	80791	1100	446.5509033203125	62.55188	100	100	Seguir processo para o cliente M
49382	971946	2.1	3104	1780	1104.6007080078125	98.05733	144.75073	144.75073	Seguir processo para o cliente A
49529	972017	3.5	80791	1100	657.553466796875	37.951817	100	100	Seguir processo para o cliente M
49441	972936	2.6	518213	1565	702.1012878417969	51.050987	177.95096	177.95096	Seguir processo para o cliente E
49446	973214	2.6	518213	1565	806.6019592285156	54.649063	206.15146	206.15146	Seguir processo para o cliente E
49440	974656	2.6	518213	1565	857.998779296875	51.851788	187.35132	187.35132	Seguir processo para o cliente E
49524	976176	4.2	5052	1530	554.8530883789062	9.54819	100	100	Seguir processo para o cliente Q
49439	977790	2.6	518213	1565	858.2975311279297	55.25101	218.45345	218.45345	Seguir processo para o cliente E
49448	977793	2.6	518213	1565	702.8993835449219	36.85079	207.55215	207.55215	Seguir processo para o cliente E
49415	977947	2.1	3104	1890	1086.6031646728516	97.650894	185.05602	185.05602	Seguir processo para o cliente B
49395	977959	2.1	3104	1840	1150.8025970458984	59.658276	146.16066	146.16066	Seguir processo para o cliente C
49443	978070	2.6	518213	1565	595.2994384765625	52.951366	206.2528	206.2528	Sucatar
49459	978137	2.6	518213	1565	795.8013763427734	22.652914	173.25172	173.25172	Seguir processo para o cliente E
49397	978221	2.1	3104	1840	906.0922546386719	60.254627	393.35538	393.35538	Seguir processo para o cliente C
49442	978281	2.6	518213	1565	576.6001586914062	52.45135	197.35052	197.35052	Sucatar
49271	978514	2.1	3104	1780	1273.9351806640625	98.25402	112.04843	112.04843	Seguir processo para o cliente A

Fonte: Elaborado pelo autor.

3.3.9 Gestão dos arquivos processados

Após processamento e armazenamento dos dados, é necessário classificar os arquivos já processados e enviá-los para um outro diretório para garantir que não haverá redundância no processamento dos e, em caso de necessidade, possibilitar o reprocessamento de maneira simples e rápida.

Nesta etapa, novamente é utilizada a estrutura criada para a identificação dos arquivos a serem processados contido no diretório destinado ao sistema, as variáveis “Filename” contendo o nome completo do arquivo e “Fileextension” contendo a extensão do arquivo.

Com essa variável já criada no início do programa, neste momento é necessário criar uma outra variável contendo a descrição do diretório de destino dos arquivos processados. Lembrando que esse diretório deve ser primeiramente criado no servidor seguindo o procedimento padrão de criação de diretórios do sistema operacional utilizado.

Com esses dados, é possível criar uma nova variável de texto originária da junção das variáveis já existentes e também incluir textos adicionais conforme a necessidade. Neste caso, a variável criada é a somatória das variáveis “Processed_Filepath”, “Filename”, “Fileextension” e do texto “_processed” para identificar através do nome que o arquivo foi processado.

Com a nova variável contendo todas as informações do arquivo, é novamente utilizada a biblioteca OS contendo recursos padrão do sistema operacional. Neste caso, é utilizada a função “rename” para alterar o nome completo do arquivo para o novo nome. Considerando o

fato de que o nome completo também contempla o diretório do arquivo, desta forma, ao renomear a variável completa, o arquivo é automaticamente transferido para o novo diretório.

Quadro 20 - Código para gerenciamento de arquivos processados

```
#Cria variável contendo o nome da pasta para armazenamento dos arquivos já processados
Processed_Filepath = 'C:/QualityMonitor/processed_files\\'
//
#Cria variável contendo o diretório de destino e nome completo do arquivo adicionando o status de arquivo já processado
Executed_FileName = Processed_Filepath+Filename+'_Processed'+Fileextension
Resultado:
'C:/QualityMonitor/processed_files\\04_3XA21_1100_34.40_4.00_1630_Processed.parquet'
//
#Renomeia o arquivo original para o novo nome e diretório do arquivo, transferindo o arquivo com o status “processed” para a pasta “processed_files”
os.rename(Complete_FileName, Executed_FileName)
```

Fonte: Elaborado pelo autor.

3.3.10 Ingestão e consulta de dados

Após a determinação das regiões de descarte é necessário armazenar os resultados em um banco de dados para garantir o histórico de produção e também, para este trabalho, utilizar o maior número de entradas possíveis para o treinamento do modelo.

Como mencionado nos capítulos anteriores, o banco utilizado é o PostgressSQL. Adicionalmente, é necessária a execução de uma tarefa para gerenciar as informações que terão interface com o banco, com as funções tanto de gravação quanto leitura, para isto, é utilizada a biblioteca “psycopg2” (Psycopg Organization, 2023).

A biblioteca psycopg2 é uma biblioteca que oferece um *framework* para a conexão com o servidor do PostgreSQL, ou seja, uma certa interface entre o que está acontecendo internamente na sua máquina e o programador. Inicialmente, é necessário importar a biblioteca para o projeto e estabelecer a conexão com o banco de dados desejado.

Quadro 21 - Código para a criação da conexão com o banco de dados

```
#Importa a biblioteca psycopg2 para o ambiente de trabalho
import psycopg2
#Cria a conexão com o banco de dados e um cursor para navegar entre as tabelas onde nos parâmetros host, database, user e password são inseridos, respectivamente, o endereço do servidor do banco de dados, o nome do banco, usuário e senha de acesso.
connection = psycopg2.connect(
    host="localhost",
    database="QualityMonitor",
    user="postgres",
    password="********")
#Cria o objeto cursor
cursor = connection.cursor()
```

Fonte: Elaborado pelo autor.

Uma vez criada a conexão, o próximo passo é criar o comando com as instruções de quais dados serão armazenados e a tabela de destino. Primeiramente, é necessário informar a o tipo da operação desejada (neste caso, é utilizado o comando “insert into” com o objetivo de inserir dados na tabela de destino) seguido das colunas em que serão inseridos os dados e da instrução complementar “values”, contemplando as extensões das respectivas colunas da tabela a ser preenchida (Psycopg Orgnization, 2023).

Em seguida, são declaradas as variáveis do programa a serem armazenadas em casa coluna, sendo necessário manter a mesma ordem de declaração contida na instrução “results_insert_query” para garantir que haja convergência entre os valores das variáveis do código e o banco de dados.

Um vez declaradas as 2 variáveis de inserção, é executada a instrução “execute”, uma extensão da função “connection” para executar ações de gravação e leitura no banco de dados, seguido do comando “commit” para efetivar a transação em andamento.

Quadro 22 - Código para a inserção de dados na tabela resultado

```
# Lista as colunas a serem preenchidas da tabela resultado
results_insert_query = INSERT INTO resultado (ot, espessura, liga, largura, comprimento_final, descarte_espessura, descarte_temperatura, descarte_final, resultado_final) VALUES (%s,%s,%s,%s, %s,%s,%s,%s)
#Lista as variáveis a serem armazenadas no banco, seguindo a mesma ordem das respectivas colunas de destino
record_to_insert=(CoilID,F4_ExitThickness_Target,Alloy,Strip_Width_Target,
Max_Exit_Length,Head_ThickDiscard,Head_TemperDiscard,Final_Disc,
Aval_Final)
# Executa os comandos para inserção de dados na tabela resultado
Cursor.execute (results_insert_query, record_to_insert)]
#Efetivação do comando em andamento
connection.commit()
```

Fonte: Elaborado pelo autor.

3.4 CONSTRUÇÃO DO MODELO

Para este trabalho foram pesquisadas algumas opções de técnicas de machine learning possíveis e de acordo com o escopo e características do sistema, foi selecionado o modelo de árvore de decisão uma vez que já temos as variáveis de especificação bem definidas e uma base de dados histórica com decisões já tomadas baseadas nestas especificações.

Analisando o modelo de árvores de decisão, encontra-se 2 métodos principais, a árvore de decisão de classificação e a árvore de decisão de regressão e 2 tipos de atributos, o atributo do tipo numérico ou ordenado onde assume-se valores reais como peso, idade, altura etc. E o atributo categórico assumindo valores em um conjunto finito, onde não há nenhuma ordem natural como, por exemplo, uma cor (LAURETTO, 2010).

A variável *target* (atributo) é quem define o tipo de árvore a ser utilizada (WANG, Y. & WITTEN, 1997). Deste modo, tendo como variável *target* um atributo categórico, foi selecionado o modelo de árvore de decisão classificatória. Para este propósito, será utilizada a função “RegressionTreeClassifier” pertencente da biblioteca Sktlearn.

3.4.1 Filtragem de dados

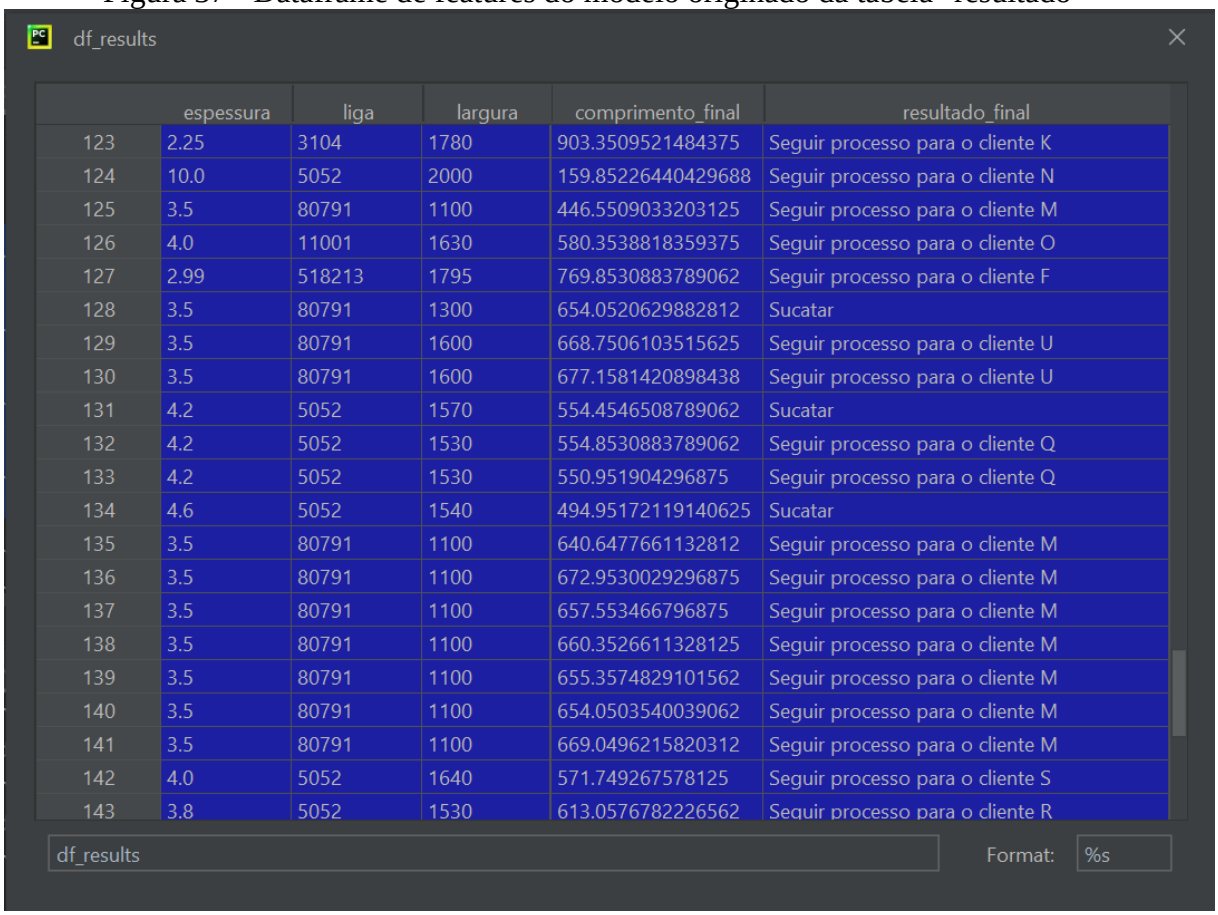
A implementação do modelo se inicia com a filtragem de dados para remover outliers (pontos fora da média) e entrada sem correlação direta para mitigar “ruídos” que possam impactar na performance do modelo. Neste caso, um novo *dataframe* contendo apenas os dados de liga, espessura, largura e comprimento válido que serão utilizadas como *features* do modelo.

Quadro 24 - Código para filtragem do dataframe “df_results”

```
#Constrói dataframe "df_results" contendo todos os dados da tabela resultado
df_results = pd.DataFrame (results_data, columns = ['ot', 'espessura', 'liga', 'largura', 'comprimento_final','descarte_espessura','descarte_temperatura','descarte_final', 'resultado_final'])
#Remove coluna "OT"
df_results = df_results.drop("ot", axis=1)]
#Remove coluna "descarte_espessura"
df_results = df_results.drop("descarte_espessura", axis=1)
#Remove coluna "descarte_temperatura"
df_results = df_results.drop("descarte_temperatura", axis=1)
#Remove coluna "descarte_final"
df_results = df_results.drop("descarte_final", axis=1)
```

Fonte: Elaborado pelo autor.

Figura 37 - Dataframe de features do modelo originado da tabela "resultado"



	espessura	liga	largura	comprimento_final	resultado_final
123	2.25	3104	1780	903.3509521484375	Seguir processo para o cliente K
124	10.0	5052	2000	159.85226440429688	Seguir processo para o cliente N
125	3.5	80791	1100	446.5509033203125	Seguir processo para o cliente M
126	4.0	11001	1630	580.3538818359375	Seguir processo para o cliente O
127	2.99	518213	1795	769.8530883789062	Seguir processo para o cliente F
128	3.5	80791	1300	654.0520629882812	Sucatar
129	3.5	80791	1600	668.7506103515625	Seguir processo para o cliente U
130	3.5	80791	1600	677.1581420898438	Seguir processo para o cliente U
131	4.2	5052	1570	554.4546508789062	Sucatar
132	4.2	5052	1530	554.8530883789062	Seguir processo para o cliente Q
133	4.2	5052	1530	550.951904296875	Seguir processo para o cliente Q
134	4.6	5052	1540	494.95172119140625	Sucatar
135	3.5	80791	1100	640.6477661132812	Seguir processo para o cliente M
136	3.5	80791	1100	672.9530029296875	Seguir processo para o cliente M
137	3.5	80791	1100	657.553466796875	Seguir processo para o cliente M
138	3.5	80791	1100	660.3526611328125	Seguir processo para o cliente M
139	3.5	80791	1100	655.3574829101562	Seguir processo para o cliente M
140	3.5	80791	1100	654.0503540039062	Seguir processo para o cliente M
141	3.5	80791	1100	669.0496215820312	Seguir processo para o cliente M
142	4.0	5052	1640	571.749267578125	Seguir processo para o cliente S
143	3.8	5052	1530	613.0576782226562	Seguir processo para o cliente R

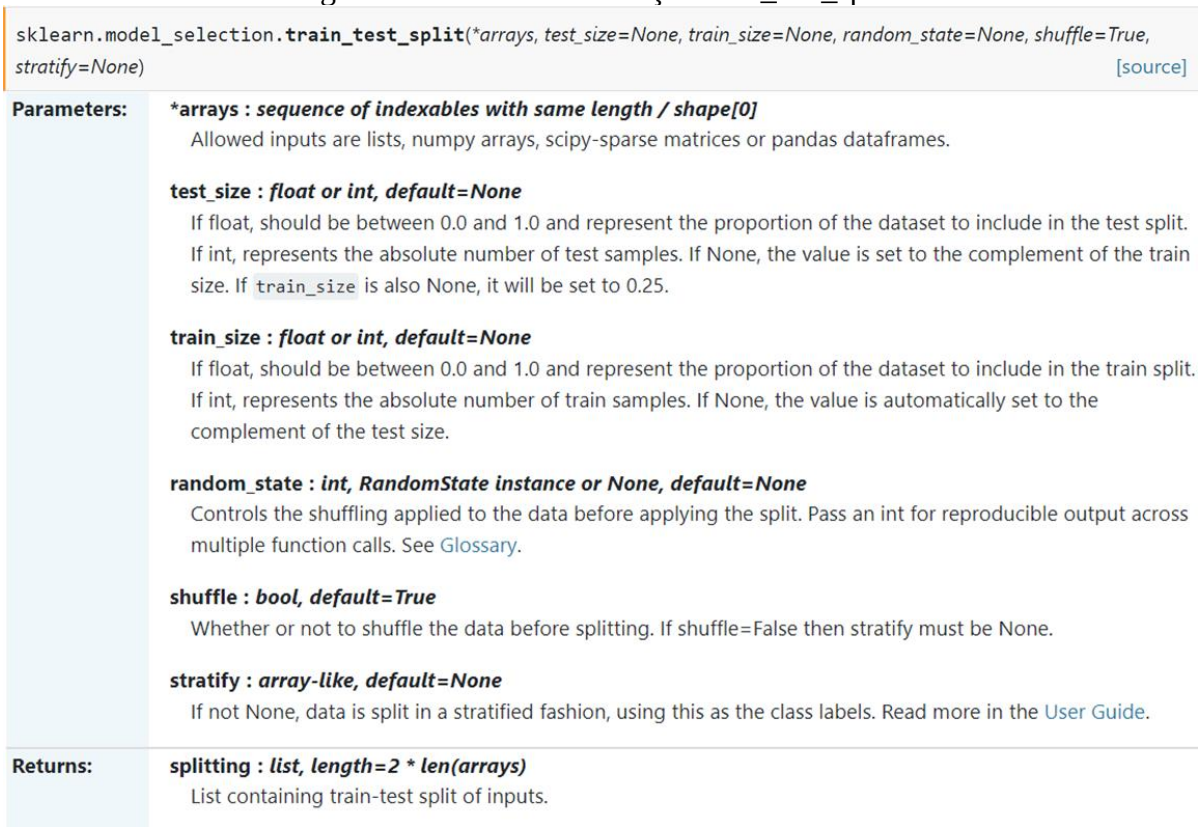
Fonte: Elaborado pelo autor.

3.4.2 Separação de dados (treinamento e teste)

Um dos principais aspectos de um sistema em *machine learning* supervisionado consiste na análise e validação do modelo. Quando se avalia a performance preditiva do modelo, é essencial que o processo seja imparcial. Usando a ferramenta “train_test_split” da biblioteca Scikit-Learn, é possível separar a base de dados em dois conjuntos que irão minimizar o potencial de parcialidade no processo de análise e validação do modelo (STOJILJKOVIĆ, 2023).

A Figura 36 mostra os detalhes da estrutura da ferramenta e a descrição de casa parâmetro de entrada e saída.

Figura 38 - Estrutura da função train_test_split



`sklearn.model_selection.`**`train_test_split`**(**arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None*) [source]

Parameters:

arrays : *sequence of indexables with same length / shape[0]
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

train_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

random_state : *int, RandomState instance or None, default=None*
Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls. See Glossary.

shuffle : *bool, default=True*
Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None.

stratify : *array-like, default=None*
If not None, data is split in a stratified fashion, using this as the class labels. Read more in the User Guide.

Returns:

splitting : *list, length=2 * len(arrays)*
List containing train-test split of inputs.

Fonte: Adaptado de scikit-learn.org (2023).

Inicialmente, é necessário importar a função train_test_split da biblioteca scikit-learn. Após esse passo é necessário determinar os arrays (vetores) de entrada e saída do modelo que serão utilizados para a separação dos dados de teste e treinamento. Nest caso, serão convencionados os vetores x (entrada) e y (saída), originados do *dataframe* "df_results" contendo os dados da tabela resultado.

Em seguida, é feita a separação destes *dataframes* em conjuntos de teste e treinamento tanto para as entradas quanto para a saída do modelo.

Neste trabalho foi informado o parâmetro "size=0.33" determinando que 33% do montante de dados total serão destinados para o conjunto de testes (X_test e y_test), sendo os outros 67% dos dados utilizados para treinamento. Já o parâmetro "*random_state=1*", determina que seja feito um embaralhamento dos dados antes da separação para reduzir potenciais ocorrências de parcialidade ou *overfitting* do modelo.

Quadro 23 - Código para criação dos conjuntos de dados de teste e treinamento

```
#Importação da train-test-split
from sklearn.model_selection import train_test_split

#Cria o dataframe "x" derivado de df_results excluindo a coluna "resultado_final"
x = df_results.drop("resultado_final",axis=1)

#Cria o dataframe "x" derivado de df_results selecionando apenas a coluna "resultado_final"
y = df_results['resultado_final']

#Cria os conjuntos de dados de teste e treinamento derivados dos dataframes x e y
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.33, random_state=1)
```

Fonte: Elaborado pelo autor.

3.4.3 Construindo a árvore de decisão classificatória

Com os dados de teste e treinamento já preparados, é feita a importação da árvore de decisão de classificação, uma função também derivada da biblioteca sklearn, subconjunto "tree" (árvore). Neste trabalho, a árvore criada será atrelada a variável "clf".

Com a árvore de decisão já criada, serão inseridos os *dataframes* contendo os dados de teste e treinamento criados no tópico anterior para de fato concretizar a criação da versão inicial árvore de decisão utilizando o comando "fit" destinado para essa função (Scikit-Learn.org, 2023).

Neste momento a árvore foi criada utilizando os parâmetros padrão, que serão otimizados de modo a buscar a maior performance de predição do modelo. Utilizando a função "get_params()" é possível capturar estes parâmetro para utilizar como ponto de partida.

Quadro 24 - Código para a criação da árvore de decisão inicial

```
# importa a árvore de decisão classificadora da biblioteca sklearn
from sklearn.tree import DecisionTreeClassifier

#Cria a árvore de decisão "clf"
clf = DecisionTreeClassifier()

#Cria a árvore de decisão baseada nos valores de treinamento
clf = clf.fit(X_train, y_train)

#Captura os parâmetros da árvore de decisão
 clf.get_params()
Resultado:
 {'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

Fonte: Elaborado pelo autor.

Com esses dados em mãos é possível notar que a árvore está utilizando o critério gini (criterion : gini). Também é possível notar que os parâmetros referentes a profundidade máxima (max_depth), número máximo de classes (max_features) e número máximo de nós de folha (max_leaf_nodes) estão com os parâmetros "none", indicando que não há nenhum valor padrão para essas variáveis. Neste trabalho, essas serão as variáveis chave a serem otimizadas.

Para levantar o índice de acurácia inicial, foi criada a variável "*predictions*" (predições) utilizando a função *predict* baseada no conjunto de dados de entrada para teste seguido da importação da ferramenta "acurance_score" do subconjunto *metrics* da biblioteca sklearn

Quadro 25 - Código para a verificação de acurácia do modelo inicial

```
# cria a variável "predictions" utilizando o conjunto de dados de teste de entrada
predictions = clf.predict (X_test)
#importação da ferramenta métrica "accurance_score"
from sklearn.metrics import accuracy_score

#Executa a avaliação de acurácia utilizando os dados de teste de entrada e saída
Accuracy_score(y_test,predictions)
Resultado : 0.8148148148148148
```

Fonte: Elaborado pelo autor.

Esse resultado de acurácia inicial significa que o modelo tem um grau de assertividade na predição de 81%, ou seja, a cada 100 bobinas avaliadas, o modelo seria capaz de tomar uma decisão coerente com a decisão tomada pelo especialista que fez a avaliação das bobinas contidas na base de dados utilizada como entrada em 81 delas.

É possível também plotar a árvore de decisão de modo a ter uma visão gráfica da estrutura do modelo, onde é possível avaliar a níveis da árvore, quebra dos nós e ramificações.

Para esse objetivo, é utilizada a bilioteca graphviz. Este pacote facilita a criação e renderização de descrições de gráficos na linguagem DOT do software de desenho de gráficos Graphviz (repositório upstream) do Python.

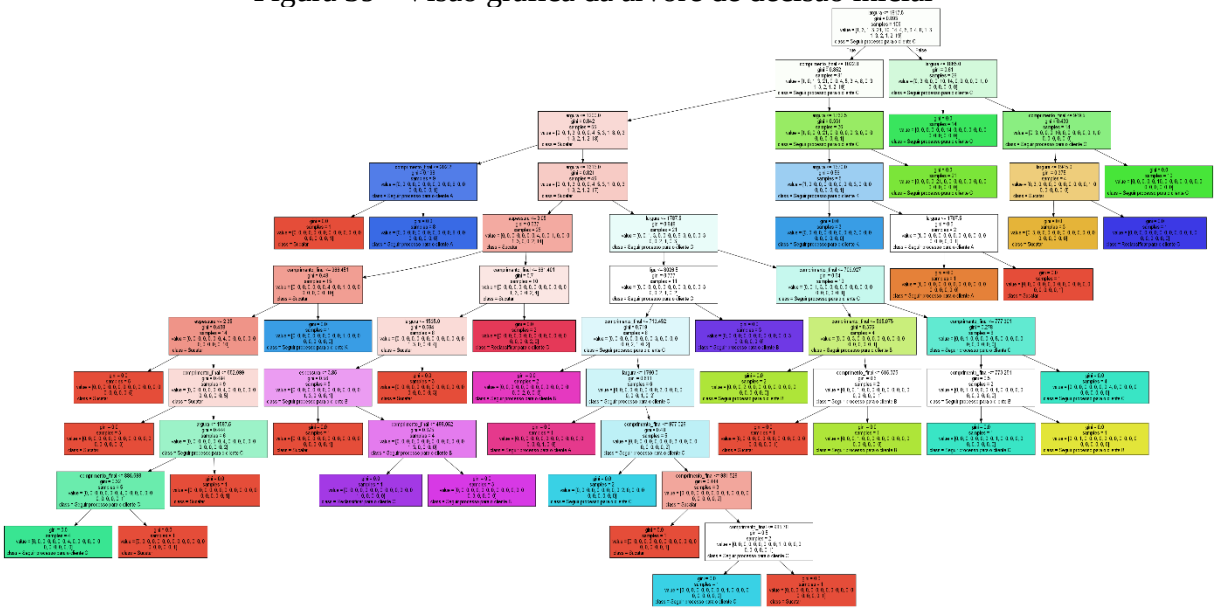
Com essa ferramenta é possível cria um objeto gráfico, montar o gráfico adicionando os nós e arestas e renderiza-lo (Pypi.org, 2023).

Quadro 26 - Código para a plotagem gráfica da árvore utilizando graphviz

```
#Importa a biblioteca de plotagem gráfica graphviz
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None, feature_names=X_train.columns, class_names= y, filled=True)

# Desenha gráfico da árvore e configura para formato .png
graph = graphviz.Source(dot_data, format="png")
graph

#Determina nome da imaem
graph.render("plotagem_árvore_decisão")
```

Fonte: Elaborado pelo autor.

Figura 39 - Visão gráfica da árvore de decisão inicial



Fonte: Elaborado pelo autor.

4 RESULTADOS E DISCUSSÕES

4.1 OTIMIZAÇÃO DO MODELO (PRUNING)

O método para a otimização do modelo adotado consiste na sequência de testes de comparação de acurácia do modelo para o mesmo conjunto de dados de teste e treinamento criados a partir da tabela “resultados”, contendo o histórico de avaliações de bobinas por especialistas.

Para esses testes, foi utilizado o mesmo comando para a criação da árvore inicial, porém, especificando os parâmetros de critério e profundidade. Para que os testes sejam efetivos, é necessário criar a árvore de decisão com os parâmetros alterados e treiná-la novamente antes de coletar a nota de acurácia do modelo.

Quadro 27 - Trecho do código para testes de profundidade com o critério Gini

```
#Teste do modelo com os critérios Gini alterando as profundidades

#Cria modelo de árvore de decisão utlizando o critério Gini e profundidade máxima de 5
clf = DecisionTreeClassifier(criterion="gini", max_depth=5)

#Treina o modelo com os conjuntos de dados e novos parâmetros
clf = clf.fit(X_train, y_train)

# cria a varíavel predictions utilizando o conjunto de dados de teste de entrada
predictions = clf.predict (X_test)

#Performance de acurácia do modelo
print("Acurácia:",accuracy_score(y_test,predictions))
Acurácia: 0.611111111
```

Fonte: Elaborado pelo autor.

O método para a otimização do modelo adotado consiste na sequência de testes de comparação de acurácia do modelo para o mesmo conjunto de dados de teste e treinamento criados a partir da tabela “resultados”, contendo o histórico de avaliações de bobinas por especialistas.

Para esses testes, foi utilizado o mesmo comando para a criação da árvore inicial, porém, especificando os parâmetros de critério e profundidade. Para que os testes sejam

efetivos, é necessário criar a árvore de decisão com os parâmetros alterados e treiná-la novamente antes de coletar a nota de acurácia do modelo.

Deste modo, é feita a repetição dos testes para diferentes profundidades buscando encontrar a combinação com o maior índice de acurácia. Neste caso, o teste foi repetido em sequência unitária até ser encerrado com a profundidade 15 devido a não haver mais tendencia de melhoria na performance do modelo a partir da profundidade 9.

Tabela 3 - Resultados de acurácia para o critério Gini

Acurácia	
Profundidade	**Gini**
5	0,611111
6	0,685185
7	0,703704
8	0,796296
9	0,814815
10	0,814815
11	0,814815
12	0,814815
13	0,814815
14	0,814815
15	0,814815

Fonte: Elaborado pelo autor.

Quadro 28 - Trecho do código para testes de profundidade com o critério Entropia

```
#Teste do modelo com critério de Entropia alterando as profundidades
#Cria modelo de árvore de decisão com os parâmetros estabelecidos
clf = DecisionTreeClassifier(criterion="entropy", max_depth=5)
#Treina o modelo com os conjuntos de dados X_train e y_train
clf = clf.fit(X_train, y_train)
# cria a varíavel predictions utilizando o conjunto de dados de teste de entrada
predictions = clf.predict (X_test)
#Performance de acurácia do modelo
print("Acurácia:",accuracy_score(y_test,predictions))
Acurácia: 0.7037037037037037
print("Acurácia:",accuracy_score(y_test,predictions))
Acurácia: 0.611111111
```

Fonte: Elaborado pelo autor.

Da mesma forma, são feitos os testes utilizando o critério de entropia buscando a melhor combinação entre a nota de acurácia e profundidade do modelo.

Tabela 4 - Resultados de acurácia para o critério Entropia

Acurácia	
Profundidade	**Entropia**
5	0,703704
6	0,814815
7	0,814815
8	0,814815
9	0,833333
10	0,833333
11	0,833333
12	0,833333
13	0,833333
14	0,833333
15	0,833333

Fonte: Elaborado pelo autor.

Utilizando o mesmo conceito, observando-se que não houve tendencia de alta, o teste foi encerrado com a profundidade 15.

Unindo os dados dos 2 testes, é selecionada a melhor combinação entre os critérios Gini e Entropia e respectivas profundidades buscando a nota mais alta em acurácia, chegando-se á combinação dos parâmetros de critério Entropia e profundidade 9, com acurácia de 83%.

Tabela 5 - Comparativo de acurácia para os critérios Gini e Entropia

	Acurácia	
Profundidade	**Gini**	**Entropia**
5	0,611111	0,703704
6	0,685185	0,814815
7	0,703704	0,814815
8	0,796296	0,814815
9	0,814815	0,833333
10	0,814815	0,833333
11	0,814815	0,833333
12	0,814815	0,833333
13	0,814815	0,833333
14	0,814815	0,833333

Fonte: Elaborado pelo autor.

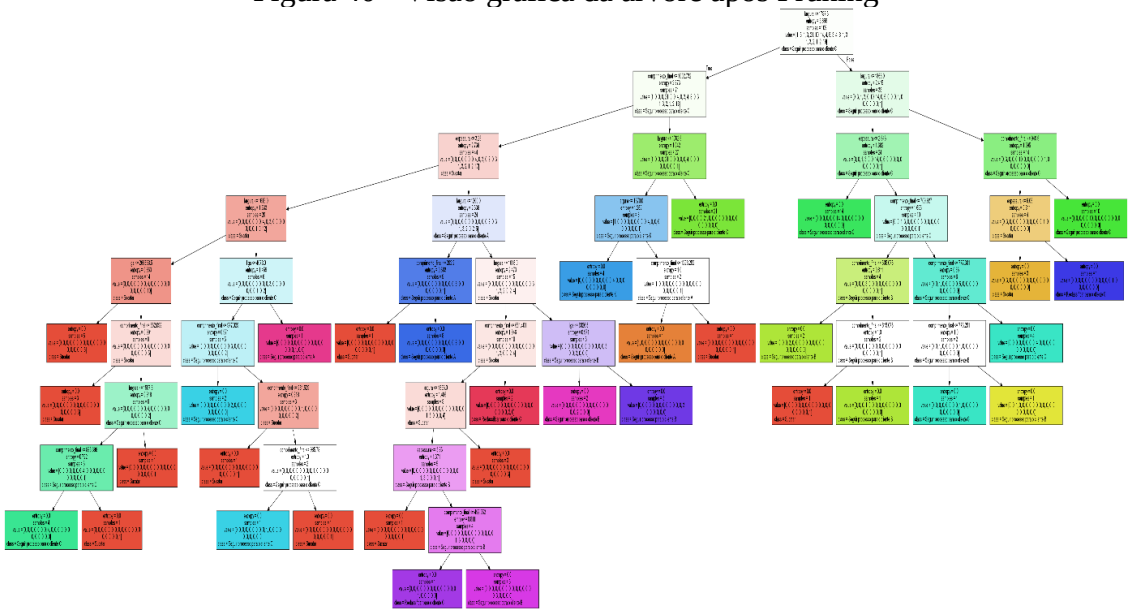
Quadro 29 – Códigos para a construção do modelo com os parâmetros otimizados e plotagem gráfica

```
#Cria modelo de árvore de decisão com os parâmetros estabelecidos
clf = DecisionTreeClassifier(criterion="entropy", max_depth=9)

#Treina o modelo com os conjuntos de dados e novos parâmetros
clf = clf.fit(X_train, y_train)

#Define o vetor de classes y
y = df_results['resultado_final']

#plota a árvore de decisão exibindo os dados de entrada e saída (classes)
tree.plot_tree (clf, feature_names = X_train.columns, class_names = y)
```

Fonte: Elaborado pelo autor.

Para visualizar a árvore após a poda ou “*pruning*”, utiliza-se novamente o comando de plotagem gráfica utilizando os dados de treinamento e classes denominadas no conjunto y, contendo os dados da coluna “resultado_final” da tabela resultados.

Figura 40 - Visão gráfica da árvore após Pruning



Fonte: Elaborado pelo autor.

Comparando a plotagem gráfica da árvore de decisão antes e após a poda é possível observar que a árvore ficou mais enxuta, apesar de alcançar uma acurácia ainda maior. Esse método é muito importante para obter a melhor performance do modelo de forma mais simples e eficaz, evitando altas cargas de processamento e *over-fitting* do modelo com regras muito específicas.

Tabela 6 - Dados do modelo antes e após *pruning*

	Pré *Pruning*	**Pós *Pruning***
Profundidade	11	9
Acurácia	61%	83,3%

Fonte: Elaborado pelo autor.

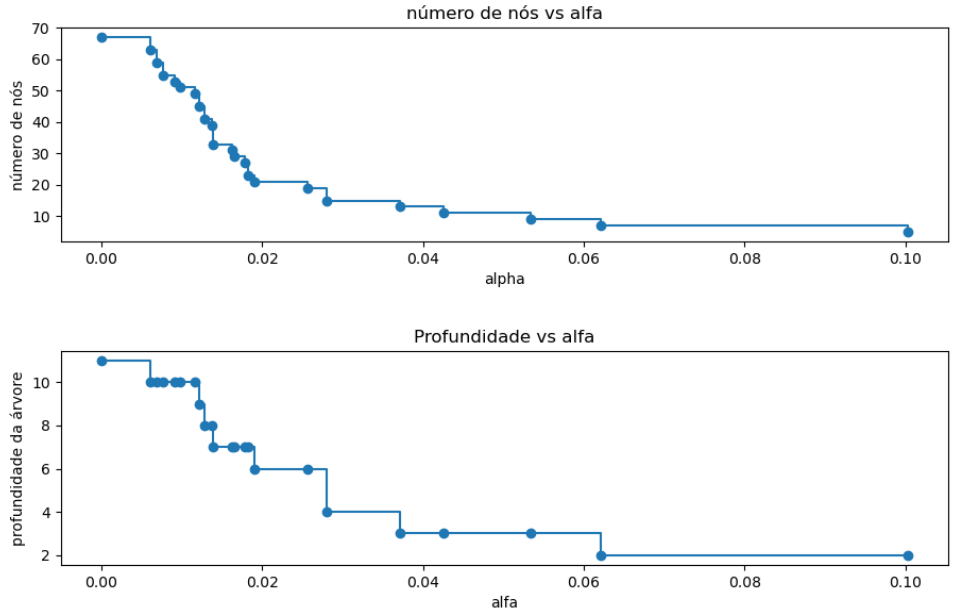
Ainda no aspecto de *over-fitting*, é possível utilizar outro recurso para verificar o processo de *prunning* e parametrizar a árvore utilizando análises mais detalhadas em diversos cenários. Este método, chamado *Cost Complexity Pruning*, consiste em determinar o fator de custo complexidade (ccp_alpha) da árvore de decisão de modo a definir o número de folhas a serem removidas de acordo com o nível de acurácia (sklearn.org, 2023).

Quadro 32 – Código para plotar a evolução da poda da árvore de acordo com os valores de ccp_alpha

```
clfs = clfs[:-1]
ccp_alphas = ccp_alphas[:-1]
node_counts = [clf.tree_.node_count for clf in clfs]
depth = [clf.tree_.max_depth for clf in clfs]
fig, ax = plt.subplots(2, 1)
ax[0].plot(ccp_alphas, node_counts, marker="o", drawstyle="steps-post")
ax[0].set_xlabel("alpha")
ax[0].set_ylabel("número de nós")
ax[0].set_title("número de nós vs alfa")
ax[1].plot(ccp_alphas, depth, marker="o", drawstyle="steps-post")
ax[1].set_xlabel("alfa")
ax[1].set_ylabel("profundidade da árvore")
ax[1].set_title("Profundidade vs alfa")
```

Fonte: Elaborado pelo autor.

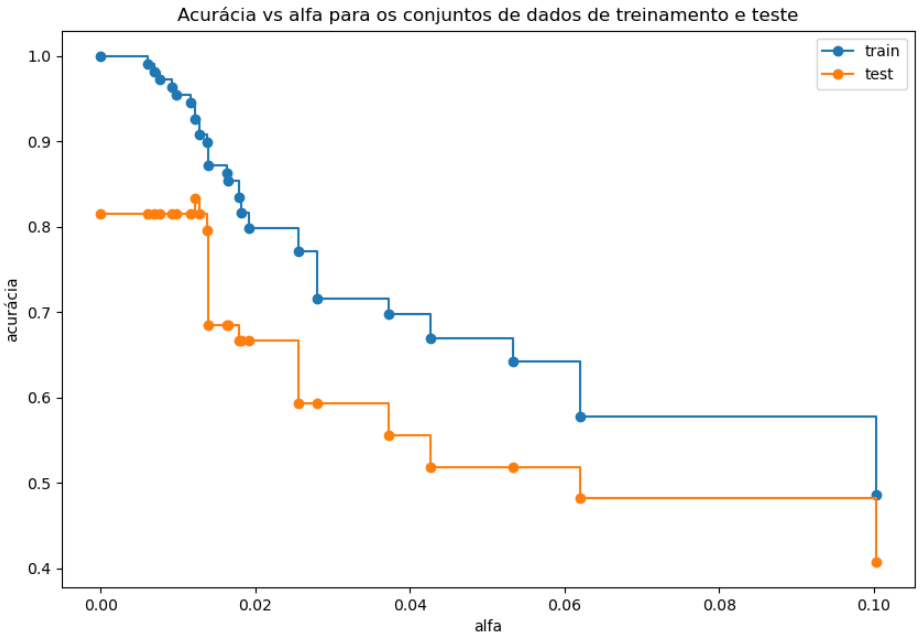
Figura 41 - Aspectos de profundidade e nós da árvore de acordo com os valores de ccp_alpha



Fonte: Elaborado pelo autor.

Em seguida, é possível comparar a acurácia dos conjuntos de dados train e test de acordo com os valores de ccp_alpha.

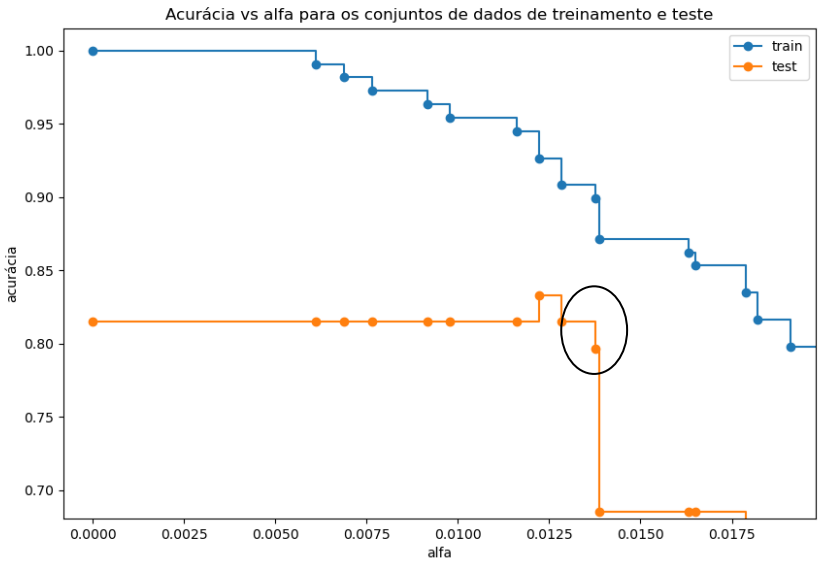
Figura 42 - Alfa vs acurácia



Fonte: Elaborado pelo autor.

Com este gráfico, é possível perceber que o valor padrão de ccp_alpha (= zero) irá fazer com que a árvore apresente uma tendência para o *over-fitting*, uma vez que neste momento a acurácia para o conjunto de dados de treinamento é de 100%. Desta forma, o valor de ccp_alpha deverá ser o valor que representa a última combinação onde a acurácia do conjunto de teste de mantém estável antes de começar a cair. Neste caso, o valor ótimo de ccp_alpha é 0,013.

Figura 43 - Alfa vs acurácia (zoom)



Fonte: Elaborado pelo autor.

O próximo passo é adicionar esse parâmetro na versão final da árvore de decisão.

Quadro 33 – Configuração final da árvore de decisão

```
clf = DecisionTreeClassifier(criterion = 'entropy', max_depth=9, ccp_alpha=0.013)
```

Fonte: Elaborado pelo autor.

4.2 MÉTRICAS DE VALIDAÇÃO

Utilizando a função metrics da biblioteca scikit learn é possível calcular, além da acurácia, os índices de precisão, *recall* e *f-score*.

Quadro 34 – Cálculo dos índices métricos da árvore pós prunning

```
from sklearn.metrics import precision_recall_fscore_support
precision_recall_fscore_support (y_test, predictions, average= 'weighted')
```

Fonte: Elaborado pelo autor.

Obtendo-se assim os valores:

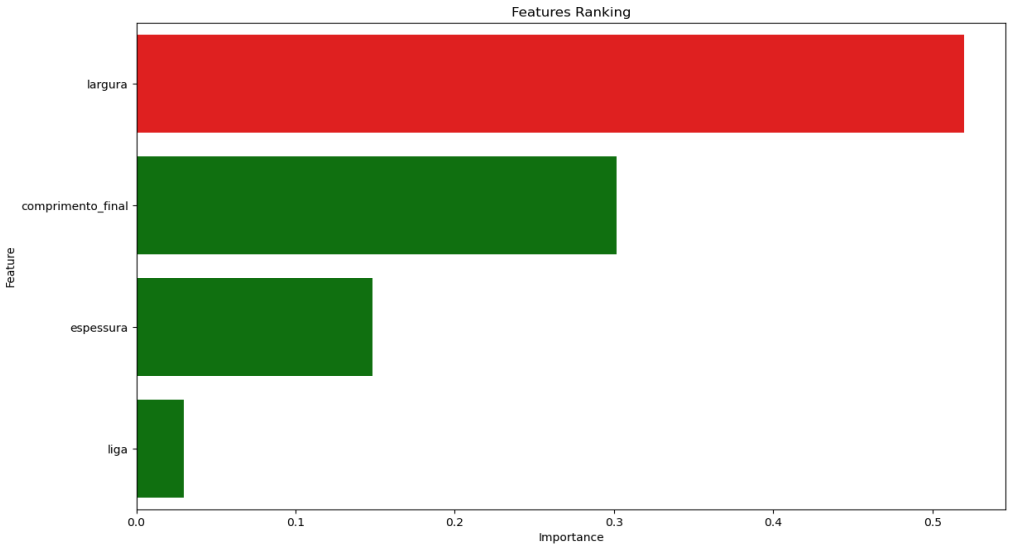
Tabela 7 - Dados de desempenho do modelo

Precisão	Recall	F-Score
83,58%	83,33%	82,07%

Fonte: Elaborado pelo autor.

Neste contexto, foi possível visualizar o ranking de *features* com as 4 variáveis de maior peso durante a tomada de decisão. Para o conjunto de dados utilizado, a largura do recurso apresentou o índice de maior importância, seguido pelo comprimento final (comprimento restante após a remoção da área de refugo usando as fórmulas de identificação de refugo apresentadas acima), liga do material e espessura do material.

Figura 44 - Ranking de importância das features



Fonte: Elaborado pelo autor.

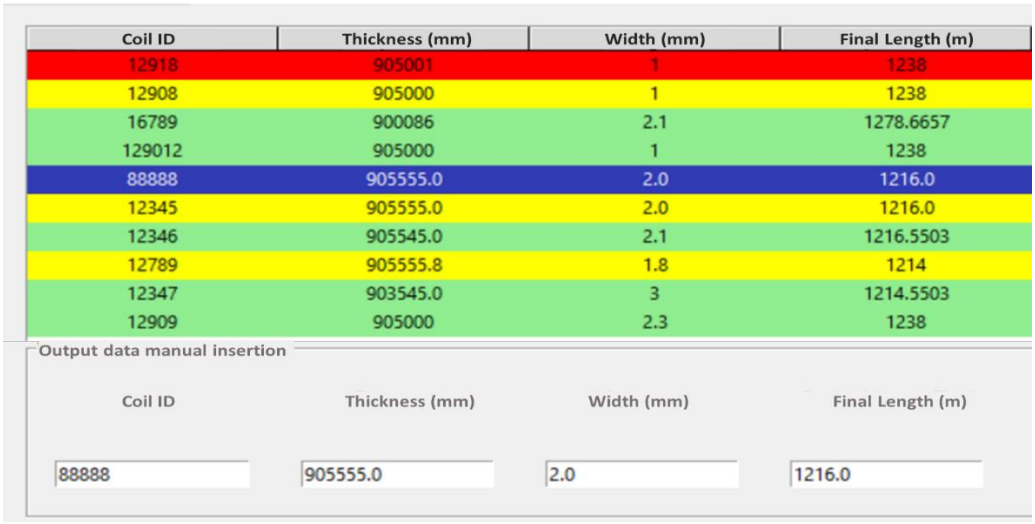
4.3 SUPERVISÓRIO ITERATIVO PARA O USUÁRIO FINAL

Para implementar o supervisório proposto, foi utilizada a biblioteca Tkinter (GRAYSON, 2020). O Tkinter é um módulo da linguagem Python, repleto de ferramentas que facilitam a criação de uma interface de usuário simples, mas de fácil compreensão. Este módulo adiciona interfaces orientadas a objetos, ou seja, nele como widget as referências são métodos e acionamos os widgets usando e seus atributos.

Neste supervisório, os usuários podem visualizar as principais saídas e decisões do modelo visualmente por cores, onde vermelho = “sucata”, amarelo = “reclassificar”, verde = “seguir processo” e azul significa que foi feita uma intervenção manual.

Além da visualização de dados, também é possível inserir novos dados manualmente ou corrigir os dados e tomadas de decisão existentes, esse recurso é uma característica chave para tratativas de *outliers* e ajustes finos nos inputs dos modelos sob a supervisão de um especialista. Em um ambiente fabril, o modelo estará em execução cíclica após o término de cada processo de laminação e o supervisório irá atualizar a lista os lotes produzidos em ordem decrescente de forma que o mais recente sempre esteja no topo. Ao visualizar os dados principais do produto a ser laminado e resultado do modelo, o especialista da equipe de operação do equipamento fará a checagem e eventual alteração necessária, sendo essas alterações salvas novamente no banco de dados e utilizado para treinamento periódico do modelo.

Figura 45 - Tela principal do supervisório



Fonte: Adaptado de Simões, M. (2023).

5 CONCLUSÃO

Neste estudo, foi possível avaliar dados de produção de uma empresa de laminação do período de 2022. Onde abordou a utilização da técnica de aprendizado de máquina conhecida como árvore de decisão classificatória para desenvolvimento de um modelo de caracterização e classificação de produtos de acordo com as variáveis de entrada liga, espessura, largura, descarte por espessura e temperatura na região final e inicial do produto, tendo como variável de saída categorias incluindo a ação tomada (seguir processo, reclassificar ou sucatear) seguidas do cliente sugerido de acordo com especificações pré definidas.

O modelo apresentou resultados satisfatórios após processo de *prunning* utilizando avaliação de acuraria e otimização do fato *ccp_alpha*. Saindo de uma acurácia inicial de 61% e profundidade de 11 para uma acurácia de 83,33% e profundidade de 9 com *ccp_alpha* de 0,012. Resultando em um modelo otimizado com precisão de 83,58%, *recall* de 83,33% e *f-score* de 82,07%. Já avaliando os pesos de cada *feature*, os resultados obtidos após a otimização do modelo de árvore de decisão mostraram a largura do material como *feature* principal, seguidos de comprimento, espessura e largura.

O desenvolvimento de um supervisório para interface e supervisão operacional possibilita a otimização constante do modelo através de entradas manuais caso necessário, além da visualização clara através de cores caracterizando cada ação tomada e atualização em tempo real após cada etapa do processo.

Como trabalhos futuros, destaca-se a possibilidade de integrar o modelo ao sistema de controle da produção de modo a otimizar a tomada de decisão observando o fluxo interno de material e estoques, tanto da fábrica quanto dos clientes em si.

Neste trabalho, observou-se que é possível construir um sistema eficiente para aplicação real na indústria usando software de código aberto e baixo custo de implementação e manutenção, incluindo uma alta flexibilidade para fazer adaptações para cada demanda de processo e com uma máquina humana amigável interface tornando-se uma operação simples de todos os níveis de usuários.

REFERÊNCIAS

BRAMER, M. **Pre-pruning classification trees to reduce overfitting in noisy domains**. Portsmouth: Faculty of Technology, University of Portsmouth, UK**,** 2002.

CASS, S. **Top programming languages 2021**. Disponível em: https://spectrum.ieee.org/top-programming-languages-2021. Acesso em: 21 set. 2023.

COUTINHO, T. **O que é python matplotlib?** conheça a biblioteca de gráficos! Disponível em: https://www.voitto.com.br/blog/artigo/o-que-e-python-matplotlib. Acesso em: 03 fev. 2023.

DESHPANDE, B. **2 main differences between classification and regression trees**. Disponível em: https://www.simafore.com/blog/bid/62482/2-main-differences-between-classification-and-regression-trees. Acesso em: 11 out. 2022.

GIL, A. C. **Como elaborar projetos de pesquisa**. 3. ed. São Paulo: Editora Atlas, 1996. 159p.

GRAYSON, J. **Python and Tkinter programming**. Greenwich: Manning Publications, 2000.

IBA AG. **Measurement systems for industry and energy**. Disponível em: https://www.iba-ag.com/en/. Acesso em: 10 mar. 2021.

IINTERNATION MACHINES BUSINESS CORPORATION. **Supervised Learning**. Disponível em: https://www.ibm.com/br pt/topics/supervised-learning. Acesso em: 18 dez. 2023.

JETBRAINS Pycharm: **o IDE Python para desenvolvedores profissionais**. Disponível em: https://www.jetbrains.com/pt-br/pycharm. Acesso em: 03 fev. 2023.

KALATHAS, I.; PAPOUTSIDAKIS, M. Predictive maintenance using machine learning and data mining: a pioneer method implemented to greek railways. **Designs**, v.5, n. 5, 2021. Disponível em: https://doi.org/10.3390/designs5010005. Acesso em: 13 abr. 2023.

KIM, M. Guaranteeing that multilevel prioritized DNN models on an embedded GPU have inference performance proportional to respective priorities. **IEEE Embedded Systems Letters**, New Jersey, v. 14, n. 2, p. 83-86, 2022. Disponível em: https://ieeexplore.ieee.org/document/9624962. Acesso em: 13 abr. 2023.

LAKATOS, E. M.; MARCONI, M. A. **Fundamentos de metodologia científica**. 7. ed. São Paulo: Atlas, 2010.

LAURETTO, M. S. **Árvores de decisão**. EACH-USP, 2010. Disponível em: https://edisciplinas.usp.br/pluginfile.php/4469825/mod_resource/content/1/ArvoresDecisao_normalsize.pdf. Acesso em: 15 mar. 2023.

LI, J. *et al*. Feature selection: a data perspective. **ACM Computing Surveys**, v.50, n.6, dez. 2016. Disponível em: https://dl.acm.org/doi/10.1145/3136625. Acesso em: 07 mar. 2024.

LOH, W. Classification and regression trees. **Wiley interdisciplinary reviews**: data mining and knowledge discovery, New Jersey, v. 1, p. 14–23, 2011.

MANUFACTURING GUIDE. **Ingot casting**. Disponível em: https://www.manufacturingguide.com/en/ingot-casting. Acesso em: 24 dez. 2023.

MILMANN K.J.; AVAIZIS M. Python for scientists and engineers. **Computing in science & engineering**, California, v.13, p. 9-12, 2011. Disponível em: https://escholarship.org/uc/item/93s2v2s7. Acesso em: 10 set. 2023.

MULINARI, B. **Pandas Python**: vantagens e como começar. Harve 2023. Disponível em: https://pypi.org/project/graphviz/. Acesso em: 12 mar. 2023.

SQL MANAGER. **Postgress database management aplication example**. Disponível em: https://www.sqlmanager.net/products/postgresql/manager. Acesso em: 05 fev. 2023.

PSYCOPG ORGANIZATION. **Basic module usage**. Disponível em: https://www.psycopg.org/docs/usage.html. Acesso em: 12 mar. 2023.

PYDATA ORGANIZATION. **Pandas**: ready parquet. Disponível em: https://pandas.pydata.org/docs/reference/api/pandas.read_parquet.html. Acesso em: 04 fev. 2023.

PYTHON ORGANIZATION. **Os miscellaneous operating system interfaces**. Disponível em: https://docs.python.org/3/library/os.html. Acesso em: 08 fev. 2023.

RUEDIGER P. *et al.* Combining visual analytics and machine learning for reverse engineering in assembly quality control. **Journal of imaging science and technology,** v. 64, n. 6, p. 060405-1-060405-13, 2020. Disponível em: https://web.cs.ucdavis.edu/~hamann/RuedigerClausHamannHagenLeitteJIST_VDA2021PaperAsPublishedB12312020.pdf. Acesso em: 06 set. 2023.

PENG, R. Nonlinear vibration characteristics and time-delayed displacement control of rolling mill under dynamic rolling force. **Journal of Vibroengineering,** v. 23**,** n. 7. p. 1535–1548, ago. 2021.

SCIKIT-LEARN ORGANIZATION. **sklearn.model_selection.train_test_split.** Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. Acesso em: 24 mar. 2023.

SCIKIT-LEARN ORGANIZATION. **sklearn.tree.plot_tree**. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html: Acesso em: 13 mar. 2023.

SCIKIT-LEARN ORGANIZATION. **Decision trees**. Disponível em: https://scikit-learn.org/stable/modules/tree.html. Acesso em: 24 ago. 2023.

SCIKIT-LEARN ORGANIZATION. **Post pruning decision trees with cost complexity pruning**. Disponível em: https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html#sphx-glr-auto-examples-tree-plot-cost-complexity-pruning-py. Acesso em: 12 dez. 2023.

SHAMRAT, F. M. J. *et al.* A comprehensive study on pre-pruning and post-pruning methods of decision tree classification algorithm. *In*: INTERNATIONAL CONFERENCE ON TRENDS IN ELETRONICS AND INFORMATICS, 5., 2021, Tirunelveli. **Proceedings […].** Tirunelveli, 2021. p. 1339-1345.

SINGER, G.; COHEN, I. **An objective-based entropy approach for interpretable decision tree models in support of human resource management**: the case of absenteeism at work. Ramat Gun: Faculty of Engineering, Bar-Ilan University, 2020.

SINGER, G.; COHEN, I. An objective-based entropy approach for interpretable decision tree models in support of human resource management: the case of absenteeism at work. **Entropy**, Calgary, v.22, n. 8, p.821, 2020. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7517405/pdf/entropy-22-00821.pdf. Acesso em: 18 set. 2023.

SONGINI, M L. Put in plain language: the high portable, object-oriented python language moves into enterprise application development. **Computerworld,** 12 set. 2005. Disponível em: http://www.computerworld.com/softwaretopics/software/story/0,10801,104484,00.html. Acesso em: 30 maio 2022.

STOJILJKOVIĆ, M. **Split your dataset with scikit-learn's train_test_split().** Disponível em: https://realpython.com/train-test-split-python-data/. Acesso em: 13 mar. 2023.

STOCK, T.; SELIGER, G. Opportunities of sustainable manufacturing in industry 4.0. **Procedia Cirp**, v. 40, p. 536-541, 2016.

TAMMINEN, S. *et al*. From measurements to knowledge: online quality monitoring and smart manufacturing. *In*: PERNER, P. (ed.) **Advances in data mining**: applications and theoretical aspects. Oulu: Springer, 2018. v. 10933, p. 17-28.

TEC-SCIENCE. **From steel to semi-finished products.** Disponível em: https://www.tec-science.com/material-science/steel-making/steel-semi-finished-products-continuous-ingot-casting. Acesso em: 13 dez. 2023.

WANG, Y.; WITTEN. Induction of model trees for predicting continuous classes. *In:* THE EUROPEAN CONFERENCE ON MACHINE LEARNING, 1997, Prague. **Proceedings […].** Prague: University of Economics, Faculty of Informatics and Statistics, 1997.

WHITE, T. **Hadoop:** the definite guide. 4th ed. O' Reilly Media, 2015.

XU, L.; FENG, L.; YAN,W. A comparative assessment of six machine learning models for prediction of bending force in hot strip rolling process. **Metals**, v.10, n.5, 2020.

YACOB, F.; SEMERE, D.; NORDGREN, E. Anomaly detection in Skin Model Shapes using machine learning classifiers. **The International Journal of Advanced Manufactoring Technology**, v.105, p. 3677-3689, 2019.