UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Adolfo Esteves Ribeiro
Eric Kenji Cordeiro

METODOLOGIA PARA DETECÇÃO DE CHATTER EM TEMPO REAL ATRAVÉS DE UM MODELO DE APRENDIZADO DE MÁQUINA

São Carlos

2023

Adolfo Esteves Ribeiro
Eric Kenji Cordeiro

METODOLOGIA PARA DETECÇÃO DE CHATTER EM TEMPO REAL ATRAVÉS DE UM MODELO DE APRENDIZADO DE MÁQUINA

Monografia apresentada ao Curso de Engenharia Mecânica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Mecânico.

Orientadora: Profa. Dra. Maíra Martins da Silva

São Carlos

2023

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues
Fontes da EESC/USP

R484m

Ribeiro, Adolfo Esteves
Metodologia para detecção de chatter em tempo real através de um modelo de aprendizado de máquina / Adolfo Esteves Ribeiro, Eric Kenji Cordeiro; orientadora Maíra Martins da Silva. -- São Carlos, 2023.

Monografia (Graduação em Engenharia Mecanica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2023.

1. Chatter. 2. Machine learning. 3. Random-forest. 4. Python. 5. Matriz de confusão. I. Cordeiro, Eric Kenji. II. Titulo.

Elaborado por Elena Luzia Palloni Gonçalves – CRB 8/4464

FOLHA DE AVALIAÇÃO

Candidatos: Adolfo Esteves Ribeiro e Eric Kenji Cordeiro

Título: Metodologia para detecção de *chatter* em tempo real através de um modelo de aprendizado de máquina

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo Curso de Engenharia Mecânica.

BANCA EXAMINADORA

Professora Dra. Maíra Martins da Silva (Orientadora)

Nota atribuída: 10,0 (dez)

Maíra M. da Silva
(assinatura)

Professor Dr. Alessandro Roger Rodrigues

Nota atribuída: 10,0 (dez)

(assinatura)

Thiago Liquita Sávio

Nota atribuída: 10,0 (dez)

Thiago L. Savio
(assinatura)

Média: 10,0 (dez)

Resultado: APROVADO

Data: 12 / 06 / 2023.

Este trabalho tem condições de ser hospedado no Portal Digital da Biblioteca da EESC

SIM ☑ NÃO ☐ Visto do orientador Maíra M. da Silva

Aos nossos pais e amigos, que nos deram forças em cada etapa dessa jornada e fizeram com que nossa formação fosse possível.

AGRADECIMENTOS

Agradecemos, primeiramente aos nossos pais e familiares que confiaram em nossos potenciais e incentivaram nossos estudos desde as primeiras etapas de nossas vidas.

Aos nossos amigos dentro e fora da universidade, com os quais compartilhamos momentos e experiências de vida.

E, por fim, aos professores que buscaram nos instruir da melhor maneira possível, de forma que nos tornássemos plenamente capazes de exercer a profissão. Em especial à professora Dr. Maíra Martins da Silva pela orientação neste trabalho, pela amizade e auxílio nesta monografia.

"A mente é como um paraquedas,
só funciona se estiver aberta."
Frank Zappa

RESUMO

RIBEIRO, A. E.; CORDEIRO, E. K. **METODOLOGIA PARA DETECÇÃO DE CHATTER EM TEMPO REAL ATRAVÉS DE UM MODELO DE APRENDIZADO DE MÁQUINA**. 2023. 58p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

Chatter é um fenômeno negativo que ocorre em processos de usinagem causado pelas vibrações auto-excitadas do sistema ferramenta-peça, provocando defeitos na peça e possíveis danos à máquina. Em um contexto industrial, o envio dessas peças para etapas posteriores da linha de produção acarreta prejuízos significativos para as empresas, comprometendo a qualidade do produto. Diante disso, este trabalho buscou desenvolver um método de identificação de *chatter* utilizando *machine learning*, visto sua capacidade preditiva através da análise de dados em tempo real. Para isso, estudou-se um modelo supervisionado classificatório utilizando como base o algoritmo de *random-forest*, em que realizou-se o tratamento dos dados, treinamento, refinamento e calibração do classificador obtido, utilizando a linguagem de programação *Python* e bibliotecas auxiliares como *pandas* e *scikit-learn*. Gerou-se a matriz de confusão e métricas do modelo, obtendo um valor de acurácia de 93% e *F-score* de 95% e 87% para os rótulos de estável e *chatter* respectivamente, indicando um bom classificador para identificação desse fenômeno.

Palavras-chave: *Chatter*. *Machine learning*. *Random-forest*. *Python*. Matriz de confusão.

ABSTRACT

RIBEIRO, A. E.; CORDEIRO, E. K. **METHODOLOGY FOR REAL-TIME CHATTER DETECTION THROUGH A MACHINE LEARNING MODEL**. 2023. 58p. Monograph (Conclusion Course Paper) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

Chatter is a negative phenomenon that occurs in machining processes caused by self-excited vibrations of the tool-workpiece system, leading to defects in the workpiece and potential damage to the machine. In an industrial context, sending these parts to subsequent stages of the production line results in significant losses for companies, compromising product quality. That said, this study sought to develop a chatter identification method using machine learning, given its predictive capability through real-time data analysis. To achieve this, a supervised classification model was studied, based on *random forest* algorithm. Data preprocessing, training, refinement, and calibration of the obtained classifier were performed using the *Python* programming language and auxiliary libraries such as *pandas* and *scikit-learn*. The confusion matrix and model metrics were generated, achieving an accuracy value of 93% and *F-score* of 95% and 87% for the stable and *chatter* labels respectively, indicating a good classifier for this phenomenon identification.

Keywords: *Chatter*. Machine learning. *Random-forest*. *Python*. Confusion matrix.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ML	*Machine Learning*
TP	*True Positive*
TN	*True Negative*
FP	*False Positive*
FN	*False Negative*
ACC	Acurácia
PRC	Precisão
REC	*Recall*
TPR	*True Positive Rate*
RFE	*Recursive Feature Elimination*
MB	Métrica de Brier
OOB	*Out-Of-Bag*

LISTA DE SÍMBOLOS

Σ	Letra grega maiúscula sigma
σ	Letra grega minúscula sigma
τ	Letra grega tau

SUMÁRIO

1 INTRODUÇÃO

A engenharia mecânica é um dos ramos da engenharia que tem como objetivo principal a aplicação de princípios da física e da matemática para a análise, fabricação e manutenção de sistemas mecânicos, estando presente ao longo da história da humanidade como uma das áreas mais importantes e influentes para o seu desenvolvimento (HOLZMANN; DALLAMUTA, 2019).

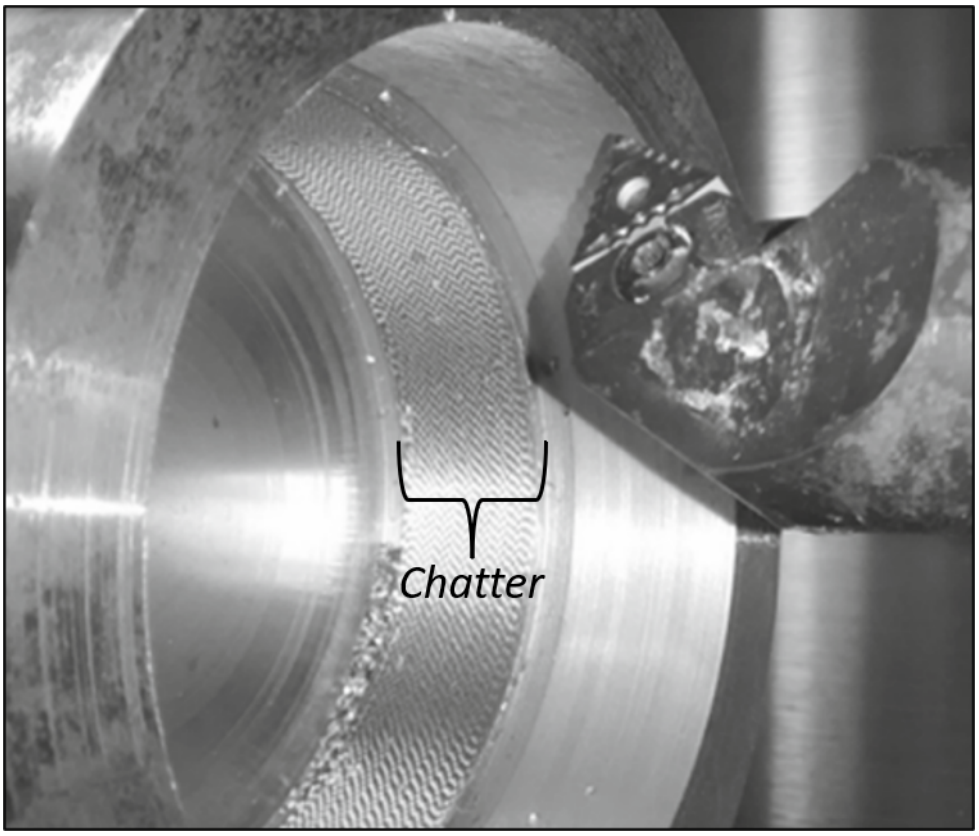
Desde a revolução industrial, época do surgimento das máquinas a vapor e dos motores a combustão, ela tem desempenhado um papel importante para impulsionar o aumento da produção de forma geral, impactando no desenvolvimento econômico e social. No século XXI, período marcado pela globalização e alto nível de consumo em diversas indústrias, como a automotiva, aeroespacial, naval e petroquímica, um outro fator tem se atrelado à busca pela alta taxa de produção: a garantia de qualidade do produto final.

O setor de usinagem, responsável pela manufatura de peças e produtos na indústria, pode ser considerado um dos mais impactados dentro das áreas da engenharia mecânica no que se refere ao quesito de busca pela qualidade, visto que um produto é oriundo do estado de suas partes iniciais, em que caso a fundação apresente defeitos, muito provavelmente seu resultado final também apresentará.

Um dos problemas mais recorrentes durante o processo de usinagem de metais é o fenômeno *chatter*, que pode ser definido como vibrações auto-excitadas causadas pelas relações dinâmicas da ferramenta com a peça, parâmetros de corte e do efeito regenerativo da peça durante o corte. Suas consequências são diversas: ruídos elevados, mau acabamento da peça, danos à ferramenta e gastos de matéria-prima e energia e, por consequência, financeiros (QUINTANA; CIURANA, 2011). A Figura 1 mostra os efeitos negativos do *chatter* sobre uma peça de metal.

Tendo em vista esse cenário, muitos estudos têm sido feitos acerca desse fenômeno, buscando entender suas causas e formas de evitá-lo, a fim de obter um equilíbrio entre uma produção acelerada e alta qualidade. Com os grandes avanços tecnológicos, a indústria 4.0, marcada pela automação e digitalização das atividades industriais, promoveu a utilização de auxílio computacional para monitoramento de diversas medidas durante o processo de usinagem em uma cadeia produtiva, fazendo uso de sensores como acelerômetros, emissores acústicos e sensores piezoelétricos (KULJANIC; SORTINO; TOTIS, 2008).

Figura 1 – Efeito *chatter* sobre a superfície metálica em uma usinagem interna



Fonte: Modificado de Venter *et al.* (2016)

Como resultado dessas implementações, vastas quantidades de dados podem ser obtidas e utilizadas para processos de análises convencionais, a fim de se obter informações relevantes na identificação de *chatter*. Contudo, em um cenário de produção autônoma, a obtenção dos dados e análise somente posterior, não é suficiente para controlar o fluxo de peças defeituosas. Dessa forma, para uma análise em tempo real dos dados obtidos pelos sensores, é possível fazer uso da programação para auxiliar esse processo.

Nos últimos anos, uma vertente da programação focada na análise de dados tem se mostrado bastante presente nos campos de pesquisa: o aprendizado de máquinas, ou *machine learning*. Essa vertente constitui um ramo da área de inteligência artificial que corresponde à criação de modelos preditivos baseados em algoritmos, capazes de classificar determinado conjunto de dados baseado em informações prévias. Assim, fazendo uso dessa ferramenta, juntamente com um processo de monitoramento e obtenção de dados em tempo real, é possível obter-se uma forma prática de identificação de *chatter*.

1.1 Objetivos

O presente trabalho busca desenvolver um modelo supervisionado de aprendizado de máquinas, através da linguagem de programação *Python*, utilizando um conjunto de dados temporais, rotulados de acordo com a presença ou não de *chatter*. Para isso, estuda-se um modelos comumente usados para problemas classificatórios, o *random-forest*, a fim de avaliar seu desempenho através da análise de suas métricas, além de determinar os melhores hiperparâmetros para tal. Assim, o modelo gerado pode ser capaz de receber como dados de entrada as medidas dos sensores em um processo de usinagem real e identificar se houve ou não *chatter* na peça.

1.2 Estrutura do Trabalho

Este trabalho encontra-se organizado da seguinte maneira: na seção 2 consta a revisão bibliográfica do trabalho, contendo conceitos relacionados a *chatter*, *machine learning* e apresentação do modelo de *random-forest*. Na seção 3 consta a metodologia utilizada, apresentando o conjunto de dados utilizado para o treinamento e teste do modelo, os hiperparâmetros fornecidos e o particionamento dos dados temporais. Os resultados do modelo são apresentados e discutidos na seção 4, com gráficos representando sua performance. Por fim, a seção 5 apresenta a conclusão do trabalho.

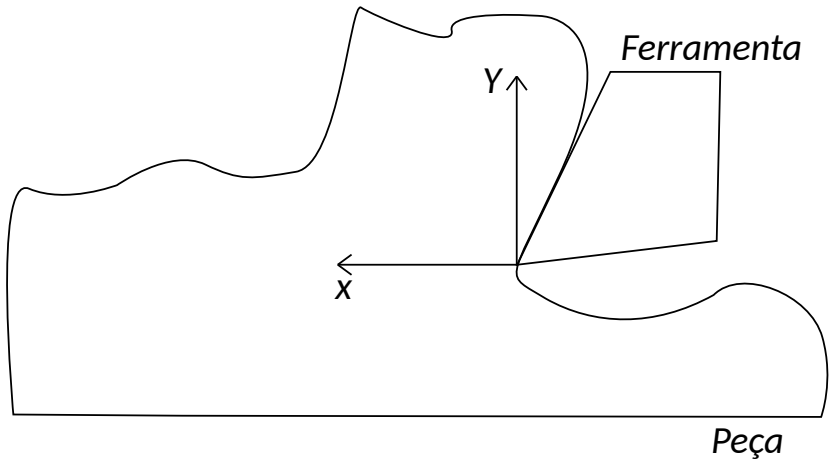
2 REVISÃO BIBLIOGRÁFICA

2.1 Fenômeno *Chatter*

Chatter consiste nas vibrações auto-excitadas do sistema ferramenta-peça e pode ser divido em *chatter* primário e secundário. O primário está relacionado ao processo de corte em si, ou seja, o cisalhamento da ferramenta com a peça, os efeitos termodinâmicos na formação do cavaco e acoplamento de modos, sendo esse último correspondente a uma vibração na direção da força de corte, gerando uma vibração na direção axial e vice-versa. Já o secundário é proveniente do efeito regenerativo da peça, causado pelas ondulações formadas durante a etapa de corte e pode ser classificado como a principal fonte de *chatter* (QUINTANA; CIURANA, 2011).

Esse *chatter* regenerativo é frequentemente observado por conta dos processos de usinagem constantemente realizarem sobreposições de corte, em que, durante o corte, a ferramenta deixa uma superfície ondulada no material e a cada passagem subsequente, um novo perfil é gerado e a diferença de fases entre elas altera a força de corte e espessura do cavaco, intensificando as vibrações (ALAMMARI *et al.*, 2015; QUINTANA; CIURANA, 2011). Esse efeito pode ser observado na Figura 2 em que a passagem da ferramenta resulta em uma superfície ondulada.

Figura 2 – Ondulações de corte causadas pela usinagem



Fonte: Elaborado pelos autores

2.2 *Machine Learning*

2.2.1 Definição

Machine learning (**ML**), ou **aprendizado de máquinas**, pode ser amplamente definido como um algoritmo capaz de processar um conjunto de dados de entrada e, como saída, entrega uma previsão de um determinado resultado de acordo com seu propósito. Sua vantagem encontra-se no fato dele não precisar ser explicitamente programado, ou seja, os modelos são construídos para se adaptarem automaticamente de acordo com os dados recebidos, através do processo de repetição, ficando mais próximo do resultado desejado (NAQA; MURPHY, 2015).

Os modelos de *machine learning* têm sido particularmente úteis para resolução de problemas mais complexos em que o desenvolvimento direto de um código de programação não era trivial ou as variáveis do problema se alteravam de forma que o código precisasse sofrer constante mudanças, visto que, a partir de um conjunto de dados de treino, ele era capaz de fornecer as respostas para novas entradas automaticamente. Alguns cenários em que são aplicados modelos de *machine learning* são:

- Detecção de fraude de crédito bancário
- Recomendação de produtos em sites de compras
- Reconhecimento de imagem e voz
- Análise de sentimentos
- Veículos autônomos
- Previsão de tráfego
- Assistente virtual
- Detecção de *spams*
- Tradução automática

2.2.2 Tipos de Modelos

O funcionamento de um modelo está diretamente relacionado com o conjunto de dados disponíveis. Existem duas principais categorias de modelos de *machine learning*, dependendo das informações presentes nos dados iniciais: **supervisionados** e **não supervisionados**. Para ambos, os dados são compostos por ***features***, que correspondem às informações de entrada para o modelo. Para o supervisionado, cada conjunto de *features* dos dados de treino está atrelado a um ***label*** ou **rótulo**, que indica o resultado que o modelo deveria prever. Já para o não supervisionado, os dados de treino não possuem esses

rótulos e o modelo deve ser capaz de identificar o padrão e relacionamento dos dados e categorizá-los. Vale citar que, para ambos os casos, uma vez que o modelo foi desenvolvido, seu papel é ser capaz de prever o rótulo corretamente a partir de um novo conjunto de dados de entrada (ZHOU, 2021).

Como exemplo, pode-se citar a criação de um modelo de identificação de *spams*, dado dois conjuntos de dados de treino:

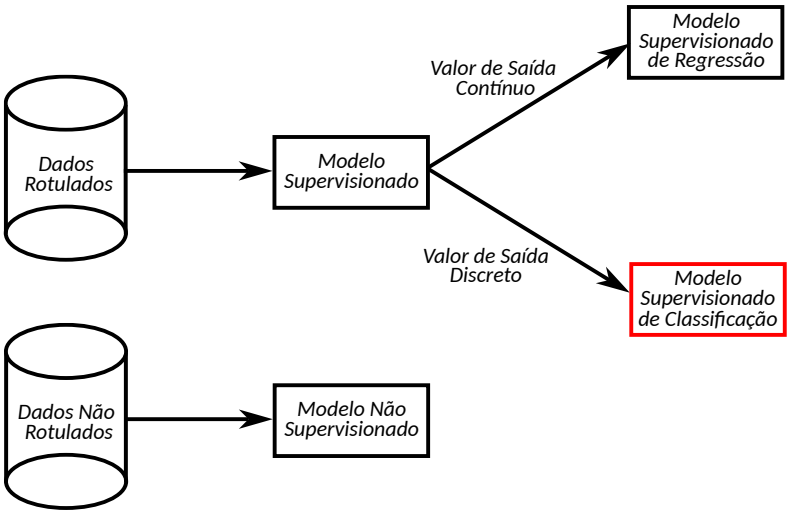
a) Um conjunto de e-mails classificados como *spam* e não *spam*;

b) Outro conjunto de e-mails mas sem nenhuma classificação;

Em ambos os cenários, os e-mails são constituídos de endereços de e-mail do remetente e destinatário, título, assunto e conteúdo. Essas informações representam as *features* que serão utilizadas no modelo. Porém, o primeiro possui um rótulo (*spam* ou não) para cada conjunto de *features* (e-mail), indicando a utilização de um modelo supervisionado. Já o segundo, apresenta apenas as *features*, sendo necessário que identifique de maneira independente a categorização de *spam*, remetendo a um modelo não supervisionado.

Dentre os modelos supervisionados, existem duas categorias: o de **regressão** e de **classificação**. O primeiro representa a previsão de valores contínuos, como a probabilidade de um e-mail ser spam. Já o segundo, representa valores discretos, como o e-mail ser ou não spam. Para este trabalho, o estudo será feito em cima de modelos classificatórios supervisionados. O fluxograma apresentado na Figura 3 apresenta de forma simplificada a divisão descrita anteriormente e o tipo de modelo a ser estudado.

Figura 3 – Fluxograma simplificado dos tipos de modelo



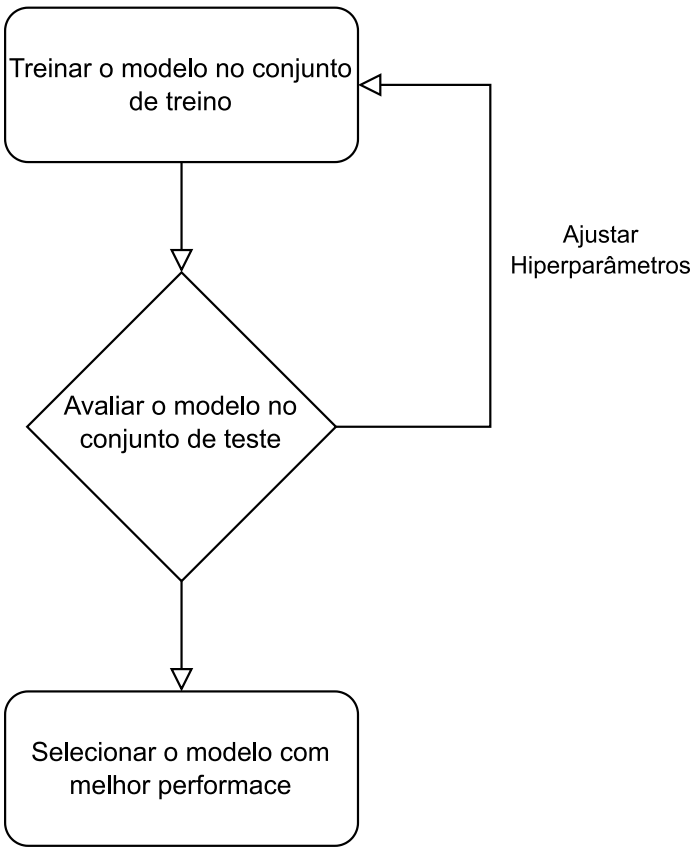
Fonte: Elaborado pelos autores

2.2.3 Conjunto de dados

Para desenvolver um modelo supervisionado classificatório de *machine learning*, é necessário entender alguns conceitos. De forma geral, dado um conjunto inicial de dados, ele deve ser tratado a fim de diminuir possíveis ruídos que possam afetar o desenvolvimento do modelo. Para isso, análises estatísticas como *boxplot* e estudo dos desvios-padrões podem ser feitas a fim de encontrar e remover dados que possuem valores muito divergentes dos demais, conhecidos como *outliers*. Além disso, padronizações da ordem de grandeza dos valores também podem ser consideradas para diminuir a variância do modelo.

Após o tratamento dos dados, realiza-se sua divisão em dois principais conjuntos: os **dados de treino**, utilizados para de fato gerar o aprendizado do modelo, e os **dados de teste**, utilizados para validar o modelo e medir sua capacidade de previsão. Isso é necessário pois, dessa forma, o modelo consegue treinar sobre um determinado conjunto de dados de forma iterativa e testar sua performance em outro conjunto não visto anteriormente. Assim, a partir dos resultados obtidos, refinamentos podem ser feitos, como por exemplo a alteração de parâmetros inseridos antes do treinamento que definem o modelo, os chamados **hiperparâmetros**, e o modelo ser retreinado, a fim de melhorar o resultado final. Esse fluxo de desenvolvimento pode ser observado na Figura 4.

Figura 4 – Fluxograma de treino e teste de um modelo

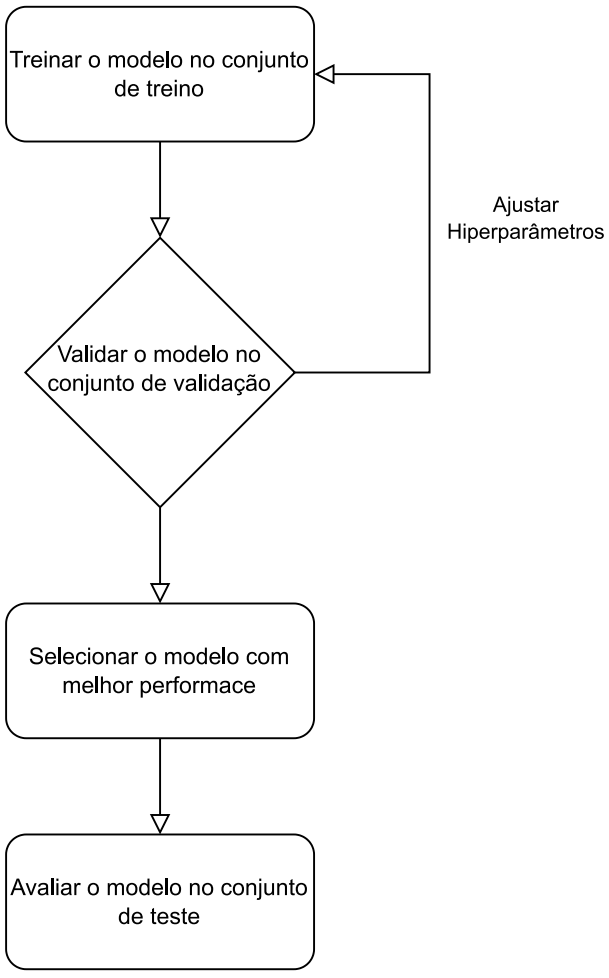


Fonte: Elaborado pelos autores

Um ponto importante desse processo é realizar o ajuste no modelo de forma a não sofrer *overfitting*, ou sobreajuste, fenômeno em que ele performa muito bem sobre os dados previamente observados, normalmente de treino, mas tem baixa performance sobre novos dados, como de teste. Contudo, um modelo que sofre vários reajustes a fim de chegar mais próximo dos resultados de teste, pode estar sofrendo um *overfitting* para esse conjunto de dados, resultando em uma baixa performance para futuros novos dados. Dessa forma, uma terceira divisão pode ser feita a partir dos dados de teste, sendo esses o conjunto de **dados de validação**.

Com essa nova repartição, o fluxo de desenvolvimento do modelo se altera, como apresentado na Figura 5. O conjunto de treino permanece o mesmo, contudo, para avaliar a performance do modelo, utiliza-se o conjunto de validação. Reajustes podem ser feitos, levando ao retreino do modelo e nova avaliação em cima dos dados de validação. Após atingir um resultado satisfatório sobre eles, o modelo é aplicado sobre os dados de teste, repartição nunca antes observada, em que de fato poderá se observar a capacidade de generalização do modelo (MALEKI *et al.*, 2020).

Figura 5 – Fluxograma de treino, validação e teste de um modelo



Fonte: Elaborado pelos autores

2.2.4 Performance de um modelo

Existem diversas maneiras de se avaliar a performance de um modelo de *machine learning*. Elas estão diretamente relacionadas ao tipo do modelo estudado (supervisionado de regressão, supervisionado de classificação, não supervisionado), em que cada um possui seus próprios indicadores. Contudo, de forma geral, um modelo possui um bom desempenho quanto menor for a diferença do resultado previsto com o resultado real, ou seja, quanto menor for sua perda. O refinamento de um modelo tem como objetivo diminuir essa perda.

Para um modelo supervisionado de classificação, um dos indicadores é a **matriz de confusão**, que corresponde a uma tabela contendo a relação entre os rótulos esperados e os previstos pelo modelo. Ele possui quatro principais categorias:

- **Verdadeiro Positivo ou *True Positive* (TP)**: Corresponde aos valores positivos previstos corretamente pelo modelo. No caso de um modelo de previsão de spam por exemplo, seria o conjunto de e-mails que eram de fato spams e o modelo os classificou dessa forma.

- **Verdadeiro Negativo ou *True Negative* (TN)**: Corresponde aos valores negativos previstos corretamente pelo modelo. No caso de um modelo de previsão de spams por exemplo, seria o conjunto de e-mails que não eram spams e o modelo os classificou dessa forma.

- **Falso Positivo ou *False Positive* (FP)**: Corresponde aos valores negativos previstos erroneamente pelo modelo. No caso de um modelo de previsão de spam por exemplo, seria o conjunto de e-mails que não eram spams e o modelo os classificou como spam.

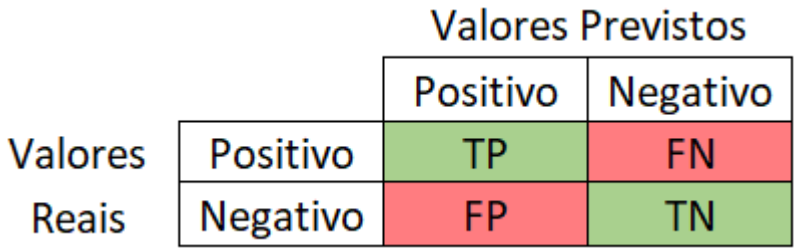
- **Falso Negativo ou *False Negative* (FN)**: Corresponde aos valores positivos previstos erroneamente pelo modelo. No caso de um modelo de previsão de spam por exemplo, seria o conjunto de e-mails que eram de fato spams e o modelo os classificou como não spam.

A Figura 6 apresenta um exemplo da matriz de confusão.

A partir dos resultados obtidos pela matriz de confusão, é possível obter métricas relevantes para análise de performance do modelo. Uma delas é a **Acurácia** (ACC) de um modelo, que representa a taxa geral de acerto do modelo, ou seja, a quantidade de previsões corretas pelo número total de previsões, como representado pela Equação 2.1.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

Figura 6 – Exemplo de Matriz de Confusão



Fonte: Elaborado pelos autores

Outra métrica é a **Precisão** (PRC), que corresponde aos valores classificados corretamente como positivos sobre total de classificações positivas, como indicado na Equação 2.2.

$$PRC = \frac{TP}{TP + FP} \tag{2.2}$$

O ***Recall***, também conhecida como **Taxa de Positivos Verdadeiros** ou ***True Positive Rate*** (TPR), corresponde aos valores classificados corretamente como positivos sobre total de positivos verdadeiros, como indicado na Equação 2.3.

$$TPR = \frac{TP}{TP + FN} \tag{2.3}$$

Nota-se que a relação entre precisão e *recall* é conflituosa, visto que o aumento de uma normalmente ocasiona o decréscimo da outra. Isso acontece pois, supondo que o modelo seja ajustado para ter maior precisão, isso significa que ele tenderá a ser mais rigoroso na quantidade de positivos classificados, aumentando a taxa de positivos acertados por positivos previstos, contudo, isso pode levar a mais verdadeiros positivos sendo erroneamente classificados, resultando em um aumento de falsos negativos e, por consequência, diminuição do *recall*. A relação entre eles pode ser determinada através do ***F-score***, apresentada na Equação 2.4.

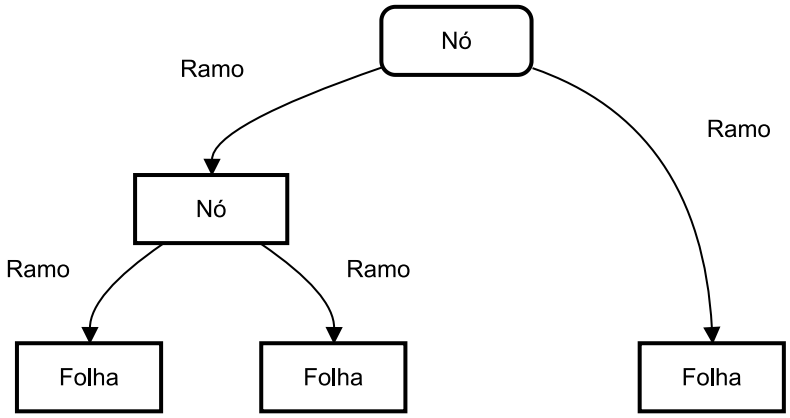
$$F_{score} = 2 \cdot \frac{PRC \cdot TPR}{PRC + TPR} \tag{2.4}$$

Para um modelo de classificação binária, é importante ressaltar que a determinação de um rótulo positivo ou negativo é dado através do cálculo da probabilidade do rótulo pertencer a uma dessas classes. O valor intermediário que determinará sua classificação é chamado de **valor de corte** ou ***threshold***. Alterar esse valor pode afetar diretamente a qualidade de um modelo.

2.3 *Random-Forest*

O modelo de *machine learning* utilizado neste trabalho é o *random-forest*. Esse modelo é comumente utilizado em problemas supervisionados de classificação e possui como base o algoritmo de **árvore de decisão**. Esse algoritmo assemelha-se a um fluxograma, compostos por vários nós, ramos e folhas, em que são executadas verificações acerca de determinadas condições, levando a diferentes caminhos. Um esquemático pode ser encontrado na Figura 7.

Figura 7 – Modelo simplificado de árvore de decisão



Fonte: Elaborado pelos autores

Os nós representam as *features* do modelo, sendo os ramos as condições baseadas nas *features* e as folhas o valor do rótulo. Para o exemplo de identificação de spams, pode se considerar um nó como a variável "usuário" e, caso esteja na lista de contatos, se direciona a um ramo, caso contrário se direciona a outro, levando a mais um nó com outra variável até chegar em uma folha que indicará se é spam ou não. A escolha dos nós é comumente baseada no índice *Gini* ou na entropia.

A desvantagem de se utilizar apenas uma árvore de decisão é sua falta de capacidade de generalização, visto que seus nós são construído com base em todas as *features* do modelo, o que responde bem aos dados de treino mas não a dados novos. O *random-forest* contorna esse problema através de sua randomização, em que, dado um conjunto de dados, uma amostra é selecionada e dela apenas duas ou mais *features* são selecionadas para o primeiro nó. Em sequência, outras duas ou mais, diferentes das previamente escolhidas, são selecionadas e esse processo se repete sucessivamente.

Além da randomização de *features*, o *random-forest* também cria diversas árvores independentes entre si, com amostras diferentes, diminuindo consideravelmente as chances de *overfit* do modelo. Essa estratégia é baseada no *Bootstrapp Aggregation* ou *Bagging*, que consiste em avaliar o resultado a partir da previsão de um conjunto de modelos, ao

invés de observar apenas o valor individual de cada. No caso de um modelo classificatório, o rótulo que mais aparece representa sua classificação final.

Tanto a escolha do número de *features* por amostra, critério de seleção, profundidade das árvores e quantidade de árvores são hiperparâmetros do modelo de *random-forest* que podem ser refinados a fim de se obter um melhor resultado.

3 METODOLOGIA

3.1 Ambiente de Desenvolvimento

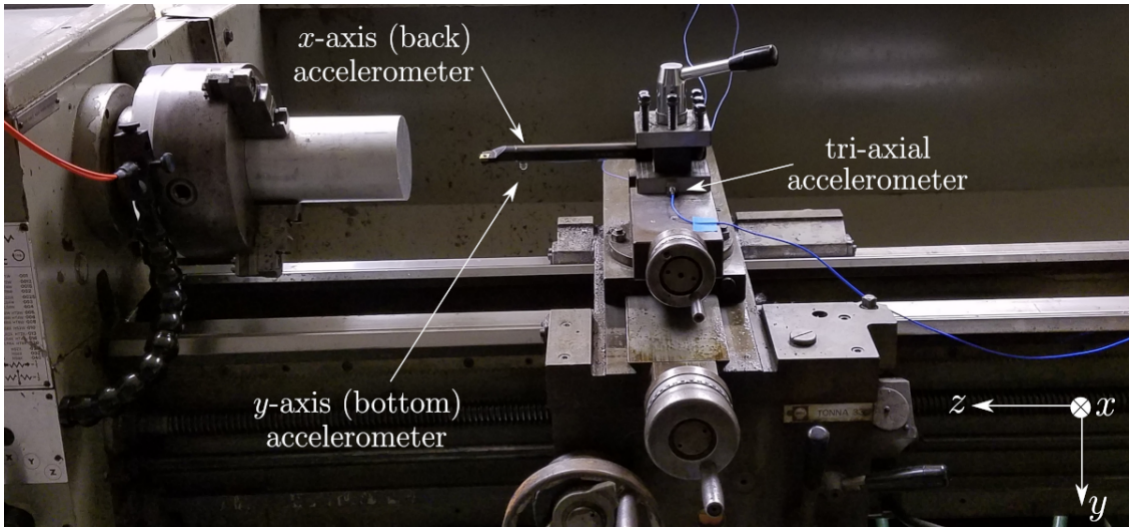
Para este trabalho, fez-se uso majoritariamente da linguagem de programação *Python* para o tratamento dos dados, desenvolvimento, treinamento e refinamento do modelo de *machine learning*, apoiando-se em diversas bibliotecas auxiliares citadas ao longo deste capítulo. O *hardware* utilizado foi concedido por um dos autores e possui as seguintes especificações: Sistema Operacional Windows 10; placa mãe MSI MAG B550 TOMAHAWK; memória RAM de 32GB DDR4; processador AMD Ryzen 7 5700X; e placa de vídeo NVIDIA GeForce RTX 3070 Ti 8GB.

3.2 Conjunto de Dados

3.2.1 Obtenção dos Dados

O desenvolvimento de um modelo de *machine learning* inicia-se através da análise do conjunto de dados que será utilizado para treinar e testar o modelo de fato. Para este trabalho, foi utilizado o conjunto extraído por Yesilli, Khasawneh e Otto (2019a, 2019b, 2019c). Esses dados são oriundos de um processo de torneamento externo de uma peça cilíndrica de alumínio 6061, cujos valores foram captados por três acelerômetros, sendo dois uniaxiais (um medindo as vibrações no eixo x e outro no eixo y) e um triaxial. A montagem desse processo de usinagem e medição pode ser observado na Figura 8.

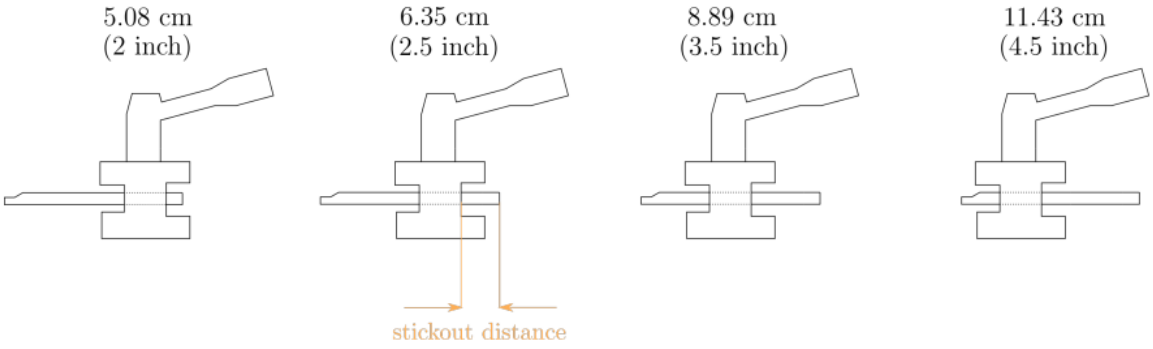
Figura 8 – Montagem do processo de usinagem e medição de *chatter*



Fonte: Yesilli, Khasawneh e Otto (2019c)

Para geração dos dados, foram realizadas quatro diferentes configurações de fixação da ferramenta, alterando seu comprimento livre, como apresentado na Figura 9. Além disso, o processo de torneamento foi feito sob diferentes combinações de velocidades e profundidades de usinagem, variando entre 320 *rpm*, 425 *rpm*, 570 *rpm* e 1030 *rpm*, e 0,254 *mm* até 12,7 *mm* respectivamente.

Figura 9 – Variação da fixação da ferramenta



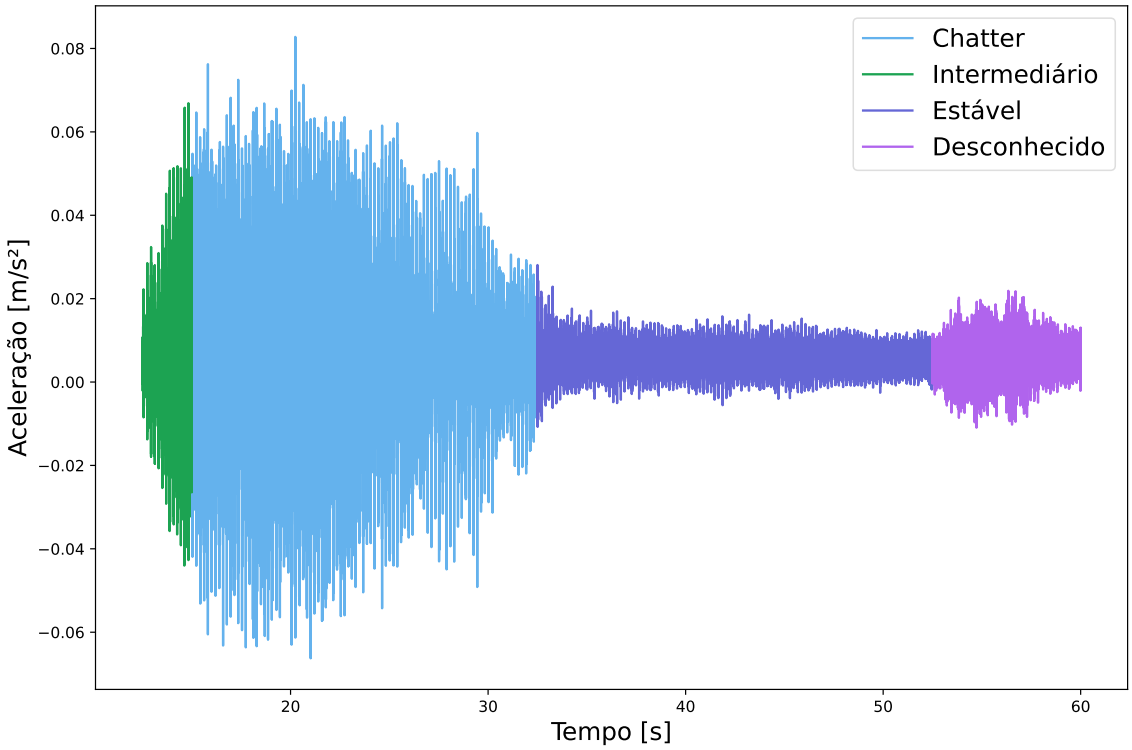
Fonte: Yesilli, Khasawneh e Otto (2019c)

Segundo Yesilli, Khasawneh e Otto (2019a), os dados obtidos originalmente foram analisados e filtrados, obtendo como melhor resultado as medidas referentes ao eixo x do acelerômetro triaxial. Acerca deles, foram criados quatro rótulos com base em suas amplitudes no domínio do tempo e da frequência:

- **Estável (s)**: Baixa amplitude no domínio do tempo e da frequência.
- ***Chatter* intermediário (i)**: Baixa amplitude no domínio do tempo e alta amplitude no domínio da frequência.
- ***Chatter* (c)**: Alta amplitude no domínio do tempo e da frequência.
- **Desconhecido (u)**: todos os outros casos.

Para este trabalho, o desenvolvimento do modelo de *machine learning* foi realizado sobre os arquivos contendo o conjunto de dados rotulado. Na Figura 10 é possível ver graficamente a relação da aceleração pelo tempo e os trechos em que houve *chatter* para uma operação a 320 *rpm*, 50,8 *mm* de comprimento livre e 1,27 *mm* de profundidade de usinagem.

Figura 10 – Gráfico rotulado contendo a medida do acelerômetro pelo tempo para uma operação a 320 *rpm*, 50,8 *mm* de comprimento livre e 1,27 *mm* de profundidade de usinagem



Fonte: Elaborado pelos autores

3.2.2 Tratamento dos Dados

Obtido o conjunto de dados, realiza-se um tratamento a fim de diminuir os ruídos presentes e melhorar a performance do modelo. Para isso, inicialmente buscou-se remover os possíveis *outliers* dos dados, analisando apenas os valores de aceleração que se encontram dentro do intervalo de confiança de três desvios-padrões (representando 99,73% dos dados), como representado na Equação 3.1, sendo x o valor da aceleração, $\overline{x}$ a média das acelerações e σ o valor do desvio padrão, calculado de acordo com a Equação 3.2, com N sendo o número de amostras.

$$\overline{x} - 3\sigma <= x <= \overline{x} + 3\sigma \tag{3.1}$$

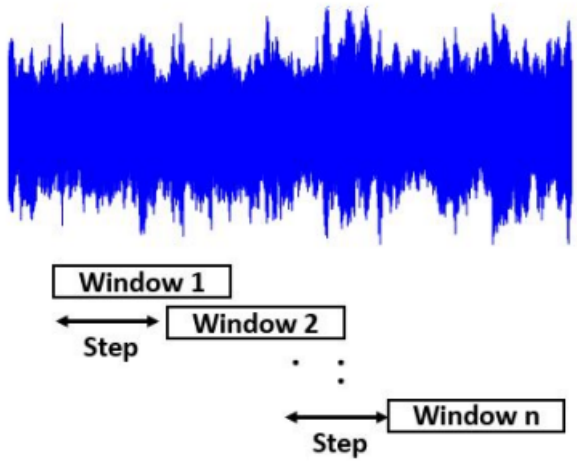
$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}} \tag{3.2}$$

Além disso, removeu-se os dados rotulados como *Desconhecido* e agrupou-se os rotulados como *Chatter* e *Chatter intermediário* em apenas *Chatter*. A leitura e tratamento dos dados foi feita utilizando-se as bibliotecas *scipy*, *numpy* e *pandas* do *Python*.

3.3 Divisão e Janelamento dos dados

Dada a natureza temporal dos dados, para a geração das *features* do modelo realiza-se o método de **Janelas Deslizantes**, ou ***Sliding Windows***, que consiste em dividir uma sequência de dados em janelas de tamanho fixo e movê-las ao longo da sequência com um passo fixo. Sobre cada janela, retira-se o conjunto de *features* utilizados no modelo. Esse processo encontra-se esquematizado na Figura 11.

Figura 11 – Método das janelas deslizantes



Fonte: Modificado de Sestito *et al.* (2022)

Os parâmetros de janelamento adotados encontram-se na Tabela 1.

Tabela 1 – Parâmetros de janelamento dos dados

Parâmetro	**Valor**
Tamanho da Janela	0,5 s
Passo	5 ms

Fonte: Elaborado pelos autores.

Antes da geração das *features*, dividiu-se os dados temporais em uma proporção 70%-30%, correspondentes aos dados de treino e de teste respectivamente, realizando o janelamento em cada um dos conjuntos separadamente. Isso foi feito para evitar que as *features* resultantes dos dados de teste tenham sido extraídas sobre dados de treino, resultando em vazamento dos dados e, por consequência, enviesamento do modelo.

3.4 *Features* do modelo

Realizado a divisão e janelamento dos dados, extraíu-se as *features* que seriam utilizadas inicialmente para o desenvolvimento do modelo. Para isso, utilizou-se as características das séries temporais canônicas (*catch22*), apresentado por Lubba *et al.* (2019), que consistem nas 22 principais medidas estatísticas que melhor performaram sobre mais de 147000 séries temporais, acrescidos da média e desvio-padrão. Essas *features* são um conjunto seleto da biblioteca de análise altamente comparativa de séries temporais (*hctsa*), mais especificamente a versão filtrada contendo 4791 *features*. Para auxiliar na aplicação do cálculo desses valores, utilizou-se a biblioteca em *Python* disponibilizada, *pycatch22*.

Além das 24 *features* provenientes do *catch22*, adicionou-se mais três elementos, sendo eles a velocidade, comprimento livre e profundidade de usinagem, totalizando 27 *features*. Elas encontram-se descritas na Tabela 2. Após a geração das *features*, dividiu-se os dados de teste em dados de validação e teste, em uma proporção de 50% para cada.

3.5 Definição do modelo

Como apresentado no Capítulo 2, o modelo adotado foi o *random-forest*, devido a sua boa aplicabilidade para casos supervisionados de classificação, além de uma implementação mais simplificada mas sem perder a eficácia. Para desenvolvê-lo, utilizou-se a biblioteca *scikit-learn* em *Python*, que, além de possuir a função do modelo em si (*RandomForestClassifier*), apresenta ferramentas que auxiliam no refinamento, geração dos resultados e performance do modelo.

3.5.1 RFE

Antes de realizar o treinamento de fato do modelo, aplicou-se o método de ***Recursive Feature Elimination*** (RFE), que consiste em avaliar as 27 *features* sobre um modelo de *random-forest* com hiperparâmetros iniciais, aplicando um processo iterativo de atribuição de pesos e ranqueamento para cada *feature*, retornando aquelas que possuem maior significância e contribuição. Dessa forma, diminui-se redundâncias e atributos que tornem mais complexo o modelo ou até mesmo impactem negativamente nos resultados. A quantidade de *features* retornadas pode ser definida como parâmetro, contudo, devido à quantidade moderada de *features*, realizou-se esse processo para todos os casos possíveis (de 1 a 26 *features*).

Para cada quantidade de *features*, dados diversos valores de *threshold* diferentes, calculou-se a precisão, *recall* e *F-score* e tomou-se os valores que maximizavam a métrica de *F-score*, para cada iteração do RFE. O conjunto que possuísse o maior *F-score*, seguido da maior precisão, seria utilizado para o treinamento do modelo.

Tabela 2 – *Features* iniciais do modelo

Feature	**Descrição**
DN_Histogram-Mode_5	Modo de distribuição com *z-score* (histograma de 5 classes)
DN_Histogram-Mode_10	Modo de distribuição com *z-score* (histograma de 10 classes)
SB_-BinaryStats_-mean_-longstretch1	Período mais longo de valores consecutivos acima da média
DN_-OutlierInclude_-p_001_mdrmd	Intervalos de tempo entre eventos extremos sucessivos acima da média
DN_-OutlierInclude_-n_001_mdrmd	Intervalos de tempo entre eventos extremos sucessivos abaixo da média
CO_f1ecac	Primeiro cruzamento 1/e da função de autocorrelação
CO_FirstMin_-ac	Primeiro mínimo da função de autocorrelação
SP_Summaries_-welch_rect_-area_5_1	Potência total no quinto mais baixo das frequências no espectro de potência de Fourier
SP_Summaries_-welch_rect_-centroid	Centróide do espectro de potência de Fourier
FC_-LocalSimple_-mean3_stderr	Erro médio de uma previsão média contínua de 3 amostras
CO_trev_1_-num	Estatística de reversibilidade de tempo, $\langle (x_{t+1}x_t)^3 \rangle_t$
CO_-HistogramAMI_-even_2_5	Informação automútua, $m = 2$, $\tau = 5$
IN_AutoMutua-lInfoStats_40_-gaussian_fmmi	Primeiro mínimo da função de informação automútua
MD_hrv_-classic_pnn40	Proporção de diferenças sucessivas superiores a 0.04σ (MIETUS, 2002)
SB_-BinaryStats_-diff_longstretch0	Período mais longo de reduções incrementais sucessivas
SB_-MotifThree_-quantile_hh	Entropia de Shannon de duas letras sucessivas em simbolização equiprovável de 3 letras
FC_-LocalSimple_-mean1_tauresrat	Alteração no comprimento da correlação após diferenciação iterativa
CO_Embed2_-Dist_tau_d_-expfit_meandiff	Ajuste exponencial para distâncias sucessivas no espaço de incorporação 2-d
SC_FluctAnal_-2_dfa_50_1_2_-logi_prop_r1	Proporção de flutuações de escala de tempo mais lentas que escalam com DFA (amostragem de 50%)
SC_FluctAnal_-2_rsrangefit_-50_1_logi_-prop_r1	Proporção de flutuações de escala de tempo mais lentas que se ajustam com escala redimensionada linearmente
SB_Transition-Matrix_3ac_-sumdiagcov	Traço de covariância da matriz de transição entre símbolos no alfabeto de 3 letras
PD_Periodi-cityWang_th0_-01	Medida de periodicidade de Wang, Wirth e Wang (2007)
DN_Mean	Média dos valores
DN_Spread_Std	Desvio-padrão
vel	velocidade de rotação da máquina
stickout_size	Comprimento livre da ferramenta
ap	Profundidade de usinagem

Fonte: Modificado de Lubba *et al.* (2019).

3.5.2 Treinamento e Refinamento do Modelo

Obtido as *features* com resultados mais significativos, realizou-se o treinamento e validação do modelo, aplicando o valor de *threshold* encontrado anteriormente e os hiperparâmetros utilizados durante o RFE, a fim de manter a consistência do processo. Após a etapa de teste, analisou-se os resultados obtidos e realizou-se o refinamento do modelo, através do processo iterativo de alteração do hiperparâmetros, como número de *features* por amostra, critério de seleção, profundidade da árvore e quantidade de estimadores, seguido da reaplicação do RFE gerando um novo conjunto de *features*, treinamento do modelo e reavaliação dos resultados.

A Tabela 3 apresenta os 15 melhores resultados do RFE, com 16 *features* apresentando o maior valor de *F-score* e *precisão* em comparação com as outras. A Tabela 4 apresenta quais foram as *features* utilizadas. Os hiperparâmetros do modelo podem ser encontrados na Tabela 5.

Tabela 3 – Resultado do RFE para seleção de 1 a 26 *features*

n_features	**threshold**	**precision**	**recall**	**f_score**
16	0,77	0,89	0,86	0,88
4	0,53	0,82	0,96	0,88
5	0,48	0,80	0,97	0,88
18	0,78	0,88	0,86	0,87
7	0,58	0,81	0,94	0,87
26	0,63	0,83	0,89	0,86
23	0,66	0,83	0,89	0,86
17	0,70	0,84	0,88	0,86
15	0,66	0,84	0,88	0,86
21	0,67	0,84	0,88	0,86
22	0,60	0,83	0,90	0,86
10	0,51	0,79	0,93	0,86
11	0,52	0,80	0,93	0,86
20	0,61	0,82	0,90	0,86
9	0,54	0,80	0,92	0,86

Fonte: Elaborado pelos autores.

Tabela 4 – 16 *features* utilizadas no modelo final

Feature	**Descrição**
SB_BinaryStats_mean_longstretch1	Período mais longo de valores consecutivos acima da média
CO_f1ecac	Primeiro cruzamento 1/e da função de autocorrelação
SP_Summaries_welch_rect_area_5_1	Potência total no quinto mais baixo das frequências no espectro de potência de Fourier
SP_Summaries_welch_rect_centroid	Centróide do espectro de potência de Fourier
FC_LocalSimple_mean3_stderr	Erro médio de uma previsão média contínua de 3 amostras
CO_HistogramAMI_even_2_5	Informação automútua, $m = 2$, $\tau = 5$
IN_AutoMutualInfoStats_40_gaussian_fmmi	Primeiro mínimo da função de informação automútua
SB_MotifThree_quantile_hh	Entropia de Shannon de duas letras sucessivas em simbolização equiprovável de 3 letras
FC_LocalSimple_mean1_tauresrat	Alteração no comprimento da correlação após diferenciação iterativa
CO_Embed2_Dist_tau_d_expfit_meandiff	Ajuste exponencial para distâncias sucessivas no espaço de incorporação 2-d
SC_FluctAnal_2_rsrangefit_50_1_logi_prop_r1	Proporção de flutuações de escala de tempo mais lentas que se ajustam com escala redimensionada linearmente
PD_PeriodicityWang_th0_01	Medida de periodicidade de Wang, Wirth e Wang (2007)
DN_Mean	Média dos valores
DN_Spread_Std	Desvio-padrão
vel	Velocidade de rotação da máquina
ap	Profundidade de usinagem

Fonte: Elaborado pelos autores.

Tabela 5 – Hiperparâmetros finais do modelo de *random-forest*

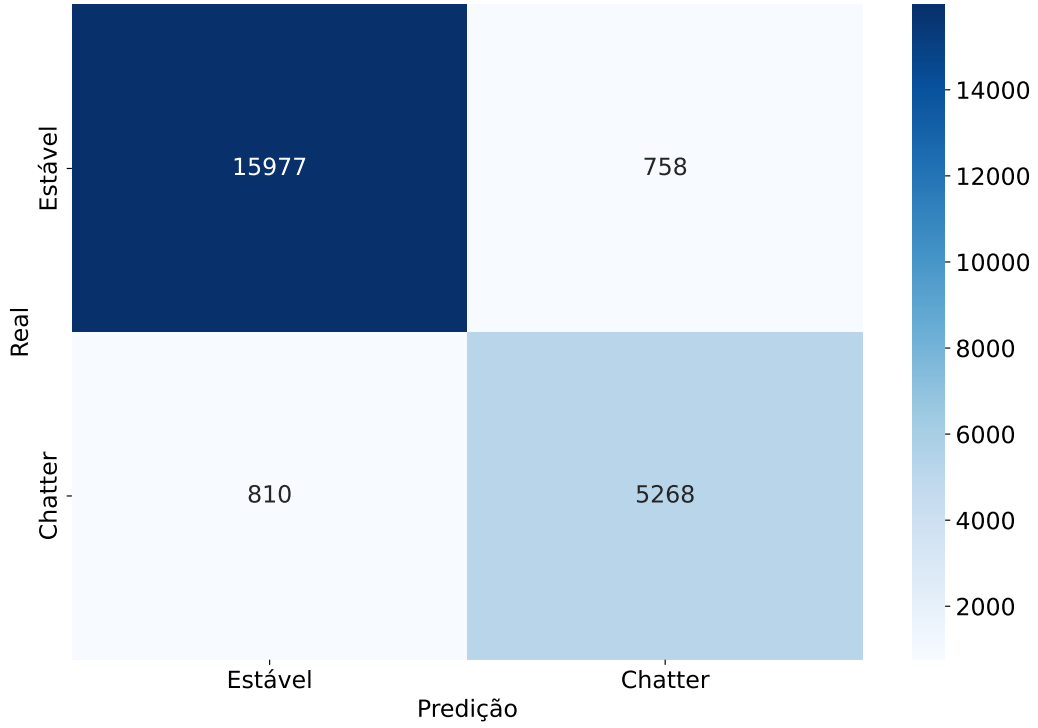
Hiperparâmetro	**Valor**
Nº de estimadores	100
Critério de seleção	Índice *Gini*
Profundidade	Máxima profundidade
Máximo Nº de *features* por árvore	4
Nº mínimo de amostras por folha	1
Nº mínimo de amostras antes de ramificar	2

Fonte: Elaborado pelos autores.

4 RESULTADOS E DISCUSSÃO

Para avaliar o modelo treinado, gerou-se a matriz de confusão em conjunto com suas métricas, correspondendo aos valores de acurácia, precisão, *recall* e *F-score*. Seus resultados podem ser observados na Figura 12 e Tabela 6.

Figura 12 – Matriz de confusão do modelo treinado



Fonte: Elaborado pelos autores

Tabela 6 – Métricas do modelo treinado

Rótulo	**Acurácia**	**Precisão**	***Recall***	***F-score***
Estável	93%	95%	95%	95%
Chatter	93%	87%	87%	87%

Fonte: Elaborado pelos autores.

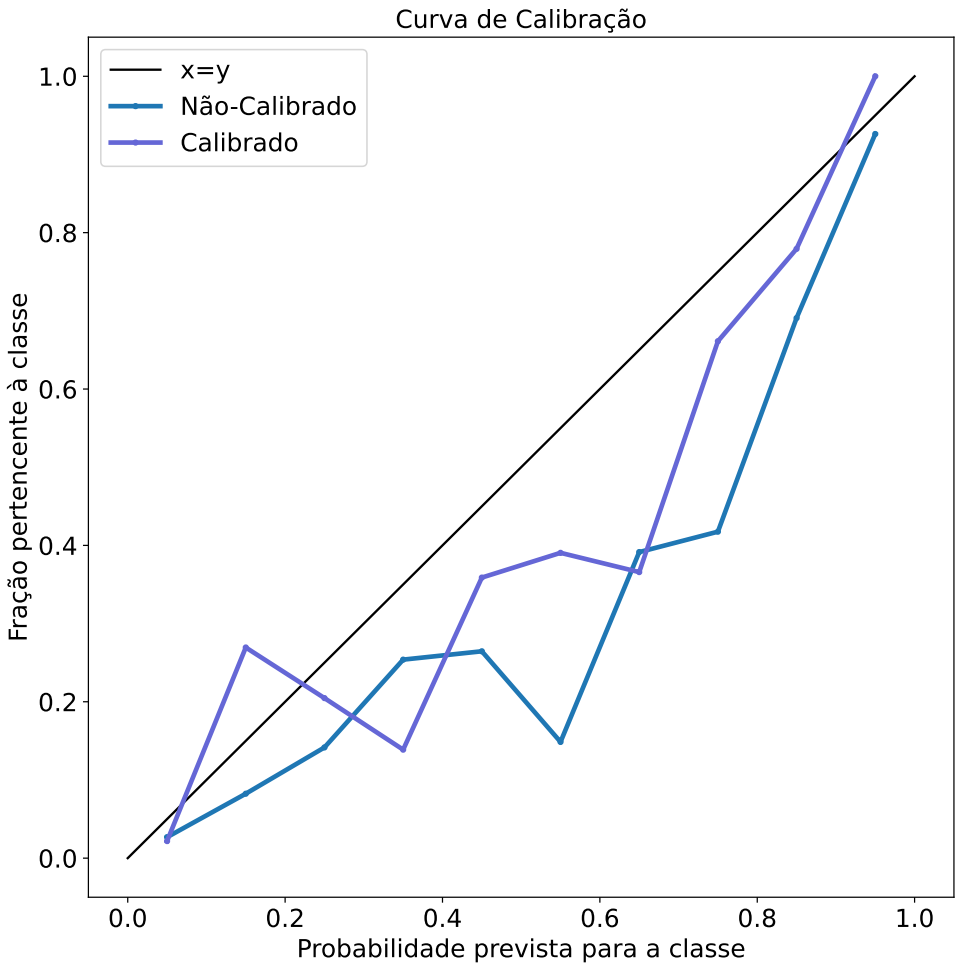
Analisando as métricas obtidas, é possível observar que o modelo apresentou uma alta taxa de acurácia (93%) com alta precisão e *recall* para os dados classificados como estáveis (95% e 95% respectivamente). Isso era esperado devido ao grande volume de dados estáveis, permitindo melhor performance nos treinos e teste. Já os valores de precisão e *recall* para o rótulo de *chatter*, apesar de satisfatórios, estão um pouco abaixo em

comparação com sua contraparte estável, mesmo possuindo menos dados disponíveis para treino e teste. Além disso, a probabilidade apresentada pelo modelo apenas treinado pode não representar na prática o resultado real, devido a uma possível tendência sistemática de superestimar ou subestimar as probabilidades.

4.1 Calibração do Modelo

Para evitar isso e melhorar os resultados do classificador, realiza-se a calibração do modelo através da função *CalibratedClassifierCV* da biblioteca *scikit-learn*. A calibração adotada é a de Platt, cuja técnica ajusta um modelo adicional logístico às saídas do classificador base. O modelo adicional é então treinado para prever a probabilidade correta das classes com base nas saídas originais. Um ponto importante é o fato dessa calibração não alterar os hiperparâmetros do modelo, visto que é uma técnica de pós-processamento. Com o modelo treinado e calibrado, o único valor alterado é o *threshold*, que passou de 0,77 para 0,72.

Figura 13 – Comparação entre as curvas de calibração



Fonte: Elaborado pelos autores

A Figura 13 apresenta as curvas referentes ao modelo não-calibrado e calibrado em comparação com a curva ideal, representada pela reta $x = y$. Nota-se que o modelo treinado e calibrado encontra-se mais próximo da reta em comparação com o modelo apenas treinado. Isso é confirmado pela métrica de Brier, representada na Equação 4.1, que mede o desempenho da probabilidade prevista (f_t) e o resultado real (o_t) de ambos os modelos, em que quanto menor o valor, melhor o resultado do modelo.

$$MB = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2 \tag{4.1}$$

Como pode ser observado na Tabela 7, sua métrica calibrada é menor que a não calibrada, indicando que houve melhoria na confiabilidade do classificador, ou seja, quando for indicado uma probabilidade de 80% do dado ser *chatter* por exemplo, sua taxa de acerto está mais próxima desses 80%.

Tabela 7 – Comparação da métrica de Brier entre os modelos

Modelo	**Métrica de Brier**
Não Calibrado	0.0605
Calibrado	0.0554

Fonte: Elaborado pelos autores.

Com o modelo treinado e calibrado, obteve-se uma nova matriz de confusão e conjunto de métricas, apresentadas na Tabela 8 e Figura 14, além da curva de precisão-*recall*, representada pela Figura 15, indicando uma boa relação entre essas duas métricas.

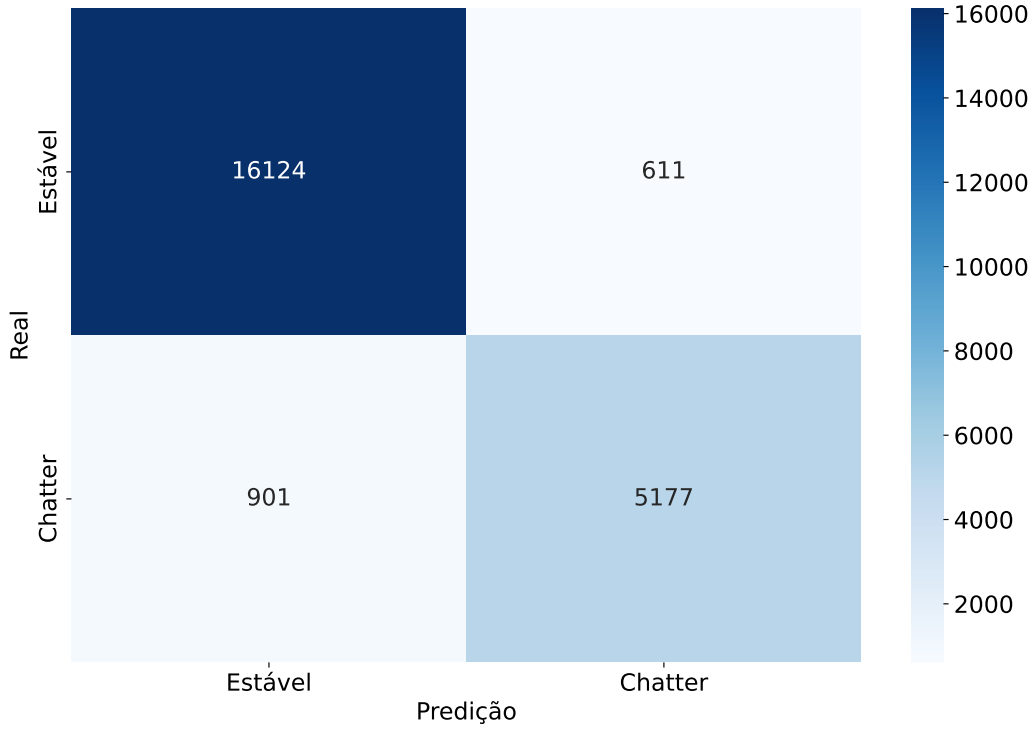
Tabela 8 – Métricas do modelo treinado e calibrado

Rótulo	**Acurácia**	**Precisão**	***Recall***	***F-score***
Estável	93%	95%	96%	96%
Chatter	93%	89%	85%	87%

Fonte: Elaborado pelos autores.

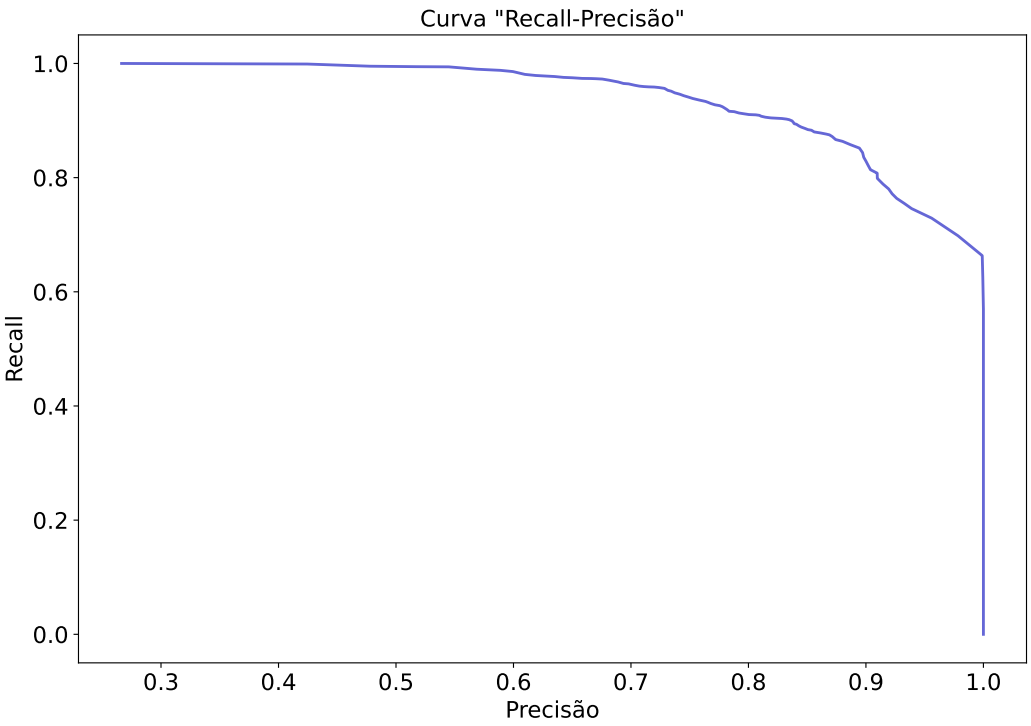
Analisando os resultados obtidos e comparando com as métricas anteriores, é possível notar que a acurácia do modelo se manteve no valor de 93%. Para a classe de *chatter*, houve um aumento de 2,3% na precisão com um decréscimo dos mesmos 2,3% para o *recall*, indicando uma manutenção na identificação de *chatter* pelo modelo. Apesar do aumento não ser muito expressivo, vale citar que esses são os resultados do modelo calibrado, ou seja, os valores apresentados possuem maior confiabilidade em relação às métricas não calibradas, sem perder eficácia, indicando um resultado final mais significativo para o modelo.

Figura 14 – Matriz de confusão do modelo treinado e calibrado



Fonte: Elaborado pelos autores

Figura 15 – Curva de precisão-*recall* do modelo treinado e calibrado



Fonte: Elaborado pelos autores

Outro ponto relevante é que houve um decréscimo de quase 20% no número de falso positivos, ou seja, o modelo passa a acusar menos dados estáveis como *chatter*. Trazendo para uma aplicação real no contexto de indústria 4.0, caso o modelo seja implementado em uma esteira de produção autônoma que retira de linha as peças acusadas com *chatter*, esse resultado reflete em menos peças normais sendo descartadas erroneamente. Em contrapartida, houve um acréscimo de 11% no número de falso negativos, que representa uma perda absoluta menor em comparação com o decréscimo de falso positivos. Contudo, em uma situação real em que não houvesse um identificador de *chatter* e o número de falsos negativos, ou seja, peças contendo *chatter*, seria de 100%, o resultado final obtido pelo modelo é satisfatório.

4.2 Teste

Terminado o processo de ajuste utilizando o conjunto de validação, é necessário avaliar a *performance* em outra parcela de dados não antes vistos pelo modelo, ou conhecidos como OOB (*Out-Of-Bag*), de forma a obter uma perspectiva mais realista sobre suas métricas, visto que os parâmetros e hiperparâmetros não foram especialmente ajustados para as particularidades do conjunto em específico. Sendo assim, registrou-se o desempenho no conjunto de teste e obteve-se a Tabela 9 e a Figura 16

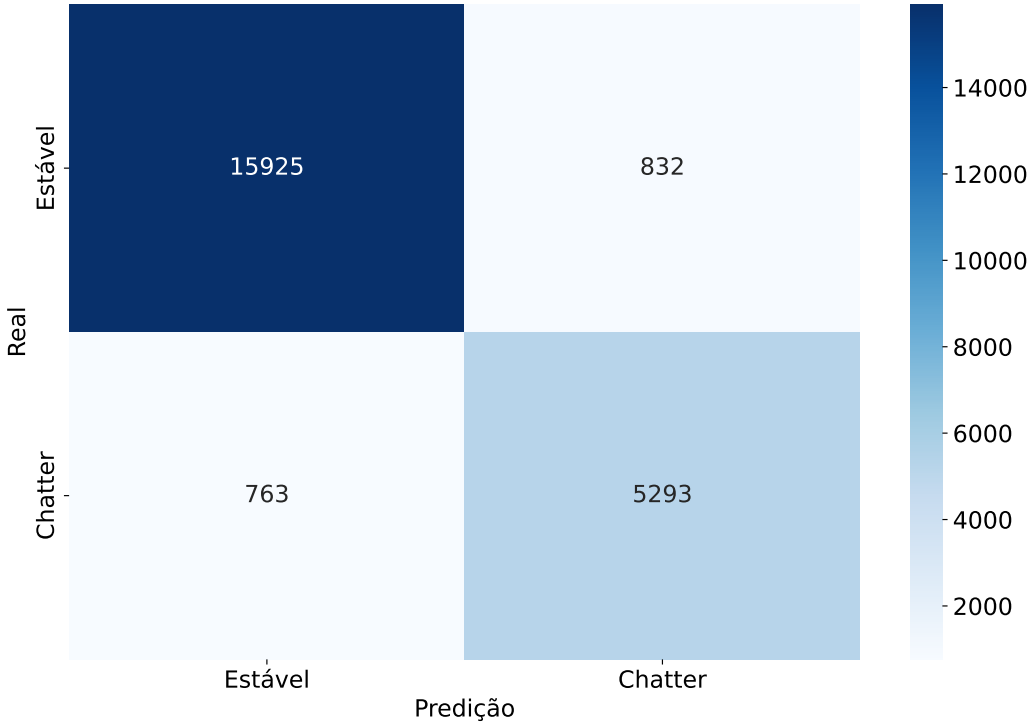
Tabela 9 – Métricas do modelo para o conjunto de teste

Rótulo	**Acurácia**	**Precisão**	***Recall***	***F-score***
Estável	93%	95%	95%	95%
Chatter	93%	86%	87%	87%

Fonte: Elaborado pelos autores.

Percebe-se que o resultado, principalmente quanto à precisão na identificação do *chatter*, é 3,4% menor em relação ao observado na Tabela 8. Isso é algo esperado, visto que a calibração e o cálculo do *threshold* são feitos tendo como base a amostra de validação, que é apenas uma representação da população toda, assim como o conjunto de treino também o é. Sendo assim, com diferentes conjuntos, presume-se que o desempenho do modelo possua uma certa diferença daquele visto nos conjuntos conhecidos.

Figura 16 – Matriz de confusão para o conjunto de teste



Fonte: Elaborado pelos autores

5 CONCLUSÃO

Neste trabalho, desenvolveu-se um modelo de *machine learning* baseado no algoritmo de *random-forest*, utilizando a linguagem de programação *Python*, para identificação do fenômeno negativo *chatter*, a partir de um conjunto de dados rotulados, caracterizando um modelo supervisionado classificatório.

Durante seu desenvolvimento, houve passos importantes como o tratamento prévio do conjunto de dados, a fim de eliminar os possíveis ruídos e amplificar a performance do modelo final. Além disso, devido à natureza temporal dos dados, dividiu-se previamente em conjunto de treino e teste e realizou-se o método de janelas deslizantes para obtenção das *features*, evitando um *overfitting* dos dados nessa etapa. Em seguida utilizou-se do método de RFE para obter as *features* mais relevantes para o modelo, evitando redundância e contribuindo para sua performance geral. Por fim, treinou-se o modelo e gerou a matriz de confusão juntamente com as métricas utilizadas para análise de performance, como acurácia, precisão, *recall* e *F-score*.

Analisando os resultados obtidos apenas com o treinamento do modelo, pode-se perceber que o modelo apresentou uma acurácia significativa de 93% e uma alta combinação de precisão e *recall* para os dados estáveis de 95% para ambos. Contudo, para identificação de *chatter* sua precisão estava um pouco abaixo, com 87%. Com o intuito de melhorar tanto a performance quanto principalmente a confiabilidade de previsão, realizou-se a calibração do modelo utilizando a técnica de Platt, em que obteve-se como saída um modelo treinado e calibrado, sendo confirmada pela métrica de Brier, que passou de 0,0605 para 0,0554. Com o modelo treinado e calibrado, novas métricas foram geradas juntamente com a matriz de confusão, resultando em um acréscimo na precisão de identificação de *chatter* para 89%, sem comprometer significativamente as métricas para identificação dos dados estáveis e acurácia, além de garantir maior confiabilidade dos resultados. Além disso, observou-se uma diminuição de 20% na classificação de falso positivos, que reflete positivamente em um caso de aplicação real em que deseja-se que o modelo não acuse peças estáveis como contendo *chatter*, descartando-as erroneamente. Por fim, para amostras OOB (*Out-Of-Bag*) no conjunto de treino, foram verificadas métricas finais de 86% de precisão, 3,4% abaixo do processo de validação, e a acurácia geral foi mantida em 93%.

Dessa forma, através dos resultados obtidos, conclui-se que o modelo de *random-forest* possui boa performance para a identificação de *chatter*, apresentando resultados satisfatórios dentro do contexto de *machine learning*. Como sugestão para estudos futuros, pode-se citar a integração do modelo obtido com uma linha de produção autônoma nos moldes da indústria 4.0, que alimenta o modelo com dados de sensores e retorna a identificação de peças sofreram *chatter*, promovendo uma aplicação prática do trabalho.

REFERÊNCIAS

ALAMMARI, Y. *et al.* Investigation of boring bar dynamics for chatter suppression. **Procedia Manufacturing**, Elsevier BV, v. 1, p. 768–778, 2015. Disponível em: https://doi.org/10.1016/j.promfg.2015.09.059.

HOLZMANN, H. A.; DALLAMUTA, J. **Impactos das Tecnologias na Engenharia Mecânica**. Atena Editora, 2019. Disponível em: https://doi.org/10.22533/at.ed.463190504.

KULJANIC, E.; SORTINO, M.; TOTIS, G. Multisensor approaches for chatter detection in milling. **Journal of Sound and Vibration**, Elsevier BV, v. 312, n. 4-5, p. 672–693, maio 2008. Disponível em: https://doi.org/10.1016/j.jsv.2007.11.006.

LUBBA, C. H. *et al.* **catch22: CAnonical Time-series CHaracteristics**. arXiv, 2019. Disponível em: https://arxiv.org/abs/1901.10200.

MALEKI, F. *et al.* Machine learning algorithm validation. **Neuroimaging Clinics of North America**, Elsevier BV, v. 30, n. 4, p. 433–445, nov. 2020. Disponível em: https://doi.org/10.1016/j.nic.2020.08.004.

MIETUS, J. E. The pNNx files: re-examining a widely used heart rate variability measure. **Heart**, BMJ, v. 88, n. 4, p. 378–380, out. 2002. Disponível em: https://doi.org/10.1136/heart.88.4.378.

NAQA, I. E.; MURPHY, M. J. What is machine learning? *In*: **Machine Learning in Radiation Oncology**. Springer International Publishing, 2015. p. 3–11. Disponível em: https://doi.org/10.1007/978-3-319-18305-3_1.

QUINTANA, G.; CIURANA, J. Chatter in machining processes: A review. **International Journal of Machine Tools and Manufacture**, Elsevier BV, v. 51, n. 5, p. 363–376, maio 2011. Disponível em: https://doi.org/10.1016/j.ijmachtools.2011.01.001.

SESTITO, G. S. *et al.* In-process chatter detection in micro-milling using acoustic emission via machine learning classifiers. **The International Journal of Advanced Manufacturing Technology**, Springer Science and Business Media LLC, v. 120, n. 11-12, p. 7293–7303, abr. 2022. Disponível em: https://doi.org/10.1007/s00170-022-09209-w.

VENTER, G. S. *et al.* Passive and active strategies using embedded piezoelectric layers to improve the stability limit in turning/boring operations. **The International Journal of Advanced Manufacturing Technology**, Springer Science and Business Media LLC, v. 89, n. 9-12, p. 2789–2801, out. 2016. Disponível em: https://doi.org/10.1007/s00170-016-9620-2.

WANG, X.; WIRTH, A.; WANG, L. Structure-based statistical features and multivariate time series clustering. *In*: **Seventh IEEE International Conference on Data Mining (ICDM 2007)**. IEEE, 2007. Disponível em: https://doi.org/10.1109/icdm.2007.103.

YESILLI, M. C.; KHASAWNEH, F. A.; OTTO, A. On transfer learning for chatter detection in turning using wavelet packet transform and empirical mode decomposition. arXiv, 2019. Disponível em: https://arxiv.org/abs/1905.01982.

YESILLI, M. C.; KHASAWNEH, F. A.; OTTO, A. Topological feature vectors for chatter detection in turning processes. arXiv, 2019. Disponível em: https://arxiv.org/abs/1905.08671.

YESILLI, M. C.; KHASAWNEH, F. A.; OTTO, A. **Turning Dataset for Chatter Diagnosis Using Machine Learning**. Mendeley, 2019. Disponível em: https://data.mendeley.com/datasets/hvm4wh3jzx/1.

ZHOU, Z.-H. **Machine Learning**. Springer Singapore, 2021. Disponível em: https://doi.org/10.1007/978-981-15-1967-3.