UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Arthur Bertoldo Benedetti

Projeto de fim de curso

Predição de falhas na produção de fios: reconhecimento de bolhas na passagem de verniz isolante

Florianópolis

2022

Arthur Bertoldo Benedetti

Predição de falhas na produção de fios: reconhecimento de bolhas na passagem de verniz isolante

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão
Orientador: Jomi Fred Hubner

Florianópolis
2022

Acordo de não divulgação

Este documento continha informações sigilosas sobre a WEG que impossibilitava sua publicação. O trabalho original pode apenas ser apresentado a banca de avaliadores do projeto de fim de curso.
Assim o documento atual teve as informações confidenciais removidas para ser publicado na biblioteca universitária da UFSC.

Arthur Bertoldo Benedetti

Predição de falhas na produção de fios: reconhecimento de bolhas na passagem de verniz isolante

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo

Florianópolis, 24 de dezembro de 2022.

Jomi Fred Hubner
Orientador
Universidade Federal de Santa Catarina

Jairo Luis Elchendor
Supervisor na Empresa
WEG

Agradecimentos

Agradeço a amigos e família por sempre me ajudarem quando necessito. A instituição por me oferecer educação, a WEG por me oferecer a oportunidade de trabalhar na empresa e por fim ao time de inteligência artificial por oferecer apoio em todo o projeto.

Resumo

O presente relatório tem por objeto um estudo feito na fábrica de fios da WEG que tem o intuito de predizer a formação de bolhas na passagem de verniz isolante nos fios. Em um mundo cada vez mais competitivo e ao mesmo tempo preocupado com seu impacto no meio ambiente a redução de gastos desnecessários e o aumento da eficiência em processos produtivos vem sendo um dos maiores focos de todos os setores da indútria. Com isso em mente a indústria 4.0 traz diversas aplicações que tem o intuito de reduzir os rejeitos além dos períodos não produtivos das empresas.

A predição de qualidade é uma das grandes promessas para as indústrias, unindo todos os grandes nomes da indústria 4.0 para resolver um problema, o de baixa qualidade. Usando da internet das coisas, integração dos dados e *machine learning* é possível identificar padrões nas linhas de produção que levam um produto a ter baixa qualidade e ser reprovado nos diversos testes de qualidade que uma empresa tem. Este projeto utiliza dos dados colhidos do gerenciador de chão de fábricas da WEG para tentar encontrar as relações entre as variáveis explicativas que podem levar a baixa qualidade da passagem de verniz nos fios da WEG. O projeto utiliza múltiplos algorítimos de inteligência artificial para prever a formação de bolhas na passagem de verniz isolante nos fios e oferece soluções objetivas para evitar a formação das mesmas.

O projeto conseguiu com alta assertividade prever com 30 minutos de antecedência a formação de bolhas e oferecer soluções para evitar as mesmas.

Palavras-chave: predição de qualidade, inteligência artificial, gerenciador de chão de fábrica

Abstract

This document reports on a study carried out at the WEG wire factory that aims to predict the formation of bubbles in the application of insulating varnishes in cables produced in the company. In a world that constantly gets more competitive and at the same time worries about it´s effects on the enviornment, reduction of unecessary spending and increase in efficiency of productive processes has become one of the major focusses of all industry sectors. With that goal in mind, Industry 4.0 brings several applications that aim to reduce waste and non-productive periods in companies.

Quality prediction is one of the great promises for factories, uniting all the great names that take part of Industry 4.0 to solve one problem, low quality. Using the internet of things, data integration and machine learning it is possible to identify patterns in production lines that lead to a low quality product and failure in the various quality tests that a company applies. This project used the data collected in WEG´s shop floor manager to try to find relationships between the explanatory variables and a dependent variable that can lead to a poor application of varnishes in WEG wires. The project uses multiple machine learning models to predict wether bubbles will be formed in the process of applicating insulating varnishes in wires and offer solutions for the problem.

The project was able to predict with high accuracy the formation of bubbles 30 minutes before it happened and offer viable solution to avoid them.

Key-words: predictive quality, artificial inteligence, shoop floor management

Lista de ilustrações

Figura 1 – *machine learning pipeline* 17
Figura 2 – Motor elétrico 22
Figura 3 – Parque fabril 1 em 1964 24
Figura 4 – Parque fabril 2 24
Figura 5 – Fios de cobre a esquerda e alumínio a direita 25
Figura 6 – Primeiro processo de trefilagem 26
Figura 7 – Segundo processo de trefilagem 27
Figura 8 – Máquina do estudo 28
Figura 9 – Aplicação de verniz 29
Figura 10 – Fornos de secagem 30
Figura 11 – Sensor de qualidade 31
Figura 12 – Carretel sendo enrolado 32
Figura 13 – Bolhas no fio 32
Figura 14 – Distribuição normal 40
Figura 15 – Gráfico de violino 42
Figura 16 – Exemplo gráfico de KNN 45
Figura 17 – exemplo de classificador por máquinas de suporte 46
Figura 18 – Exemplo de árvore de decisão 47
Figura 19 – Exemplo de matriz de confusão 50
Figura 20 – MotorScan 52
Figura 21 – Distribuição das temperaturas 58
Figura 22 – Distribuição das temperaturas sem os 3 σ 59
Figura 23 – Correlação das variáveis de temperatura utilizando kendall-tau 60
Figura 24 – P-valor das variáveis de temperatura utilizando kendall-tau 61
Figura 25 – Matriz de correlação das variáveis do SFM adicionadas depois 62
Figura 26 – F1 score da árvore de decisão em relação aos *datasets* e *samplings* . . . 65
Figura 27 – Evolução do erro das florestas randômicas pela função de minimização 65
Figura 28 – F1 score das florestas rândomicas em relação aos *datasets* e *samplings* . 66
Figura 29 – Evolução do erro das florestas randômicas pela função de minimização 67
Figura 30 – F1 score do *gradient boosting* em relação aos *datasets* e *samplings* . . . 68
Figura 31 – Evolução do erro do *gradient boosting* pela função de minimização . . . 68
Figura 32 – F1 score do *xtreme gradient boosting* em relação aos *datasets* e *samplings* 69
Figura 33 – Evolução do erro do *xtreme gradient boosting* pela função de minimização 70
Figura 34 – Evolução do erro dos modelos em sequência florestas randômicas, árvores de decisão, *gradient boosting* e *Xtreme gradinet boosting* 71

Figura 35 – Acurácia do modelo meta com a modificação dos diversos parâmetros utilizados. 72
Figura 36 – Minimização do erro do modelo meta 72
Figura 37 – matriz de confusão do modelo meta 73
Figura 38 – matriz de confusão da árvore de decisão 75
Figura 39 – Árvore de decisão inteira 76
Figura 40 – Zoom em uma parte do caminho de uma amostra 76
Figura 41 – tela da fabrica de fios do SFM com o ícone 80
Figura 42 – tela das regras 81
Figura 43 – Distribuição aceleração *motorscan* eixo X 88
Figura 44 – Distribuição aceleração *motorscan* eixo X limpo com 3 σ 89
Figura 45 – Distribuição aceleração *motorscan* eixo Y 90
Figura 46 – Distribuição aceleração *motorscan* eixo Y limpo com 3 σ 91
Figura 47 – Distribuição aceleração *motorscan* eixo Z 92
Figura 48 – Distribuição aceleração *motorscan* eixo Z limpo com 3 σ 93
Figura 49 – Distribuição das temperaturas adicionadas antes 94
Figura 50 – Distribuição das temperaturas adicionadas antes limpas com 3 σ 95
Figura 51 – Distribuição das temperaturas adicionadas depois 96
Figura 52 – Distribuição das temperaturas adicionadas depois limpos com 3 σ . . . 97
Figura 53 – Distribuição dos dados dos ventiladores 98
Figura 54 – Distribuição dos dados dos ventiladores limpas com 3 σ 99
Figura 55 – Distribuição do resto das variáveis 100
Figura 56 – Distribuição do resto das variáveis limpas com 3 σ 101
Figura 57 – Correlação ventiladores Kendal-tau 102
Figura 58 – Correlação temperaturas *motorscan* Kendal-tau 103
Figura 59 – Correlação acelerações *motorscan* Kendal-tau 103
Figura 60 – Correlação temperaturas Kendal-tau 104
Figura 61 – Correlação venitladores spearman 104
Figura 62 – Correlação acelerações *motorscan* spearman 105
Figura 63 – Correlação temperaturas *motorscan* spearman 105
Figura 64 – Correlação temperaturas spearman 106
Figura 65 – P-valor ventiladores spearman 106
Figura 66 – P-valor acelerações *motorscan* spearman 107
Figura 67 – P-valor temperaturas *motorscan* spearman 107
Figura 68 – P-valor temperaturas spearman 108
Figura 69 – P-valor venitladores kendall-tau 108
Figura 70 – P-valor acelerações *motorscan* kendall-tau 109
Figura 71 – P-valor temperaturas *motorscan* kendall-tau 109
Figura 72 – P-valor temperaturas kendall-tau 110

Figura 73 – Matriz de correlação das variáveis do SFM adicionadas antes 110
Figura 74 – Matriz de correlação das variáveis do *motorscan* 111

Lista de tabelas

Tabela 1 – Transformação do SFM 54
Tabela 2 – Adição de eventos ao SFM 54
Tabela 3 – Adição de eventos ao SFM 56
Tabela 4 – *Dataset* sujo 56
Tabela 5 – Tabela de correlação de forças 60
Tabela 6 – Tabela de regras 78

Lista de abreviaturas e siglas

SOAP	*Simple Object Access Protocol*
REST	*Representational State Transfer*
HTTP	*Hypertext Transfer Protocol*
SMTP	*Simple Mail Transfer Protocol*
XML	*extensible markup language*
WSDL	*Web Services Description Language*
JSON	*JavaScript Object Notation*
CSV	*Comma Separated Values*
A.M.	Aprendizado de Máquina
ML	*Machine Learning*
SFM	*Shop FLoor Management*
GCF	Gerenciador do Chão de Fábrica
ORM	*Object Relational Mapper*
BD	Banco de Dados
DBA	*Data Base Admnistrator*
ENS	Sistema de Ensaios
URI	*uniform resource identifier*

Lista de símbolos

σ	distribuição normal
λ	parâmetro de potência

Sumário

Introdução **15**

0.1 Pipelines em *machine learning* **16**

1 PREDIÇÃO DA BAIXA QUALIDADE **20**

2 WEG E O PROCESSO DE FABRICAÇÃO DE FIOS **23**

3 TEORIAS E MÉTODOS ESTUDADOS **34**

3.1 Introdução **34**

3.2 Coleta e integração **35**

3.3 Pre processamento e análise **36**

3.3.0.1 Bibliotecas 36

3.3.0.2 Métodos estatísticos 38

3.3.1 *Over* e *under sampling* de dados 42

3.4 Algorítimos de previsão **44**

3.4.1 Regressão logística 44

3.4.2 Redes Neurais 44

3.4.3 *K Nearest Neighbors* 45

3.4.4 Máquinas de Suporte 45

3.4.5 Modelos Singulares 46

3.4.6 Modelos *Ensemble* 47

3.4.7 Otimização de Parâmetros 48

3.5 Avaliação de Modelos **48**

4 MODELOS E ACURÁCIA **51**

4.1 Introdução **51**

4.2 Coleta e integração **51**

4.3 Pré-processamento **56**

4.4 Algorítimos de previsão **63**

4.4.1 Árvore de decisão 64

4.4.2 Florestas randômicas 66

4.4.3 *Gradient boosting* 67

4.4.4 Xtreme gradient boosting 69

4.4.5 avaliação final 70

4.4.6 modelo meta 71

4.5 Retorno das regras **74**

4.6 API 78
4.7 Banco de dados 79
4.8 *Front-end* 79

5 CONCLUSÃO 82
5.1 O que foi feito 82
5.2 Trabalhos futuros 83

REFERÊNCIAS 85

APÊNDICE A – DISTRIBUIÇÃO DAS VARIÁVEIS DO PROBLEMA 88

APÊNDICE B – CORRELAÇÃO DAS VARIÁVEIS DO PROBLEMA102

Introdução

Com a extensa competitividade do mercado atual empresas precisam se inovar cada vez mais para conseguir alcançar as altas demandas exigidas de qualidade, prazo e preço imposta pelos clientes. Aliado a isso estão as necessidades de sustentabilidade e legislações para a preservação do meio ambiente. Nesse contexto as indústrias são cada vez mais direcionadas a reduzir seus gastos aumentando a eficiência em todos os seus processos, reduzindo o consumo de matéria prima e produção de resíduos.

Dentre os diversos fatores que influenciam os objetivos acima um se destaca devido a grande possibilidade de atuação: A qualidade de produção. Ironicamente uma das variáveis que mais influenciam no custo de fabricação é gratuita, parafraseando (CROSBY, 1979) a qualidade é gratuita, o que custa são todas as ações subsequentes de não ter feito o processo da forma correta pela primeira vez. Assim, o custo da falta de qualidade pode ser expresso pela soma de todos os custos causados a clientes e empresa devido ao produto não ter alcançado as especificações de ambos (HARRINGTON, 1987). Estes custos podem ser subdivididos em três tipos: os internos, externos e ocultos. (MAHMOOD; KURESHI, 2015).

Os custos internos são todos aqueles que a empresa despende antes do produto ser entregue ao cliente. Ou seja, todos os gastos na linha de produção devido a uma peça defeituosa. Esses incluem: o tempo desperdiçado pelas máquinas e trabalhadores no retrabalho da peça, o tempo desperdiçado em inspeção e reteste, a matéria prima que é descartada ou consumida no caso de retrabalho, a parada da linha devido a criação de um gargalo pela falta da peça na máquina subsequente da linha e consequentemente custos com atrasos caso estes se desenvolvam.

Os problemas internos acabam por acarretar em custos externos. Estes podem ser definidos como todos os custos com o cliente após a entrega da peça defeituosa. Aqui entram: ativar a garantia, custos com mão de obra nos setores de reclamação, inspeção e correção, custos com a deterioração do produto final, transporte no caso de um *recall* e até no pior dos casos com ações judiciais.

Por último todos esses problemas acabam por se desenvolver em diversas despesas implícitas a empresa. Um produto de baixa qualidade acaba por piorar a impressão da mesma no mercado. Neste caso é muito difícil estimar o quanto uma companhia perde ao perder a confiança de seu consumidor. Mesmo sendo árduo calcular o valor exato de oneração causado pela degradação de reputação, é possível perceber a dimensão do problema com um pequeno exemplo: imagine que uma pessoa comprou um celular e ele quebra logo na primeira semana. É bem provável que não só a pessoa nunca mais compre

da marca, mas também faça um esforço extra para garantir que outras pessoas próximas a ela também não comprem. Esse efeito cresce exponencialmente podendo alcançar perdas astronômicas a empresa.

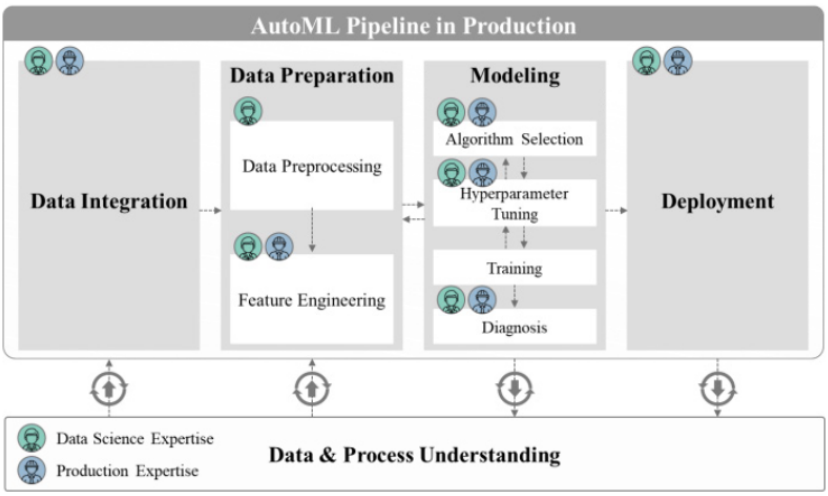
Dentro das diversas formas de atenuar os problemas supracitados, a predição de qualidade se apresenta como uma potencial solução. De maneira breve, a predição de qualidade utiliza de dados coletados na linha de produção para fazer inferências sobre os produtos sendo produzidos. Assim, a predição de qualidade tem como interesse prever com antecedência problemas de produção para evita-los economizando recursos das empresas que aplicam esta técnica. Para conseguir fazer este prognóstico se utiliza de técnicas de aprendizado de máquina e estatística para correlacionar as variáveis explicativas coletadas no chão de fábrica com as variáveis resposta que mensuram o acontecimento de falhas na produção.

Este trabalho tem como interesse a predição de formação de bolhas na passagem de verniz isolante em fios além do retorno de certos procedimentos que possam evitar a formação das mesmas antes que elas aconteçam. O tema está intrinsecamente relacionado ao curso devido a programação, utilização e leitura de sensores, assim como o estudo de uma linha de produção completamente automatizada.

0.1 Pipelines em *machine learning*

Para desenvolver uma solução para o problema foram utilizados de modelos de aprendizado de máquina para prever as bolhas e oferecer os procedimentos supracitados. Modelos de *machine learning* seguem certas etapas padrão que aparecem em qualquer problema resolvido com inteligência artificial (KRAUSS et al., 2020). Esta série de etapas em sequência que manipulam dados são conhecidas na computação como *pipelines*. No caso de uma solução de aprendizado de máquina quatro grandes processos fazem parte da *pipeline* do sistema, sendo que cada um pode ter uma outra *pipeline* interna que realiza a tarefa em questão. Integração dos dados, preparação dos dados, modelagem e aplicação são as quatro etapas que serão comentadas abaixo.

Figura 1 – *machine learning pipeline*



Fonte: (KRAUSS et al., 2020)

A integração é o primeiro passo e um dos mais demorados ela tem como interesse conseguir os dados necessários e integra-los para futuramente ser utilizado no algorítimo. Tabelas de origens diferentes são unidas e seus dados formatados para a estrutura mais conveniente. Essa etapa pode demorar devido ao tempo despendido com solicitações de acesso aos dados para cada parte responsável da empresa.

Com os dados formatados é necessário a preparação dos mesmos. Esta etapa consiste em duas etapas menores, o pré-processamento e a extração das características importantes. No primeiro passo são feitas as limpezas inicias. Dados nulos, ou valores sem sentidos são removidos, as variáveis são formatas para os tipos corretos e transformadas de acordo com o problema. Com isso feito é o momento de encontrar as características relevantes para o problema. As variáveis explicativas são estudadas e aquelas não importantes para a solução são removidas. Além disso são feitas a união e transformação de diversas das medições para conseguir características melhores que então serão aplicadas no modelo.

Desenvolvido um bom *dataset* é necessário a modelagem do sistema. Esta etapa consiste em selecionar um ou vários algorítimos, otimizar os hiper parâmetros, treinar os algorítimos e avaliar os resultados. A seleção dos modelos dependem de três critérios, se o problema é supervisionado, se é um problema de regressão ou classificação e por fim da performance do modelo em si. modelos supervisionados são aqueles que se tem a variável resposta, assim o modelo pode utiliza-la para ser treinado, modelos não supervisionados tentam encontrar correlações entre variáveis sem uma variável resposta. Problemas de classificação são aqueles onde a variável de interesse é categórica enquanto os de regressão a variável é contínua. Por fim existe a performance do modelo, isso depende de coisas como acurácia, custo computacional e até afinidade com o problema, já que modelos diferentes

podem performar melhor com *datasets* diferentes e selecionar modelos que trabalhem em conjunto pode influenciar nas escolhas dos mesmos. Selecionados os modelos é necessário realizar um ajuste dos hiper parâmetros. Estes são parâmetros ajustáveis que influenciam como o modelo funciona, isso varia de modelo para modelo e é necessário escolher os parâmetros que melhor resolvam o problema em questão. Por fim se avalia a solução antes de passar para a última etapa de implementação.

No estágio final o programa é colocado no servidor correto, as API´s para consumo da aplicação são criadas, restrições de tempo e responsividade são atendidos. Caso a aplicação necessite uma interface de usuário ela é feita nesta etapa.

Devido a posse dos dados da geração de bolhas foi escolhido por utilizar de modelos supervisionados para o problema. Assim deve se decidir se o problema se encaixa como um modelo de classificação ou como um modelo de regressão.

No caso da classificação um período próximo a acontecer uma bolha será classificado como um estado "próximo de dar bolha"e se utiliza de um modelo para classificar se o sistema está neste estado ou não. Um estudo parecido foi feito para revestimentos em pó, onde se utilizou árvores de decisão para prever com 88% de acurácia se o revestimento ficaria bom ou não (PACHNER; MIETHLINGER, 2019). Diversos estudos foram feitos na área de classificação de qualidade nas linhas de produção com o intuito de minerar dados importantes para o processo, como demonstrado em (VAZAN et al., 2017).

No caso de regressão se tenta prever a evolução das variáveis e assim definir quanto tempo falta para a formação de uma bolha. Devida a necessidade de alta qualidade de dados, rápido tempo de aquisição e grande capacidade computacional um modelo de regressão para prever a evolução das variáveis explicativas e assim poder fazer inferências sobre as bolhas foi considerado muito complexo.

Sendo assim o projeto segue os passos descritos na *pipeline de machine learning* com o intuito de classificar se em um dado período de tempo haverá bolha ou não e em caso positivo oferecer certas soluções para evitar o problema antes que ele aconteça. Para retornar regras que são passíveis de entendimento é necessário utilizar modelos de caixa branca ou seja modelos que é possível investigar as decisões internas do algorítimo que levaram a decisão final. Geralmente modelos caixa branca são menos precisos por isso foi utilizado um conjunto de modelos caixa branca e preta para a resolução do problema. Em conjunto dos modelos foram utilizados métodos de *over e under sampling* para melhor equilibrar os *datasets* de treino e evitar que o algorítimo tenha um viés a classificar a classe mais comum, a classe sem bolhas.

Este relatório é dividido do seguinte modo. No primeiro capítulo será apresentado uma introdução a predição de qualidade da manufatura, modelos já usados em outras indústrias e sua importância atualmente. No segundo os processos da fábrica de fios é

explicado e o problema em questão apresentado. No terceiro capítulo será apresentado todos os modelos e teorias que foram explorados para a resolução do problema. Então é apresentado o que foi feito no projeto para por fim desenvolver as conclusões finais sobre o mesmo.

1 Predição da baixa qualidade

Com a revolução digital, dados se tornaram o petróleo do mundo moderno, para ser utilizado em todo seu potencial precisa ser explorado e refinado. Na perspectiva da indústria, a quarta revolução industrial trouxe pela primeira vez através da internet das coisas uma união entre diversos processos e camadas das empresas abrindo um leque variado de opções para a utilização deste novo recurso.

Entre as diversas aplicações dos dados na indústria moderna a predição de qualidade da produção se apresenta como um promissor investimento prometendo amplos retornos em corte de gastos, redução de rejeitos e aumento da eficiência.

Qualidade pode ser definida de diversas maneiras. Segundo (CHANDRA, 2001) a definição da qualidade de um produto pode variar entre satisfazer os requisitos do consumidor, estar apto para uso ou se conformar aos requisitos previamente estabelecidos para o produto. Estando claro que a qualidade tem como objetivo explicito satisfazer o cliente, meta principal para qualquer empresa. Com este objetivo em mente diversos métodos foram desenvolvidos com o intuito de aprimorar a qualidade dos processos produtivos nas empresas, entre eles estão: *six-sigma* e Kaizen.

Dos modelos citados apenas o *six-sigma* utiliza de métodos matemáticos para a melhoria da qualidade dos processos produtivos da empresa. O *six-sigma* é uma abordagem disciplinada, orientada por projetos e estatisticamente baseada de reduzir variabilidade, remover defeitos e eliminar desperdício de produtos, processos e transações (MONTGOMERY; WOODALL, 2008).

O Kaizen, do japonês melhoria continua é uma filosofia de continuamente despender esforços para melhorar os processos produtivos de uma empresa (SINGH; SINGH, 2009). Ele vem como uma via de duas mão colocando mais confiança no trabalhador, prometendo mais estabilidade empregatícia e distribuição de benefícios para a força trabalhadora e recebendo em troca a experiência dos mesmos para melhor desenvolver os padrões de atuação nos processos da empresa.

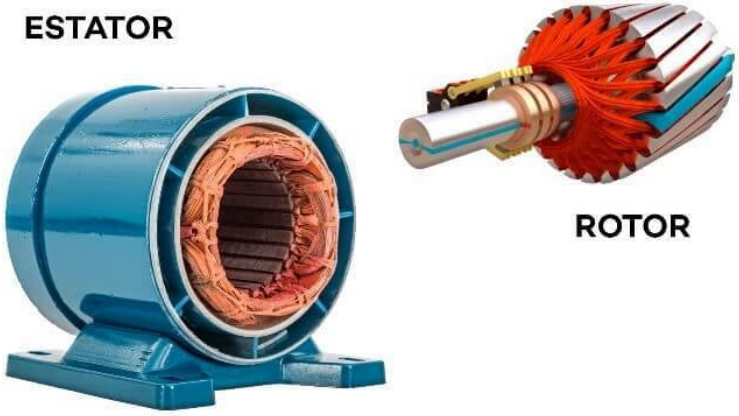
A predição de qualidade tem uma abordagem diferente pois suas soluções "são construídas sobre os dados do processo de manufatura. Extraindo padrões recorrentes de dados e relacionando-os com medições de qualidade. A predição de qualidade possibilita a estimação da qualidade do processo baseado nos dados do processo"(TERCAN; MEISEN, 2022). Assim esta não depende de colaboradores para encontrar uma falha, basicamente depende apenas do histórico das variáveis de processo e dos indicadores de qualidade do produto sendo assim uma grande aliada aos processos de melhoria de qualidade já existentes.

Essa independência de mão de obra no processo traz certas vantagens que os outros métodos citados não oferecem. Entre elas se destacam encontrar padrões não evidentes em certos problemas e também evitar respostas tendenciosas de especialistas. As linhas de produção são complexas e diversas vezes processos anteriores afetam a qualidade de um processo posterior, por causa dessa interdependência as vezes fica complicado encontrar onde exatamente uma falha se origina, portanto não é evidente para um operador como acontecem estas correlações. Dificultando ainda mais o processo de identificação de problemas, profissionais acabam por ter um certo viés a opiniões, podendo ignorar evidências que levem a indicar que a origem do problema de qualidade do produto tem uma razão distinta da prevista pelo profissional. Portanto a predição de qualidade pode ser uma ferramenta poderosa a ser aliada com a experiência dos especialistas para melhor apontar áreas passiveis de melhoria, que são abundantes.

Para se ter uma melhor noção da abrangência do impacto que a falta de qualidade pode acarretar em uma empresa toma-se de exemplo a Toyota, empresa conhecida por sua confiabilidade estima uma perda de 770 a 880 milhões de dólares na América do norte apenas em 2012 com *recalls* (TELI et al., 2012). O problema se expande muito além da indústria automobilística sendo previsto por muitos que o custo da falta de qualidade se encontra entre os 15 e 33 % apenas avaliando os custos explícitos (MAHMOOD et al., 2014). Ou seja, desconsiderando todos os custos em sofrimento causado por acidentes e outros custos não evidenciados nos cálculos.

Com o tema geral do projeto discorrido é possível se concentrar no problema particular da WEG. Um motor elétrico é de forma simplificada dois componentes que interagem entre si, o estator e o rotor. O estator nada mais é que um esqueleto de metal que carrega as diversas bobinas de fios isolados que quando são expostas a uma corrente alternada geram um campo magnético que se desloca de forma rotativa. Esse campo por sua vez movimenta o rotor que é um ímã (permanente ou gerado por bobinas) que gira produzindo movimento. Para produzir as bobinas os fios de cobre ou alumínio precisam ser cobertos em um verniz isolante que evita que eles conduzam entre si. A problemática do trabalho se foca nas falhas ao passar o verniz nos fios, mais especificamente na formação de bolhas no verniz que estraga o fio e leva o mesmo ao sucateamento.

Figura 2 – Motor elétrico



Fonte:(NEOCHARGE, 2022)

O trabalho começa introduzindo a fábrica de fios, o que ela faz e como segue a linha de produção da mesma. Depois serão apresentadas as técnicas e teorias de *machine learning* e *datascience* que foram estudadas para o projeto. Então serão apresentados todos os passos que levaram a solução seguindo a *pipeline* de *datascience* apresentada na introdução. Com o projeto devidamente explicado serão discutidos os resultados, se foi possível ou não aplicar o projeto, o porque dele ter funcionado ou não e etc. Por fim, será feita uma conclusão do trabalho como um todo abrangendo globalmente tudo que foi visto.

2 WEG e o processo de fabricação de fios

A WEG é a maior produtora de motores elétricos da America Latina, sediada em Jaraguá do Sul SC é uma das poucas empresas brasileiras multinacionais a produzir bens de consumo. Começou apenas na área de máquinas elétricas porém com o passar do tempo expandiu seus negócios para englobar a automação industrial e residencial, geração, distribuição e transmissão de energia, além de tintas e vernizes (WEG, 2022).

A história da WEG começa em 1964 com a aquisição do parque fabril 1 em Jaraguá do Sul SC pelos fundadores da empresa Werner Ricardo Voigt, Eggon João da Silva e Geraldo Werninghaus, cujo inicias formam o nome da empresa. O negócio cresceu rapidamente iniciando exportações para o Uruguai, Paraguai, Equador e Bolívia na década de 70. Foi na mesma década que a WEG adquiriu seu parque fabril 2 e entra na bolsa de valores. Nos anos 80 a companhia expande sua área de atuação criando a WEG transformadores, energia, automação e química, ganhando seu primeiro prêmio "maiores e melhores"da revista exame. Em 90 a WEG abre suas filiais nos Estados Unidos e em diversos países da Europa solidificando ainda mais a empresa como uma multinacional. Essa tendência de internacionalização de seus negócios se fortalece na década subsequente com a abertura de fábricas no México, Portugal e Argentina. Atualmente a empresa vem expandindo seus negócios com a aquisição de diversas empresas, adicionando nomes ao grupo WEG se solidificando cada vez mais como uma gigante não só na área de motores elétricos, mas em todas as áreas que envolvem a energia elétrica, fabricando desde tintas, vernizes para fios e interruptores até grandes equipamentos para geração eólica, turbinas hidráulicas entre outros (WEG, 2022).

Figura 3 – Parque fabril 1 em 1964

Fonte: (WEG, 2022)

Das diversas unidades produtoras da WEG o parque fabril 2 é o maior de todos, responsável pela produção de diversos dos motores de uso genérico da companhia. O parque contém 63 prédios diferentes responsáveis pelos processos de fabricação dos motores como estamparia, usinagem.

Figura 4 – Parque fabril 2

Dos diversos processos de fabricação, um deles é de extrema importância devido ao seu alto impacto na qualidade final das máquinas elétricas, a produção dos fios. A fábrica

de fios é responsável por essa tarefa, modelando os fios em seus respectivos diâmetros e aplicando os diversos vernizes utilizados para isolar os condutores. A fábrica de fios produz fios de dois materiais, cobre e alumínio, devido ao alto valor destas matérias primas o aumento da qualidade e redução de descartes é vigorosamente almejado.

Figura 5 – Fios de cobre a esquerda e alumínio a direita

Fonte: André Correa Mafra, 2022

Diversas máquinas são utilizadas na fábrica de fios, isso acontece pois fios de materiais e espessuras diferentes dependem de processos e por consequência máquinas diferentes para realizar sua produção. Neste capítulo serão estudados os procedimentos da máquina em questão, além dos processos que a precedem.

A WEG começa sua fabricação com a chegada de fios muito grossos de cobre ou alumínio vindos de diversos fornecedores em grandes carretéis. Estes fios então passam pelo primeiro processo de trefilagem. A trefilagem é um processo para a redução de diâmetro dos fios que utiliza de tração junto ou não de aquecimento para aumentar o comprimento do material sendo trefilado. O resultado da primeira trefilagem é um fio mais fino que é armazenado em carretéis mostrados na figura 6.

Figura 6 – Primeiro processo de trefilagem

No segunda trefilagem o fio grosso passa por diversas roldanas sendo devidamente lubrificado até alcançar o diâmetro requerido pelo lote solicitado pelos consumidores das outras fábricas da empresa.

Figura 7 – Segundo processo de trefilagem

Ao sair da segunda máquina de trefilagem o fio passa por roldanas no teto chegando a máquina (figura 8). A máquina é responsável por aplicar verniz nos fios, este é responsável pelo isolamento do mesmo para então o fio ser utilizado nas diversas bobinas dos produtos WEG. A máquina contém 3 passos principais que são repetidos diversas vezes para formar as inúmeras camadas de verniz.

Figura 8 – Máquina do estudo

Primeiro o fio passa por uma fieira de esmaltação que aplica uma fina camada de verniz sobre o metal.

Figura 9 – Aplicação de verniz

Então o fio passa pela etapa de secagem. Nele os fios com verniz já aplicados passam por fornos (figura 10) que retiram o solvente e curam o verniz selando-o no fio. Após passarem pelos fornos os fios se deslocam pela área de secagem a frio, onde o fio aberto ao ambiente é resfriado com diversos exaustores retirando o vapor dos fios e circulando ar, facilitando a secagem (figura 8).

Figura 10 – Fornos de secagem

Depois de todas as camadas de verniz terem sido aplicadas e devidamente secas, o fio passa pelo sensor de qualidade (figura 11). Este é uma combinação de dois sensores, um sensor de continuidade que testa através do campo elétrico se o fio está corretamente isolado e um sensor de relevo, basicamente uma placa que movimenta caso exista alguma rugosidade. Esse sensor é responsável por checar a geração de bolhas nos fios.

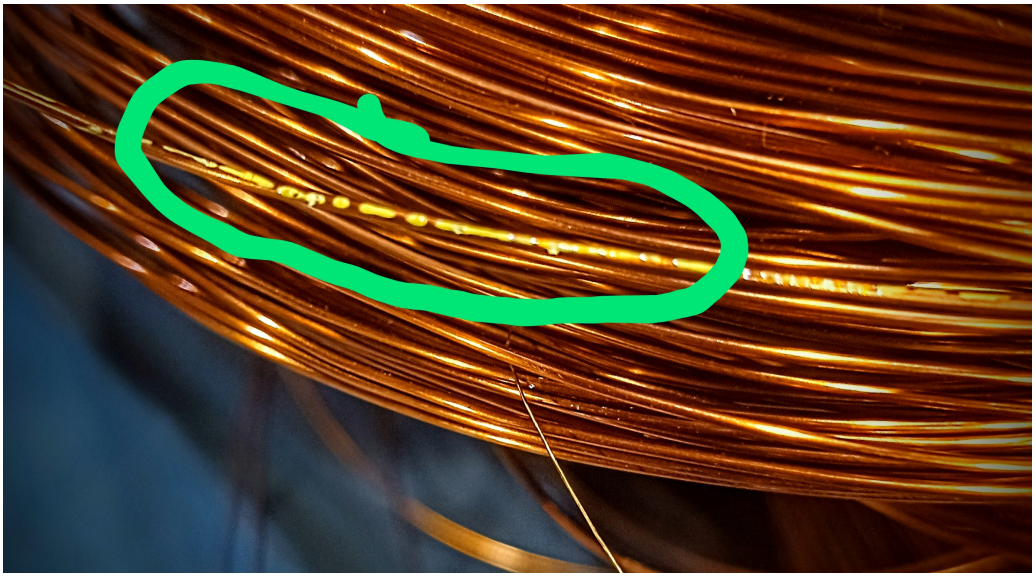
Figura 11 – Sensor de qualidade

Por fim o fio vai para a última máquina que enrola o fio no carretel. Essa máquina é interessante pois quando um carretel é completo não é necessário parar a produção, a máquina troca de carretel automaticamente, cabendo ao colaborador apenas mover o carretel cheio para o local adequado.

Figura 12 – Carretel sendo enrolado

Enquanto os fios são produzidos imperfeições e falhas são coletadas. Estas informações reprovam ou não um carretel e são armazenadas no banco de dados de ensaios WEG (ENS). Outros testes também são feitos para aprovar ou reprovar o carretel porém estes são feitos em laboratório.

Figura 13 – Bolhas no fio



Todo o histórico de qualidade, incluindo as falhas supracitadas são registradas, assim existe uma larga quantia de dados para explorar os motivos das falhas de produção que levam uma bolha a ser formada. As falhas de bolhas alvo do presente estudo são de grande importância devido a serem o motivo mais comum de descartes de fios, para 2022 foi calculado um custo de R$ 52.193,14 por ano por máquina segundo o analista André Correa Mafra, são cinco máquinas de fios de diâmetro menor e três de diâmetro maior, em outras palavras existe um grande incentivo financeiro para resolver o problema.

Portanto é possível avaliar as variáveis de processo e a qualidade da produção com o objetivo de correlacionar as variáveis de processo que possam levar a geração de bolhas. O projeto será focado em prever a geração das mesmas através das variáveis resposta encontradas em diversos bancos de dados da WEG. Para fazer isso será utilizado de classificação para identificar estados logo antes de acontecer as bolhas e assim poder prevê-las gerando economia de gastos com falhas de produção.

3 Teorias e métodos estudados

3.1 Introdução

Para tentar correlacionar as variáveis de processo da máquina com os dados de qualidade dos produtos será seguido os passos da *pipeline* de *machine learning* demonstrados em Krauss(KRAUSS et al., 2020). Assim este capítulo será subdividido em cada passo apresentado na *pipeline* com todas as teorias e modelos que foram explorados explicados em suas respectivas etapas. O capítulo começa com a coleta e integração de dados, então é apresentado o pre processamento e análise dos mesmos, em seguida é demonstrado as técnicas e algorítimos para treino para por fim apresentar os métodos de avaliação de qualidade dos algorítimos.

Para desenvolver a aplicação duas linguagens de programação se apresentaram como opções viáveis de escolha: R e Python. R é uma linguagem muito conhecida no meio acadêmico, foi desenvolvida especificamente para estatística e assim é muito completa nessa área. Gráficos oferecem uma melhor visualização dos dados com as bibliotecas padrão que associados a grande velocidade da linguagem e por consequência a capacidade de lidar com *datasets* massivos fazem do R uma linguagem bastante adequada. Porém a linguagem é complexa e muito específica, o R não lida muito bem com consumo de serviços web e não é muito utilizado na indústria. Python é uma linguagem mais versátil (OZGUR et al., 2017), trabalha muito bem com outras linguagens e tem uma plataforma muito boa para consumir serviços web. Além disso por mais que a visualização de gráficos no Python é mais difícil que em R diversas bibliotecas de terceiros amenizam este problema. Outro grande benefício do Python é sua ampla utilização e facilidade de aprendizado, a linguagem é de alto nível e é amplamente utilizada na indústria tendo uma comunidade que oferece amplo suporte para os mais variados temas. portanto foi decidido que a linguagem Python é mais apta para o problema e oferece maior suporte tanto na web como dos colegas de trabalho para o desenvolvimento da aplicação em questão.

Por fim é necessário gerar um esqueleto para a aplicação. Os chamados *frameworks* são estruturas onde programadores podem desenvolver seus programas seguindo a formatação do *framework*. Existem diversos destes para *machine learning*. Devido a experiência da equipe foi escolhido utilizar o Kedro. Kedro é um *framework* para criar código de ciência de dados previsível, reprodutível e sustentável (ALAM et al., 2022). Existem diversas vantagens do kedro além da organização e de seguir os padrões da indústria para este tipo de programa. Umas delas é a criação do diagrama da *pipeline* do projeto que traz uma forma visual bastante interessante do que o programa faz, recurso bastante utilizado no projeto.

3.2 Coleta e integração

Para qualquer projeto de *data science* é necessário acessar os dados aos quais se deseja utilizar. Geralmente são armazenados em bancos de dados que utilizam de linguagens específicas dependendo da relação entre os objetos do banco. Os bancos não relacionais como o próprio nome diz não relacionam os objetos armazenados enquanto os relacionais fazem o oposto. Os bancos de dados não relacionais não foram utilizados no projeto logo não serão explorados nesta seção enquanto o acesso e manipulação de dados dos bancos relacionais será explorado abaixo.

Devido a ampla utilização e maturidade da tecnologia, bancos de dados relacionais tem uma linguagem padrão com normas já bem estabelecidas. O SQL (Structured Query Language) é como o próprio nome diz uma linguagem de consulta estruturada. Desenvolvida nos anos 70, foi padronizada em 1986 e atualizada em 1989, em 1992 a linguagem foi finalizada se tornando a linguagem padrão para utilizar este tipo de banco de dados (MELTON; SIMON, 1993). Diversas companhias utilizam do modelo para criar, acessar e modificar bancos de dados. Entre elas duas foram estudadas nesse projeto.

A primeira é a Oracle SQL. Uma das primeiras arquiteturas para lidar com SQL é um serviço pago que oferece amplo suporte da empresa para facilitar sua utilização, "uma arquitetura fácil, elegante e de alto desempenho para acessar, definir e manter dados"(ORACLE, 2022). A ferramenta também conta com a biblioteca CX_oracle que possibilita a utilização dos bancos Oracle no formato de orientação a objetos do Python. A Oracle SQL armazena um dos bancos da WEG utilizados no projeto e por isso foi bastante explorada.

A segunda é o PostGresSQL. Assim como a arquitetura da oracle é rápido e muito bom para armazenar uma grande quantia de dados, sua interface é simples com um *design* moderno. Porém diferente do *Oracle* é gratuito e é um grande candidato a guardar os dados do projeto quando o mesmo for para produção.

Outra forma utilizada no trabalho de se obter dados são os *webservices*. Um serviço web é um programa que realiza alguma tarefa disponível através da internet (TIHOMIROVS; GRABIS, 2016). No caso em específico o serviço oferecido são a base de dados armazenados nos bancos. Assim para se comunicar com estes serviços se utiliza de um protocolo SOAP ou REST para acessar as funcionalidades do programa.

O protocolo SOAP é o mais antigo dos dois, utiliza os protocolos de comunicação HTTP e SMTP. Os dados são enviados através de um arquivo XML, utilizando uma série de padrões no arquivo para comunicação, sendo o mais comum o WSDL (TIHOMIROVS; GRABIS, 2016). No WSDL é onde as funções e seus parâmetros utilizados na aplicação são especificados. O usuário utiliza dessas funções para consumir o serviço requisitado. Devido ao grande número de padrões o XML é um arquivo relativamente pesado e o protocolo

SOAP vem perdendo espaço para o REST. Ainda assim foi utilizado no projeto já que alguns dos serviços WEG foram desenvolvidos nesse padrão.

O protocolo REST é mais utilizado recentemente, sendo mais flexível já que aceita dois tipos de arquivo o XML e o JSON. Os arquivos de tipo JSON tem *headers* menores e consomem menos processamento. O REST utiliza padrões definidos pelo fornecedor da aplicação e utiliza dos protocolos HTTP para acessá-las. Devido a essa flexibilidade os protocolos REST vem dominando cada vez mais o mercado. Para o consumo de certas api´s da weg são utilizados os protocolos rest, recebendo como resposta um JSON, que podem ser transformados em dicinários do python com um pequeno esforço.

Por fim é necessário armazenar as informações dos arquivos produzidos pelo programa. Para os dados crus inicias foram utilizados arquivos .csv. São arquivos muito simples com valores separados por vírgula, e, por isso são muito rápidos do python ler e modificar. Entretanto esta simplicade tem um preço. Arquivos csv não são tipadosm ou seja, uma vez salvos o python não saberá que tipo de dado cada coluna era podendo gerar erros de tipagem, como, por exemplo, tratar uma coluna de datas como se ofsse uma coluna de *strings*. Para evitar isso os dados tipados foram salvos em arquivos diferentes discutidos abaixo.

Para salvar os dados entre os diversos passos do trabalho dois tipos de arquivos foram considerados. Os arquivos .xlsx (excel) e os arquivos .parquet (parquet). Os arquivos do excel tem certas vantagens como uma visualização mais fácil e a possibilidade de mexer nos dados pela própria plataforma excel. Porém os mesmos são lentos para serem lidos e são limitados a aproximadamente um milhão de linhas. Os dados do tipo parquet foram feitos para se trabalhar com ciência de dados no python são extremamente rápidos de ler e carregar e também são tipados apenas perdendo no aspecto de visualização para o excel. Com isso em mente foi estabelecido que seria uma melhor opção utilizar o parquet do que o excel, entretanto um problema no python evitou que isso fosse uma possibilidade. Mesmo com as bibliotecas que lidam com parquet instaladas não foi possível acessá-las pelos ambiente virtual, por fim não foi possível arrumar o problema com o curto período de tempo disponível para o término do trabalho. Por causa disso os arquivos intermediários na construção do modelo utilizaram de arquivos excel.

3.3 Pre processamento e análise

3.3.0.1 Bibliotecas

Preprocessamento de dados é um dos grandes passos no *pipeline* de *datascience*. Os dados geralmente são coletados de fontes diferentes sendo apresentado de diversas formas. Portanto os dados nunca foram modificados e podem conter erros ou dados nulos (GARCÍA; LUENGO; HERRERA, 2015). Logo é necessário remodelar as informações

com o intuito de lidar com estas inconsistências. No caso do Python existe uma biblioteca muito utilizada nas áreas que envolvem *data science*. A biblioteca Pandas é um conjunto de módulos que oferecem diversas funções para mexer com tabelas. A biblioteca é baseada em dicionários Python e trabalha incrivelmente bem tanto com arquivos e bancos de dados como com as bibliotecas de *machine learning* disponíveis para a linguagem de programação. Portanto é um grande facilitador para fazer as operações necessárias para limpar os *datasets*, possibilitando resolver com mais facilidade os diversos problemas habituais de preprocessamento de dados.

Analisar as informações obtidas antes de passá-las a um algorítimo de A.M. é um dos grandes trabalhos do cientista de dados. Para avaliar um grande número de informações é muito recorrente a utilização de métodos estatísticos e gráficos.

A biblioteca padrão de gráficos do Python é a Matplotlib, é uma biblioteca poderosa que confere ao usuário um largo leque de opções para diversos tipos de gráficos e layouts, oferecendo ao usuário uma experiência um pouco trabalhosa porém muito completa. Outra opção interessante é o Seaborn. Construído utilizando o Matplotlib, o Seaborn oferece uma alternativa que reduz o esforço para montar gráficos, oferecendo layouts já prontos mais bonitos e complexos. O seaborn foi utilizado para criar uma ponte entre o matplotlib e o pandas construindo os gráficos utilizando as próprias tabelas do pandas. Por causa disso é uma ótima biblioteca para gráficos de distribuição, assertividade entre outros.

Em conjunto com métodos visuais os métodos estatísticos tem uma função indispensável na interpretação das informações de qualquer problema de ciência de dados ou previsão de resultados. Uma biblioteca muito comum para estes métodos é o Scipy. Utilizado inclusive na criação do Pandas, esta biblioteca é muito poderosa para a aplicação de diversos métodos matemáticos e científicos. Foi muito utilizada neste trabalho devido a facilidade de uso e amplo suporte da comunidade e da documentação, que se encontram em abundância online.

Por fim com as informações devidamente interpretadas é necessário aplicar as diversas transformações necessárias para adequar melhor os dados aos algorítimos de A.M.. A Sickit-learn é uma biblioteca com foco em aplicação de algorítimos de A.M., porém a mesma tem diversos módulos voltados a modificação de *datasets* para melhor tratá-los para aplicação dos modelos. Esta flexibilidade da biblioteca possibilita utilizar a mesma para diversos passos da pipeline de *machine learning* evitando problemas de compatibilidade e oferecendo diversos métodos para a preparação dos dados para futuramente aplicar nos modelos. Por causa destas inúmeras vantagens a biblioteca é a mais utilizada em A.M. no python e por isso tem uma comunidade muito ativa que facilita a resolução de problemas.

No caso do processamento e preparação dos dados duas expansões da biblioteca scikit foram amplamente utilizadas no trabalho. A sickit-optimize e a imbalanced-learn. A sickit-optimize é uma biblioteca de otimização de hiper parâmetros. Ela utiliza de

diversos algorítimos, que podem ser escolhidos de acordo com o problema para reduzir uma função resposta (atrelada a assertividade do modelo de A.M.) e assim melhorar os hiper parâmetros de um algorítimo de *machine learning* aumentando a assertividade do modelo sem a necessidade de mexer nos dados. A imbalanced-learn é outra extensão da sickit-learn. Esta bilioteca oferece métodos para equilibrar *datasets* desequilibrados, ou seja, onde uma classe é mais frequente que outra. Isso pode gerar problemas pois um modelo pode se tornar tendencioso e acabar por frequentemente classificar a classe majoritária o que causa baixa assertividade na classificação da classe minoritária. Para evitar isso a biblioteca oferece diversos métodos de preencher os dados da classe minoritária ou remover dados da majoritária para melhor equilibrar o *dataset*.

3.3.0.2 Métodos estatísticos

Para conseguir um bom modelo de inteligência artificial é necessário isolar as variáveis que tem mais influência no problema em questão. É válido ressaltar que todos os testes com as variáveis resposta são correlações entre variáveis categóricas e variáveis contínuas o que limita os modelos que podem ser utilizados no problema.

O primeiro modelo estatístico estudado foi a correlação bilateral de ponto. Esta correlação é o valor de Pearson para variáveis dicotômicas e contínuas. Assim como em Pearson os valores da correlação variam entre -1 e 1, onde -1 é uma correlação negativa perfeita e 1 o inverso (KORNBROT, 2014). Além disso valores abaixo de 0.3 são considerados de correlação fraca ou inexistente. Este tipo de medição foi considerado para aplicação no projeto, porém certas restrições interferem no projeto. Uma delas é que as variáveis de interesse estejam próximas de uma distribuição normal. O problema é que isso foi vigorosamente testado e sempre com resultado negativo. As distribuições das variáveis explicativas estão bastante distantes de um distribuição normal, tendo diversos picos de concentração de valores e frandes caudas causadas por um grande número de *outliers*. Por isso o método não pode ser utilizado.

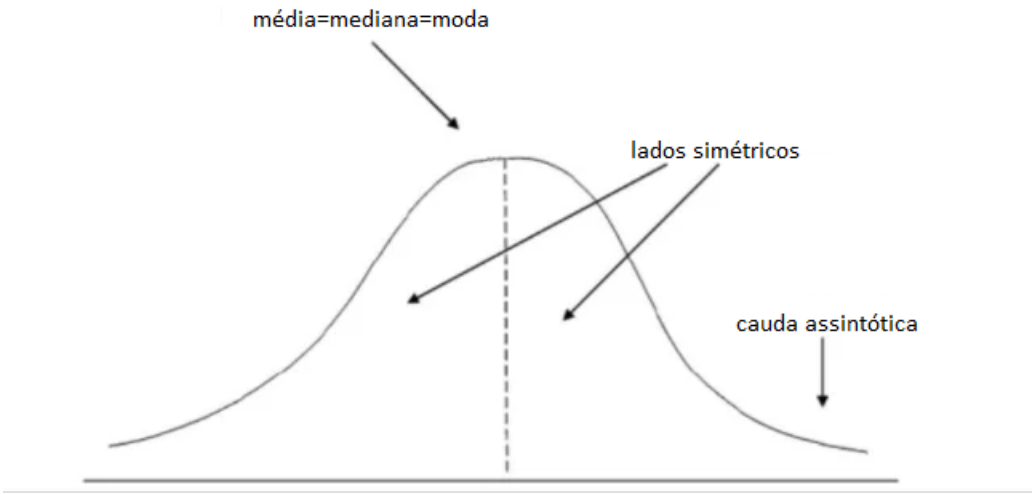
Outros métodos estudados avaliam a variância da população, estes conferem a dispersão das variáveis com interesse de classificar se as variações entre as variáveis independente e dependente são randômicas ou correlacionadas. O primeiro método estudado é muito conhecido na estatística, é a análise de variância (ANOVA). O método avalia as médias de cada grupo classificado para tentar estimar se existe ou não uma correlação. Todavia o procedimento necessita de uma distribuição normal. O teste de Kruskall-Wallis não necessita de uma distribuição normal devido a utilizar as variáveis ranqueadas em vez de usar os valores em si (HECKE, 2012), tendo apenas a necessidade de que os indivíduos da população sejam verdadeiramente randômicos e que existam pelo menos 3 grupos diferentes. Devido a o número de categorias na variável resposta do problema ser igual ou menor ao mínimo necessitado pelo método para calcular correlação, foi estabelecido que o

método não seria utilizado. Por fim seria necessário encontrar modelos não paramétricos que consigam classificar classes binárias.

Para seguir as restrições acima dois métodos recorrentes na literatura foram estudados: a correlação ranqueada de Spearman e τ de Kendall. A correlção de Spearman é a versão não paramétrica da correlação de Pearson. Ou seja, a correlação de Spearman calcula a relação linear entre os ranques das variáveis estudadas, como em Pearson os valores variam entre -1 e 1, onde o sinal indica direção da correlação e seu tamanho indica a "força". É valido comentar que valores idênticos das variáveis paramétricas receberão o mesmo ranking. A correlação de Kendall é bem parecida com a de Spearman, também é não paramétrica e também serve para correlação entre variáveis contínuas e categóricas. Porém a correlação de Kendall avalia o número de pares concordantes e discordantes. Pares concordantes e discordantes podem ser definidos comparando duas medidas das amostras ranqueadas i e j. Um par de observações é dito concordante caso $x_i > x_j$ e $y_i > y_j$ ou $x_i < x_j$ e $y_i < y_j$, sendo x e y as variáveis correlacionadas. Pares são ditos discordantes no caso contrário as desigualdade supracitadas (PUTH; NEUHÄUSER; RUXTON, 2015).

Para lidar com as variáveis de forma paramétrica é necessário que os dados tenham uma distribuição normal, por isso é importante testar as distribuições das variáveis envolvidas no estudo. Um modo muito simples de testar normalidade é avaliar a frequência da distribuição dos dados graficamente, porém este método depende da interpretação do avaliador e pode oferecer um resultado não satisfatório. Uma forma de complementar as interpretações gráficas são os métodos matemáticos, A grande maioria deles checam a distorção e a curtose da curva. A distribuição normal é perfeitamente simétrica ou seja, quando seccionada no meio as duas partes da curva são espelhadas de forma idêntica. A distorção mede o quão assimétrica é esta divisão, sendo zero caso a curva seja perfeitamente simétrica. A curtose é um parâmetro que define a cauda assintótica da distribuição normal, ou seja as pontas onde a distribuição vai tendendo a zero. Quanto maior a curtose de uma curva mais distantes as suas pontas estão do formato padrão da distribuição normal. O teste de D'agostino-pearson avalia estes parâmetros tendo como única restrição que o número de amostras não seja muito pequena (menor que 20), regra que o modelo do projeto obedece muito bem (ZAIONTZ, 2022). Os modelos de teste para distribuição normal retornam o p-valor com a hipótese nula ($p>0.05$) sendo a distribuição normal dos dados.

Figura 14 – Distribuição normal



Fonte: (PSICOMETRIA ONLINE, 2022)

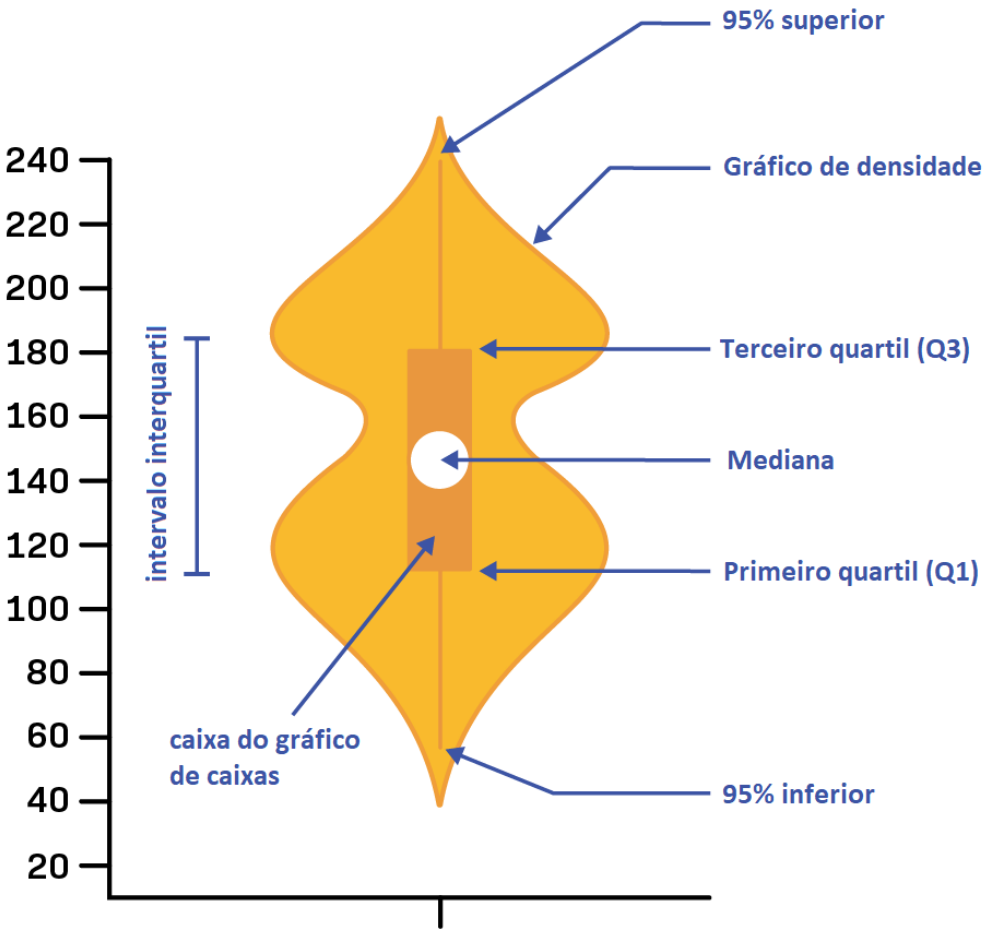
Caso uma distribuição não seja normal ainda é possível aplicar métodos paramétricos através da transformação das amostras para forçá-las a esta distribuição. Dependendo de quais são os problemas que levam um arranjo de dados a se afastarem da normal é possível aplicar métodos específicos que atenuem este afastamento. Um caso muito comum é a existência de *outliers* que acabam por expandir a cauda assintótica da distribuição normal. Uma forma muito utilizada para evitar isso é utilizar do desvio padrão (σ) para cortar dados muito distantes da média. É frequente observar valores de 3 σ na literatura pois estes mantém 99.7% dos dados o que remove dados muito ruins mas mantém causas não usuais de variação no sistema (BAKAR et al., 2006). Outro motivo de não conformidade muito comum são distorções ou curtoses na curva de distribuição. Diversas vezes os dados ainda podem ser utilizados em modelos paramétricos porém precisam ser ajustados para se conformar melhor a disposição de dados requerida. Por consequência foi criada a transformação box-cox, que utiliza de diversas potências diferentes para melhor encaixar a curva estudada a padrão. A única restrição da transformação é que todos os números das amostras sejam positivos. Ambos modelos foram utilizados no estudo de dados do projeto já que os mesmos se adéquam a todas as restrições. Por mais que os mesmo não tenham oferecidos resultados satisfatórios.

Antes de treinar um modelo existem algumas boas práticas que auxiliam na acurácia dos algorítimos, entre elas a escalação dos dados. Que é um processo com o intuito de colocar as variáveis independentes na mesma escala. Em diversos projetos variáveis com escalas muito diferentes podem acabar distorcendo os resultados de um algorítimo. Existem duas características importantes para as variáveis quando se fala em escalas, a

magnitude que é o valor da variável e a unidade. É possível perceber que por causa destas características as mudanças de mesma magnitude em variáveis diferentes tem influências completamente diferentes, por exemplo 1 quilo representa muito menos massa do que uma tonelada. Portanto para evitar que estas diferenças distorçam os algorítimos de A.M. duas transformações são utilizadas: a normalização e a padronização. A normalização escala todos os valores entre 0 e 1, transformando o maior valor de uma variável em 1 e o menor em 0. Este método é um pouco problemático pois é bastante suscetível a influência de *outliers* e por isso não foi utilizado no projeto. A padronização transforma todas as variáveis de acordo com uma curva normal de média 1 e desvio padrão 0, este método é mais resistente a *outliers* e é mais utilizado para *machine learning* como um todo. O sickit learn tem um módulo para este tipo de transformação chamado *prerpocessing*.

Por último será comentado sobre gráficos de violino. Não são métodos estatísticos mas são formas visuais de se avaliar a distribuição de alguma variável qualquer. O gráfico de violino é composto de um corpo onde sua espessura representa a frequência de observações e uma barra interna composta de três partes. A linha fina representa onde estão 95% da distribuição da variável. A barra mais grossa representa os 50% da distribuição mais comuns e por fim tem o ponto dentro da barra que é a mediana. As características explicadas no texto podem ser vistas na figura 15.

Figura 15 – Gráfico de violino



Fonte: (LABXCHANGE (HARVARD ONLINE), 2021)

3.3.1 *Over* e *under sampling* de dados

Qualquer algorítimo de aprendizado de máquina supervisionado utiliza de dados de treino para aprender e assim conseguir desempenhar a função para a qual foi programado. Porém se os dados oferecidos ao programa forem muito desbalanceados o programa terá uma certa tendência a classificar a classe majoritária. Este é um problema sério para a classificação de bolhas devido a grande maioria dos dados coletados não darem bolhas levando a A.M. a ser tendenciosa para este estado não classificando as bolhas em potencial que é exatamente seu propósito. Para evitar isso existem diversas estratégias de balanceamento de *dataset* que agem removendo dados da classe majoritária (*undersampling*) ou gerando dados a mais para a classe minoritária (*oversampling*). Existem diversos tipos de algorítimo para ambas as estratégias e as utilizadas no trabalho serão comentadas a seguir.

Uma das formas mais simples de equilibrar um *dataset* seria criar ou remover dados

aleatoriamente. Esta estratégia randômica pode parecer muito ruim, porém além de sua simplicidade esta estratégia traz algumas vantagens extras. Entre elas uma que se destaca é redução de *overfitting*. Overfitting é quando um modelo de A.M. se adapta demais aos dados de treino e acaba por utilizar ruídos na classificação, isso leva a uma baixa acurácia no *dataset* de teste e pode estragar com um modelo. Já que a estratégia cria/tira dados de forma randômica, o algorítimo é obrigado a se adaptar flexibilizando suas regras e reduzindo a chance de *overfitting*.

Near Miss é um método de *undersampling* que remove os dados da classe majoritária que tem a menor distância média entre os outros pontos da mesma classe (MANI; ZHANG, 2003). O modelo faz isso com o intuito de retirar dados que não contém grandes quantidades de informação útil para o modelo.

Neighbourhood cleaning rule é outro modelo de *undersampling*. Ele também utiliza das distâncias entre as amostras porém ele remove as amostras da classe majoritária que estão na borda de decisão (LAURIKKALA, 2001), basicamente a ideia contrária do método de *Near Miss*. A justificativa deste método é remover ruído e dados mal classificados para melhor gerar a fronteira de decisão entre as classes. Este método não tem como objetivo principal deixar as classes do mesmo tamanho, tem apenas como onjetivo deixar a borda entre as amostras mais clara. *One sided selection* segue a mesma estratégia porém este tem a diferença de balançear completamente as classes, não apenas deixando a fronteira de decisão mais clara como terminando com um mesmo número de amostras em todas as classes.

SMOTE é uma técnica de *oversampling* que tem como estratégia utilizar dos vizinhos de uma amostra para gerar amostras falsas com características parecidas com a verdadeira. Este método tem como outra ideia interessante utilizar áreas pouco populadas do espaço de estados das amostras, em outras palavras o modelo adiciona amostras em lugares onde as amostras são raras porém a classificação é bastante clara (CHAWLA et al., 2002). Outro método muito parecido é o de *SVMSMOTE*, ele faz a mesma coisa que o modelo de SMOTE porém utiliza da técnica de A.M. de máquinas de vetores de suporte para gerar as amostras raras.

Por último tem-se o modelo de *ADASYN* assim como o modelo de *SMOTE* também utiliza dos vizinhos para gerar novas amostras, porém tem uma grande diferença. O modelo decide quantas amostras devem ser geradas de acordo com a dificuldade de aprendizado daquela parte do espaço de estados, utilizando da distribuição das classes para gerar estas amostras falsas (HE et al., 2008) nas áreas mais críticas.

3.4 Algorítimos de previsão

Com os dados devidamente limpos e reduzidos deve-se estudar quais modelos podem ser utilizados para prever o resultado de qualidade da máquina. Para isso é necessário avaliar qual problema se tem em mãos, quais suas características e quais são suas restrições. Sabe-se que a aprendizagem será supervisionada, já que os dados da máquina tem a variável resposta no seu histórico. A variável resposta é categórica, por isso apenas modelos de classificação funcionam com este tipo de problema, já que os mesmos tentam dividir em grupos as amostras de acordo com as variáveis explicativas. Modelos de regressão servem para tentar prever a evolução de variáveis contínuas logo não se aplicam ao problema. Por fim deve se avaliar se o projeto tem muitos ou poucos dados? se são confiáveis e se representam bem a população geral que os mesmos tentam emular. Com essas questões em mente foram considerados diversos algorítimos discutidos abaixo.

3.4.1 Regressão logística

A Regressão Logística é um dos modelos mais conhecidos em *machine learning* graças a sua simplicidade. Diferente de outros modelos este é um método estatístico que mede a probabilidade de uma variável categórica ter um certo valor baseado em um número de variáveis explicativas. É chamada de logística pois utiliza do logaritmo natural para calcular a probabilidade do valor da variável dependente. Existem três tipos de regressão logística: a binária, multinomial e multinomial ordinária. Onde a primeira consiste em uma variável resposta de apenas dois valores enquanto as outras aceitam múltiplos valores, sendo que a ordinária tem os valores ranqueados enquanto a multinomial comum não tem ordem. Devido ao formato não linear da resposta deste tipo de algorítimo, é comum que os métodos computacionais retornem a exponencial da saída para tornar o entendimento do valor mais claro (IBM, 2022). A fórmula que calcula a probabilidade da regressão logística é dada na equação 3.1:

$$\ln\left(\frac{P_i}{1-P_i}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_k K_k \tag{3.1}$$

Equação 3.0 – Equação da regressão logística

3.4.2 Redes Neurais

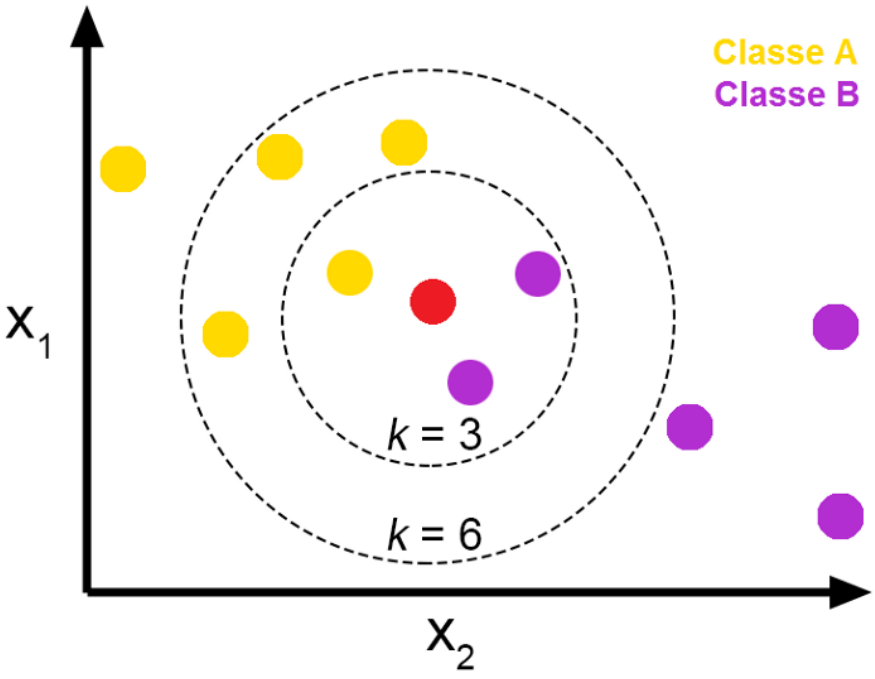
Um modelo extremamente famoso na literatura são as redes neurais. Inspiradas pelos primeiros modelos de processamento sensorial do cérebro uma rede neural pode ser criada simulando redes neurais em um computador (KROGH, 2008). Assim como nosso cérebro é ótima para identificar padrões em imagens e linguagens. Devido a sua fama diversos modelos de redes foram desenvolvidos no passar dos anos para as mais diferentes tarefas variando em tipos e números de camadas, número de neurônios e tipo de resolução

de problema. Infelizmente a flexibilidade de redes neurais exige uma grande quantia de estudo para ser bem adaptada e exige um grande número de dados para ser corretamente treinada. Por causa destes problemas redes neurais podem obter resultados bem distantes dos satisfatórios.

3.4.3 *K Nearest Neighbors*

K Nearest Neighbors (KNN) é um algorítimo que tenta encontrar um número pré definido de amostras de treino mais próximas da amostra de teste utilizando qualquer forma de medida de distância métrica entre os pontos de treino e teste para classificá-lo (PEDREGOSA et al., 2011). Basicamente tenta relacionar as classes dos vizinhos mais próximos a amostra desconhecida. É um algorítimo muito simples em essência, porém como é o caso de diversos algorítimos de inteligência artificial classifica muito bem diversos tipos de problemas.

Figura 16 – Exemplo gráfico de KNN



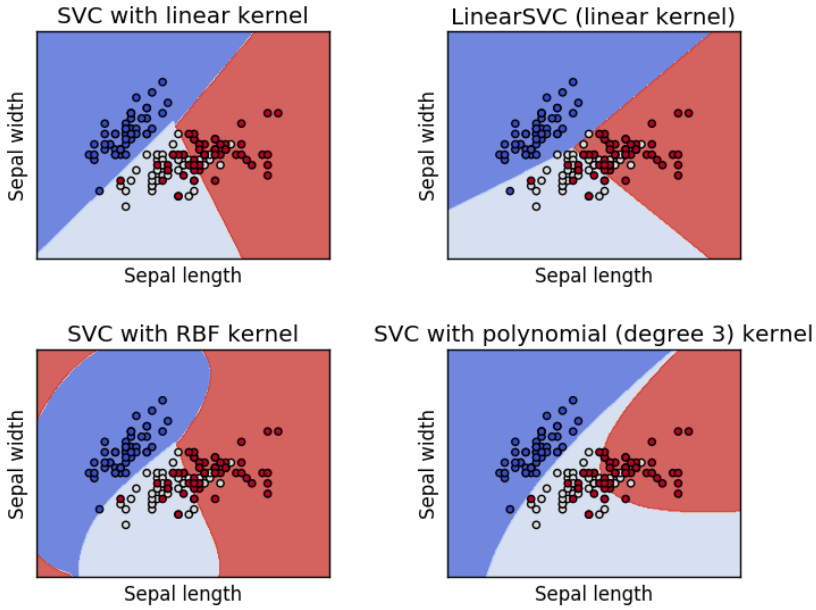
Fonte: (JOSé, 2018)

3.4.4 Máquinas de Suporte

Máquinas de suporte são outro algorítimo extremamente comum em *machine learning*. É um modelo que tenta desenhar diversas curvas de grau pré-estabelecido para dividir as classes de um modelo. É um algorítimo de alta acurácia em especial para *datasets*

pequenos. Porém esta alta assertividade vem com um alto custo computacional de treino que pode tornar inviável em certas aplicações.

Figura 17 – exemplo de classificador por máquinas de suporte



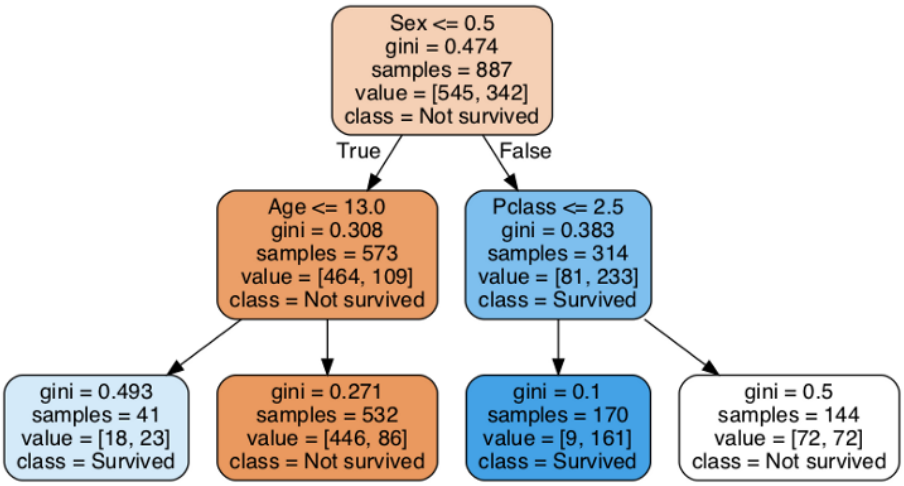
Fonte: (PEDREGOSA et al., 2011)

3.4.5 Modelos Singulares

Os modelos simples são aqueles que quando unidos de alguma forma formam os modelos *ensemble* que foram de extrema importância para o desenvolvimento do projeto. Este tipo de modelo em média não tem uma grande acurácia sozinho porém pode se tornar um algorítimo poderoso quando utilizado em conjunto.

As árvores de decisão são um modelo que recebe diversas *features* de entrada e divide os dados de acordo com certas perguntas geradas pelo algorítimo. Estas perguntas acabam por dividir os dados de treino até um ponto que as folhas da árvore se tornem satisfatoriamente puras, ou seja tenham a grande maioria dos indivíduos na folha de apenas uma classe (KINGSFORD; SALZBERG, 2008). Árvores de decisão podem ser utilizadas para regressão e classificação e são utilizadas como modelo base para diversos algorítimos *ensemble* como *XGBoost*, *GradientBoosting*, *RandomForests* e *AdaBoosting*, sendo basicamente o pilar de todos os modelos ensemble do trabalho.

Figura 18 – Exemplo de árvore de decisão



Fonte: (DUIF, 2020)

3.4.6 Modelos *Ensemble*

Modelos *ensemble* são um conjunto de modelos mais simples que de alguma forma se unem para gerarem uma previsão mais precisa após o treino. 3 técnicas de *ensemble* existem na literatura: *bagging*, *boosting*, *stacking*.

O tipo *bagging* pode ser pensado como uma votação. Vários modelos simples treinados com parte ou todo o *dataset* "votam"em qual previsão acreditam ser a correta e então pela soma de todos os "votos"a previsão com mais probabilidade é a qual é retornada pelo algorítimo *ensemble*. Um destes algorítimos de classificação utilizado no trabalho foi o de florestas de decisão. Este método se utiliza de diversos modelos de árvores de decisão treinados com partes diferentes do *dataset*. No fim estes modelos "votam"e a decisão da floresta será a categoria que foi classificada mais vezes (BREIMAN, 2001).

O método de *boosting* é diferente, em vez de serem previsões em paralelo são previsões em série onde cada algorítimo simples utiliza de um anterior para alterar sua previsão até chegar a uma previsão final. Em geral estes modelos utilizam do erro do algorítimo passado ou dos exemplos difíceis de se classificar para melhorar o estimador final. Exemplos deste tipo de modelos são o *AdaBoosting*, *GradientBoosting* e seu irmão mais potente o *XGBoost*. O *AdaBoosting* se baseia nas classificações erradas para ajustar seu algorítimo enquanto o *boosting* de gradientes utilizam os exemplos difíceis de se classificar para aumentar sua acurácia. A grande vantagem do *XGBoost* é sua capacidade de guardar as variáveis pré ordenadas e sua grande capacidade de aparar nós ruins o que faz do algorítimo um dos mais bem sucedidos em diferentes aplicações (CHEN et al., 2015).

Por último se tem o *stacking*. Este tipo de modelo tem a ideia de utilizar a saída de

outros modelos como entrada para si (PAVLYSHENKO, 2018). A grande vantagem deste tipo de estratégia é conseguir unir as forças de modelos diferentes para melhor classificar as amostras na saída. Devido a necessidade grande de dados para treinar os múltiplos modelos sem utilizar dados de teste que foram usados no treino dos modelos de entrada, este tipo de estratégia geralmente utiliza algorítimos simples que não necessitam de uma grande quantia de dados para ser bem ajustado.

3.4.7 Otimização de Parâmetros

Após a escolha de um modelo para solucionar o problema proposto é necessário ajustar seus parâmetros. Estes parâmetros mudam certos comportamentos como o passo de reajuste do modelo, número de estimadores em um modelo *ensemble* entre outros. A forma mais comum de ajustar estes parâmetros é através da experiência e avaliação dos modelos do cientista de dados que com seu trabalho aumenta a assertividade de previsão do *dataset* de teste. Entretanto esta forma demanda muito tempo e pode ser utilizada em conjunto com outras. Pode-se pensar em um ajuste grosso e outro fino, onde o ajuste grosso é feito por algum tipo de algorítimo e o ajuste fino é feito pelo cientista de dados ou vice-versa.

Um método muito comum de força bruta é o *grid search*. Este é um método que recebe uma lista de valores para cada parâmetro que se deseja otimizar e então o modelo testa todos. É um modelo para ajuste grosso do modelo conseguindo resultados bons mas não ótimos.

Outros modelos se baseiam em minimizar uma função de custo, como por exemplo o negativo da assertividade de um algorítimo, para tentar encontrar um conjunto de parâmetros ideal para o modelo. Diversos algorítimos podem ser utilizados para isso entre eles os mais comuns são, *gradient boosting*, *Baeysian optimization* e árvores de decisão. Ambos modelos de árvores ja foram discutidos no trabalho só sobrando o modelo de estatística Bayesiana. Este método estatístico basicamente se baseia que os parâmetros do modelo são independentes entre si. Isso pode parecer uma simplificação grosseira porém em grande parte dos casos pouco afeta a acurácia final de um modelo otimizado desta forma.

3.5 Avaliação de Modelos

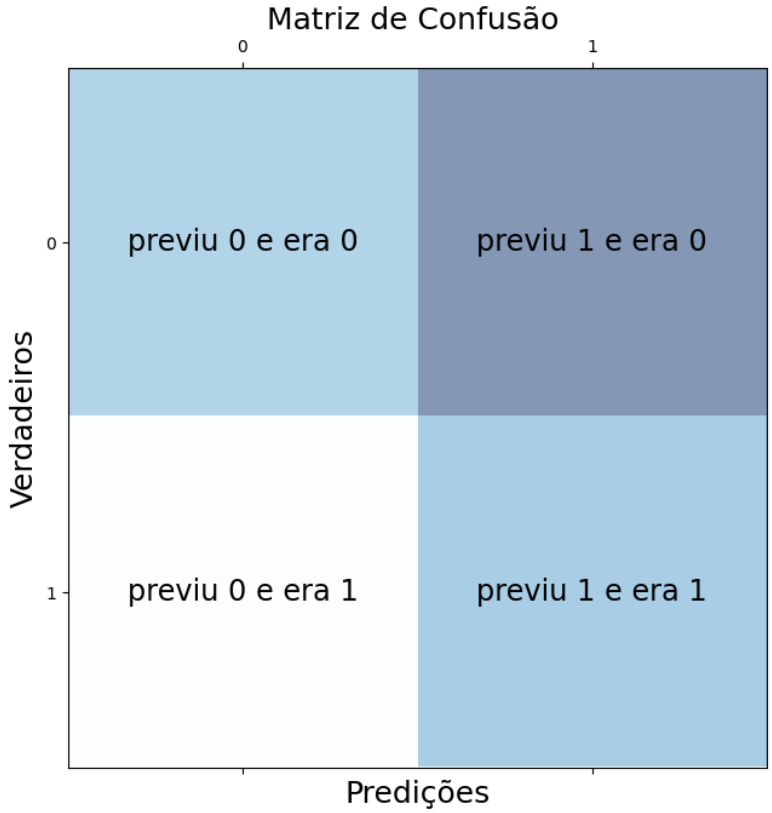
Um trabalho de A.M. é um ciclo constante de teste e melhoria. Por causa disso é necessário ter uma boa forma de medir o quão bom um modelo é. Isto pode ser bem mais complexo do que parece, um exemplo bom disso é a acurácia geral para um modelo muito desbalanceado. No caso das bolhas raramente elas acontecem logo se um algorítimo de A.M. chutasse sempre que não fosse dar bolha a acurácia seria bastante alta porém

o modelo não estaria prevendo nada. Pensando nisso foram utilizadas diversas métricas de sistema levando em consideração a qualidade, custo computacional e até magnitude das medições de qualidade de um algorítimo, sendo assim os seguintes modelos foram utilizados para avaliar os modelos gerados.

O primeiro deles é o *F-1 Score*. Este método faz a média do *recall* com a acurácia para retornar um valor entre 0 e 1, sendo quanto mais alto melhor. O *Recall* é a medida de quantos valores foram classificados corretamente em relação a quantos não foram para a mesma classe. Portanto é a medida de valores verdadeiros positivos divididos pelos valores verdadeiros positivos mais os falsos negativos para uma determinada classe. Então se faz a média deste valor com a acurácia, total de acertos dividido pelo total de observações da classe estudada, para retornar o valor final. O bom deste tipo de método é que ele é muito bom para avaliar modelos que tratam de *datasets* desbalanceados.

Outro método muito importante são as matrizes de confusão. É formada uma matriz onde as linhas são os valores reais das observações e as colunas os valores preditos pelo algorítimo. Quanto mais valores na diagonal principal melhor o algorítimo classificou as classes. É um modelo poderoso pois oferece uma forma visual de comparar todas as previsões (corretas ou não) do algorítimo. Assim se existe uma classe que não pode ser falsa negativa por exemplo, no caso de diagnóstico de câncer, a matriz de confusão deixa bem evidente.

Figura 19 – Exemplo de matriz de confusão



Todos os modelos anteriores são evoluções de algum modelo de acurácia. Sendo esta apenas uma relação entre valores que o algorítimo previu corretamente e o número total de observações.

4 Modelos e acurácia

4.1 Introdução

Neste capítulo será apresentado o que foi feito no projeto. Com toda a base teórica apresentada no Capítulo 3 é possível discorrer sobre as atividades que levaram a conclusão do projeto. Serão apresentadas as linhas de raciocínio, os erros e mudanças, assim como partes do código que foram importantes ou interessantes no desenvolvimento do programa. Como os capítulos anteriores este também receberá a divisão baseada nos passos da *pipeline* apresentada na introdução. Portanto começarão sendo apresentadas as fontes de onde os dados foram retirados além do processo para obtê-los e integrá-los. Será discutido o pré processamento dos dados, que inclui a remoção de dados ruins ou irrelevantes além de diversas estatísticas para melhor limpar e compreender o modelo, em seguida serão apresentados os algorítimos testados e por fim serão avaliados os modelos comparando-os por sua acurácia e desempenho.

4.2 Coleta e integração

Buscando a melhoria continua a WEG coleta informações sobre basicamente todos os processos que tem informações relevantes para empresa. Isso vai de informações de vendas e estoque, até a qualidade e velocidade dos processos. Porém, devido ao tempo e motivo pelos quais esses dados começaram a ser coletados, os mesmos se encontram em bancos de dados distintos com formatação diferente e com diversas formas de consumir suas informações. Assim o primeiro passo do processo foi conseguir acesso a esses bancos, para então combinar as informações em um *dataset* que pode ser utilizado na resolução do problema.

O primeiro banco acessado foi o gerenciador de chão de fábrica ou SFM. O gerenciador de chão de fábrica da WEG é uma ferramenta desenvolvida para a coleta de dados nas linhas produtivas da empresa. Ela tem como objetivo proporcionar o acompanhamento em tempo real da produção, gerando a integração entre a camada gerencial e de produção da empresa. O GCF tem como objetivo principal reduzir os custos indiretos de obtenção de dados, proporcionando assim um ambiente mais propenso para tomada de decisões e melhoria contínua. Para alcançar estes objetivos o SFM coleta diversos dados, para o caso específico da fábrica de fios os dados são divididos em três tabelas principais: as variáveis de processo, eventos e dados de produção.

As variáveis de processo são os dados coletados de todos os sensores das máquinas

da fábrica de fios. Aqui são medidas todas as temperaturas e velocidades das linhas, todos os dados de produção são armazenados neste banco com exceção dos dados do *motorscan*.

O *motorscan* é um sensor da WEG desenvolvido para coletar as informações de motores elétricos. Este tipo de sensor mede a temperatura, velocidade e vibração com o intuito de oferecer informações úteis sobre a saúde do motor. A WEG tem um banco de dados de manutenção onde são armazenados os dados dos *motorscan*. Existem duas formas de consumir estes dados, pedindo para os responsáveis e recebendo um arquivo .csv do último ano dos motores que foram pedidos ou acessando a aplicação via WEB.

Figura 20 – MotorScan

É possível acessar a aplicação do *motorscan* através de uma API em formato REST. A aplicação contém duas requisições para consumir seus dados. A primeira é uma requisição do tipo *post* para mandar os dados de autenticação do usuário e receber um *token* que libera o usuário a consumir os dados dos *motorscans* por um dia. Se utiliza este token no cabeçalho da segunda requisição para ter acesso aos dados. Esta requisição de tipo *get* recebe a data de inicio e fim da pesquisa além do ID do motor, que seleciona o motor correto no meio dos milhares de motores que tem este sensor instalado. Quando isso é feito a API retorna um arquivo do tipo JSON com as informações do motor naquele período que então pode ser integrado aos dados do SFM.

Diferente dos dados dos *motorscans*, o SFM foi oferecido através de um usuário que se conectava diretamente ao banco de dados da aplicação. O banco Oracle é acessado em Python através de uma biblioteca chamada cx_oracle, que oferece diversas aplicações como cursores e ferramentas de pesquisa para acessar a linguagem relacional com linguagem orientada a objetos. Porém seu uso é relativamente complicado e assim junto ao cx_oracle foi utilizada a biblioteca SQlAlchemy que é uma ORM que facilita a utilização das ferramentas de banco de dados no python. Foi criada uma classe onde sua instância é a conexão com o BD. Para consultar o banco através do SQLAlchemy é necessário a criação de uma *engine* que é a conexão propriamente dita. Esta *engine* pode ser passada ao Pandas como argumento da função *pd.read_sql* que junto do argumento de consulta retornam um *dataframe*, esta estrutura de dados que a biblioteca pandas utiliza para manipulação dos mesmos, é basicamente uma tabela excel para Python.

Com acesso ao SFM é necessário consultar a máquina utilizada no projeto. Já que existem as *views* pré-programadas pelo DBA do SFM, a consulta da máquina foi relativamente simples, coletando apenas as colunas de informações importantes. Entretanto ao consultar os dados, seu formato não era ideal para a utilização em um algorítimo de *machine learning*. Cada variável de processo estavam em linhas diferentes, sendo diferenciadas por uma coluna chamada "ProcessVariableDescription"com sua hora de leitura em outra coluna. Assim cada variável de processo da máquina tomava uma linha diferente com tempos parecidos, porém não iguais, e foi trabalhoso unir estas variáveis corretamente. Foi feito primeiro uma divisão entre cada variável de processo pegando cada tipo e separando em um *dataframe*. Assim a coluna de tempo de cada *dataframe* era comparada linha a linha com as dos outros. Caso a diferença entre elas fosse suficientemente pequena (menores que o intervalo de tempo entre as aquisições do SFM) elas eram modificadas recebendo um tempo único, como se tivessem sido medidas exatamente no mesmo momento. Com isso realizado agora era possível unir cada *dataframe* em um único, transformando as datas em apenas uma coluna e oferecendo a cada variável de processo uma coluna com seu respectivo nome.

Tabela 1 – Transformação do SFM

ProcessVariableDescription	Avg_value	Date
Variável 1	Valor 1	1/10/2020 11:10:15
Variável 2	Valor 2	1/10/2020 11:10:13
Variável 3	Valor 3	1/10/2020 11:11:05
Variável 1	Valor 4	1/10/2020 11:25:04
Variável 2	Valor 5	1/10/2020 11:24:13

↓

Variável 1	Variável 2	Variável 3	Date
Valor 1	Valor 2	Valor 3	11:10:15
Valor 4	Valor 5	...	11:25:04

Com a tabela de variáveis de processo devidamente formatada foi consultada a tabela de eventos. Os eventos da máquina são muito importantes pois eles que informam qual a situação da máquina em uma faixa de tempo. Nem sempre a máquina está produzindo, em diversos momentos ela está parada para manutenção, ou por troca de *setup*. Quando ela não está operante as temperaturas dos fornos caem, os ventiladores podem ser desligados e etc. Assim deve-se analisar estes dados para remover estes períodos não desejados além dos momentos próximos a eles quando a máquina ainda não alcançou seu regime de equilíbrio. Os dados da tabela de eventos vêm com cada linha representando um evento, nesta linha tem o nome do evento, a data do seu início e do seu término. Para unir os dados desta tabela as variáveis de processo, foram comparadas as datas da tabela formatada das variáveis de processo ao início e término do evento. Todas as linhas que se encaixassem neste período de tempo foram rotuladas com o nome do evento.

Tabela 2 – Adição de eventos ao SFM

Variável 1	Variável 2	Variável 3	Date	Eventos
Valor 1	Valor 2	Valor 3	11:10:15	Operando
Valor 4	Valor 5	Valor 6	11:25:04	Operando
Valor 7	Valor 8	Valor 9	11:40:12	Manutenção

Coletados os dados do SFM é necessário conseguir os dados de qualidade dos fios. O ENS é o sistema de ensaios WEG, nele são encontrados todos os dados de qualidade de todos os lotes produzidos na empresa. No caso da máquina, cada carretel produzido é armazenado com seus dados de qualidade. Para acessar o ENS primeiramente é necessário pedir acesso ao gerente da seção. Cada funcionário da WEG recebe um cadastro e senha

que podem ser utilizados para diversos serviços dentro da empresa, quando o acesso ao ENS é liberado o cadastro do funcionário se torna um usuário e senha válidos para acessar os ensaios de qualidade. Para utilizar os dados do ENS é oferecida uma API em formato SOAP que tem suas requisições especificadas em um wsdl. Para consumir este tipo de API com o python se utiliza a biblioteca zeep. Para comunicar a biblioteca com o servidor utilizando SOAP primeiro se cria uma instância de transporte, que é passada como parâmetro junto com as autenticações de usuário para uma instância de cliente utilizada no zeep para acessar a API. Com acesso a API é necessário oferecer o URI para especificar no arquivo XML qual recurso da API se deseja consumir. No caso da aplicação, foi utilizado primeiro a requisição *TesteSearchRequest*. Para ligar o zeep a ela se utiliza uma instância de *bind*. Com todas as instâncias feitas é possível fazer a requisição a API. Esta requisição recebe como parâmetro as datas, o código do parque fabril, o código do módulo, da máquina e da linha. O recurso retorna um arquivo XML com todos os testes feitos no período entre as datas oferecidas nos parâmetros. Para evitar congelar o servidor com um pedido de informações muito grande foram feitos diversos pedidos pequenos dentro de um *for* com um tempo de espera entre cada iteração. Os testes de cada carretel tem um ID que foram armazenados em um arquivo.

Com estes IDs é possível utilizar a segunda requisição do ENS. A *ExportTestMeasuredRequest* é uma requisição para retornar todos os dados de um teste ela recebe como parâmetro um "*group_id*"e um "*characteristics_id*". O "*group_id*"é um parâmetro para estabelecer o que você quer do teste, como por exemplo, nome, medições, *status* e etc. O "*characteristics_id*"é um parâmetro para escolher o que pegar do "*group_id*", pois, por exemplo, dentro de medições existe o nome da medição, o valor e a unidade. Para o projeto foi necessário escolher as medições e pegar aquelas medições de bolhas grandes e pequenas e o horário de cada teste. Para fazer isso foi necessário fazer a requisição para cada teste e armazena-los em outro arquivo .csv.

Então é necessário unir os dados do ENS com os dados do SFM. Para fazer isso primeiro foram criadas duas colunas na tabela do SFM, uma de bolhas pequenas e outra de bolhas grandes. Essas colunas foram todas preenchidas com zero. Então foi feito um laço nos dados do ENS que pegavam a data e o número de bolhas e a somava na linha do SFM com a data mais próxima a data da bolha no ENS. Este processo foi feito para todas as linhas do ENS até unir ambas as tabelas em uma única.

Por fim é necessário gerar as variáveis resposta do trabalho. A ideia do problema é pegar quando aconteceram bolhas e classificar os tempos prévios a elas como um estado de "próximo de gerar bolhas". Foram utilizados quatro períodos anteriores as bolhas para a correlação das variáveis resposta e explicativas, 10 minutos, 30 minutos, uma hora e duas horas. Múltiplos períodos foram escolhidos para encontrar uma boa balança entre avisar com antecedência o suficiente entretanto não ficar longe demais das bolhas a ponto

de classificar estados normais como próximos de estados defeituosos.

Tabela 3 – Adição de eventos ao SFM

Variável N	Date	bolhas grandes	bolhas pequenas	10 minutos	30 minutos
Valor 1	2021/04/08 10:55:23	0	0	0	1
Valor 2	2021/04/08 11:00:10	0	0	0	1
Valor 3	2021/04/08 11:05:15	0	0	0	1
Valor 4	2021/04/08 11:10:04	0	0	1	1
Valor 5	2021/04/08 11:15:12	0	0	1	1
Valor 6	2021/04/08 11:20:15	0	2	1	1
Valor 7	2021/04/08 11:25:43	0	0	0	0

Assim a etapa de coleta e integração dos dados foi concluída. O diagrama do caminho e transformação dos dados desta etapa estão disponíveis na figura do pipeline de coleta e integração.

4.3 Pré-processamento

Com todos os dados em apenas um *dataframe* é possível salvar estas informações em excel para então passar para o preprocessamento dos dados. O excel salvo tem 120702 linhas com mais de cinquenta variáveis. Uma visualização do mesmo nos mostra diversos problemas.

Tabela 4 – *Dataset* sujo

Tensão no fio	umidade relativa	temperatura	Date	30 minutos	eventos
0		420,2	2021/04/08 10:55:23	1	OPERANDO
0		420,1	2021/04/08 11:00:10	1	OPERANDO
0		420,9	2021/04/08 11:05:15	1	OPERANDO
0		419,5	2021/04/08 11:10:04	1	OPERANDO
0		419,8	2021/04/08 11:15:12	1	OPERANDO
0		420,1	2021/04/08 11:20:15	1	OPERANDO
0		418,4	2021/04/08 11:25:43	0	OPERANDO

Como se percebe na tabela 4 existem diversas variáveis nulas. Isso acontece pois certas variáveis da máquina foram adicionadas posteriormente ao inicio dos dados no SFM. Outro problema é que os dados do *motorscan* tem apenas alguns meses de antecedência a quando foram coletados e por isso existem diversas linhas nulas nas colunas provindas do mesmo. Para evitar uma perda grande demais de dados foi decidido por separar o *dataset* em três para fazer a limpeza e avaliação dos mesmos. Foram separados em um *dataset* todas as variáveis que existiam desde o início dos dados (sfm_early), as variáveis

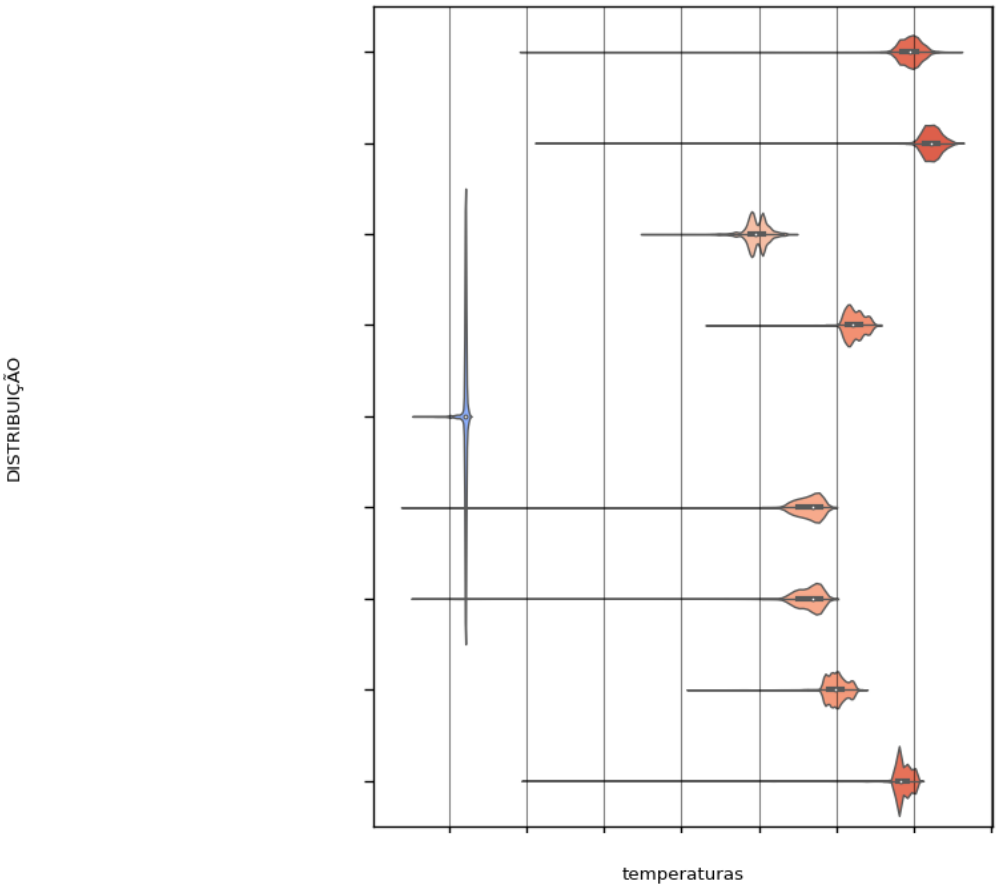
adicionadas posteriormente ao SFM (sfm_later) e por fim as variáveis do *motorscan*. Após estes *datasets* serem limpos serão avaliadas as importâncias e correlações entre as variáveis para por fim separar e limpar o *dataset* original apenas com as variáveis que tenham relevância para o problema.

Todos os dados supracitados passaram por uma *pipeline* de limpeza. Primeiramente foram removidas as linhas em que a máquina não estava operando ou estava em regime transiente. Para decidir o que era regime transiente foi oferecido apoio de um do analistas responsáveis pela fábrica de fios André Correa Mafra. Segundo ele os eventos de manutenção, falta de programação e feriados/concessões eram eventos que tomavam mais tempo para retornar a máquina ao seu regime permanente. Assim foram removidas todas as linhas do arquivo 4 horas após e antes destes eventos, é válido remover os períodos antes pois a máquina passa por um tempo de desativação. Outros eventos que não representam a operação da máquina também foram removidos, como por exemplo a troca de ferramental. Estes são eventos que não desativam a máquina completamente e foram removidos apenas 30 minutos após estes eventos, já que não necessariamente existe uma operação prévia a estas ocorrências, o tempo anterior a estes eventos não foi removido.

Depois da remoção destes eventos foram retiradas as linhas nulas de cada *dataset*. Seguido pela remoção de todas as variáveis negativas. Foram removidas todas as variáveis abaixo de zero pois nenhuma grandeza medida na máquina faria sentido ser negativa dada as condições da máquina. Em seguida foram removidas as colunas constantes, obviamente valores constantes não irão oferecer informação útil ao problema. Por fim foram removidas as variáveis que eram zero. Neste caso nem todas as colunas passaram por esta operação, já que é possível a vibração de um motor no *motorscan* ser zero. Esta afirmação foi confirmada por um dos trabalhadores do *motorscan* Lucas Henrique dos Santos Tavares.

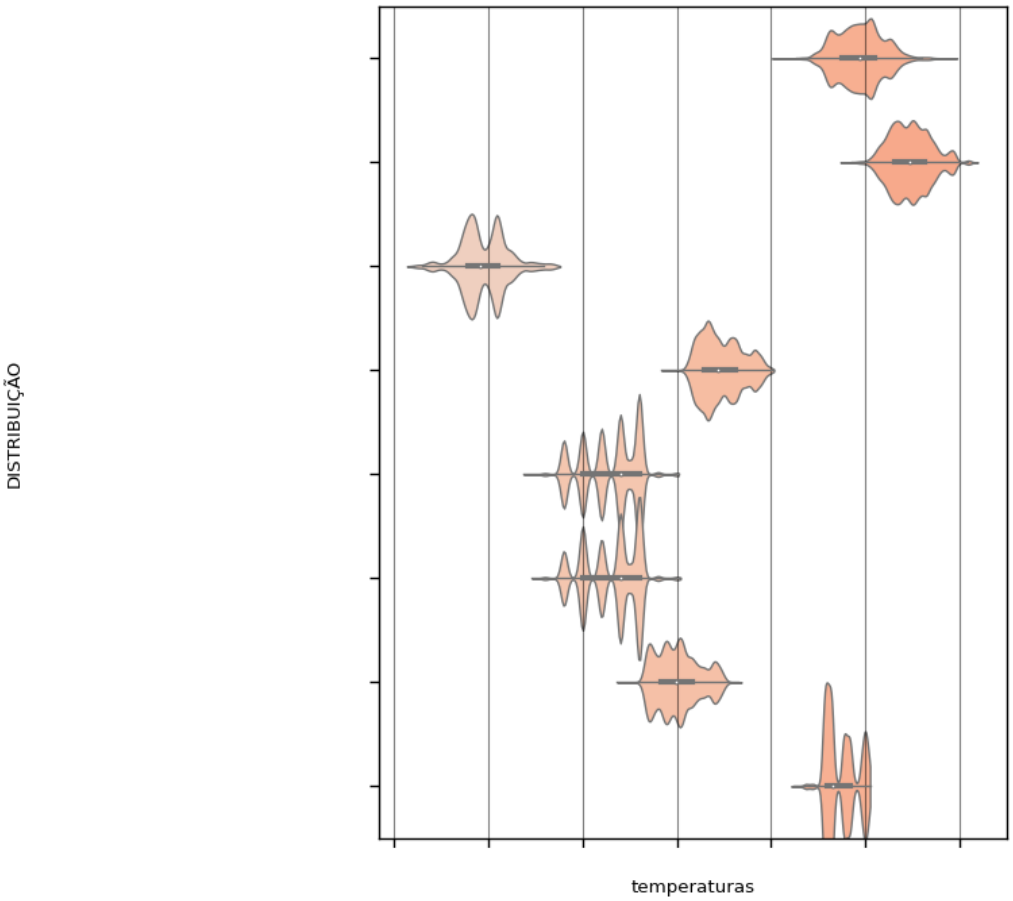
Depois de passar os *datasets* pelo *pipeline* de limpeza as distribuições foram avaliadas através de gráficos de violino. Na figura 21 é demonstrado um exemplo, as outras distribuições se encontram no apêndice Apêndice A.

Figura 21 – Distribuição das temperaturas



Como se percebe as distribuições tem uma longa cauda o que denuncia *outliers* extremamente foras do comum, estes valores provavelmente são resquícios de quando a máquina não estava operando, ou momentos que a máquina não estava operando porém o evento não foi indicado no SFM. Para limpar estes valores foi utilizada da técnica de 3 σ. É uma técnica de remover *outliers* que mantém causas infrequentes que possam atuar na variável resposta sem manter dados ruins (BAKAR et al., 2006). Assim pode-se notar a redução das caudas com esta estratégia na figura 22.

Figura 22 – Distribuição das temperaturas sem os 3 σ



Como fica evidente nas imagens não seguem de forma alguma uma distribuição normal porém devido as vantagens dos métodos estatísticos paramétricos ainda foi tentado modelos de normalização de distribuição. O modelo escolhido foi o box-cox, diversos λ´s foram escolhidos com o intuito de aproximar os arranjos a uma normal, porém sem sucesso. Por causa da baixa qualidade da distribuição dos valores o método de ANOVA para testar correlação entre variáveis foi descartado.

No lugar dos métodos paramétricos, modelos não paramétricos menos precisos tiveram de ser utilizados. Para conseguir correlacionar variáveis continuas com uma variável categórica de resposta dois modelos se mostraram como boas opções dadas as restrições do problema: o teste de Spearman, e o de Kendall-tau. Caso o p-valor de ambos seja menor que 0,5 é possível dizer com 95% de certeza que existe uma correlação entre a variável de processo medida e a variável de resposta. Os testes também retornam um valor que indica a força da relação entre as variáveis. Foi considerado utilizar o teste de Kruskal-Wallis, porém seus valores não adicionaram muita informação em relação aos outros testes e por isso não foi considerado nas decisões subsequentes aos testes de correlação. A tabela 5 demonstra as relações de força seguindo (MASTROTHANASIS, 2020) para o teste de

Kendall-tau e (STATSTUTOR, 2022) para o de Spearman.

Força	Spearman	Kendall-tau
muito fraco	<0.19	<0.10
fraco	0.2-0.39	0.10-0.19
médio	0.4-0.59	0.2-0.29
forte	>0.6	>0.3

Tabela 5 – Tabela de correlação de forças

Para fazer os testes de correlação foram utilizadas todas as variáveis resposta (10, 20, 60 e 120 minutos). Um exemplo das correlações entre as variáveis de temperatura e respostas é demonstrado na figura 23. Na figura 24 são demonstrados os p-valores para as mesmas. O resto das correlações e p-valores estão no Apêndice B

Figura 23 – Correlação das variáveis de temperatura utilizando kendall-tau

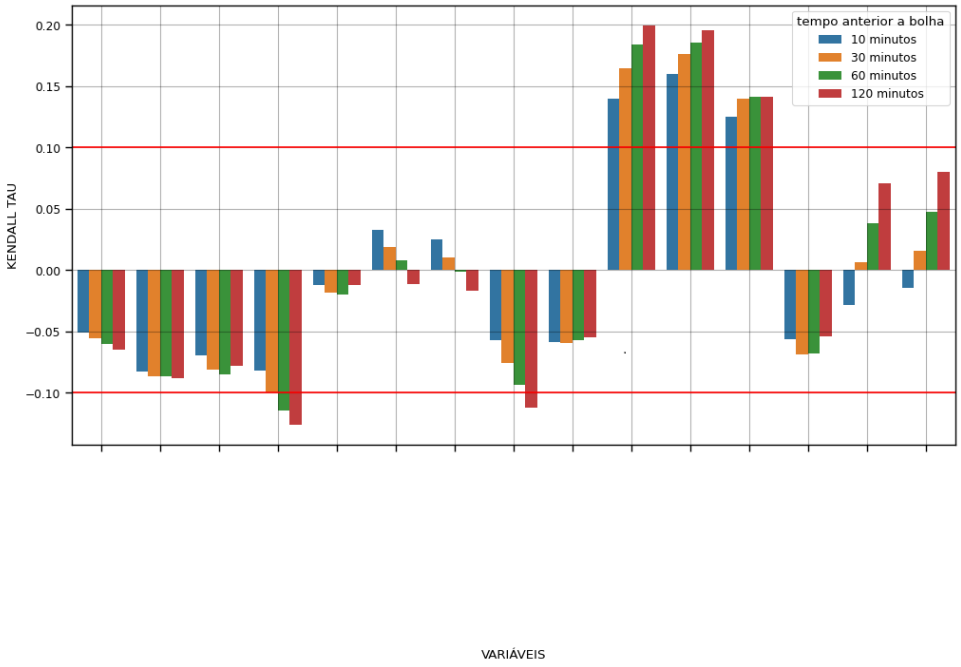
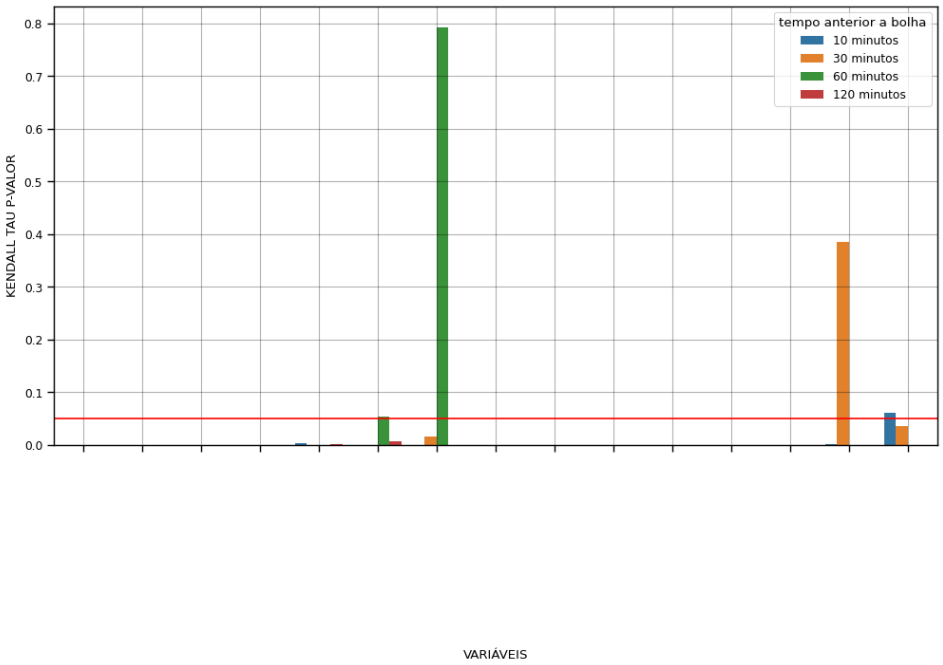


Figura 24 – P-valor das variáveis de temperatura utilizando kendall-tau



Como é possível perceber existe uma grande discrepância entre a correlação para cada variável explicativa. Pensando nisso se teve a ideia de separar em diversos *datasets* levando em consideração a força de correlação e quando as variáveis foram adicionadas. Disso foram formados 6 grupos diferentes de variáveis explicativas.

O primeiro grupo foi formado por todas variáveis explicativas cujo o p-valor é menor que 0.05 para todas variáveis resposta e que o sinal das correlações concordem entre si. Foi pensado que faria pouco ou nenhum sentido uma variável explicativa aumentar a chance de bolhas com 10 minutos de antecedência porém aumentar a chance de bolhas 60 ou 120 minutos antes. O segundo é a mesma coisa porém sem as variáveis adicionadas posteriormente. Isso foi feito com o intuito de manter um maior número de dados após limpar os dados novamente.

O terceiro grupo é formado de todas as variáveis cujo pelo menos uma das correlações entre as variáveis resposta ultrapassassem a faixa muito fraca de correlação de kendall-tau. Isso foi feito porque se acreditou que seria um bom comprometimento entre número total de dados e o número de variáveis explicativas para o problema. Novamente o 4 grupo é a mesma coisa porém sem as variáveis adicionadas posteriormente.

Por fim o quinto grupo foi mantido apenas as variáveis onde todas as correlações ultrapassavam a faixa muito fraca. Aqui o intuito foi manter apenas as variáveis bastante explicativas maximizando a quantidade de dados para algorítimos que necessitam de uma grande quantia dos mesmos. Novamente o sexto grupo apenas remove as variáveis

adicionadas posteriormente.

Com as variáveis explicativas separadas foi feito um último tipo de limpeza. É muito comum em uma máquina industrial existir diversas medições que mensuram o mesmo fenômeno. Sendo assim as variáveis são extremamente correlacionadas entre si e não oferece uma grande quantia de informação extra manter as duas. Portanto para descobrir estas relações foi feita uma matriz de correlação para cada *dataset* separadamente. Seria melhor fazer apenas uma para os três *datasets* existentes, porém para correlacionar as variáveis os dados devem estar limpos e nesse caso sobraria pouquíssimas observações no final. Na figura 25 é demonstrada umas destas matrizes de correlação, as outras duas se encontram no Apêndice B.

Figura 25 – Matriz de correlação das variáveis do SFM adicionadas depois

É interessante observar que existem diversas variáveis fortemente correlacionadas, com um kendall-tau de 0.88 apenas na imagem 25. Porém com o intuito de não remover informações úteis, em especial de variáveis importantes para o problema foi escolhido retirar apenas aquelas com um kendall-tau igual ou superior de 0.9. Assim os 6 grupos de variáveis foram reformatados sem estas variáveis e assim a seleção das variáveis explicativas estava completo.

Com as variáveis escolhidas foi pego novamente o *dataset* original sem qualquer

forma de limpeza. As variáveis dos 6 grupos geraram novos *datasets* que foram passados novamente pela *pipeline* de limpeza, incluindo o filtro de 3 σ para gerar as entradas dos modelos de inteligência artificial. A *pipeline* de preprocessamento pode ser vista na figura.

4.4 Algorítimos de previsão

A seleção de modelos é um dos grandes passos do projeto. Existem certas restrições que devem ser seguidas para fazer o sistema funcionar. É válido lembrar que apenas prever bolhas não é o suficiente para a WEG. Do que adianta prever uma falha se o modelo não consegue oferecer uma solução para o problema? Pensando nisso se estabeleceu que o algorítimo de previsão precisa ter em sua composição pelo menos um modelo que consiga estabelecer regras claras de decisão que possam ser interpretadas por um colaborador para então o mesmo tomar uma ação.

Para estabelecer quais modelos seriam utilizados foi usada uma técnica de força bruta para testar os modelos. O problema em questão é um problema de classificação desbalanceada, assim existem 3 grandes características que devem ser avaliadas para encontrar a melhor assertividade para um modelo dado o nosso problema.

A primeira delas é a seleção de um modelo de *sampling*. Como foi dito o problema não é balanceado, ou seja, existem mais exemplos da classe sem bolha do que da classe com bolha. Os modelos de *sampling* tentam igualar as duas classes utilizando alguma estratégia para a diminuição ou aumento das classes. Como é possível imaginar cada modelo de A.M. terá algum modelo de *sampling* que deixará o *dataset* com as correlações mais claras e assim uma assertividade maior.

A segunda característica é qual *dataset* melhor funcionará com um algorítimo qualquer. As características dos dados acaba por ter uma forte conexão com a assertividade de um algorítimo, isso acontece pois certos algorítimos funcionam com um número maior de dados porém menor de variáveis explicativas, enquanto outros algorítimos funcionam ao contrário. Assim é necessário testar quais dados melhor exploram a capacidade de um algorítimo.

Por fim existe a otimização de hiperparâmetros. Cada modelo de aprendizado de máquina tem certos parâmetros que podem ser variados para aumentar a assertividade do modelo. Estes parâmetros modificam a estrutura do algorítimo para melhor se encaixar no problema em específico e assim evitar *over e underfitting* além de outros problemas que um modelo possa ter. Exitem diversas estratégias para otimizar hiperparâmetros, algumas manuais, outras de força bruta e por fim técnicas mais refinadas. Estas últimas são modelos que tentam reduzir uma função através da modificação dos hiperparâmetros dos modelos. Diversas operações podem ser utilizadas para este fim, as utilizadas no trabalho foram descenso do gradiente, árvores de decisão e árvores de decisão impulsionadas pelo

gradiente. Como todas as características passadas a escolha destes algorítimos também depende do modelo de A.M. que está se tentando melhorar a assertividade.

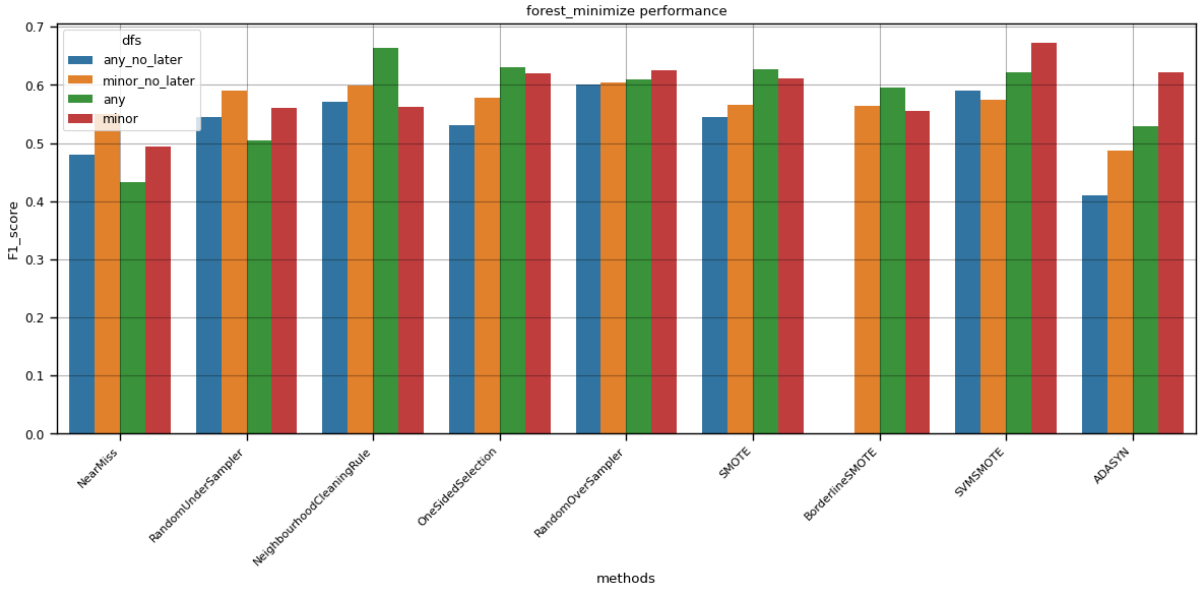
Sendo assim a ideia foi testar todas as combinações destas três características diferentes para encontrar o melhor modelo. Esta técnica será utilizada apenas uma vez para encontrar as melhores combinações, posteriormente, caso o modelo seja retreinado será utilizada apenas a melhor combinação destas características para retreinar o modelo. Abaixo serão demonstrados apenas os modelos que tiveram uma assertividade boa ou que tem uma aplicação útil devido a transparência do modelo.

No treino dos modelos se percebeu que dois *datasets* consistentemente performavam abaixo dos outros. Ambos os modelos que só pegavam as variáveis com maior influência nas bolhas foram descartados devido a baixa assertividade de todos os modelos testados com eles. Sendo assim sobraram apenas quatro *datasets* para serem testados.

4.4.1 Árvore de decisão

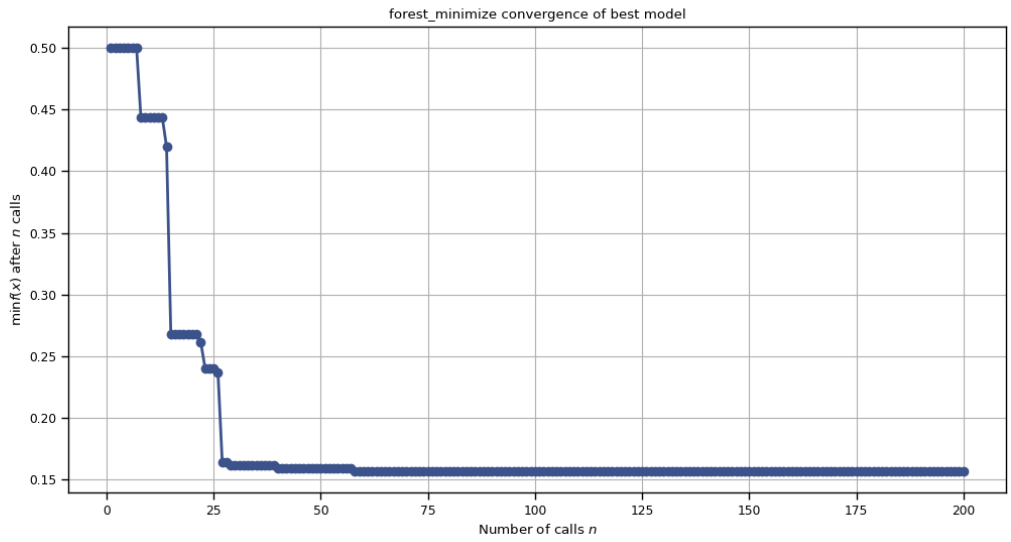
O modelo base para os modelos *ensemble* a árvore de decisão oferece uma ótima relação entre visualização e assertividade. É um algorítimo que gera nós de decisão de acordo com alguma regra que tem o intuito de dividir as classes. Ou seja no caso de bolhas são oferecidos um caminho que leva a causar bolhas, algo bastante útil para o problema. Porém a simplicidade do algorítimo tem um preço sendo um dos algorítimos de pior performance utilizado. Para avaliar a melhor combinação de características de *sampling* e *dataset* é apresentado o gráfico de barras (figura 26) da combinação de todos esses modelos com o melhor algorítimo de minimização encontrado. Isso foi decidido de acordo com a maior assertividade global do sistema.

Figura 26 – F1 score da árvore de decisão em relação aos *datasets* e *samplings*



Como se percebe, por ser um modelo mais simples a redução de variáveis explicativas do *dataset minor* ofereceu uma melhor assertividade para o modelo. Pode-se avaliar a evolução da função de custo para minimização na figura 27. É possível perceber que ainda muito próximo das últimas iterações o modelo continuava tendo melhorias (redução na função de minimização).

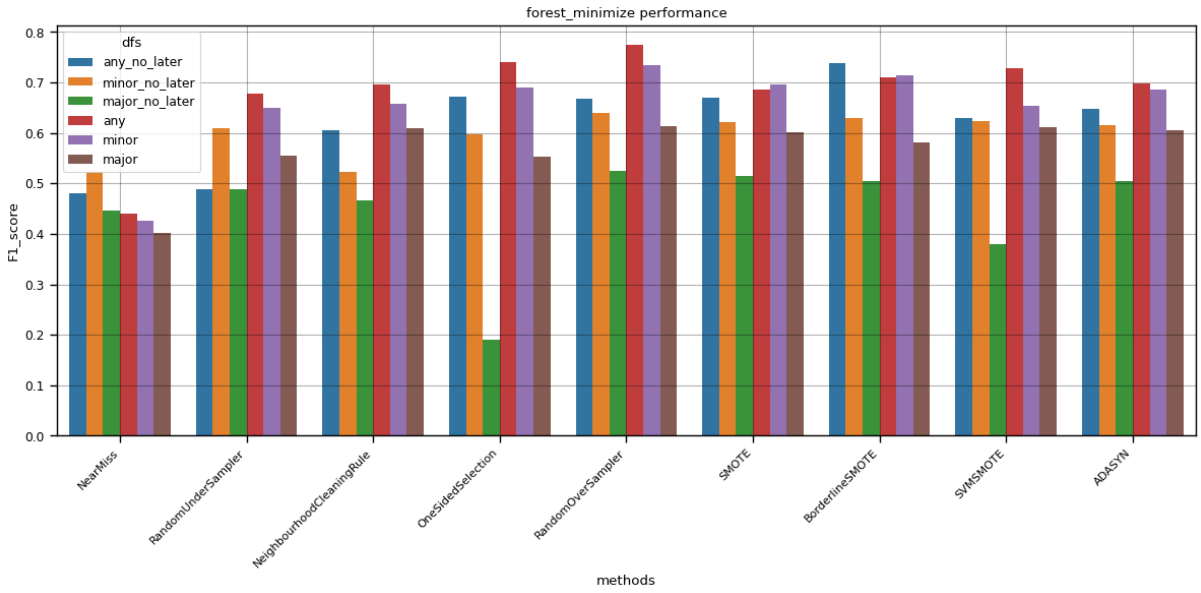
Figura 27 – Evolução do erro das florestas randômicas pela função de minimização



4.4.2 Florestas randômicas

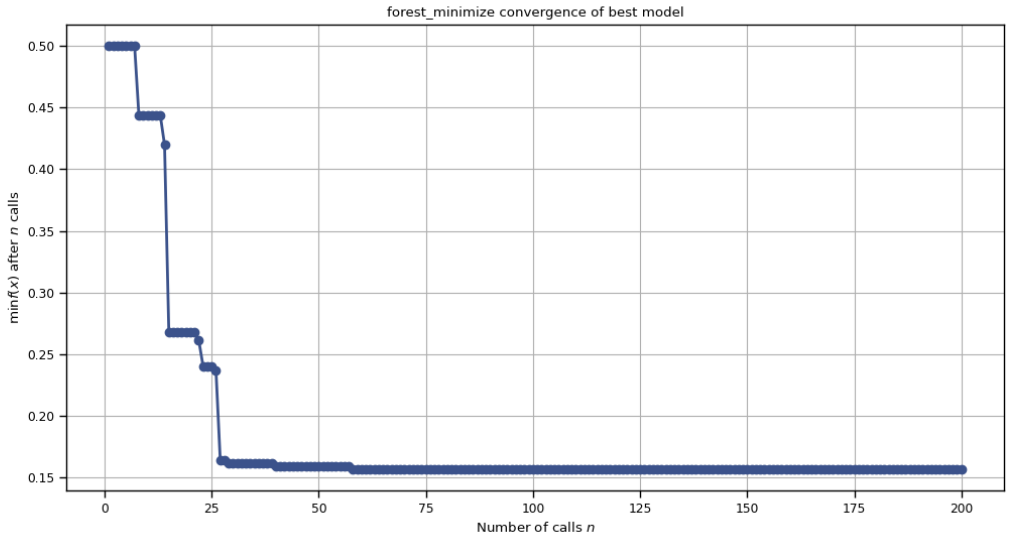
O primeiro modelo testado, florestas randômicas são modelos *ensemble* de árvores de decisão. Tem uma dificuldade de visualização devido ao grande números de árvore independentes porém não é um modelo caixa preta como outros modelos que tiveram uma assertividade bastante alta. A combinação das características foi feita da mesma forma que as árvores de decisão, o gráfico de barras das florestas é demonstrado na figura 28.

Figura 28 – F1 score das florestas rândomicas em relação aos *datasets* e *samplings*



Para avaliar o modelo de minimização pode-se avaliar a evolução da função de minimização utilizada. A função utilizada foi a média do erro absoluto, é uma função relativamente simples e por isso é mais fácil de calcular, entretanto devido a não dividir as classes, pode levar o modelo ao *overfitting*, ainda assim caso a função continue sendo minimizada o modelo continuará melhorando. A evolução da função de minimização é mostrada na figura 29.

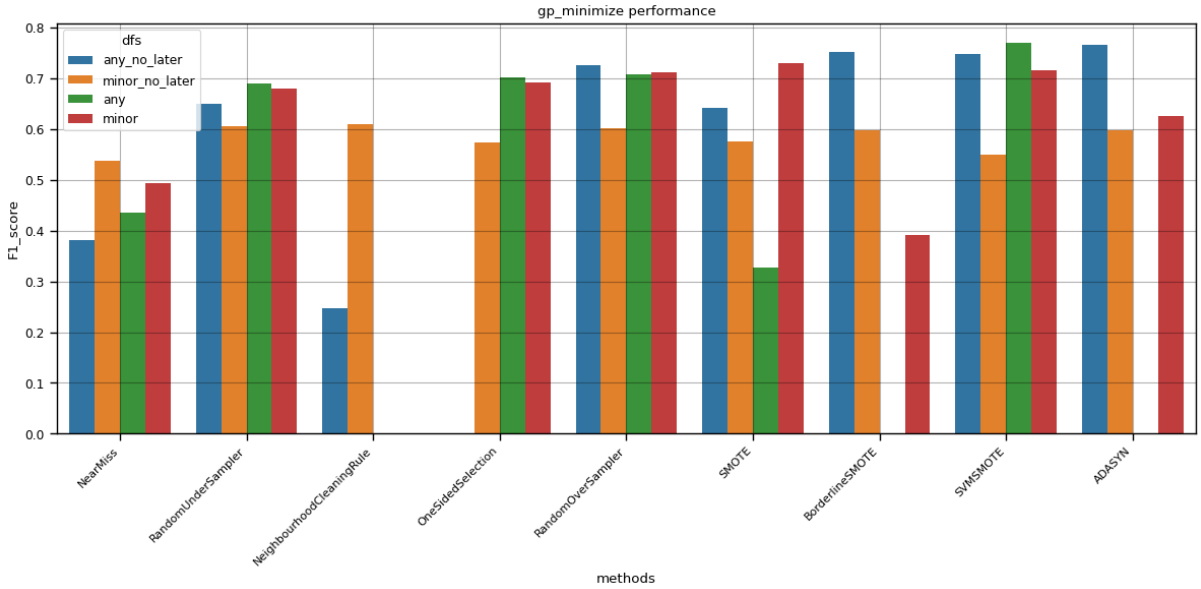
Figura 29 – Evolução do erro das florestas randômicas pela função de minimização



4.4.3 *Gradient boosting*

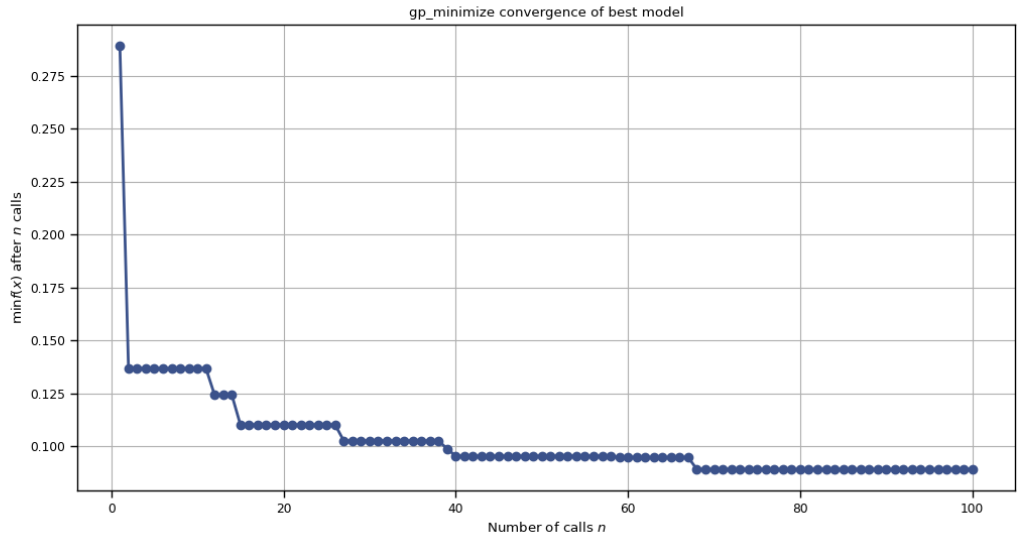
Como o modelo de florestas randômicas também utiliza de um *ensemble* de árvores de decisão para alcançar o modelo final. Porém por utilizar as árvores em série e uma árvore ser treinada no erro da passada não faz sentido puxar os estimadores separadamente como nas florestas. Por isso, o modelo pode ser considerado caixa preta e serve exclusivamente para melhorar a performance de previsão mas não para oferecer soluções ao problema. Seguindo a mesma ideia dos modelos anteriores foi testado as combinações supracitadas na figura 30.

Figura 30 – F1 score do *gradient boosting* em relação aos *datasets* e *samplings*



A evolução do modelo de minimização pode ser observada na figura 31. Foi reduzido o número de iterações com o intuito de diminuir o tempo de treino do programa. É difícil dizer se esta redução teve algum impacto na assertividade final do algorítimo sem um teste. A redução das iterações foram feitas de acordo com o número de hiperparâmetros que deveriam ser otimizados.

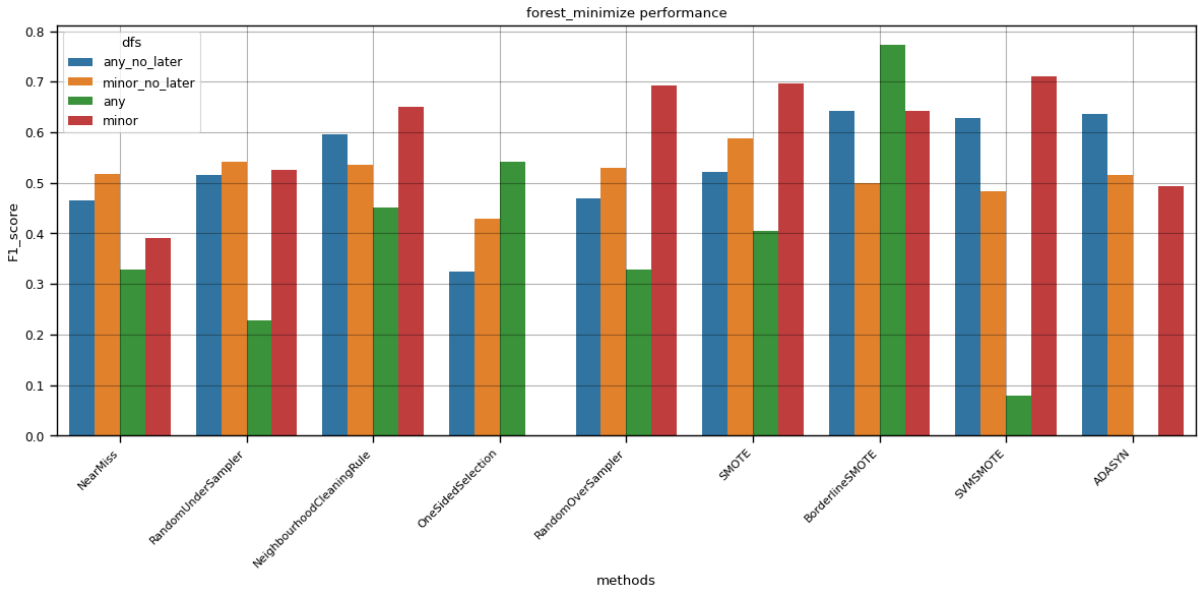
Figura 31 – Evolução do erro do *gradient boosting* pela função de minimização



4.4.4 Xtreme gradient boosting

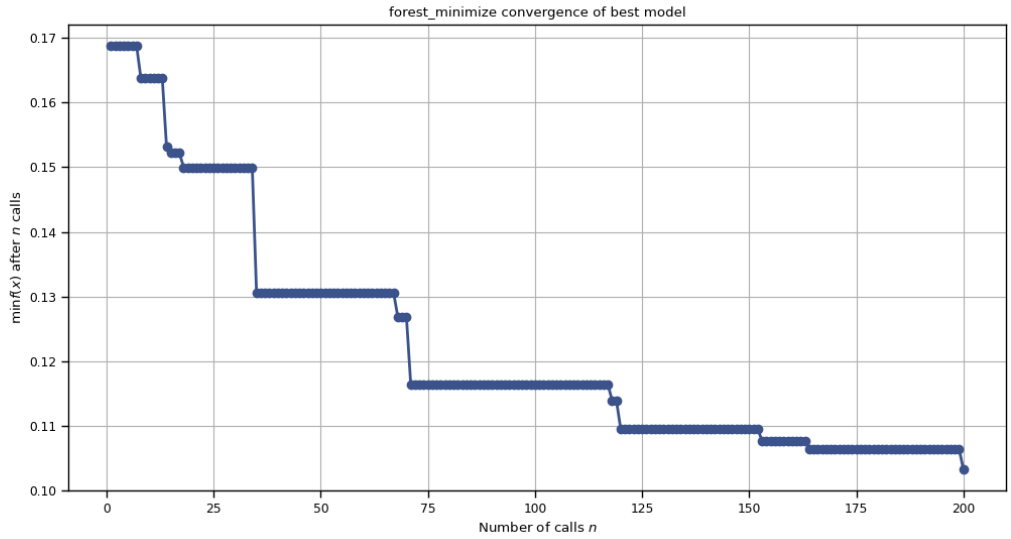
Muito parecido com o *gradient boosting* este modelo oferece algumas diferenças em regularização. o modelo basicamente poda diversas árvores utilizando os métodos de regularização L1 e L2. Isso da ao modelo uma melhor capacidade de generalização sendo uma ótima adição aos modelos de caixa preta para previsão do problema. Seu teste dos *datasets* e métodos de *sampling* é demonstrado na figura 32.

Figura 32 – F1 score do *xtreme gradient boosting* em relação aos *datasets* e *samplings*



Como se percebe na figura 33 mesmo com 200 iterações o modelo ainda melhorou na última iteração trazendo dúvidas sobre quantas iterações é necessário para cada modelo alcançar o mínimo real da função de otimização.

Figura 33 – Evolução do erro do *xtreme gradient boosting* pela função de minimização

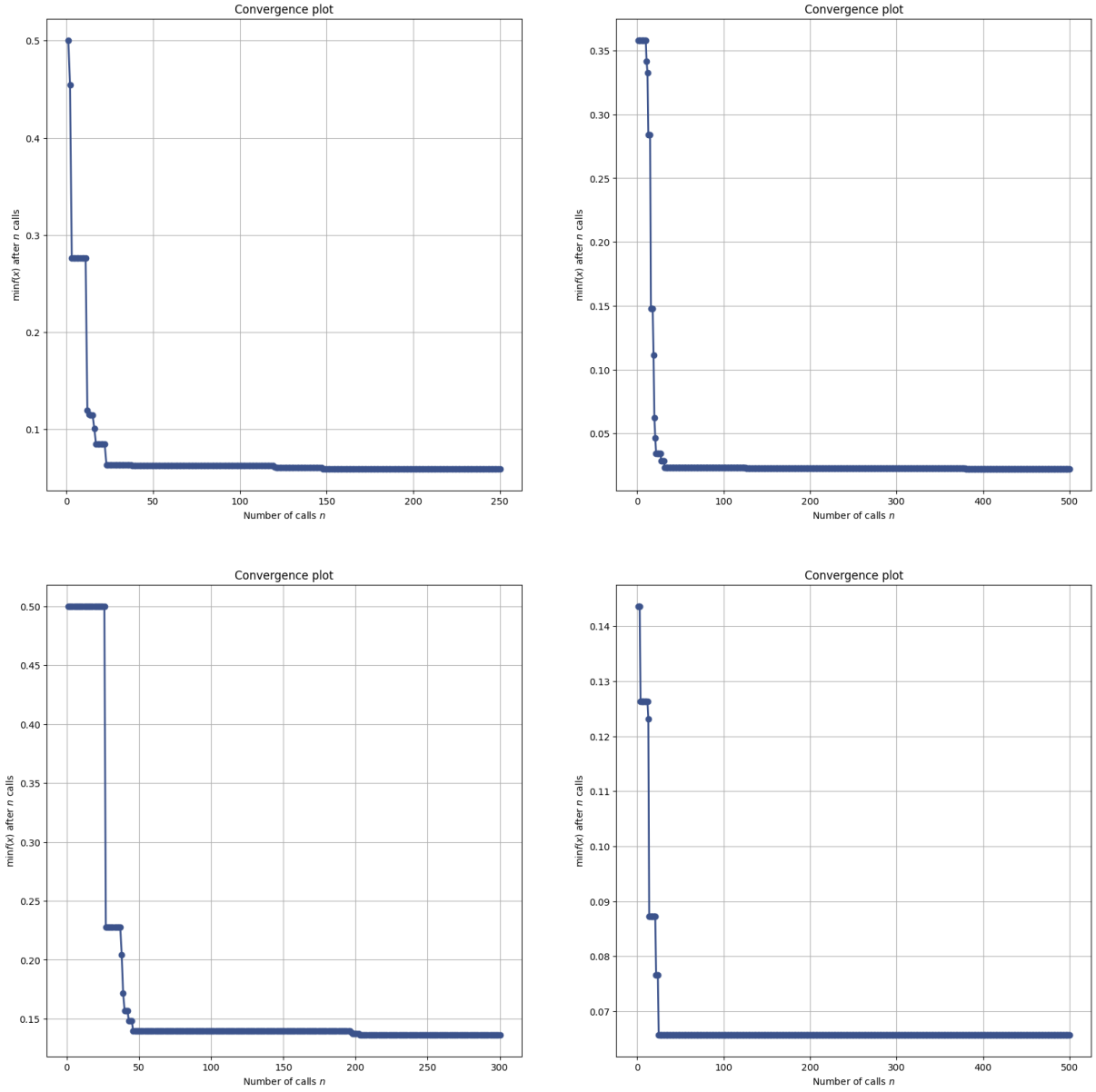


Com as melhores combinações encontradas para cada modelo é necessário resolver dois problemas observados. O primeiro é ter amostras que nunca foram utilizadas por nenhum modelo para treinar o modelo meta, o segundo é aumentar o número de iterações do modelo de minimização para garantir que o mesmo alcançou verdadeiramente seu mínimo. Ambos podem ser feitos retreinando os modelos utilizados apenas com as combinações ótimas e apenas com uma parte do *dataset* escolhido, salvando outra parte para o modelo meta.

4.4.5 avaliação final

Os modelos escolhidos para serem utilizados no projeto foram retreinados e re-otimizados com mais iterações. Porém não foi possível melhorar os modelos e infelizmente as características de antes se mantiveram ou pioraram. O que leva a entender que existe uma faixa de acurácia que os modelos verdadeiramente estão e dependendo das amostras de teste a acurácia do modelo varia. Ainda assim a minimização das funções de erro em relação as iterações é demonstrada na figura 34.

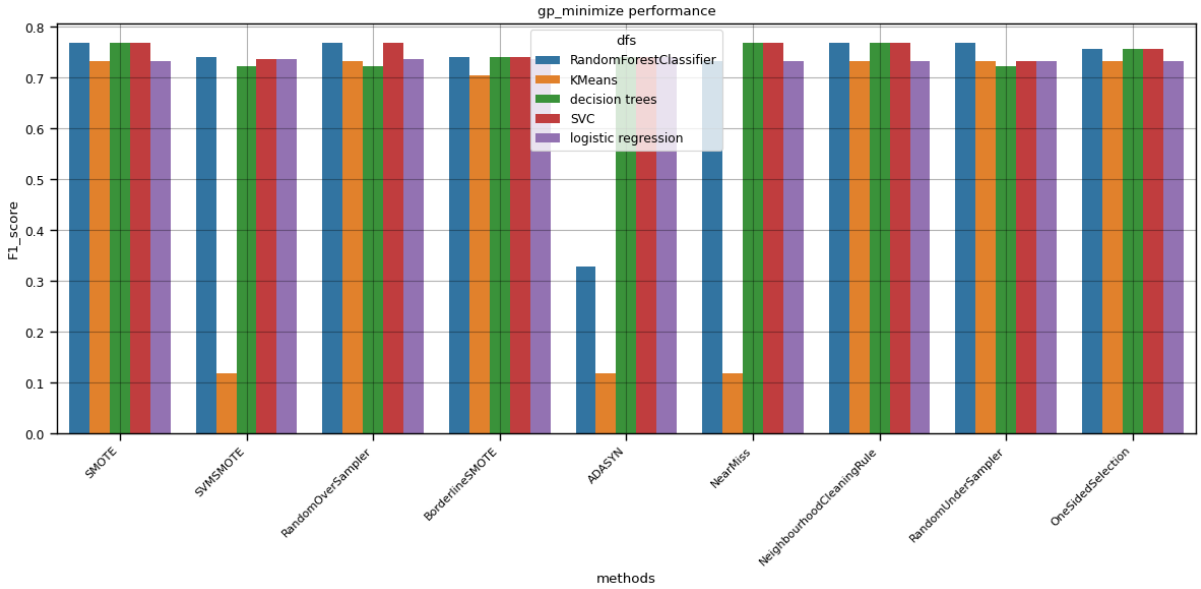
Figura 34 – Evolução do erro dos modelos em sequência florestas randômicas, árvores de decisão, *gradient boosting* e *Xtreme gradinet boosting*



4.4.6 modelo meta

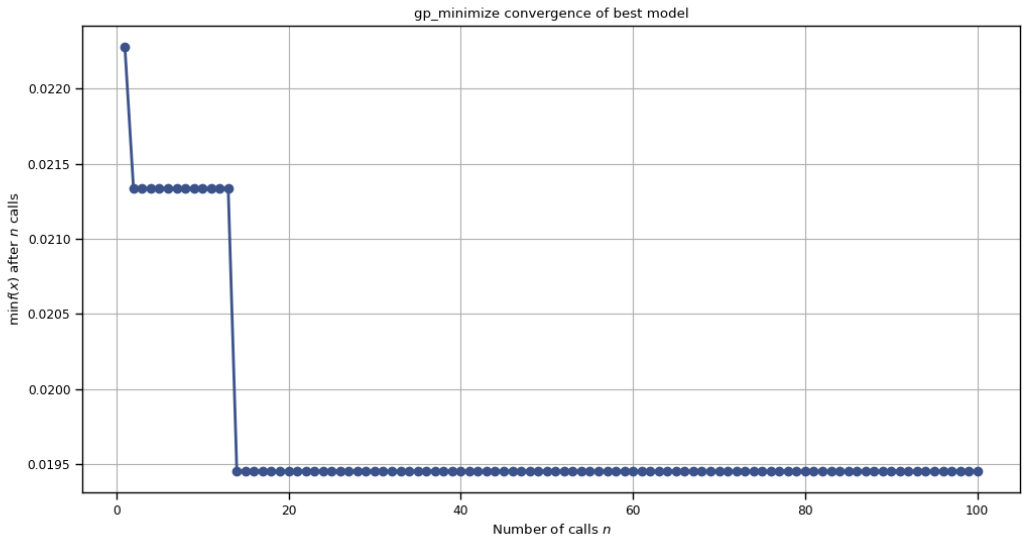
O modelo meta é aquele que recebe a previsão dos outros modelos e desenvolve a própria conclusão na base da conclusão dos outros modelos. Devido ao baixo número de variáveis explicativas e de amostras, foram escolhidos algorítimos mais simples para este modelo. Assim como os outros, o modelo é escolhido se testando múltiplas combinações de estratégias envolvendo técnicas de equilíbrio de *dataset* em conjunto da otimização do mesmo. O teste dos modelos com sua acurácia é demonstrado na figura 35.

Figura 35 – Acurácia do modelo meta com a modificação dos diversos parâmetros utilizados.



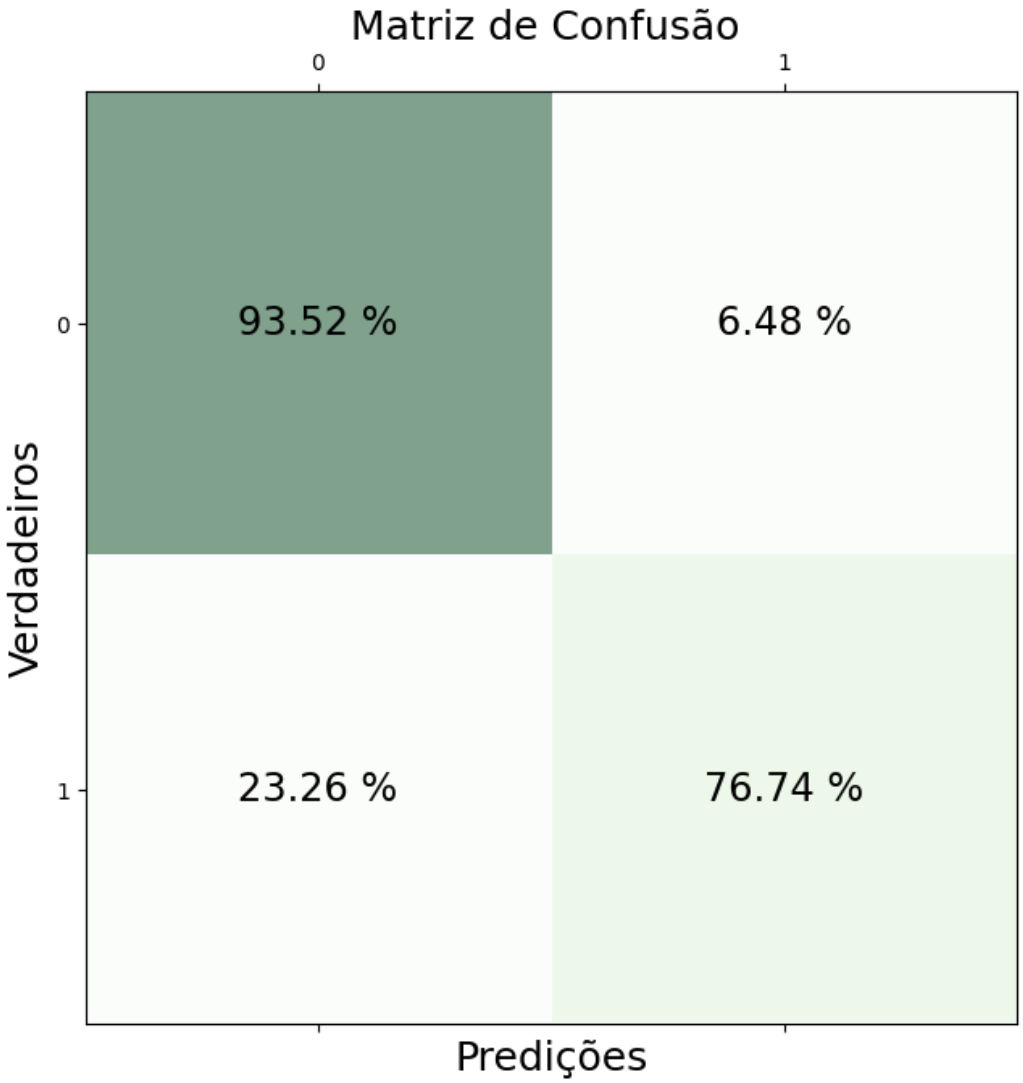
Como se percebe independente do método, o modelo de estimação fica bastante parecido. Sendo o maior *F1 score* de 0.771 utilizando florestas randômicas equilibrando o *dataset* utilizando a técnica de *undersampling neighborhood cleaning rule*. Devido ao baixo número de *features* que diferenciam pouco entre si a otimização dos hiper parâmetros não precisou de um grande número de iterações para alcançar seu mínimo como é demonstrado na figura 36.

Figura 36 – Minimização do erro do modelo meta



Assim pode-se avaliar o modelo final utilizando uma matriz de confusão apresentada na figura 37.

Figura 37 – matriz de confusão do modelo meta



Como se percebe o modelo tem uma altíssima assertividade para prever a não existência de bolhas, porém para prever a existência das mesmas o modelo acerta apenas um pouco mais que 3/4 das vezes. O que da de se concluir que o modelo meta não levou a grandes melhorias em acurácia. Entretanto isso não é motivo para descartar a união dos modelos, por utilizar diversos modelos em conjunto, um modelo meta é mais robusto que modelos simples mesmo que isso não se tenha resultado em grandes melhorias na acurácia apresentada. O baixo aumento em assertividade também levanta o questionamento sobre a explicação do problema em si. É bem provável que as variáveis explicativas utilizadas

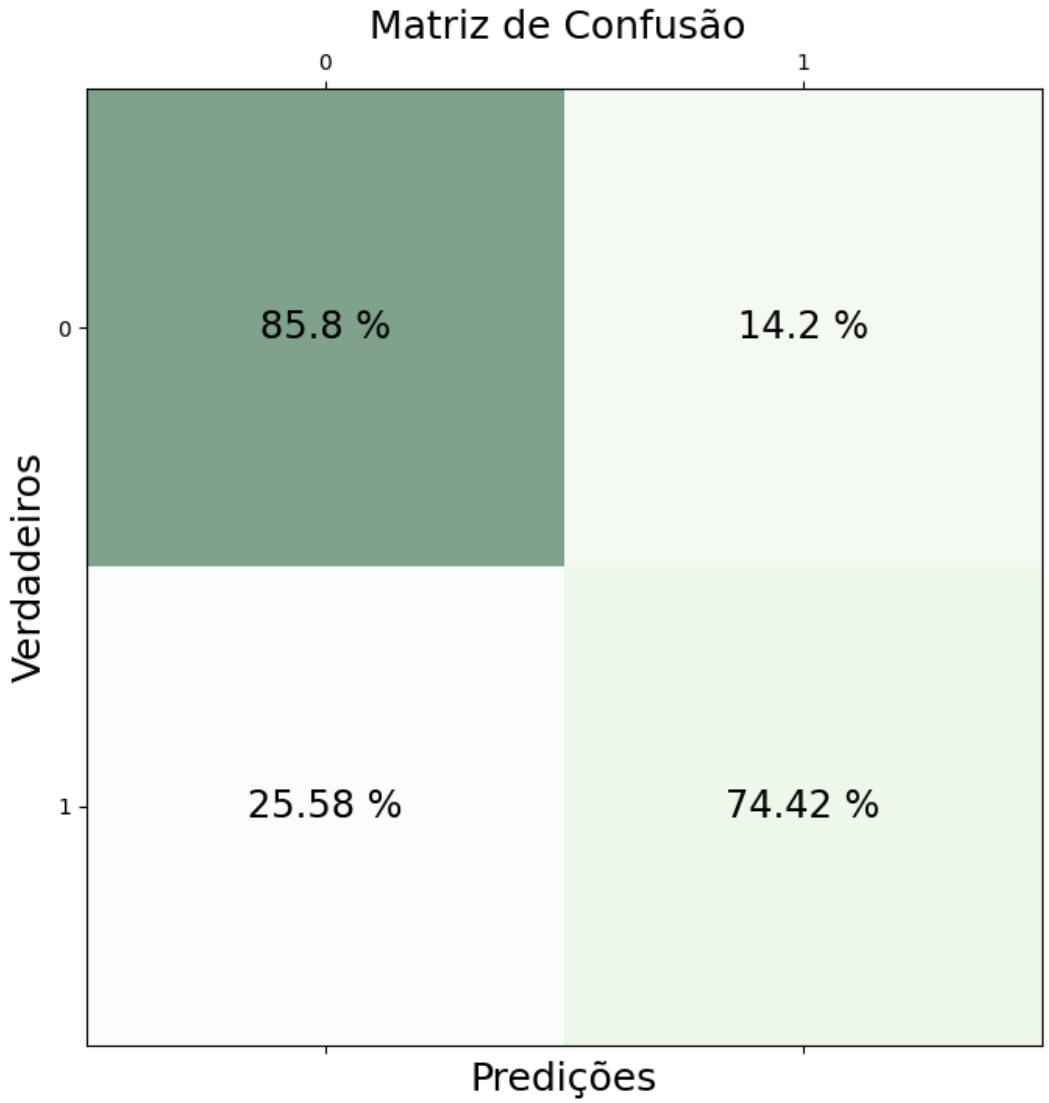
no momento não avalia todos os fenômenos físicos que agem na formação de bolhas na passagem de verniz nos fios.

O pipeline de treino dos modelos de A.M. pode ser avaliado na figura.

4.5 Retorno das regras

Com o modelo treinado, o próximo passo é avaliar as condições que levam o sistema a formar bolhas. Para avaliar isto foi utilizada a árvore de decisão da seção passada para retornar soluções, como é visto na figura 38 o modelo de árvore não é tão preciso como o modelo meta para a previsão de variáveis, porém o mesmo consegue retornar regras mais claras para a atuação nas variáveis de processo.

Figura 38 – matriz de confusão da árvore de decisão



Para conseguir retornar a decisão tomada pelo algorítimo seria necessário armazenar todos os nós que uma amostra percorreu na árvore. Feito isso foi retornada todas as regras de decisão de cada nó, puxando a variável independente utilizada no nó e seu limite de decisão. Na biblioteca do *sickit-learn* a divisão é sempre feita comparando se o valor da variável da amostra é menor ou igual a do nó. Caso isso seja verdade a amostra caminha para o próximo nó que tem o índice do nó anterior mais um. No caso contrário o valor do nó depende de quantos outros nós existem pelo caminho verdadeiro do nó pai. Assim é possível retornar o caminho da amostra e comparar com seus nós adjacentes e avaliar qual a probabilidade de formar bolhas caso se mude o caminho de um nó qualquer. As

probabilidades são calculadas através do *dataset* de treino e são utilizadas para avaliar o quão bom seria mudar a decisão de um nó e assim o caminho da amostra. Para tornar este caminho de decisões mais claro pode-se ver a árvore de decisão em sua totalidade na figura 39 e um zoom da escolha de uma amostra na figura 40, os nós em azul fazem parte do caminho tomado pela amostra. Enquanto os nós vermelhos são mais propensos a bolhas e os verdes menos.

Figura 39 – Árvore de decisão inteira

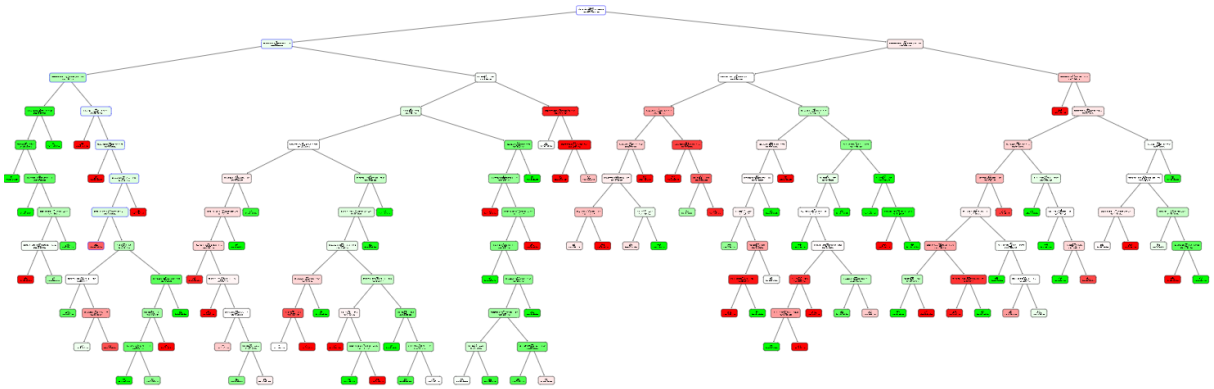
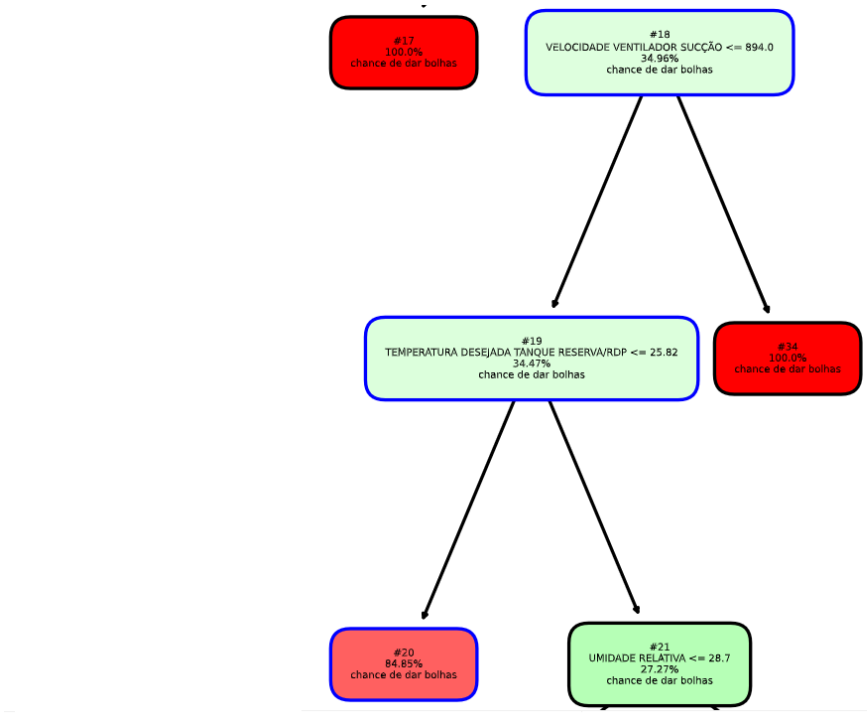


Figura 40 – Zoom em uma parte do caminho de uma amostra



Observando a figura 40 a amostra em específico acaba em uma folha com 84.85% de chance de causar bolhas, porém com uma mudança da "TEMPERATURA DESEJADA

TANQUE RESERVA/RDP"para um valor maior que 25.82 °C a chance de uma bolha se formar cai consideravelmente para 27.27% segundo o algorítimo. Dessa forma é possível retornar uma série de regras claras para serem tomadas com o intuito de reduzir a chance de formar bolhas.

Este retorno de regras resulta em dois problemas que merecem ser comentados. O primeiro deles é a mudança para um caminho que continue ruim. Já que apenas os filhos mais próximos do nó pai são avaliados, não se sabe em que folha a amostra verdadeiramente cairá. O segundo problema é que não necessariamente a mudança de um nó será possível já que em diversos casos isso depende de fatores externos que não podem ser controlados pelo colaborador responsável pela máquina como temperatura ambiente e umidade.

Para resolver o primeiro problema duas novas formas de procurar soluções na árvore foram testadas. A primeira delas seria avaliar a última divisão feita antes de alcançar uma folha da árvore, avaliar todos os nó que descendem deste nó pai e modificar as variáveis para alcançar uma folha que prevê um estado normal. Porém, devido a árvore reutilizar *features* na sua escolha, uma mudança nos nós debaixo podem acabar por mudar um nó em cima que não foi avaliado, mudando completamente o caminho final e gerando um erro nó método.

Devido a este problema o terceiro método foi utilizado, é basicamente um método de força bruta para decidir a melhor mudança a ser tomada. O que o método faz é percorrer o caminho feito pela amostra da raiz até a folha modificando cada nó onde o caminho original passa e avaliando em qual folha o novo caminho termina. Caso todas as mudanças singulares de *features* resulte em uma folha com bolha o modelo passa a tentar modificar duas *features*. Ele faz isso primeiro gerando um novo caminho mudando um nó do caminho original e retornando o novo caminho gerado. Então o método aplica o mesmo método que utilizou no caminho original e vai modificando nó a nó o caminho gerado e retorna apenas aqueles que resultam em um caminho sem bolha.

O problema deste último método é que o número de caminhos percorridos na árvore cresce ao quadrado do número de nós percorridos pela amostra, já que cada mudança em um caminho gerará um novo caminho com diversos novos nós que podem ser alterados para gerar novos caminhos. Por causa disso o método foi limitado a apenas duas mudanças de *features*. Ainda assim duas mudanças geram centenas de novos caminhos onde dezenas deles são viáveis para evitar bolhas e cabe ao colaborador decidir qual dos caminhos viáveis é o mais fácil de se seguir.

para ser legível para o operador o que fazer as mudanças dos nós foram transformadas em texto legível como é demonstrado na tabela 6.

- Regule a variável ventilador exaustão abaixo de 1504.52, regule a variável temperatura desejada tanque amida abaixo de 20.0.
- Regule a variável temperatura zona de cura abaixo de 523.28, regule a variável temperatura zona de cura abaixo de 688.01.
- Regule a variável temperatura zona de cura abaixo de 688.01, regule a variável ms_accelerationx_insuflador_polia abaixo de 0.16.
- Regule a variável temperatura ar de exaustão abaixo de 485.49.

Tabela 6 – Tabela de regras

4.6 API

Com a previsão de bolhas e soluções desenvolvidas é necessário oferecer uma forma do usuário do programa consumir os dados fornecidos pelo programa. Uma das formas mais comuns de oferecer serviços na WEG é utilizar a intranet para oferecer uma API, foi utilizada a biblioteca do FastAPI que oferece um serviço leve e de rápida programação.

Foi feito uma API do tipo REST com 3 requisições do tipo *get* para consumir os dados do programa. A primeira delas é a *root* está é a requisição padrão do programa e apenas retorna um pequeno texto que explica o que a aplicação faz:

"welcome to the bubble prevention system"

A segunda requisição chamada de *"predict_bubble"* pega uma amostra aleatória do banco de dados criado para o programa e a classifica retornando um booleano sendo 0 o estado que não terá bolhas e 1 o contrário. É valido lembrar que quando o programa entrar em produção, não será retirada uma amostra aleatória do banco de dados. Serão feitas todas as requisições para os respectivos serviços que armazenam as variáveis necessárias para o programa, com estas informações será gerada a amostra que após ser classificada deverá ser armazenada no banco de dados da aplicação para um futuro retreinamento.

Por fim a última requisição é a de retorno de soluções, chamada de *"solutions"*, esta requisição obtém a mesma amostra da requisição *"predict_bubble"* e a passa no algoritmo de soluções. O mesmo retorna uma string com as diversas regras a serem aplicadas para evitar as bolhas. No caso de fazer a requisição em uma amostra que não gera bolhas a requisição retorna a seguinte mensagem:

"no solutions found"

4.7 Banco de dados

Para guardar os dados já extraídos e para armazenar dados futuros foi criado um banco de dados *postgres* com duas tabelas simples, ambas foram colocadas em um *schema* chamado dev para teste da aplicação.

A primeira tabela armazena todos os dados crus extraídos do programa. Foi pensado que seria interessante manter os dados crus caso num futuro outro tipo de limpeza ou outras variáveis sejam utilizadas no programa, além disso é bom no caso de uma amostra não poder ser classificada por valores nulos ou algo do tipo. A tabela foi chamada de *raw_data* e tem 120701 linhas.

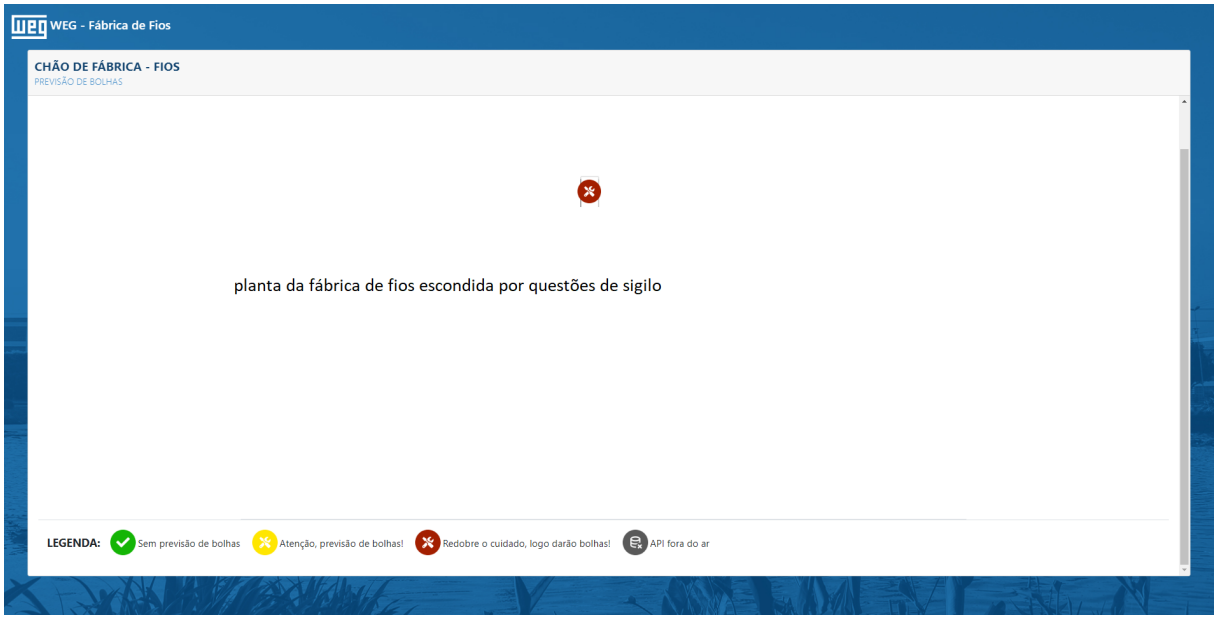
A segunda tabela é a dos dados verdadeiramente utilizados no programa. Nela estão apenas as variáveis consideradas na aplicação do programa e contém apenas dados limpos prontos para serem utilizados. A tabela foi chamada de *cleaned_data* e ela é utilizada para fazer as requisições da API.

Atualmente as tabelas não estão correlacionadas com chaves estrangeiras mesmo que ambas tenham correlação já que todas as linhas da tabela *cleaned_data* se encontram na tabela *raw_data*.

4.8 *Front-end*

Por fim a API foi passada para a desenvolvedora do *front-end* Ingrid Ester Zimmermann. Ela adicionou as requisições da API na tela do SFM da fábrica de fios com um ícone que mostra se terá bolhas ou não que caso se clique em cima retorna as soluções oferecidas pelo programa.

Figura 41 – tela da fabrica de fios do SFM com o ícone



Fonte: Ingrid Ester Zimmermann, 2022

Figura 42 – tela das regras

Fonte: Ingrid Ester Zimmermann, 2022

5 Conclusão

5.1 O que foi feito

A inteligência artificial está em suas décadas douradas. Diversas aplicações que a poucos anos jamais seriam consideradas para uso em A.M. estão sendo usadas hoje em dia. Esta forte expansão vem alcançando a indústria, sendo utilizada em diversos segmentos da produção, de controle de qualidade a predição da manutenção.

Dentro desta grande área este projeto demonstra as capacidades que estes tipos de algoritmo podem trazer á indústria e sua competitividade no mercado. Para realizar este projeto de A.M. primeiro foi avaliado o processo produtivo da indústria. Basicamente quais são os passos que a máquina faz, foi visto todos os processos de trefilagem, passando pela aplicação do verniz e secagem do mesmo nos fios. Além disso foi visto quais indicadores de qualidade existiam, quais sensores e variáveis poderiam ser utilizados para melhor avaliar o processo e entender o que estava errado e como poderia se atuar.

Depois de ter uma ideia melhor do processo foi buscado a aquisição dos dados, onde as variáveis de interesse estavam e como seria a melhor forma de acessá-las. Com os dados em mãos existe a discussão de como integra-los, quais são as características positivas e negativas de juntar os dados naquela maneira? Existe alguma perda de informação? os dados representam bem o problema? É possível tirar algo útil com os dados formatados da maneira que está sendo pensado?

Com os dados bem formatados é hora de manipular as variáveis de interesse. Separar o que é lixo do que não é, saber diferenciar o que é uma informação útil. A limpeza dos dados é uma das partes mais importantes do projeto, limpar demais pode retirar exatamente as informações que se procura e limpar de menos acaba por oferecer ao algorítimo uma população de dados que não representa o problema fielmente.

Por fim vem a escolha do algorítimo. Dentro dos inúmeros modelos de A.M. disponíveis qual se encaixa melhor no problema? qual respeita as restrições e limitações que o projeto inerentemente tem e quais acabam por desrespeitar estas regras e resultar em uma baixa acurácia? Todas estas perguntas foram respondidas através dos testes feitos que selecionou a melhor combinação de algorítimos encontrada para resolver o problema.

Assim da pra se dizer que foi bem sucedida a tentativa de previsão e solução de bolhas para a máquina em questão. Segundo os cálculos do analista André Correa Mafra a máquina estaria gerando R$ 52.193,14 de custos com a formação de bolhas. No caso das soluções oferecidas pelo algorítimo serem implementadas pelos colaboradores da fábrica se espera uma economia entre R$ 33 e 37 mil por máquina. É valido lembrar que na

fábrica de fios contém outras 4 máquinas do mesmo tipo da máquina estudada e outras três diferentes que também passam verniz em fios, porém em fios de diâmetros maiores. É uma grande oportunidade de economia devido a um investimento inicial extremamente baixo o que resulta numa balança muito positiva entre valor investido e retorno gerado. Porém existe muito trabalho antes de explorar este potencial.

5.2 Trabalhos futuros

Existem diversos passos a serem compridos para deixar o programa pronto para produção. É necessário consumir todos os dados por API´s sem utilizar acesso direto ao banco de dados das aplicações externas. Isso é importante pois o programa irá fazer requisições de forma frequente, e ter que filtrar os dados por data direto do banco é uma má pratica de programação.

O segundo passo que precisa ser feito é modificar o banco de dados para conseguir ser utilizado com os dados sendo consumidos em tempo real. É valido lembrar que a variável de previsão e os eventos do SFM para limpeza utilizam do tempo e precisam ser levados em consideração.

Em seguida precisam ser criadas as requisições corretas puxando os dados das API´s externas no tempo atual sem utilizar os dados já coletados puxados aleatoriamente como foi feito na demonstração do programa.

Com os passos supracitados o programa estará pronto para ser utilizado na máquina do presente estudo, porém diversas outras questões podem ser melhoradas para fazer do programa o preditor de falhas da WEG.

O primeiro é a auto detecção de *features* pelo programa. No programa atual foi feito um amplo estudo estatístico para tentar identificar quais variáveis seriam úteis para o problema. Porém técnicas estatísticas não demonstram a melhor assertividade para este tipo de problema. Além disso é um passo extremamente custoso do ponto de vista de mão de obra para um retorno relativamente pequeno. Diversos modelos de ML conseguem classificar importância de *features* como árvores de decisão que poderiam ser utilizadas para cortar em uma grande quantia o tempo necessário para aplicar o programa a uma nova máquina.

Outro passo importante seria a automatização mesmo que parcial do acesso aos dados da máquina. Atualmente cada banco de dados foi acessado manualmente e suas variáveis extraídas. Talvez seja possível automatizar parte deste processo através do ID da máquina para mais rapidamente fazer a extração e junção dos dados.

Com estas mudanças feitas é possível utilizar o programa em outros problemas de classificação de falhas na WEG que podem resultar em diversos cortes de gastos para a

empresa externos a formação de bolhas em fios.

Referências

ALAM, S. et al. *Kedro*. 2022. Disponível em: <https://github.com/kedro-org/kedro>. Citado na página 34.

BAKAR, Z. A. et al. A comparative study for outlier detection techniques in data mining. In: IEEE. *2006 IEEE conference on cybernetics and intelligent systems*. [S.l.], 2006. p. 1–6. Citado 2 vezes nas páginas 40 e 58.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 47.

CHANDRA, M. J. *Statistical quality control*. [S.l.]: CRC Press, 2001. Citado na página 20.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002. Citado na página 43.

CHEN, T. et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, v. 1, n. 4, p. 1–4, 2015. Citado na página 47.

CROSBY, P. B. Quality is free-if you understand it. *Winter Park Public Library History and Archive Collection*, p. 4, 1979. Citado na página 15.

DUIF, M. *An Introduction to Decision Trees with Python and scikit-learn*. 2020. Disponível em: <https://towardsdatascience.com/an-introduction-to-decision-trees-with-python-and-scikit-learn-1a5ba6fc204f>. Citado na página 47.

GARCÍA, S.; LUENGO, J.; HERRERA, F. *Data preprocessing in data mining*. [S.l.]: Springer, 2015. v. 72. Citado na página 36.

HARRINGTON, H. J. *Poor-quality cost*. [S.l.]: CRC Press, 1987. Citado na página 15.

HE, H. et al. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: IEEE. *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. [S.l.], 2008. p. 1322–1328. Citado na página 43.

HECKE, T. V. Power study of anova versus kruskal-wallis test. *Journal of Statistics and Management Systems*, Taylor & Francis, v. 15, n. 2-3, p. 241–247, 2012. Citado na página 38.

IBM. *What is logistic regression?* 2022. Disponível em: <https://www.ibm.com/topics/logistic-regression>. Citado na página 44.

JOSé, I. *KNN (K-Nearest Neighbors) #1*. 2018. Disponível em: <https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e>. Citado na página 45.

KINGSFORD, C.; SALZBERG, S. L. What are decision trees? *Nature biotechnology*, Nature Publishing Group, v. 26, n. 9, p. 1011–1013, 2008. Citado na página 46.

KORNBROT, D. Point biserial correlation. *Wiley StatsRef: Statistics Reference Online*, Wiley Online Library, 2014. Citado na página 38.

KRAUSS, J. et al. Automated machine learning for predictive quality in production. *Procedia CIRP*, Elsevier, v. 93, p. 443–448, 2020. Citado 3 vezes nas páginas 16, 17 e 34.

KROGH, A. What are artificial neural networks? *Nature biotechnology*, Nature Publishing Group, v. 26, n. 2, p. 195–197, 2008. Citado na página 44.

LABXCHANGE (HARVARD ONLINE). *How to Interpret Violin Plots*. 2021. Disponível em: <https://www.labxchange.org/library/items/lb:LabXchange:46f64d7a:html:1>. Citado na página 42.

LAURIKKALA, J. Improving identification of difficult small classes by balancing class distribution. In: SPRINGER. *Conference on artificial intelligence in medicine in Europe*. [S.l.], 2001. p. 63–66. Citado na página 43.

MAHMOOD, S. et al. Determining the cost of poor quality and its impact on productivity and profitability. *Built Environment Project and Asset Management*, Emerald Group Publishing Limited, 2014. Citado na página 21.

MAHMOOD, S.; KURESHI, N. A literature review of the quantification of hidden cost of poor quality in the historical perspective. *Journal of Quality and Technology Management*, v. 11, n. 1, p. 1–24, 2015. Citado na página 15.

MANI, I.; ZHANG, I. knn approach to unbalanced data distributions: a case study involving information extraction. In: ICML. *Proceedings of workshop on learning from imbalanced datasets*. [S.l.], 2003. v. 126, p. 1–7. Citado na página 43.

MASTROTHANASIS, K. *How can I make interpretation of kendall's Tau-b correlation magnitude?* 2020. Citado na página 59.

MELTON, J.; SIMON, A. R. *Understanding the new SQL: a complete guide*. [S.l.]: Morgan Kaufmann, 1993. Citado na página 35.

MONTGOMERY, D. C.; WOODALL, W. H. An overview of six sigma. *International Statistical Review/Revue Internationale de Statistique*, JSTOR, p. 329–346, 2008. Citado na página 20.

NEOCHARGE. Website, *Como funciona o motor de um carro elétrico*. 2022. Disponível em: <https://www.neocharge.com.br/tudo-sobre/carro-eletrico/motor-como-funciona>. Citado na página 22.

ORACLE. Website, *SQL para Acessar, Definir e Manter Dados*. 2022. Disponível em: <https://www.oracle.com/br/database/technologies/appdev/sql.html>. Citado na página 35.

OZGUR, C. et al. Matlab vs. python vs. r. *Journal of data Science*, Chinese Data Mining Association, v. 15, n. 3, p. 355–371, 2017. Citado na página 34.

PACHNER, S.; MIETHLINGER, J. Smart data analysis for optimized manufacturing of powder coatings on co-rotating twin screw extruders. In: AIP PUBLISHING LLC. *AIP Conference Proceedings*. [S.l.], 2019. v. 2055, n. 1, p. 070010. Citado na página 18.

PAVLYSHENKO, B. Using stacking approaches for machine learning models. In: IEEE. *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*. [S.l.], 2018. p. 255–258. Citado na página 48.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado 2 vezes nas páginas 45 e 46.

PSICOMETRIA ONLINE. Website, *Distribuição Normal*. 2022. Disponível em: <https://psicometriaonline.com.br/distribuicao-normal/>. Citado na página 40.

PUTH, M.-T.; NEUHÄUSER, M.; RUXTON, G. D. Effective use of spearman's and kendall's correlation coefficients for association between two measured traits. *Animal Behaviour*, Elsevier, v. 102, p. 77–84, 2015. Citado na página 39.

SINGH, J.; SINGH, H. Kaizen philosophy: a review of literature. *IUP journal of operations management*, IUP Publications, v. 8, n. 2, p. 51, 2009. Citado na página 20.

STATSTUTOR. *Spearman's correlation*. 2022. Disponível em: <https://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf>. Citado na página 60.

TELI, S. N. et al. Assessment of cost of poor quality for automobile industry. *International Journal of Engineering Research and Applications*, v. 2, n. 6, p. 330–336, 2012. Citado na página 21.

TERCAN, H.; MEISEN, T. Machine learning and deep learning based predictive quality in manufacturing: a systematic review. *Journal of Intelligent Manufacturing*, Springer, p. 1, 2022. Citado na página 20.

TIHOMIROVS, J.; GRABIS, J. Comparison of soap and rest based web services using software evaluation metrics. *Information technology and management science*, v. 19, n. 1, p. 92–97, 2016. Citado na página 35.

VAZAN, P. et al. Using data mining methods for manufacturing process control. *IFAC-PapersOnLine*, Elsevier, v. 50, n. 1, p. 6178–6183, 2017. Citado na página 18.

WEG. Website, *Site da WEG*. 2022. Disponível em: <https://www.weg.net>. Citado 2 vezes nas páginas 23 e 24.

ZAIONTZ, C. Website, *D'Agostino-Pearson Test*. 2022. Disponível em: <https://www.real-statistics.com/tests-normality-and-symmetry/statistical-tests-normality-symmetry/dagostino-pearson-test/>. Citado na página 39.

APÊNDICE A – Distribuição das variáveis do problema

Figura 43 – Distribuição aceleração *motorscan* eixo X

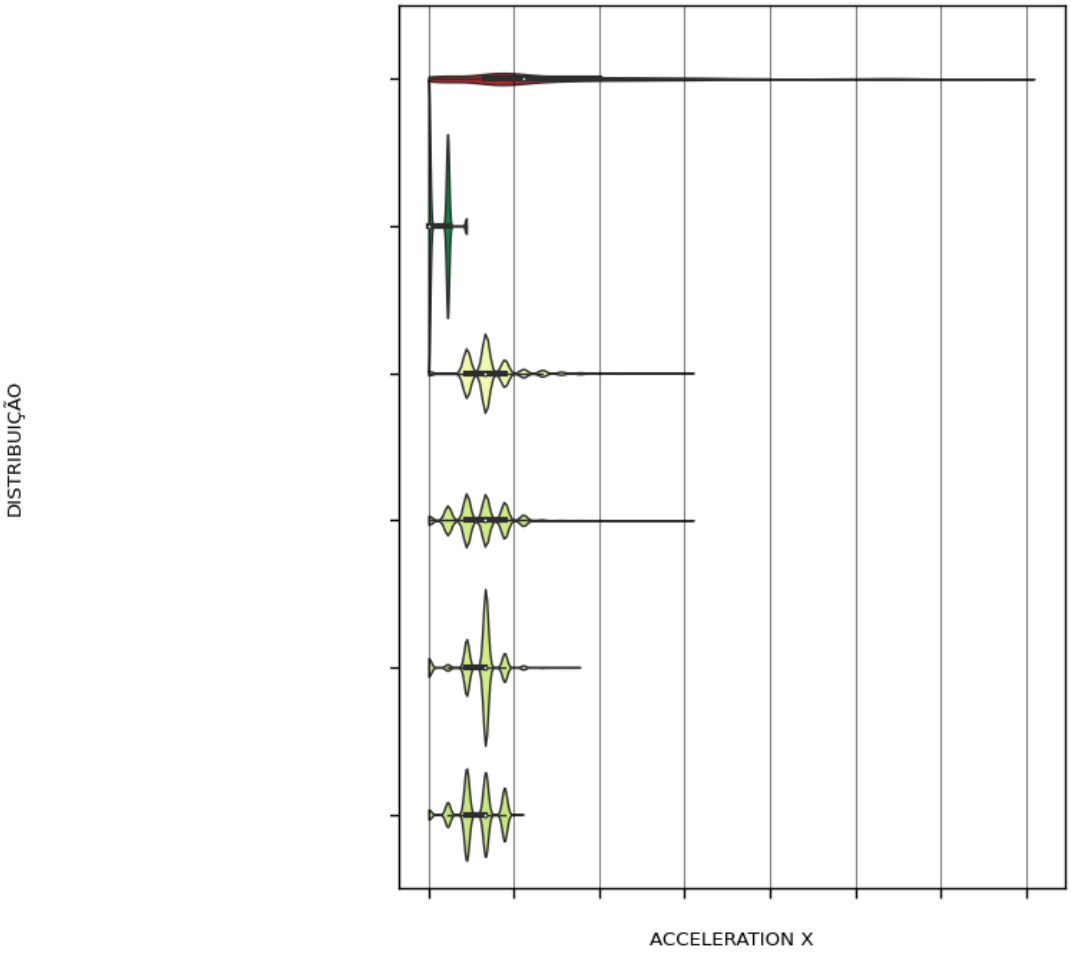


Figura 44 – Distribuição aceleração *motorscan* eixo X limpo com 3 σ

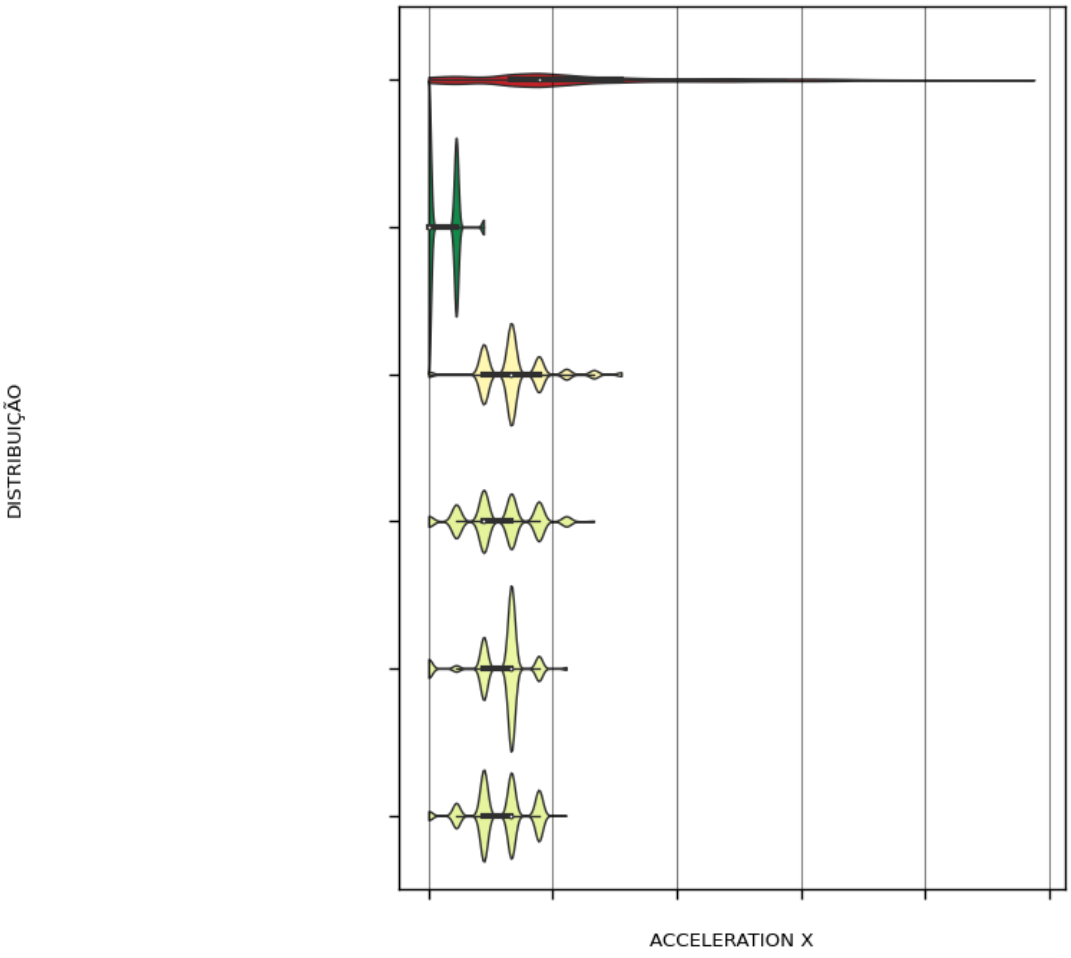


Figura 45 – Distribuição aceleração *motorscan* eixo Y

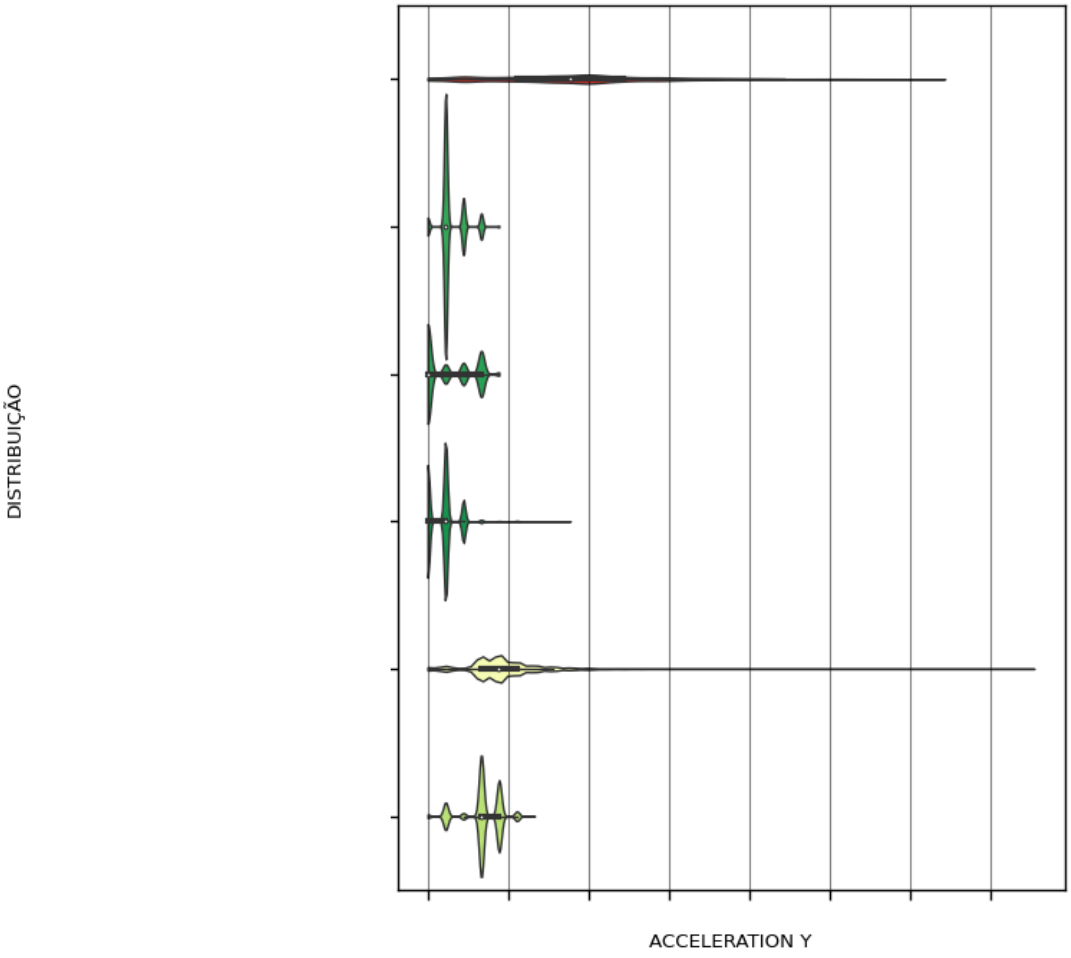


Figura 46 – Distribuição aceleração *motorscan* eixo Y limpo com 3 σ

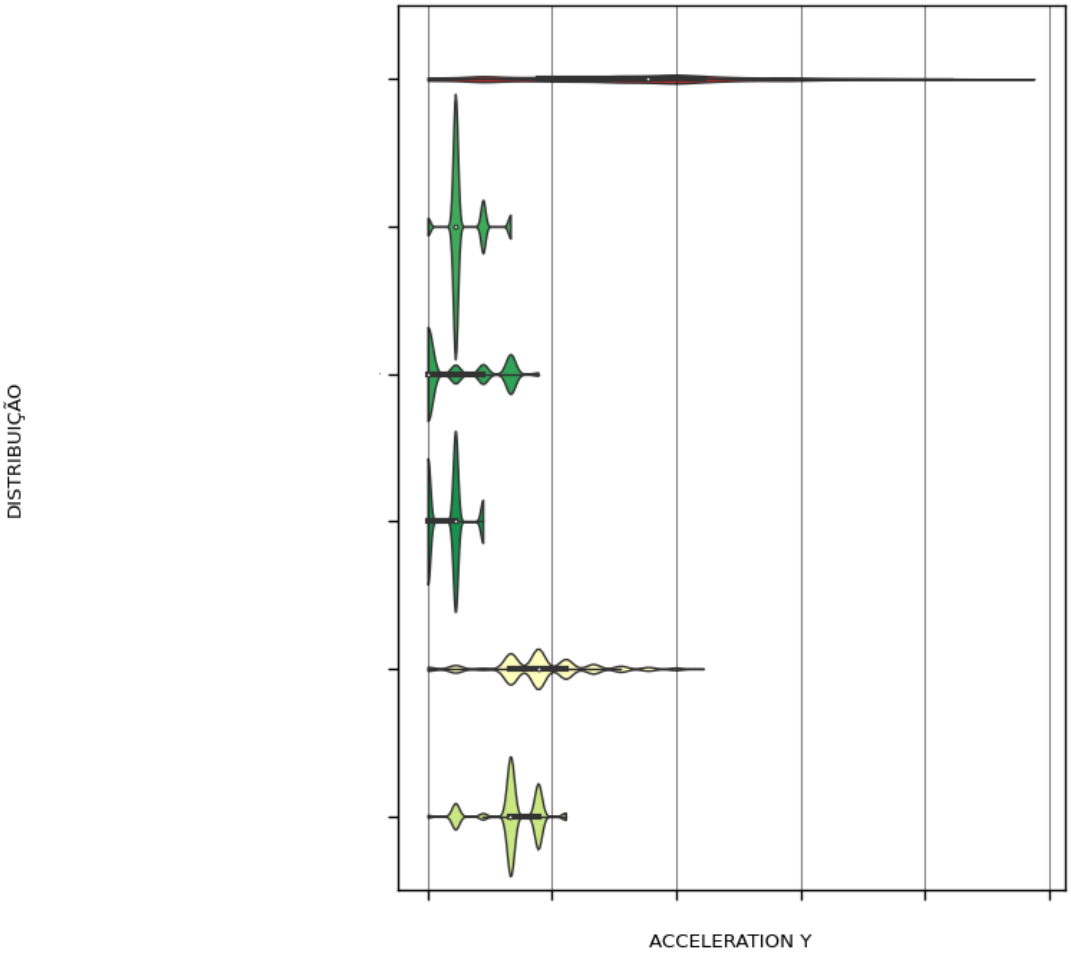


Figura 47 – Distribuição aceleração *motorscan* eixo Z

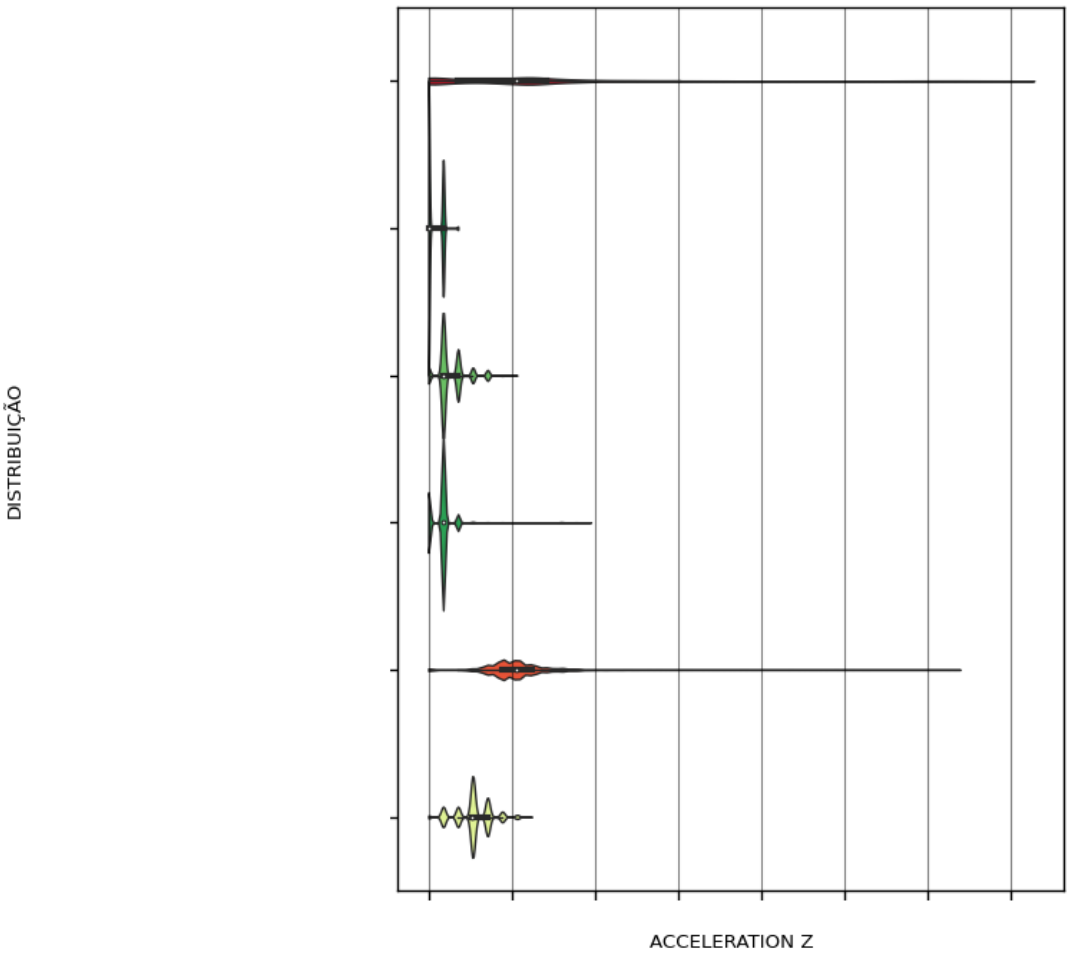


Figura 48 – Distribuição aceleração *motorscan* eixo Z limpo com 3 σ

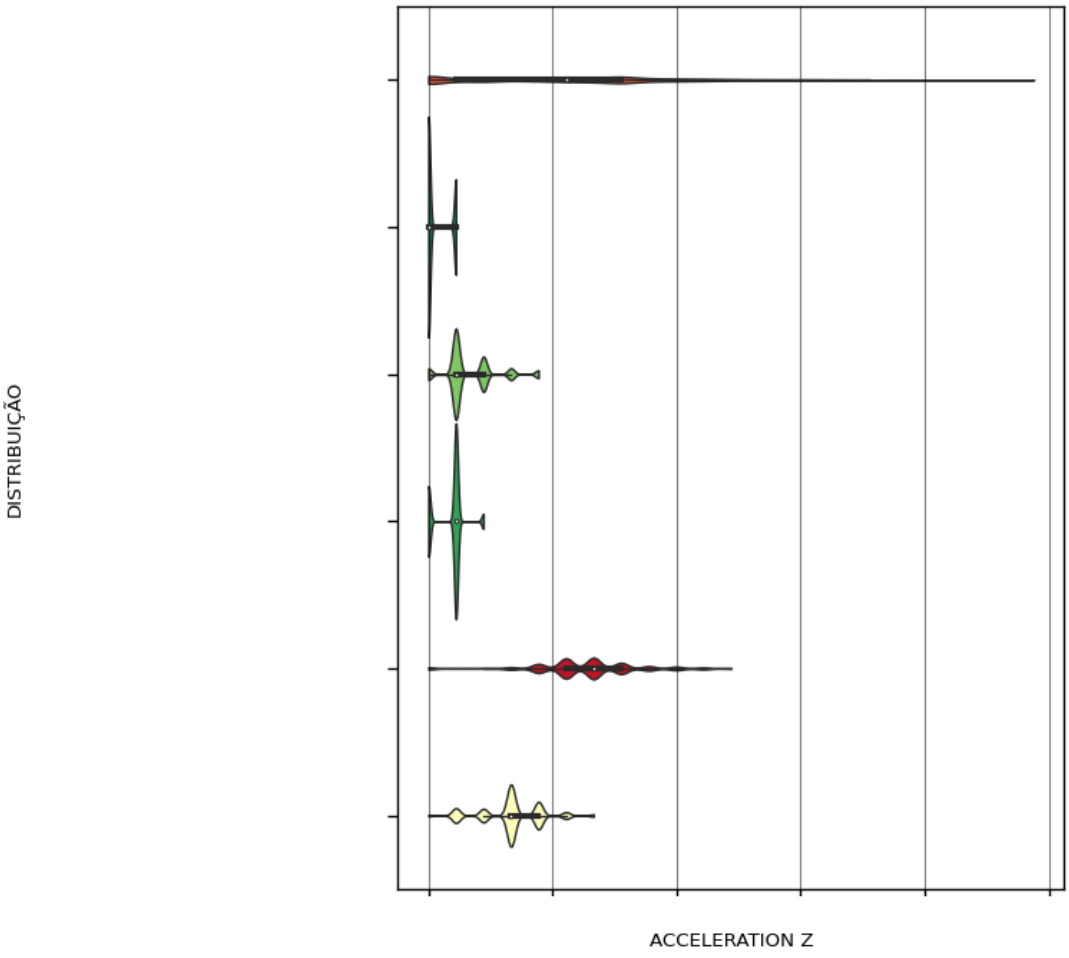


Figura 49 – Distribuição das temperaturas adicionadas antes

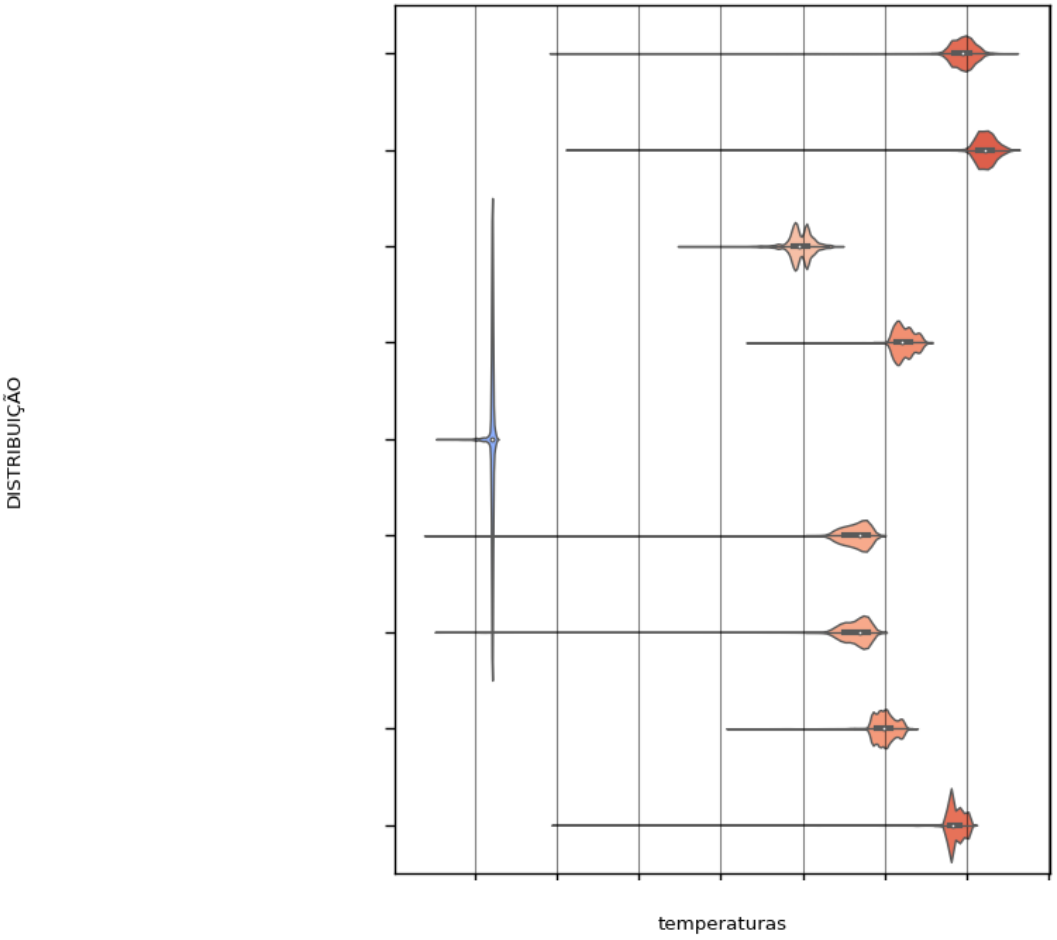


Figura 50 – Distribuição das temperaturas adicionadas antes limpas com 3 σ

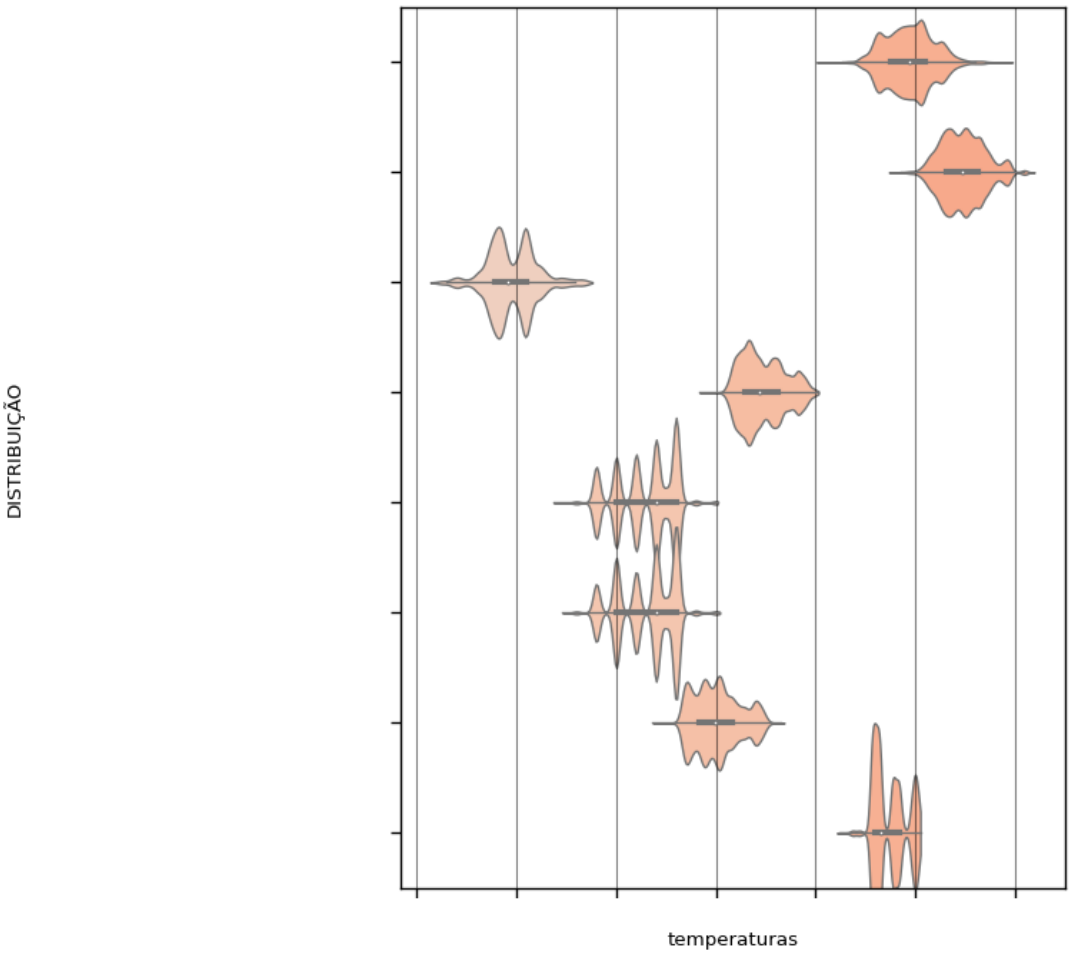


Figura 51 – Distribuição das temperaturas adicionadas depois

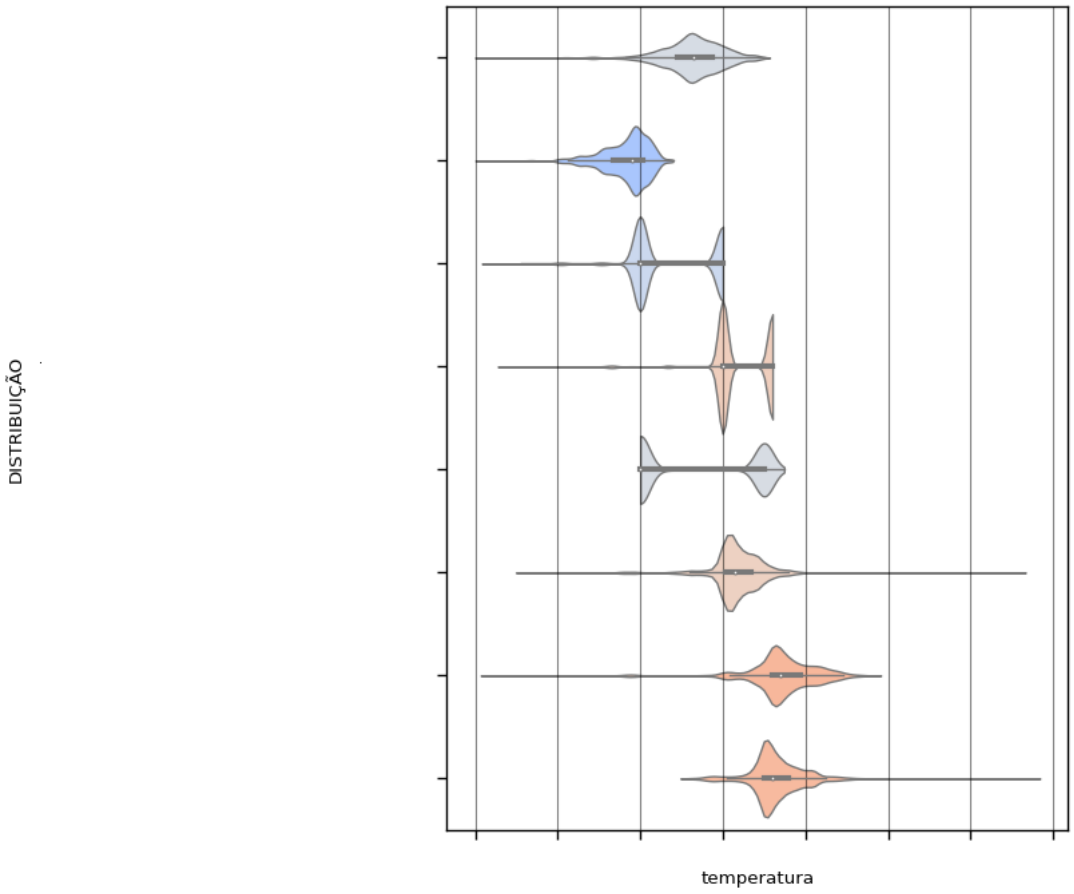


Figura 52 – Distribuição das temperaturas adicionadas depois limpos com 3 σ

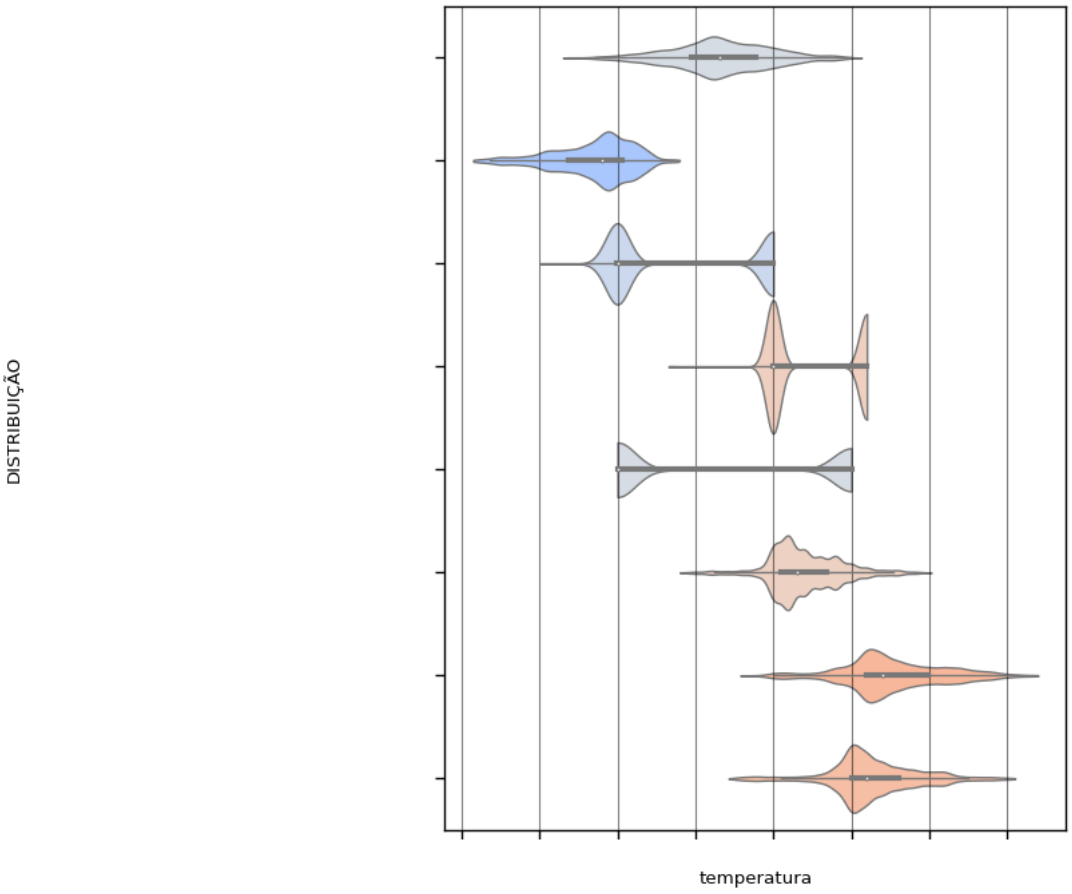


Figura 53 – Distribuição dos dados dos ventiladores

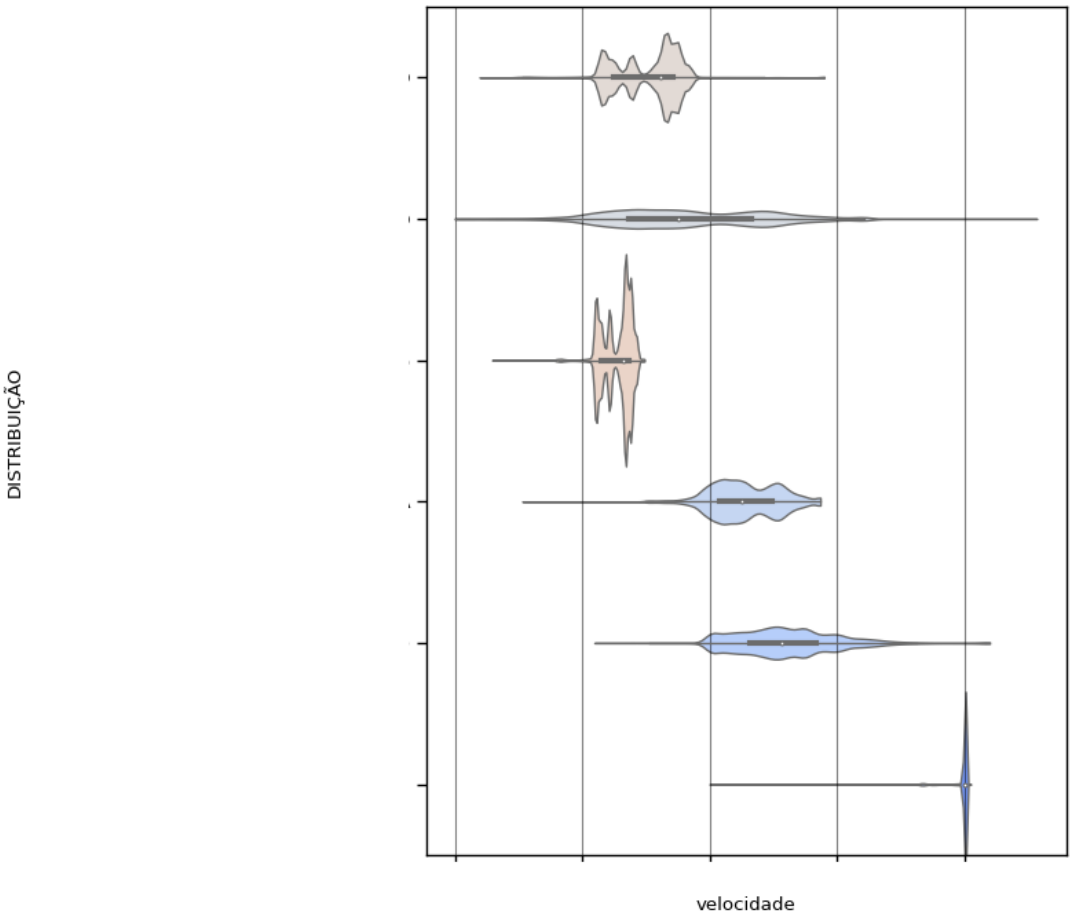


Figura 54 – Distribuição dos dados dos ventiladores limpas com 3 σ

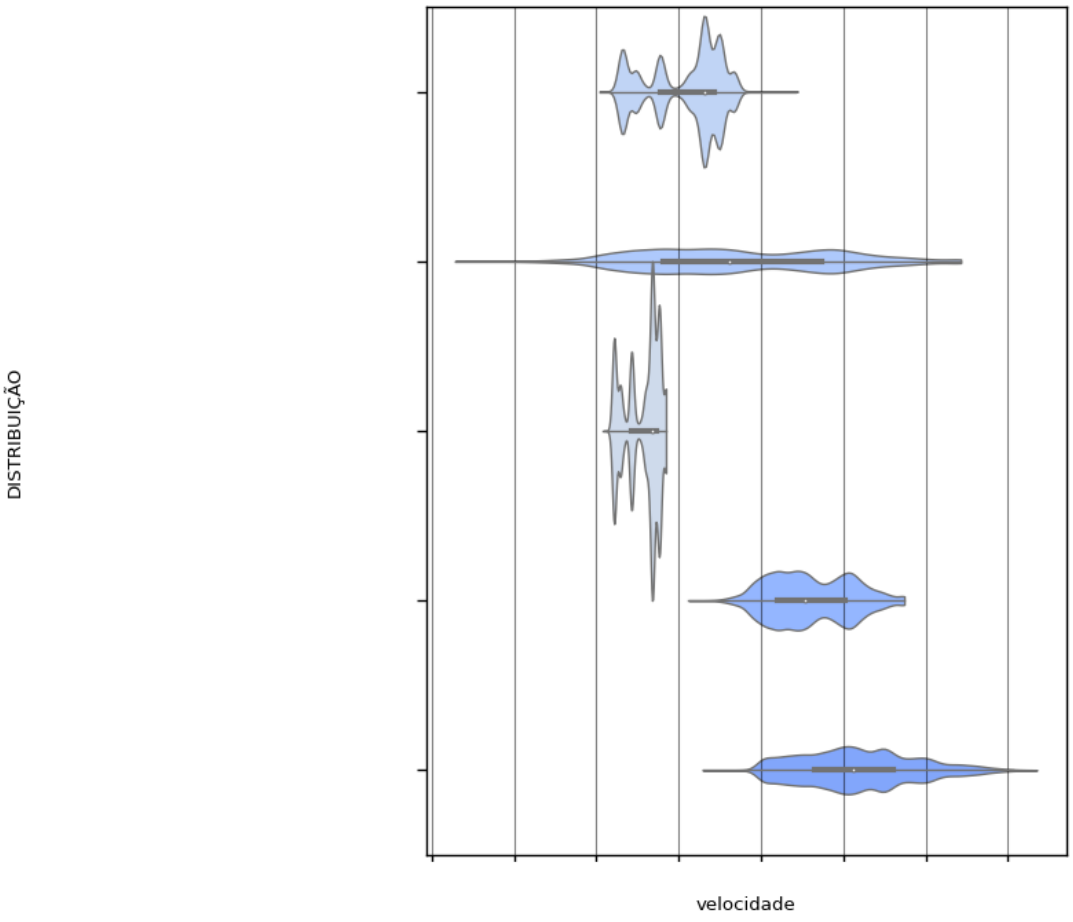


Figura 55 – Distribuição do resto das variáveis

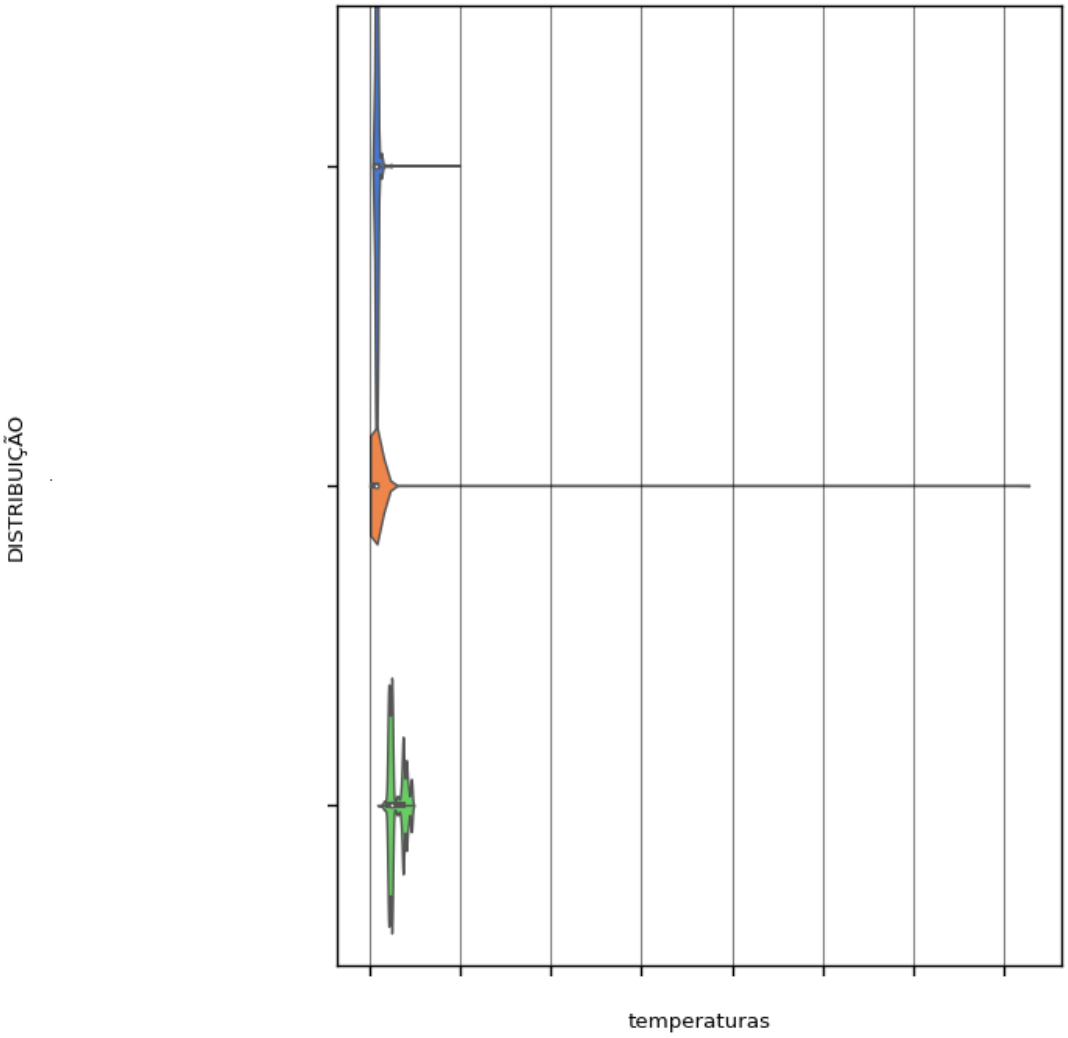
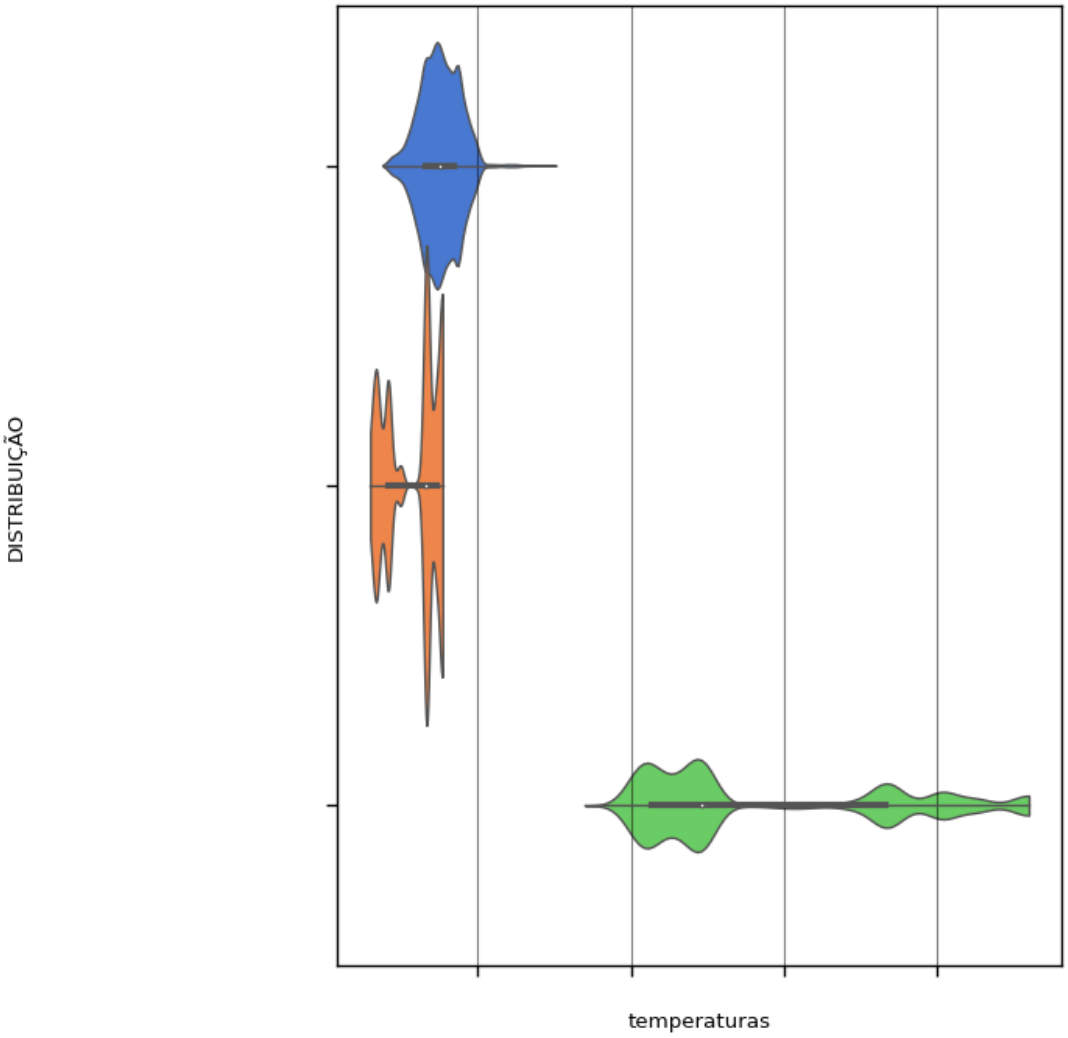


Figura 56 – Distribuição do resto das variáveis limpas com 3 σ



APÊNDICE B – Correlação das variáveis do problema

Figura 57 – Correlação ventiladores Kendal-tau

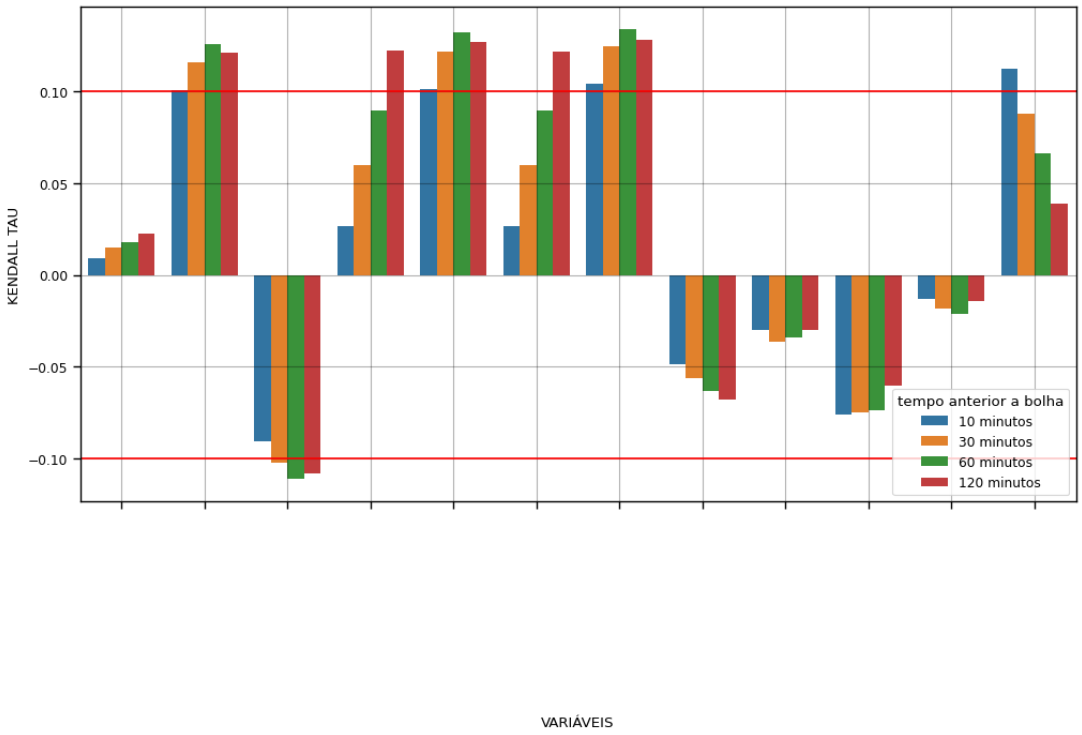


Figura 58 – Correlação temperaturas *motorscan* Kendal-tau

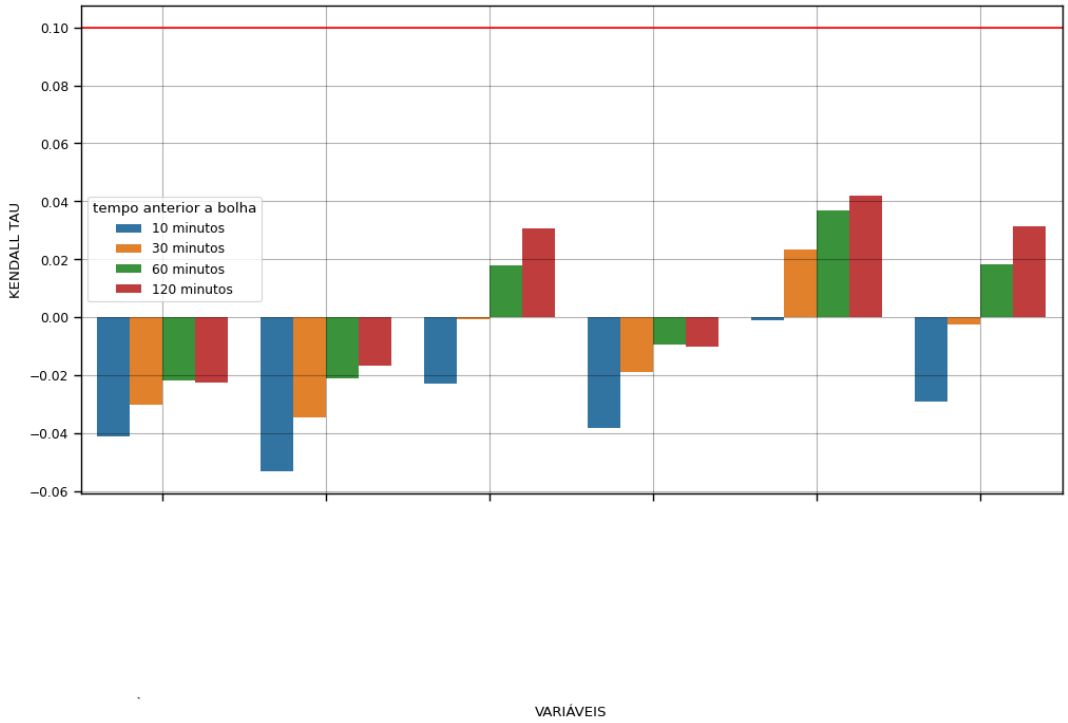


Figura 59 – Correlação acelerações *motorscan* Kendal-tau

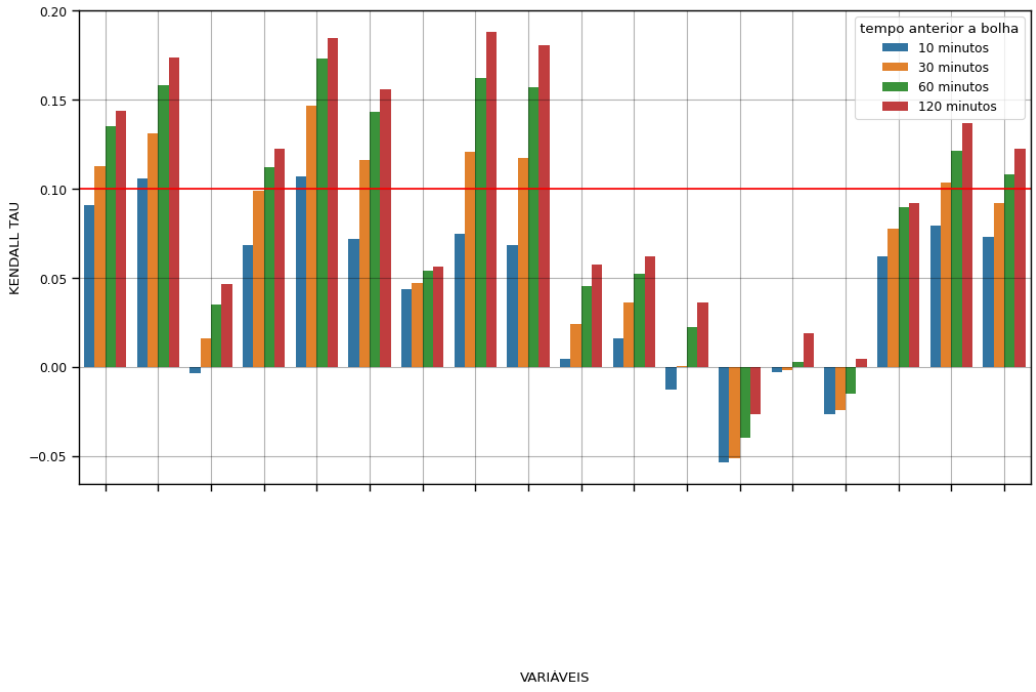


Figura 60 – Correlação temperaturas Kendal-tau

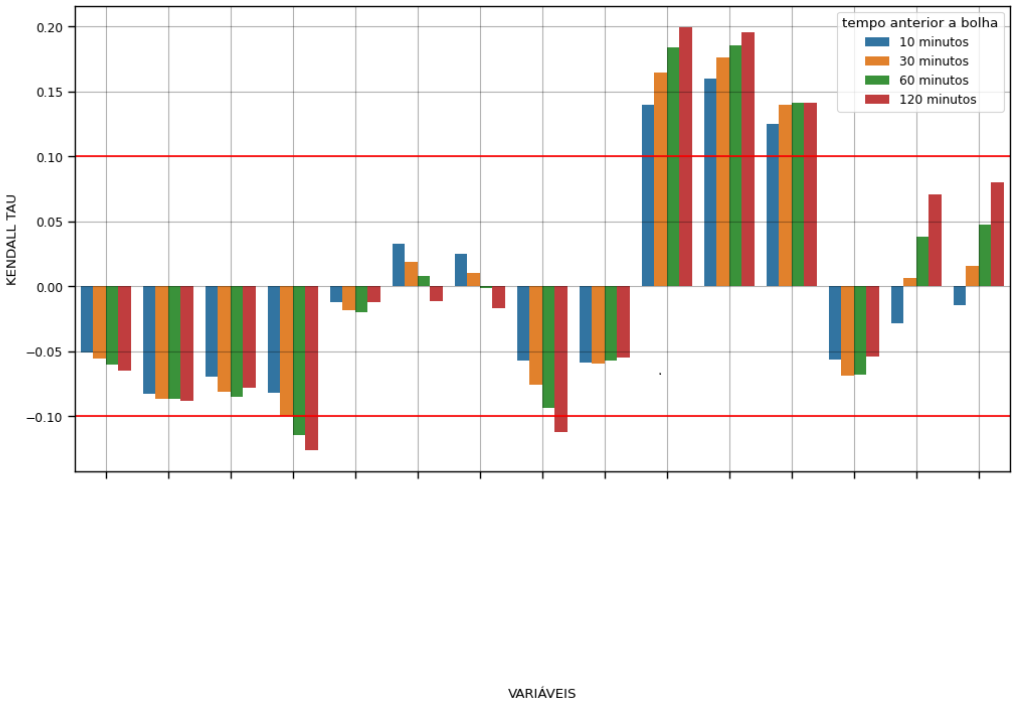


Figura 61 – Correlação venitladores spearman

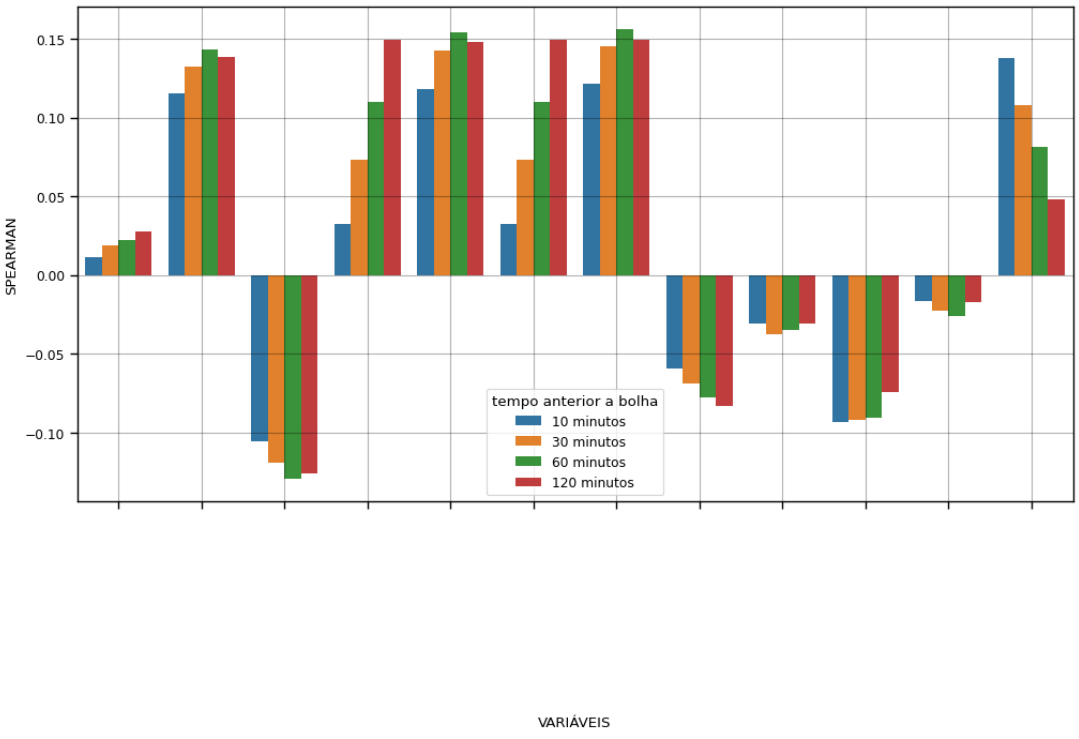


Figura 62 – Correlação acelerações *motorscan* spearman

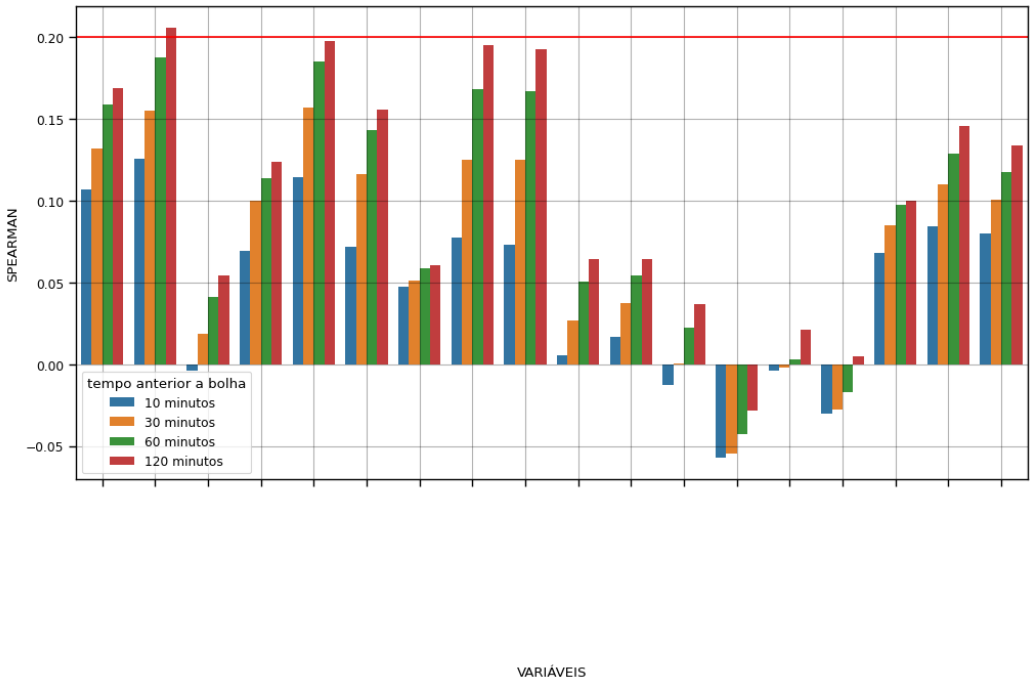


Figura 63 – Correlação temperaturas *motorscan* spearman

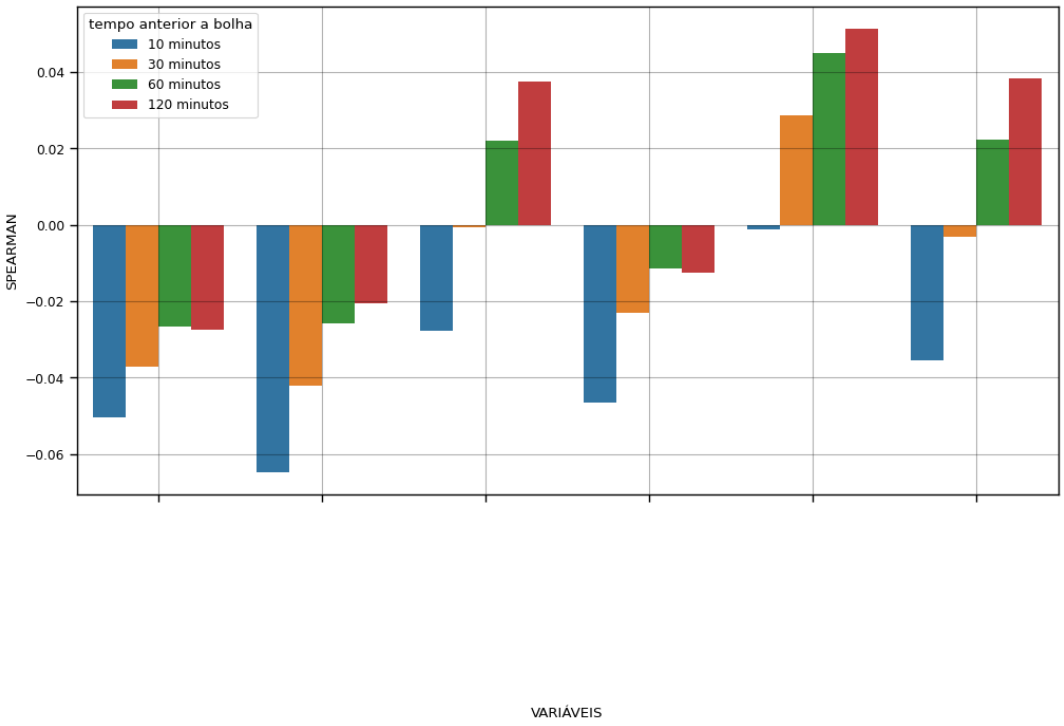


Figura 64 – Correlação temperaturas spearman

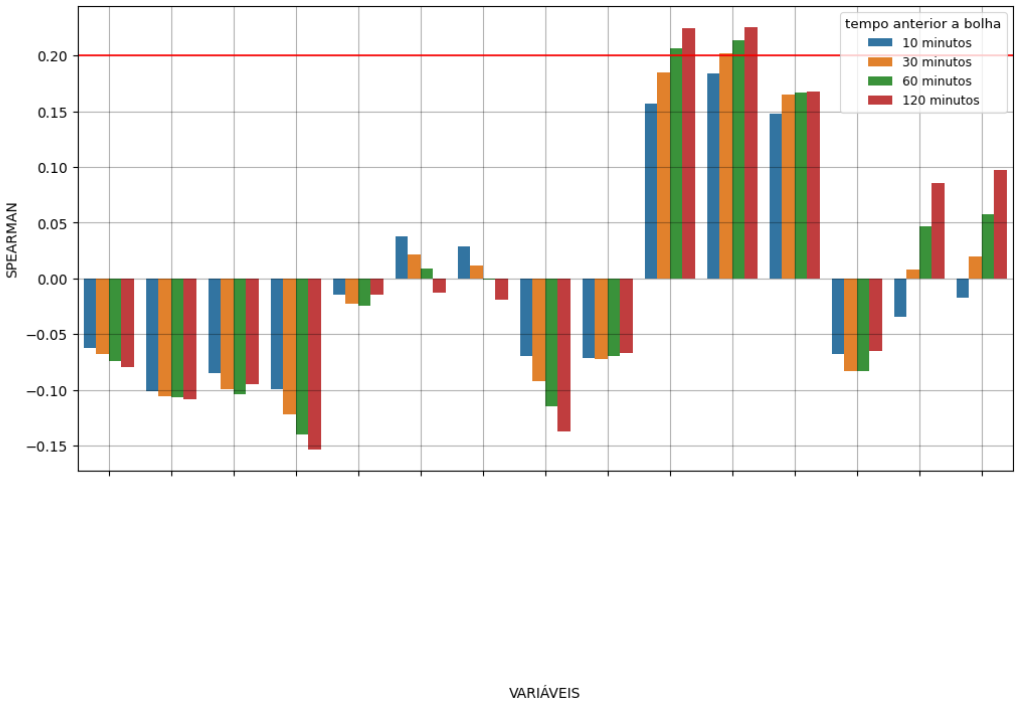


Figura 65 – P-valor ventiladores spearman

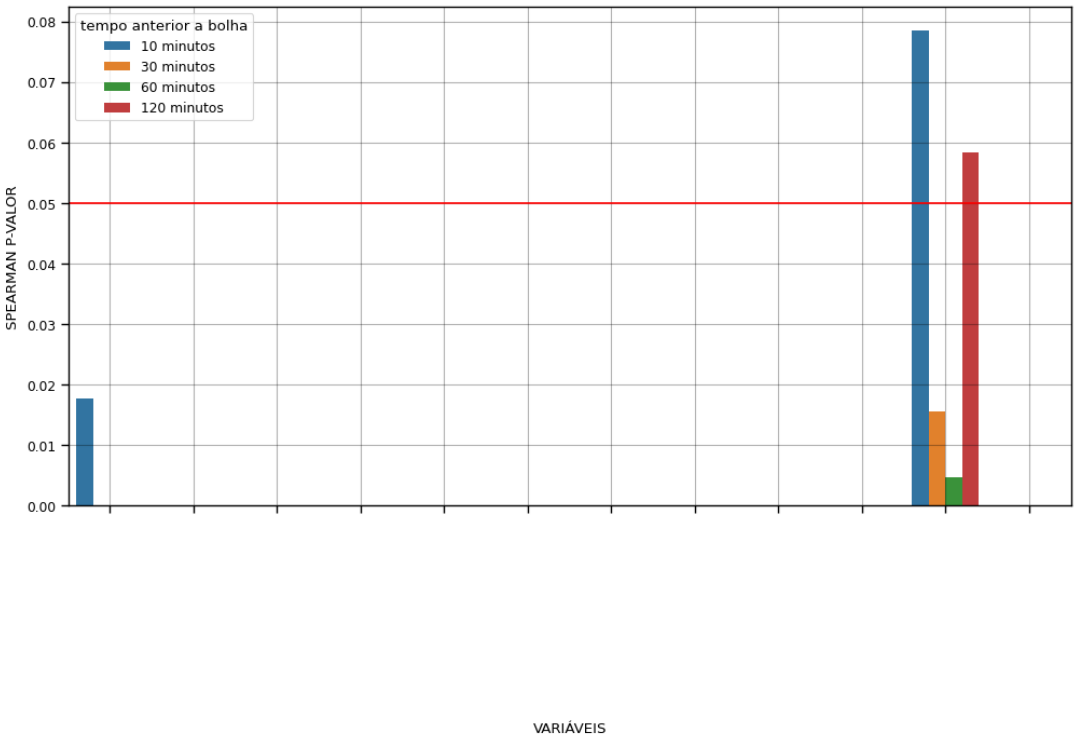


Figura 66 – P-valor acelerações *motorscan* spearman

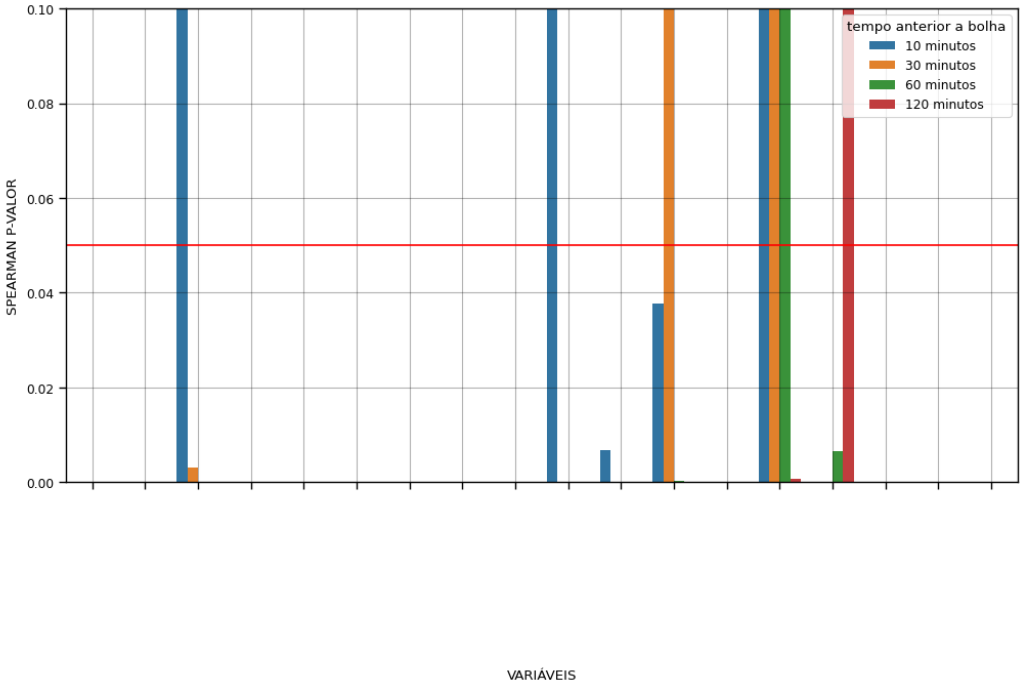


Figura 67 – P-valor temperaturas *motorscan* spearman

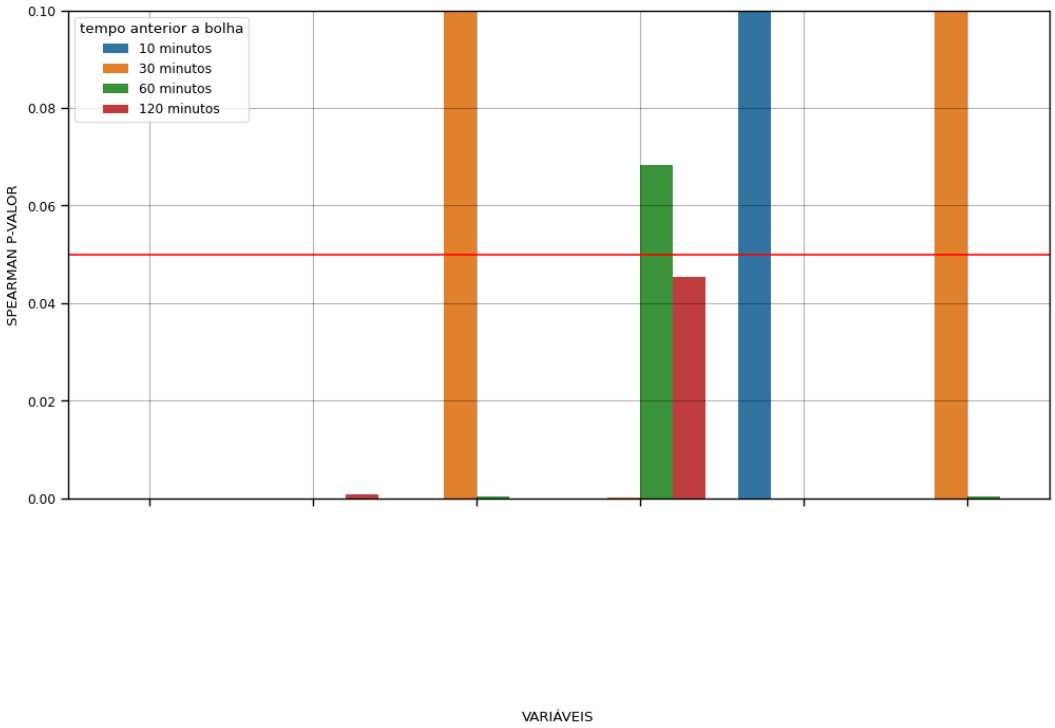


Figura 68 – P-valor temperaturas spearman

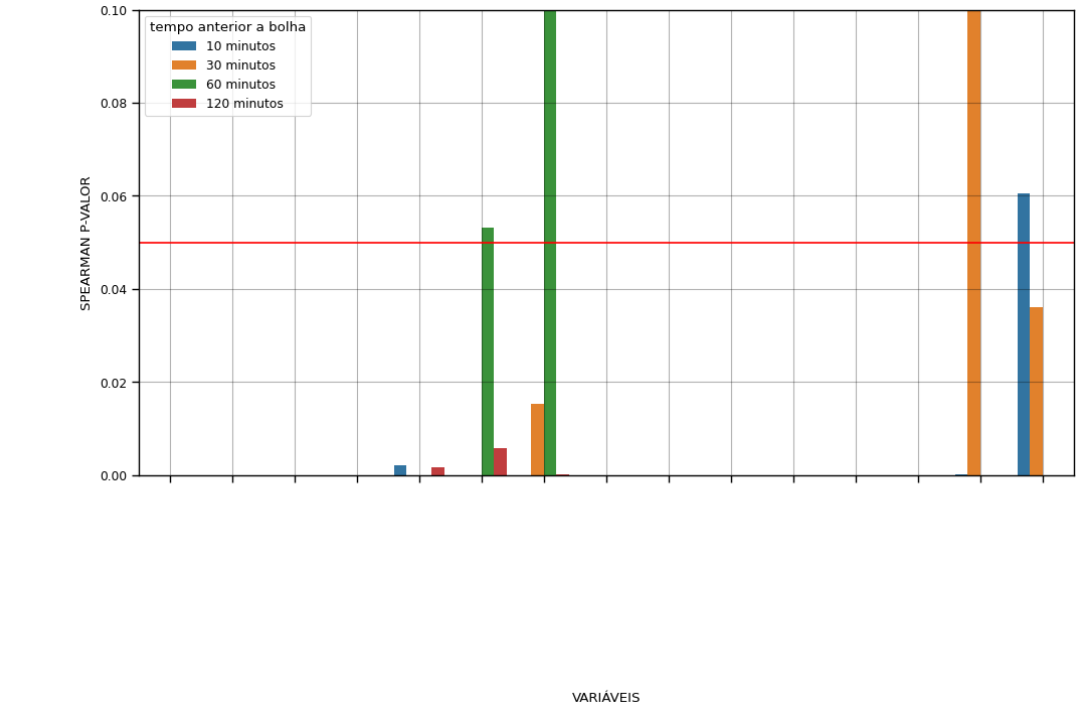


Figura 69 – P-valor venitladores kendall-tau

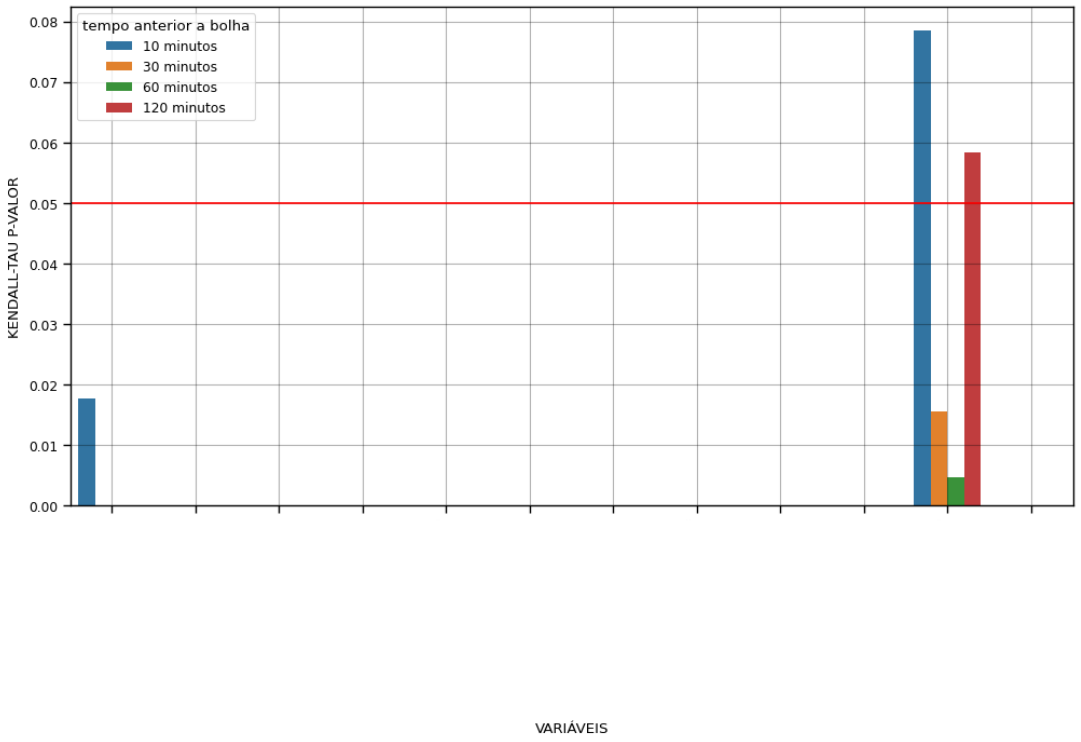


Figura 70 – P-valor acelerações *motorscan* kendall-tau

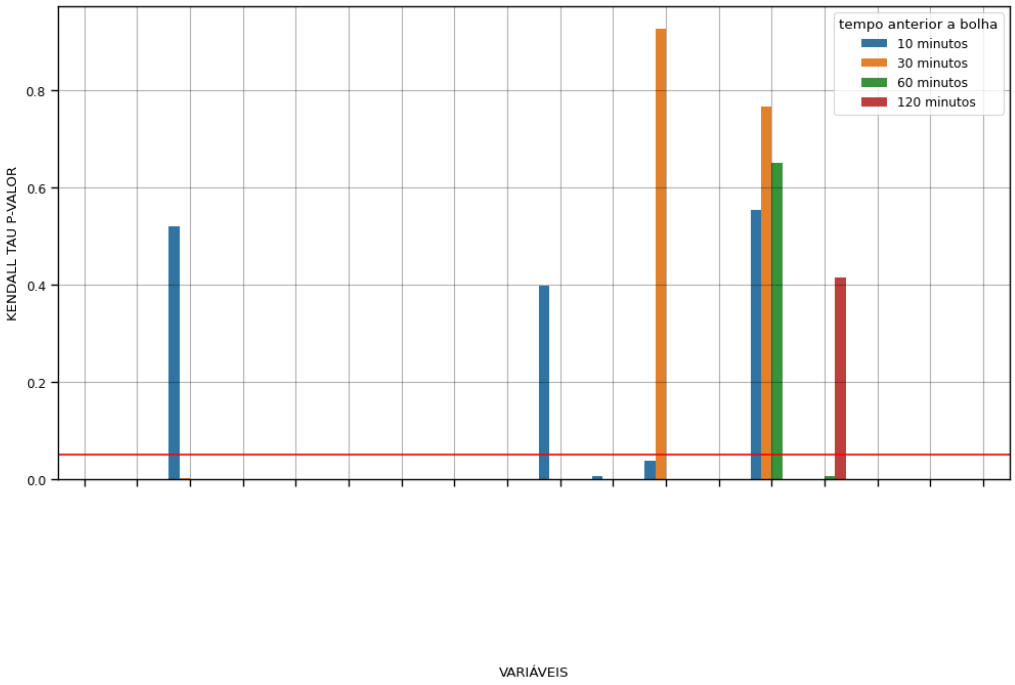


Figura 71 – P-valor temperaturas *motorscan* kendall-tau

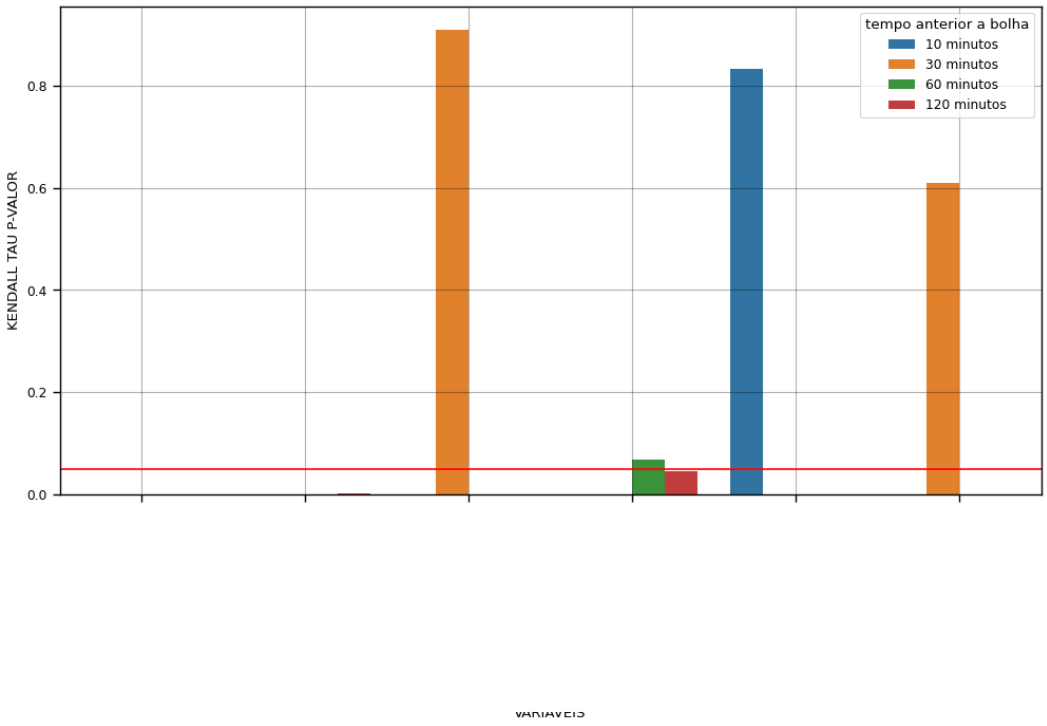


Figura 72 – P-valor temperaturas kendall-tau

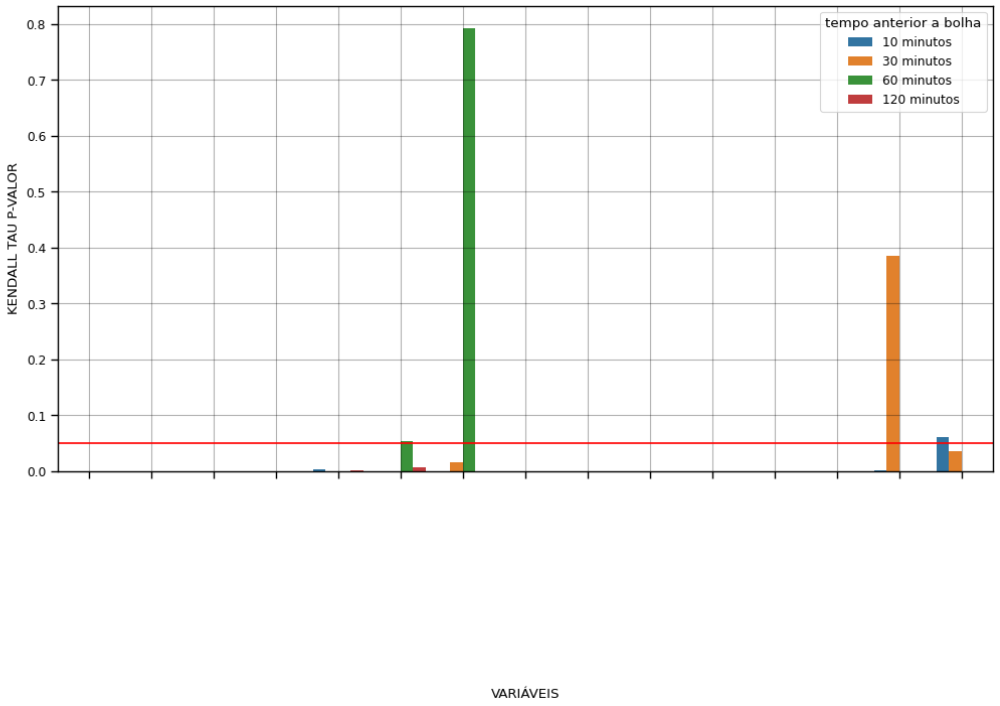


Figura 73 – Matriz de correlação das variáveis do SFM adicionadas antes

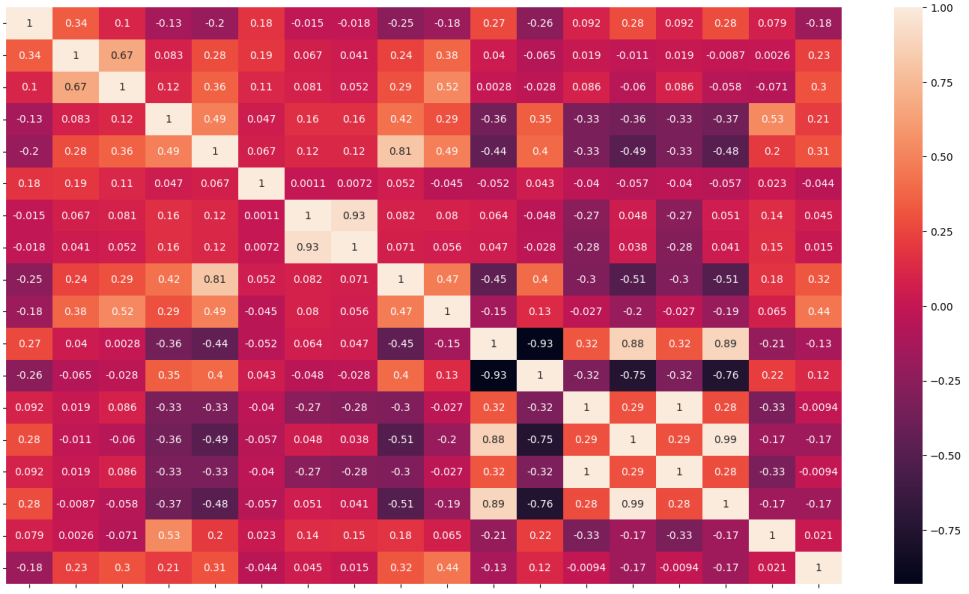


Figura 74 – Matriz de correlação das variáveis do *motorscan*