UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ANA CAROLINA BRANCO NEUMANN
MATHEUS RONELLE RODRIGUES

APLICAÇÃO DE TÉCNICAS DE MACHINE LEARNING NA ANÁLISE PREDITIVA DE FALHAS PARA OTIMIZAÇÃO DA MANUTENÇÃO INDUSTRIAL

PONTA GROSSA
2023

ANA CAROLINA BRANCO NEUMANN
MATHEUS RONELLE RODRIGUES

APLICAÇÃO DE TÉCNICAS DE MACHINE LEARNING NA ANÁLISE PREDITIVA DE FALHAS PARA OTIMIZAÇÃO DA MANUTENÇÃO INDUSTRIAL

Application of Machine Learning techniques in predictive analysis of failures for industrial maintenance optimization

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Mecânica da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientadora: Marcella Scoczynski Ribeiro Martins.
Coorientador: Roger Navarro Verastegui.

PONTA GROSSA
2023

4.0 Internacional

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

CONSTRUÇÃO DE UM PROGRAMA DE MANUTENÇÃO PREDITIVA POR MEIO DA CLASSIFICAÇÃO DE PARÂMETROS DE FALHA COM MACHINE LEARNING

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Mecânica da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 01 de dezembro de 2023

Marcella Scoczynski Ribeiro Martins
Doutorado
Universidade Tecnológica Federal do Paraná

Virgínia Helena Varotto Baroncini
Doutorado
Universidade Tecnológica Federal do Paraná

Oscar Regis Junior
Doutorado
Universidade Tecnológica Federal do Paraná

PONTA GROSSA

2023

AGRADECIMENTOS

Agradecemos primeiramente a Deus, por nos dar a oportunidade de encontrarmos um ao outro e de concluirmos esta etapa importante de nossas vidas. Sua força e orientação nos sustentaram em todos os momentos de dificuldade.

Agradecemos também aos nossos pais, Cristiana Gonçalves Rodrigues, Ubiratan de Almeida Oliveira, Márcia Helena Branco Neumann e Egon Geraldo Neumann, pelo amor, apoio e incentivo incondicional. Sem vocês, nada disso seria possível.

Agradecemos à Universidade Tecnológica Federal do Paraná, pela formação de qualidade que nos proporcinou. Agradecemos a todos os professores que fizeram parte de nossa formação, que nos ajudaram a crescer intelectualmente e a desenvolver habilidades necessárias para o mercado de trabalho.

A conclusão deste trabalho é uma realização pessoal e profissional muito importante para nós. É resultado de muito esforço, dedicação e apoio de muitas pessoas.

RESUMO

Com a difusão da Indústria 4.0, o modo de operação das indústrias tem sido impactado consideravelmente pela transformação digital. A utilização de tecnologias avançadas, como o Machine Learning, aparece como uma ferramenta poderosa para conduzir a uma maior eficiência e competitividade entre as empresas. Otimizando, assim, a programação de manutenção preditiva, permitindo antecipar falhas em máquinas e equipamentos através de dados históricos, evitando paradas não programadas e maximizando a disponibilidade dos ativos.

É nesse contexto, que a utilização de técnicas de Machine Learning se justifica, permitindo a análise de grande volume de dados de forma automatizada e inteligente, possibilitando a identificação de padrões e correlações de difícil detecção. O presente trabalho possui o objetivo principal de identificar incidência de parâmetros de modos de falha em um maquináro, auxiliado por inteligência artificial, no âmbito da Engenharia Mecânica. A base sintética de dados utilizada foi a "*Machine Predictive Maintenance Classification*", do site Kaggle, a qual simula os parâmetros de uma máquina real, em seu estado de operação e falha. Os modelos de Machine Learning utilizados para predição de falhas foram *Random Forest, Support Vector Machine e Multilayer Perceptron*, e suas acurácias foram comparadas para seleção do melhor modelo.

Palavras-chave: Aprendizado de máquina; Manutenção preditiva; Modelo preditivo; Análise de falha.

ABSTRACT

With the diffusion of Industry 4.0, the operational mode of industries has been significantly impacted by digital transformation. The use of advanced technologies, such as Machine Learning, emerges as a powerful tool to drive greater efficiency and competitiveness among companies. This optimizes the scheduling of predictive maintenance, allowing for the anticipation of machine and equipment failures through historical data, thus avoiding unplanned downtime and maximizing asset availability.

In this context, the utilization of Machine Learning techniques is justified, enabling the automated and intelligent analysis of large volumes of data, facilitating the identification of patterns and correlations that are difficult to detect.

The main objective of this work is to identify the incidence of failure mode parameters in machinery, aided by artificial intelligence, within the scope of Mechanical Engineering. The synthetic data used for this purpose was obtained from the "Machine Predictive Maintenance Classification" dataset on the Kaggle website, which simulates the parameters of a real machine in both its operational and failure states. The Machine Learning models employed for failure prediction were Random Forest, Support Vector Machine and Multilayer Perceptron, and their accuracies were compared to select thebest model.

Keywords: Machine Learning; Predictive Maintenance; Predictive Model; Failure Analysis.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação de um modelo de árvore de decisão 20
Figura 2 – Representação de um modelo de SVM..... 21
Figura 3 – Representação teórica de análise de vibração 23
Figura 4 – Código Python para geração do gráfico de distribuição de tempratura do ar 27
Figura 5 – Código do modelo Random Forest..... 37
Figura 6 – Código do modelo Support Vector Machine..... 38
Figura 7 – Código do modelo Multilayer Perceptron 39
Figura 8 – Matriz de Confusão do modelo Random Forest..... 40
Figura 9 – Matriz de Confusão do modelo Support Vector Machine..... 41
Figura 10 – Matriz de Confusão do modelo Multilayer Perceptron..... 42

LISTA DE GRÁFICOS

Gráfico 1 – Distribuição da temperatura do ar 31
Gráfico 2 – Distribuição da temperatura do processo 32
Gráfico 3 – Distribuição do torque..... 32
Gráfico 4 – Distribuição da velocidade rotacional 33
Gráfico 5 – Gráfico de correlação entre as variáveis..... 33
Gráfico 6 – Gráfico de correlação entre Temperatura do Ar e Temperatura de Processo 34
Gráfico 7 – Gráfico de correlação entre Torque e Velocidade Rotacional..... 34
Gráfico 8 – Distribuição do Desgaste de Ferramenta por Variável Target 35
Gráfico 9 – Distribuição da Temperatura do Ar por Variável Target 35
Gráfico 10 – Distribuição da Temperatura do Processo por Variável Target 36
Gráfico 11 – Distribuição da Velocidade Rotacional por Variável Target..... 36
Gráfico 12 – Distribuição da Torque por Variável Target 36

LISTA DE TABELAS

Tabela 1 – Listagem de TMAs de componentes..... 22
Tabela 2 – Ferramentas utilizadas 26

LISTA DE ABREVIATURAS E SIGLAS

AED	Análise Exploratória de Dados
BI	Business Inteligence
IoT	Internet of Things
KNN	K-nearest neighbors
MLP	Multilayer Perceptron
ML	Machine Learning
PCA	Análise de Componentes Principais
SVM	Support Vector Machine
TMA	Temperatura Máxima Admissível
t-SNE	t-Distributed Stochastic Neighbor Embedding

SUMÁRIO

1 INTRODUÇÃO 13
1.1 Contextualização 13
1.2 Justificativa 14
1.3 Objetivo Geral 14
1.4 Objetivos específicos 15
2 REVISÃO BIBLIOGRÁFICA 16
2.1 Análise Exploratória dos Dados 16
2.1.1 Técnicas de Análise Exploratória de Dados 16
2.2 Machine Learning 17
2.2.1 Paradigmas de Aprendizado 17
2.2.2 Treinamento e teste de modelos de Machine Learning 18
2.2.3 Métricas de classificação 18
2.2.4 Modelos de regressão 18
2.2.5 Modelos de predição 19
2.2.6 K-nearest neighbors (KNN) 19
2.2.7 Árvore de decisão 19
2.2.8 Anáilse de regressão 20
2.2.9 Classificadores de Naïve Bayes 20
2.2.10 Agrupamento 20
2.2.11 Redução de dimensionalidade 20
2.2.12 Redes neurais 20
2.2.13 Random Forest 21
2.2.14 Support Vector Machine 21
2.3 Manutenção preditiva 22
2.3.1 Análise de temperatura 22
2.3.2 Análise de vibração 23
2.4 Machine Learning aplicado à manutenção preditiva 24
3 METODOLOGIA 26
3.1 Dataset e Ferramentas 26
3.2 Procedimentos de análise e preparação dos dados 27

3.2.1 Análise exploratória 27
3.2.2 Preparação dos dados 29
3.3 Seleção dos algoritmos de Machine Learning 29
3.3.1 Random Forest 29
3.3.2 Support Vector Machine 30
3.3.3 Multilayer Perceptron 30
3.4 Avaliação do desempenho dos modelos 30
4 RESULTADOS 32
4.1 Análise Exploratória dos Dados (AED) 32
4.1.1 Análise da distribuição de temperatura do ar e do processo 32
4.1.2 Ánálise da distribuição de torque e velocidade rotacional 33
4.1.3 Análise de correlação entre as variáveis analisadas 34
4.1.4 Análise da distribuição dos dados por Variável Target 36
4.2 Construção e Treinamento dos Modelos de Machine Learning 38
4.2.1 Random Forest 38
4.2.2 Support Vector Machine (SVM) 38
4.2.3 Multilayer Perceptron (MLP) 39
4.3 Avaliação do desempenho dos modelos de Machine Learning 40
4.3.1 Random Forest 40
4.3.2 Support Vector Machine (SVM) 41
4.3.3 Multilayer Perceptron (MLP) 42
5 CONCLUSÃO 44
REFERÊNCIAS 46
APÊNDICE A 48

1 INTRODUÇÃO

1.1 Contextualização

A manutenção preditiva permite a detecção de problemas antes de ocorrerem falhas através do monitoramento de desempenho e condições de máquinas e equipamentos em tempo real. Isso significa evitar paradas imprevistas que causam perda de produção e gastos adicionais com manutenção corretiva, ajudando a prolongar a vida útil dos ativos. Segundo Kardec e Nascif (2009), esta foi a primeira quebra de paradigmas na Manutenção, enfatizando seu avanço com o avanço tecnológico. Assim, se tornando uma forte aliada das empresas que buscam se manter competitivas, produzindo mais com menos paradas, reduzindo custos de manutenção e aumentando a disponibilidade. Disponibilidade a qual, segundo a Associação Brasileira de Normas Técnicas (1994,p.8), é a “capacidade de um item estar em condições de executar certa função em um dado instante ou durante um intervalo de tempo determinado”.

Em adição, com o avanço da indústria 4.0 para novos ramos do mercado, incluindo a manutenção preditiva, é possível obter modelos de machine learning com dados históricos de máquinas, através de uma coleta de dados cada vez mais precisa e automatizada, permitindo, através de ferramentas mais sofisticadas, a detecção de problemas de forma ágil. Machine learning é a prática de aplicação de modelos algorítmicos de maneira iterativa em dados (PIERSON, 2019). Ao invés de coletar os dados em medições periódicas, é possível também, a implementação da monitoração contínua, coletando e processando o estado do equipamento em tempo real (KARDEC; NASCIF, 2009).

A construção de um modelo de machine learning para classificação de falhas em máquinas abrange algumas etapas, com a coleta de dados até a implementação do modelo em um sistema de gerenciamento. Primeiro, após a instalação de sensores em equipamentos da operação, se faz necessária a coleta e processamento desses dados em um banco de dados. O próximo passo é treinar um modelo de machine learning usando esses dados históricos coletados via sensores. Esse modelo de machine learning será capaz de identificar correlações e prever padrões que indicam a possibilidade de uma falha. Finalmente, o modelo machine learning é implementadoem um sistema de gerenciamento de manutenção.

Um questionamento que vem à tona são os desafios de investir em indústria 4.0, uma vez que envolve custos e desafios técnicos de implementação, como investimento em novas tecnologias - sensores, plataformas e banco de dados-, investimento em conhecimento técnico dos funcionários e habilidade de adaptação dos gestores. Já que, numa indústria que aplica os conceitos de indústria 4.0 as máquinas estão conectadas como em uma comunidade, utilizando as ferramentas de *Big Data, Cloud computing, IoT (Internet Of Things)* e inteligência artificial (MAURA, 2019).

1.2 Justificativa

O meio industrial segue enfrentando dificuldades na implementação de programas de manutenção eficientes, de forma que a utilização das manutenções corretivas e preventivas continua sendo largamente utilizada nas fábricas. No entanto, essas técnicas possuem várias limitações e acarretam em vários impactos negativos, tais como um elevado custo associado e uma baixa confiabilidade e disponibilidade dos ativos. A falta de uma abordagem proativa e eficiente para lidar a respeito dessa questão impactam diretamente nas paradas de produção e satisfação dos clientes.

Como um meio de revolucionar os processos industriais e otimizar a eficiência operacional, várias técnicas e ferramentas da Indústria 4.0 ganham destaques significativos no cenário atual. A aplicação dessas tecnologias no cenário da manutenção pode auxiliar de forma expressiva na forma que a gestão de ativos é realizada.

É nesse contexto que a utilização de técnicas de Machine Learning se justifica, permitindo a análise de grande volume de dados de forma automatizada e inteligente. A análise exploratória de dados (AED) desempenha um papel importante nesse processo, pois serve como uma etapa essencial para o entendimento e preparação dos dados para o treinamento do modelo. Dessa forma, possibilitando a identificação de padrões e correlações de difícil detecção, acarretandoem uma redução dos custos de manutenção e aumento de sua confiabilidade e disponibilidade.

1.3 Objetivo geral

O presente trabalho possui o objetivo principal de identificar e classificar falhas em equipamentos através da análise histórica de dados, auxiliado por inteligência artificial, no âmbito da Engenharia Mecânica.

1.4 Objetivos específicos

Para cumprir o objetivo geral deste estudo, será necessário desenvolver os seguintes passos:

i. Realizar a análise exploratória, preparar os dados e avaliar os modos de falha da máquina;
ii. Selecionar o(s) algoritmo(s) de Machine Learning mais adequado(s) para o desenvolvimento do plano de manutenção;
iii. Treinar o modelo selecionado a partir dos dados históricos do equipamento;
iv. Avaliar o desempenho do modelo na classificação de falhas mecânicas.

2 REVISÃO BIBLIOGRÁFICA

2.1 Análise Exploratória dos Dados

A análise exploratória dos dados envolve a exploração e compreensão dos dados do conjunto, por meio de gráficos e estatística. Trazendo assim, informações sobre padrões, tendências e insights que podem orientar a aplicação de algoritmos de modelagem apropriados e tomar decisões informadas.

Essa fase é iniciada após a preparação dos dados, envolvendo a visualização de dados de uma forma intuitiva e eficiente para maior conhecimento dos dados (Escovedo e Koshiyama, 2020). A análise exploratória de dados não tem a finalidade de confirmar conceitos pré-definido, mas sim de formular hipóteses para estudos futuros (Faceli, 2011). E, vale constatar que a análise exploratória não tem uma forma linear pré-definida, mas sim, que é construída a medida que novas percepções são obtidas do conjunto de dados.

i. Técnicas de Análise Exploratória de Dados

A extração de algumas medidas estatísticas descritivas básicas é uma das formas de se explorar o conjunto de dados e resumir suas principais características, como média, mediana, soma, mínimo, máximo e desvio padrão (Myatt, 2007). Além dessas medidas, tem-se a análise de frequência, localização e dispersão.

A visualização dos dados é outro fator importante na análise exploratória. Histogramas, polígonos de frequência, gráficos de dispersão e gráficos de caixa permitem a identificação de outliers, entendimento da distribuição dos dados, detecção de padrões e relações entre variáveis (Faceli et al., 2021).

Há, ainda, a análise de correlações entre variáveis, que permite a identificação de uma associação linear entre as variáveis, medindo a força e direção da associação (Harrison, 2020). Nesse meio, está a correlação de Pearson, variando de -1 a 1, onde na extremidade inferior -1 estão os casos de correlação negative forte, valores próximos a 0 indicam a inexistência de uma correlação e, valores próximos a 1, indicam uma correlação positive forte.

Por fim, a análise exploratória ainda é composta de inúmeras outras técnicas como PCA (Análise de Componentes Principais), análise de clusters e técnicas de redução de dimensionalidade. Possibilitando, assim, a identificação de relações complexas entre as variáveis e identificar padrões ocultos nos dados.

2.2 Machine Learning

Aprendizado de máquinas (machine learning) é quando se faz uso de uma máquina para coleta, aprendizado e análises, a partir de dados, para determinação de algo previamente estipulado, Filatro (2020). É uma subárea da inteligência artificial que possibilita o desenvolvimento de modelos computacionais capazes de aprender a partir de dados, fazer previsões e possibilitar a tomada de decisões sem a necessidade, propriamente dita, de programação explícita.

Surgiu com o intuito de melhorar o desempenho em várias tarefas práticas de soluções de problemas (Mitchell, 1997). Em comparação com os métodos estatísticos tradicionais, pode-se assumir que o aprendizado de máquinas tem demonstrado desempenho superior, levando em consideração o advento do *Big Data* e avanços em hardware (Beghi, 2015).

i. Paradigmas de Aprendizado

O aprendizado de máquinas é subdividido em dois paradigmas principais: aprendizado supervisionado e aprendizado não supervisionado (Bengio, 2016). No aprendizado supervisionado, o algoritmo é fornecido de dados que contêm a rotulagem (resposta) correta, fazendo com que o algoritmo seja treinado nos padrões aprendidos. Por outro lado, o aprendizado não supervisionado não fornece a rotulagem dos dados, fazendo com que o algoritmo busque por conta propria padrões e estruturas inerentes aos dados sem orientação específica (Piassa, 2019).

Há um terceiro modo de aprendizado, o aprendizado por reforço, o qual trabalha com a tomada de decisões em ambientes dinâmicos (Sutton e Barto, 2018). Essa modalidade trabalha com feedback em forma de recompensas ou penalidades. Por meio de tentativa e erro, o agente tende a buscar tomar ações que maximizem as recompensas e diminua as penalidades a longo prazo.

Já, o aprendizado profundo é um subconjunto de aprendizado, concentrado em redes neurais com inúmeras camadas, permitindo representações complexas de padrões hierárquicos para aprendizados (Goodfellow et al., 2016). Utilizado com grande sucesso e amplamente, nos dias atuais, em visão computacional, processamento de linguagem natural e reconhecimento de fala.

ii. Treinamento e teste de modelos de Machine Learning

Os processos de treinamento e teste são essenciais para a construção de um modelo de machine learning. O processo de treinamento se baseia na alimentação do modelo com dados de treinamento, que contém exemplos rotulados. O modelo aprende com esses dados a relação entre entradas e saídas, ajustando os parâmetros para minimização das saídas desejadas para as saídas que de fato estão acontecendo, através de um algorítmo de otimização. O treinamento de modelos de machine learning é um processo fundamental para a construção de sistemas inteligentes. Segundo De Almeida e De Souza (2021).

Após o treinamento, o modelo é submetido a testes em conjuntos de dados que nunca teve contato. De Almeida e De Souza (2021) destacam que o processo de treinamento é seguido pelo teste do modelo em um conjunto de dados de teste separado. O desempenho do modelo é avaliado conforme métricas de classificação. As métricas fornecem dados para identificação se o modelo está superajustado ao conjunto de teste apenas, ou se o desempenho do modelo é satisfatório.

iii. Métricas de classificação

A fim de avaliar a qualidade e desempenho de um modelo de machine learning, as métricas de classificação atuam avaliando a capacidade de um modelo em prever corretamente as classes de um conjunto de dados. As métricas mais comumente utilizadas são a acurácia, precisão, *recall, F1-score* e matriz de confusão. Vale citar que cada métrica fornece uma perspectiva sobre o desempenho do modelo, e segundo Powers (2011) as métricas comumente utilizadas podem ser tendenciosas se usadas sem uma compreensão clara de sua função e limitações.

iv. Modelos de regressão

De acordo com Shen e Zhang, em seu artigo *"The research of regression model in machine learning field"* (2018), a definição de modelos de regressão é uma técnica de análise estatística que permite estudar a relação entre uma variável dependente (variável prevista) e uma ou mais variáveis independentes (preditores). Em outras palavras, são algorítmos usados para entender a relação entre uma variável de saída com uma ou mais variáveis de entrada. Alguns dos tipos de regressão mais conhecidos são a regressão linear (simples e múltipla), regressão logística e regressão polinomial. Para escolha de um modelo de regressão, alguns

fatores são levados em consideração, como o tipo de variável dependente, relação linear ou não linear, número de variáveis independentes, escalabilidade e objetivo do modelo.

Sua usabilidade, segundo Hair Jr. et al. (2017), é ampla em diversos campos, como marketing, finanças, economia, entre outras, para prever comportamentos e tendências futuras.

v. Modelos de predição

A definição de um modelo preditivo, de acordo com o artigo *"Two Cultures of Statistical Modeling"* (Breiman, 2001), é uma técnica que utiliza algoritmos para analisar dados e identificar padrões, com o objetivo de fazer previsões precisas sobre novos dados. Os modelos de predição são treinados em conjuntos de dados que possuem entradas e saídas rotuladas, ajustando seus parâmetros durante o treinamento para reduzir a diferença entre o resultado real e observados no conjunto de treino. Alguns dos tipos de modelos preditivos mais comuns são: regressão, *Deep Learning*, séries temporais e classificação. Geralmente, os modelos preditivos aprendem através de supervisionamento, onde entradas e saídas são conhecidas. Mas, existem técnicas como o *clustering* que é construído usando aprendizado não supervisionado, onde é identificados padrões sem dados de saída rotulados.

vi. K-nearest neighbors (KNN)

O algoritmo *K-nearest neighbors (KNN)*, ou em português, vizinhos mais próximos, é um algoritmo usado para classificação e regressão (Mitchell, 1997). Ele olha para os K pontos mais próximos verificando a classe que possui a maior frequência, essa classe então, é atribuída ao ponto em questão ao qual se está observando. Esse algoritmo não possui parâmetros fixos, usando a distância para determinar a similaridade entre pontos fornecidos.

vii. Árvore de decisão

É utilizada para classificação e regressão, utilizando características para construção de uma estrutura de decisões hierárquica (Mitchell, 1997). Tendo a árvore como uma simbologia, cada um de seus nós internos representa uma característica, e cada um de seus nós folha representa um valor previsto. Esse método pode ser utilizado tanto em dados categóricos como numéricos.

viii. Análise de regressão

É uma técnica utilizada para prever valores contínuos de saída com base em características dos valores de entrada fornecidos (Filatro, 2020). Uma de suas técnicas mais conhecidas é a regressão linear, a qual graficamente ajusta em uma relação linear as características de entrada e da variável alvo. Com base na força e direção do relacionamento, é possível obter insights para realização de previsões.

ix. Classificadores *Naïve Bayes*

Utilizam o teorema de Bayes para, dado as características observadas, determinar a probabilidade de uma classe (Mitchell, 1997). Esse método é particularmente eficaz quando analisados dados de alta dimensionalidade e classificações de texto.

x. Agrupamento

Busca a classificação em clusters de pontos de dados com base em similaridades (Bengio, 2016). Tem como objetivo principal garantir a similaridade entre os pontos do mesmo cluster em relação a outros *clusters*. Alguns algoritmos comumente utilizados nesse intuito são o k-means e o agrupamento hierárquico.

xi. Redução de dimensionalidade

A fim de manter as informações essenciais porém reduzir o número de características ou variáveis, são implementadas as técnicas de redução de dimensionalidade (Bengio, 2016). Duas técnicas de redução de dimensionalidade amplamente utilizadas são a Análise de Componentes Principais (PCA) e *t-Distributed Stochastic Neighbor Embedding (t-SNE)*, as quais visam extrair representações significativas e ajudar na visualização de dados de alta dimensionalidade.

xii. *Multilayer Perceptron* (Redes neurais)

As redes neurais são um modelo computacional inspirado no funcionamento e estruturação do cérebro humano (Goodfellow et al., 2016). São consistidas em nós, que representam os neurônios, interconectados, formados em camadas: entrada, ocultas e de saída. Cadan ó funciona com a recepção de entradas, aplicando uma função de ativação e fornecendo sua saída para a próxima camada. Redes neurais profundas possuem várias camadas ocultas, podendo aprender representações hierárquica de dados.

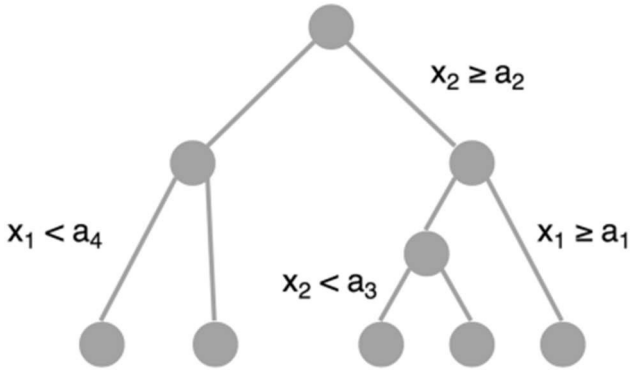
xiii. Random Forest

Em português, Floresta Aleatória, é um método de aprendizado de máquina que realiza tarefas de classificação ou regressão. O objetivo do *Random Forest* é criar várias árvores de decisão usando um subconjunto aleatório de atributos selecionados a partir do conjunto original, que contém todos os atributos (Neto, 2014). As árvores de decisão são compostas por três principais tipos de nós:

- Nó raiz: divisão inicial dos dados baseada em uma característica principal. Determina, com base em uma condição, qual ramo seguir;
- Nós da divisão: nós internos que representam condições que determinam subconjuntos menores de dados. Igualmente ao nó raiz, há prosseguimento para o próximo ramo com base nos resultados das condições;
- Folhas (nós terminais): representam o resultado final do modelo.

Abaixo, pode-se encontrar a representação de uma árvore de decisão para classificação.

Figura 1 – Representação de um modelo de árvore de decisão.



Fonte: Schonlau (2020).

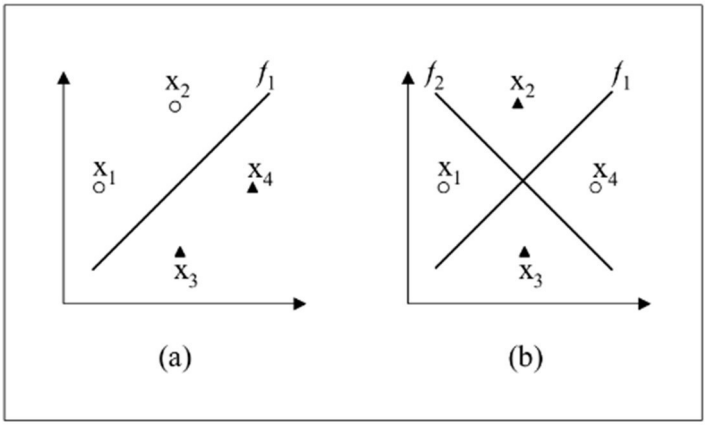
xiv. Support Vector Machine

O SVM é um classificador binário que busca encontrar uma fronteira ótima entre duas classes. Utiliza um hiperplano para separar os dados em classes pré-definidas. Os SVMs maximizam a margem de separação usando vetores de suporte, que são amostras mais desafiadoras de classificar. A fronteira de decisão ótima é determinada minimizando os erros de classificação durante o treinamento. O processo de aprendizagem do SVM envolve a seleção de vários hiperplanos e a escolha do ótimo, maximizando a margem de separação. Essas informações são relevantes para entender o funcionamento e a aplicação do SVM em tarefas de classificação

(Sheykhmousa, 2020).

Abaixo, pode-se encontrar um exemplo da forma em que a classificação das classes é realizada.

Figura 2 – Representação de um modelo de SVM.



Fonte: Lorena (2007).

2.3 Manuteção preditiva

A principal premissa da manutenção preditiva é o monitoramento regular da condição mecânica real, de sua eficiência operacional, além de outros indicadores do estado operacional de um conjunto de máquinas e equipamentos. (Mobley, 2002).

De acordo com Efthymiou (2012), a manutenção de sistemas de fabricação está se tornando cada vez mais importante, uma vez que, em muitas plantas industriais, os custos de manutenção frequentemente ultrapassam 30% dos custos operacionais e, no contexto do ciclo de vida dos sistemas de fabricação, a manutenção e o suporte representam até 60 a 75% dos custos totais do ciclo de vida. Nesse cenário, a utilização da manutenção preditiva se mostra uma solução viável para redução de custos e sua utilização no contexto da Indústria 4.0 se justifica, visto que apenas recentemente as tecnologias se tornaram aparentemente capazes e baratas o suficiente para tornar a manutenção preditiva amplamente acessível (Colemen, 2017, p. 7).

i. Análise de temperatura

A temperatura desempenha um papel essencial na manutenção preditiva, tanto em componentes elétricos quanto mecânicos. O monitoramento constante deste parâmetro auxilia a detectar de forma precoce quaisquer alterações no modo de operação de máquinas e equipamentos, de forma a identificar um possível modo de

falha. Isso é fundamental para evitar falhas e minimizar perdas de produtividade (Fluke, 2009).

Segundo Caramalho (2012), um dos parâmetros mais relevantes para uma correta implementação de um programa de manutenção preditiva em equipamentos e componentes elétricos é a temperatura máxima admissível (TMA). Essa variável expressa a temperatura máxima na qual um equipamento deve operar, e a mesma não deve ser ultrapassada. O valor da TMA pode ser definida com base em especificações técnicas dos componentes ou segundo orientações do fabricante. No quadro abaixo, pode-se consultar informações da TMA segundo especificações técnicas de seus componentes.

Tabela 1 – Listagem de TMAs de componentes.

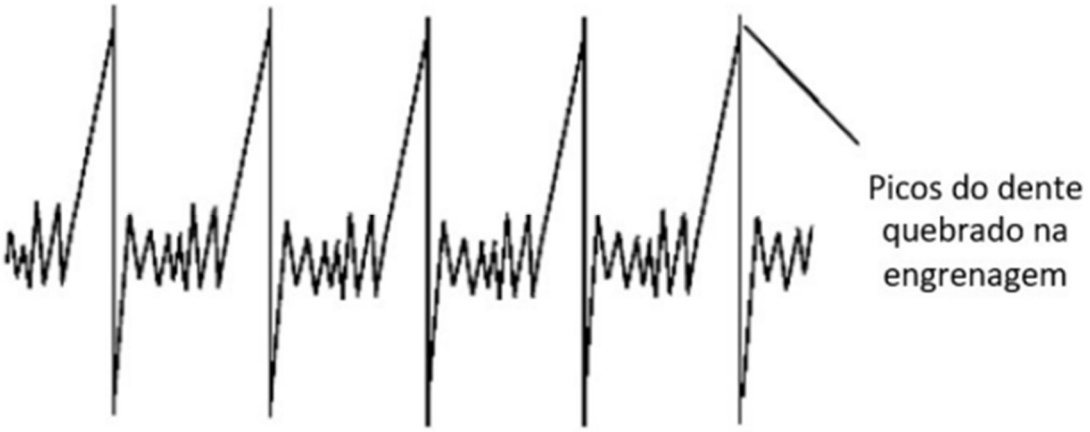
Componente	**TMA (°C)**
Cabo condutor com isolamento em PVC (cloreto de polivinil)	70
Cabo condutor com isolamento em EPR (borracha etileno propileno)	90
Cabo condutor com isolamento em XLPE (polietileno reticulado)	90
Ligações com parafusos	70
Ligações e barramentos de baixa tensão (BT)	90
Ligações recobertas de prata ou níquel	90

Fonte: Caramalho (2012).

ii. Análise de Vibração

A análise de vibração é um método utilizado para examinar o desempenho de componentes móveis de um dispositivo. Essa técnica é de extrema importância, especialmente no contexto industrial, pois permite identificar possíveis falhas nos equipamentos antes que essas falhas causem a paralisação total do dispositivo, impactando negativamente as operações em andamento (Ribeiro, 2017).

Figura 3 – Representação teórica de análise de vibração.



Fonte: Siemann (2021).

No exemplo da imagem anterior, os picos de vibração são indicativos de potenciais modos de falha no componente analisado, evidenciando a necessidade de intervenção com uma parada programada para a manutenção do equipamento (Siemann, 2021).

2.4 Fundamentos de Machine Learning aplicados à manutenção preditiva

Abordagens de Machine Learning têm a capacidade de lidar com dados multidimensionais e multivariados, e extrair relacionamentos ocultos dentro de dados em ambientes complexos e dinâmicos, como ambientes industriais (Wuest, 2016). Dessa forma, a utilização de modelos de Machine Learning oferece uma abordagem interessante no que tange à identificação antecipada de falhas em componentes e equipamentos.

Essa tecnologia vêm ganhando cada vez mais espaço no âmbito da manutenção no meio automobilístico. Um estudo realizado por pesquisadores (Husni, 2016) apresentou o desenvolvimento de um sistema inovador para automóveis chamado *Smart Car*. Esse sistema utiliza monitoramento contínuo e geração de relatórios sobre a condição do veículo, permitindo prever manutenções de rotina e fornecer orientações para aprimorar a forma de dirigir. Os parâmetros monitorados pelo *Smart Car* são armazenados de forma segura em um servidor na nuvem. Essa abordagem tem como objetivo principal obter informações relevantes para a realização de manutenção preditiva e monitoramento constante do veículo.

De acordo com (Gama, 2017), o projeto intitulado "O uso de Machine Learning na implementação de manutenção preditiva em usinas termelétricas" aborda um estudo de caso baseado na PETROBRAS, abordando o monitoramento

da manutenção em usinas termelétricas. É elaborada uma proposta de solução para manutenção preditiva em nuvem, com base em dados históricos de falha, a transformação digital nas empresas e a crescente utilização de ferramentas de *Business Intelligence (BI)* nesse contexto. As principais ferramentas de BI, em termos de uso, são apresentadas, bem como as tecnologias emergentes que acompanham essas ferramentas como *big data* e machine learning.

Em um terceiro projeto, pesquisadores conduziram um estudo (Çınar, 2020) com o objetivo de prever a condição de um equipamento usando dados de vibração. A pesquisa propôs uma abordagem que envolveu a conversão dos dados de medição em um sistema de vibração simulado de um motor, a fim de treinar e testar um modelo de Rede Neural Artificial. O objetivo final era prever a condição futura do equipamento e identificar possíveis falhas. No estudo, o desempenho do algoritmo de Redes Neurais Artificiais foi comparado com outros modelos, como Árvore de Regressão e Random Forest. O modelo baseado em Redes Neurais Artificiais apresentou os melhores resultados em termos de eficiência e precisão, especialmente em médio e longo prazo.

Já, no trabalho de (Macêdo, 2020), de tema "Manutenção Preditiva no Contexto da Indústria 4.0: um Modelo Preditivo em uma Fábrica do Ramo Metalúrgico", tem-se a aplicação da estratégia da manutenção preditiva, com conceitos da Indústria 4.0 em uma empresa metalúrgica. O modelo preditivo desse trabalho faz a utilização de um filtro discreto de Bayes, devido a seu desempenho com dados flutuantes e ruidosos, integrando dados de diferentes fontes e trabalhando com a incerteza inerente das medições de sensores. Esse modelo fornece uma estratégia prescritiva de manutenção, que permite a previsão da degradação das máquinas e alternativas de solução para a tomada de decisão. A análise prescritiva é um conceito da Manutenção no meio da Indústria 4.0, tema que não apenas prevê eventos futuros, mas também oferece alternativas para resolução. É ideal para quando se lida com falhas aleatórias em equipamentos. Esse trabalho não abordará conceitos de manutenção prescritiva, devido a isso o tema não será apresentado com profundidade.

3 METODOLOGIA

Para atingir os objetivos propostos, é necessário utilizar ferramentas de análise de dados e executar uma pesquisa que é dividida em etapas bem definidas, as quais são descritas mais detalhadamente abaixo.

3.1 Dataset e Ferramentas

A base sintética de dados atualmente utilizada é a *"Machine Predictive Maintenance Classification"* do site Kaggle, a qual simula os parâmetros de uma máquina real, em seu estado de operação e falha. A escolha da base de dados sintética se dá pela dificuldade de identificação de indústrias que extraem e armazenam seus dados de operação de máquinas (SEEBO, 2018), somado ao fato do sigilo industrial. Contudo, a mesma modelagem pode ser aplicada para conjuntos de dados oriundos de máquinas reais, com suas devidas adaptações.

Em relação às ferramentas, está sendo utilizado o PostgreSQL como SGBD (Sistema Gerenciador de Banco de Dados), o Visual Studio Code como IDE (Ambiente de Desenvolvimento Integrado) e a Linguagem de Programação Python.

A seguir, apresentam-se os dados contidos na base de dados:

- UDI: Identificadores exclusivos para cada registro;
- *Type:* Consiste de uma letra L, M ou H para baixa (50% de todos os produtos), média (30%) e alta (20%) como variantes de qualidade do produto;
- *Air temperature [K]:* Temperatura do ar em Kelvin;
- *Process temperature [K]:* Temperatura do processo em Kelvin;
- *Rotational speed [rpm]:* Velocidade de rotação do eixo do motor, calculado a partir de uma potência de 2860W;
- Torque [Nm]: Torque gerado pelo eixo do motor;
- *Tool wear [min]:* Tempo de desgaste da ferramenta, calculada a partir das variantes de qualidade H, M e L, as quais adicionam 5, 3 e 2 minutos, respectivamente, de desgaste à ferramenta utilizada no processo;
- *Target:* Classificação do registro em “Falha” ou “Não falha”;
- *Failure Type:* Classificação do tipo de falha ocorrida.

A seguir, podem-se encontrar maiores detalhes das ferramentas que estão sendo utilizadas para a análise dos dados e construção do modelo de classificação de falhas:

Tabela 2 – Ferramentas utilizadas.

Nome	Versão	Descrição do uso
PostgreeSQL	15	Armazenamento da base de dados
Visual Studio Code	1.79	Ambiente para desenvolvimento dos códigos
Linguagem Python	3.11.4	Linguagem de programação para análise e desenvolvimento do modelo

Fonte: Autoria própria.

3.2 Procedimentos de análise e preparação dos dados

i. Análise exploratória

Na etapa de análise exploratória, para realizar a investigação dos dados do conjunto, são utilizadas um conjunto de bibliotecas em Python para geração de gráficos e medidas estatísticas.

Ao iniciar, os dados são importados como um data frame em Pandas, transformando os dados em uma tabela para investigação. A biblioteca Pandas fornece alguns códigos para descrição e informações gerais sobre a base de dados, como descrição das colunas, tipo de dados de cada coluna, quantidade de números únicos por coluna, estrutura geral de linhas e colunas da base, número de valores ausentes por coluna e algumas estatísticas descritivas básicas, como média, contagem, máximo e mínimo.

Em seguida, as bibliotecas Seaborn e Matplotlib são utilizadas para geração de gráficos para melhor visualização dos dados. Com a biblioteca Seaborn, tem-se a geração do gráfico em si, e com a biblioteca Matplotlib, as configurações de plotagem do gráfico, como tema e títulos, área de plotagem do gráfico. Nessa etapa, são compreendidas algumas informações de distribuição e correlação dos dados, tais como:

- Análise de distribuição dos valores das colunas numéricas com histograma – entendimento da incidência dos dados, como faixa de distribuição da temperatura do processo conforme abaixo descrito na Figura 4;

Figura 4 - Código Python para geração do gráfico da distribuição de temperatura do ar.

Fonte: Autoria Própria.

- Análise da incidência de falhas através da biblioteca em Pandas – entendimento da quantidade de falhas por categoria (falha ou não falha) e tipo de falha;
- Análise dos valores médios das colunas numéricas por tipo de falha via biblioteca em Pandas – entendimento da variação dos parâmetros do processo para cada tipo de falha ocorrida;
- Distribuição do tipo de falha em gráfico de colunas com Seaborn e Matplotlib – gráfico comparativo da quantidade dos valores de tipo de falha;
- Análise de correlação entre variáveis do data frame com matriz de correlação em Seaborn e Matplotlib – entendimento de correlação entre os parâmetros do processo, como correlação entre velocidade rotacional e torque;
- Análise de outliers por meio da geração de gráfico bloxpot com Seaborn e Matplotlib para todas as colunas numéricas do data frame – identificação se há valores dissonantes com a média geral por parâmetro, como por exemplo se há algum valor muito elevado de velocidade rotacional;
- Análise de tendência entre as variáveis do data frame através de Seaborn e Matplotlib – entendimento se por exemplo, com o aumento da temperatura do ar há aumento na temperatura do processo;
- Análise dos valores médios das colunas numéricas por falha e não falha *(target)* via Pandas – entendimento das características gerais dos parâmetros em falha e não falha.

A partir dos resultados das análises obtidos, verifica-se as necessidades da etapa de preparação de dados para aplicação dos modelos de machine learning.

ii. Preparação dos dados

A etapa de preparação dos dados é facultativa, dependendo dos resultados da análise exploratória dos dados. Caso seja verificado um dos casos abaixo descritos, medidas de preparação para treino do modelo devem ser tomadas:

- Desbalanceamento entre falha e não falha *(target)* ou, entre tipos de falha: caso seja verificada diferença significativa entre os valores de falha e não falha, a aplicação de um método de augmentation (aumento de quantidade artificialmente) de dados deverá ser implementado para balanceamento de valores entre as classes;
- Remoção de colunas não utilitárias na aplicação do modelo: remoção de colunas classificatórias, como o UDI (chave-primária) o qual fornece um valor unitário para cada registro da tabela, id do produto e tipo do produto. Essas colunas não trazem implicações técnicas sobre quais parâmetros podem ocasionar falha.
- Segmentação da base de dados em conjunto de treino para modelo e conjunto de teste.
- Normalização de outliers, identificando e tratando valores atípicos de maneira apropriada caso seja necessário.

Após a etapa de preparação de dados, poderão ser iniciados os testes com os modelos escolhidos e, então, a validação da acurácia do modelo.

3.3 Seleção dos algoritmos de Machine Learning

i. Random Forest

O modelo de machine learning *Random Forest* é utilizado através da biblioteca scikit-learn (sklearn), em Python, no Visual Studio Code. A classe RandomForestClassifier, da biblioteca, é utilizada para determinação do classificador Random Forest e, com ela, é determinado o parâmetro de quantas árvores de decisão serão utilizadas no modelo através do n_estimators.

O modelo é treinado com a base de teste definida, anteriormente, através da função ‘fit()’, com os dados do treinamento e rótulos correspondentes. As predições são realizadas através da função ‘predict()’ no conjunto para obter as previsões do modelo.

ii. Support Vector Machine

Para construção do modelo *Support Vector Machine (SVM)* é utilizada a biblioteca scikit-learn (sklearn), mais especificamente a classe *SVC (Support Vector Classifier)*, em Python, no Visual Studio Code.

Os testes e predições do modelo utilizam as mesmas funções que o item anterior de *Random Forest*, a função ‘fit()’ e a função ‘predict()’ nos conjuntos de dados apropriados.

iii. Multilayer Perceptron

Para a construção do modelo *Multilayer Perceptron (MLP)*, é utilizada a biblioteca scikit-learn (sklearn), especificamente o uso da classe ‘MLPClassifier’ em ambiente Python, por meio do Visual Studio Code, que é apropriada para tarefas de classificação.

Os procedimentos de teste e previsão com o modelo MLP seguem uma abordagem semelhante àquelas empregadas anteriormente com os morelos anteriores. São utilizadas as funções 'fit()' para treinar o modelo com os dados de treinamento e 'predict()' para realizar previsões com base nos conjuntos de dados pertinentes.

3.4 Avaliação do desempenho dos modelos

Para verificar a acurácia dos modelos de Machine Learning, é realizada uma técnica de segmentação dos dados, dividindo-os em dois conjuntos: dados de treinamento e dados de teste.

No conjunto de treinamento, o modelo é alimentado com os dados e ajustado para aprender os padrões de falha e não falha presentes nos dados. Após o treinamento, é necessário avaliar o desempenho do modelo com o conjunto de teste, que contém dados não vistos pelo modelo durante o treinamento. Esse procedimento é realizado fornecendo os dados de teste ao modelo e comparando as previsões resultantes com os rótulos já conhecidos dos dados de teste. A acurácia é calculada como a proporção de previsões corretas em relação ao total de previsões realizadas pelo modelo. Essa métrica fornece uma medida geral da capacidade do modelo em classificar corretamente as amostras.

Para obtenção da precisão dos modelos, é utilizada a função 'accuracy_score()'. Essa função compara as previsões do modelo com os rótulos verdadeiros dos dados de teste e calcula a proporção de previsões corretas.

Após os cálculos anteriores, é realizada uma comparação entre os modelos para selecionar aquele que possui maior acurácia, ou seja, o modelo que possui maior eficiência na classificação de falhas.

4 EXPERIMENTOS E RESULTADOS

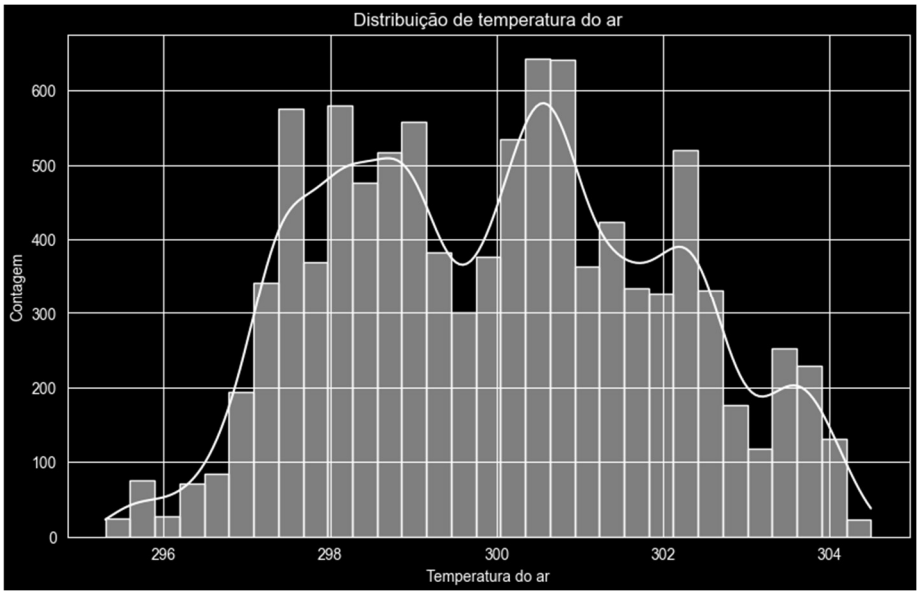
Na presente seção, são apresentados e discutidos os resultados obtidos no desenvolvimento no modelo de Machine Learning para classificação de falhas em maquinários.

4.1 Análise exploratória dos dados (AED)

i. Análise da distribuição de temperatura do ar e do processo

O Gráfico 1 motra a distribuição dos registros da temperatura do ar durante a operação da máquina.

Gráfico 1 - Distribuição da temperatura do ar

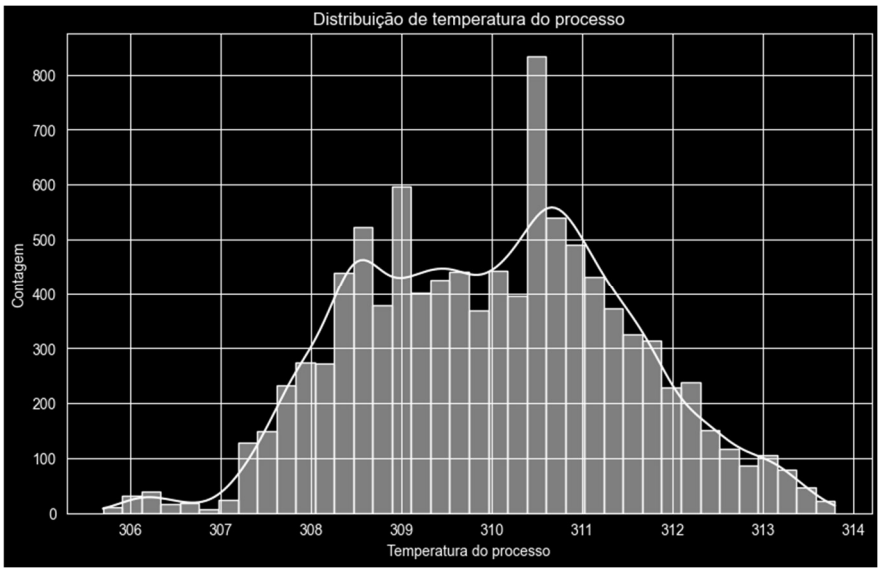


Fonte: autoria própria.

Nota-se uma distribuição bimodal dos dados, caracterizado, principalmente, por dois picos proeminentes, no qual cada um representa a faixa de valores da temperatura mais frequentes.

Em relação à temperatura do processo, a qual espera-se ter correlação com a temperatura do ar, nota-se um comportamento semelhante caracterizado por dois picos com a faixa de temperatura mais frequente, no entanto, menos explíticos em relação à amostragem de dados anterior, como pode ser consultado no Gráfico 2, logo abaixo.

Gráfico 2 - Distribuição da temperatura do processo

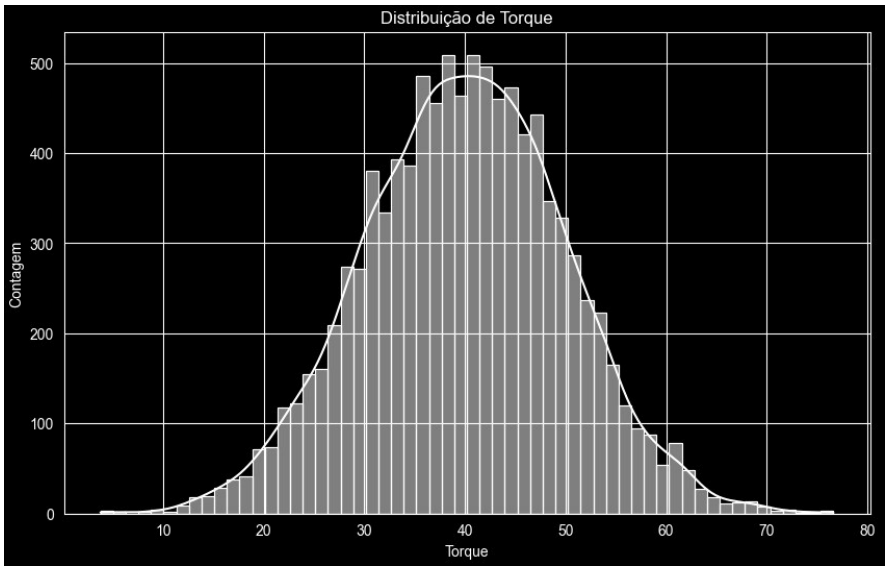


Fonte: autoria própria.

ii. Análise da distribuição de torque e velocidade rotacional

O Gráfico 3 mostra a distribuição dos registros do torque durante a operação da máquina.

Gráfico 3 - Distribuição do torque



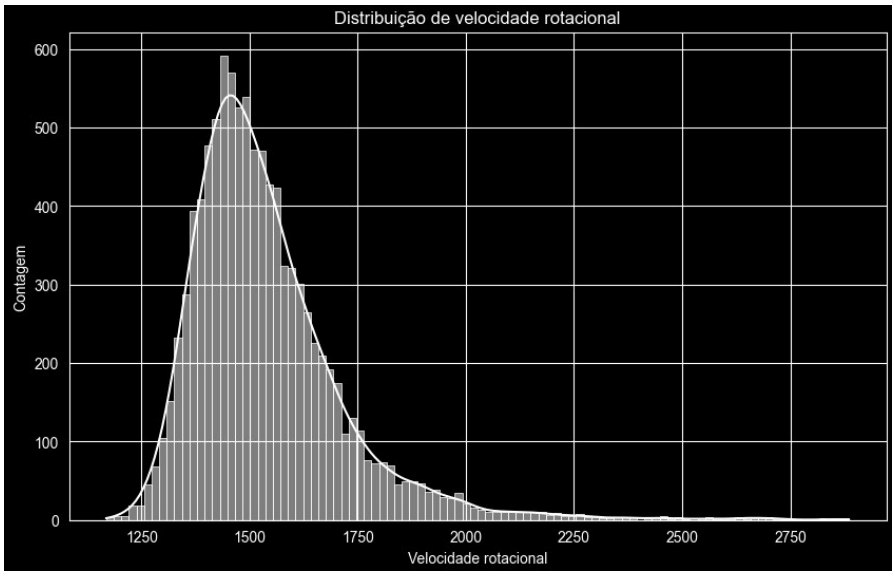
Fonte: autoria própria.

Nota-se uma distribuição normal dos dados, caracterizada principalmente por sua aparência no formato de “sino” de forma simétrica, devido à coincidencia da média, moda e mediana do conjunto de dados.

Devido a correlação existente entre o torque e a velocidade rotacional, visto que o torque aplicado está diretamente ligado à aceleração angular, e a velocidade rotacional é diretamente ligada à essa aceleração, espera-se um comportamento

semelhante na distribuição dos dados da velocidade rotacional, a qual se confirma, de acordo com o gráfico 4:

Gráfico 4 - Distribuição da velocidade rotacional

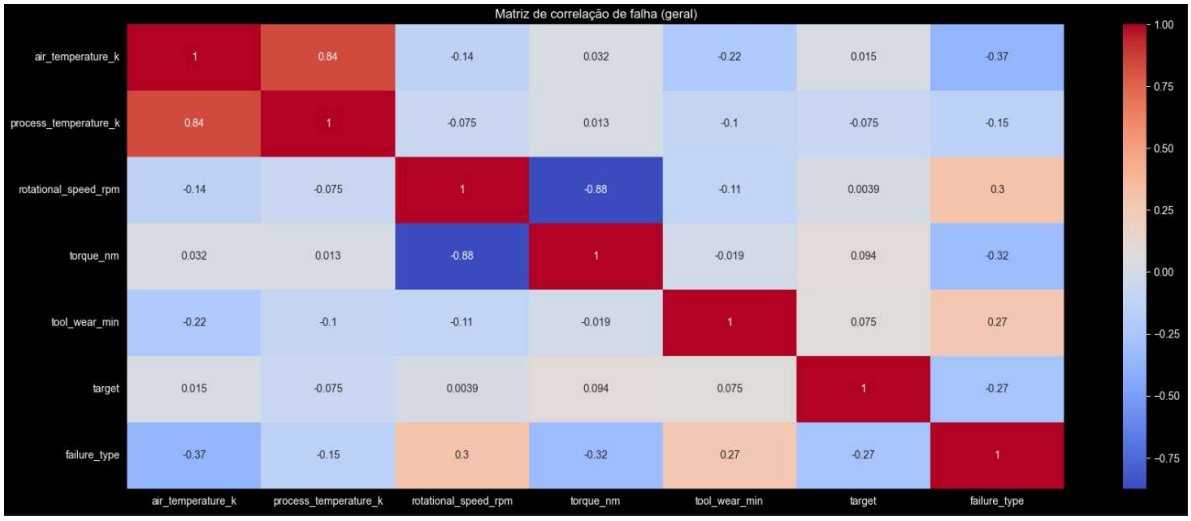


Fonte: autoria própria.

iii. Análise de correlação entre as variáveis analisadas

Através da análise do gráfico de correlação entre as variáveis (Gráfico 5), fica evidente que as correlações previamente esperadas foram confirmadas. Isso inclui a forte correlação positiva entre a temperatura do ar e a temperatura de processo (0.84), bem como a correlação negativa entre o torque e a velocidade rotacional (-0.88).

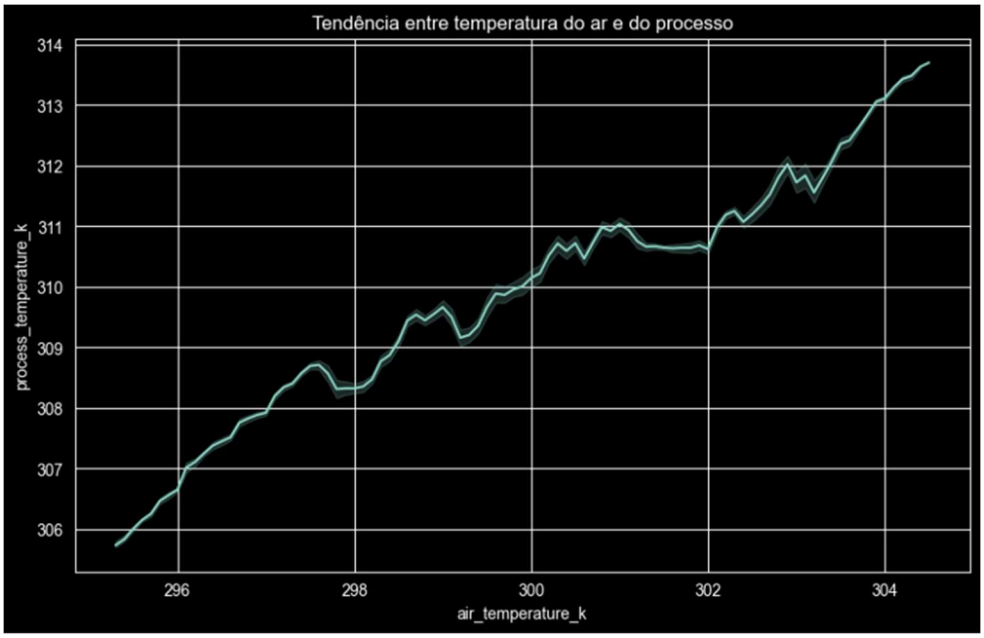
Gráfico 5 - Gráfico de correlação entre as variáveis



Fonte: autoria própria.

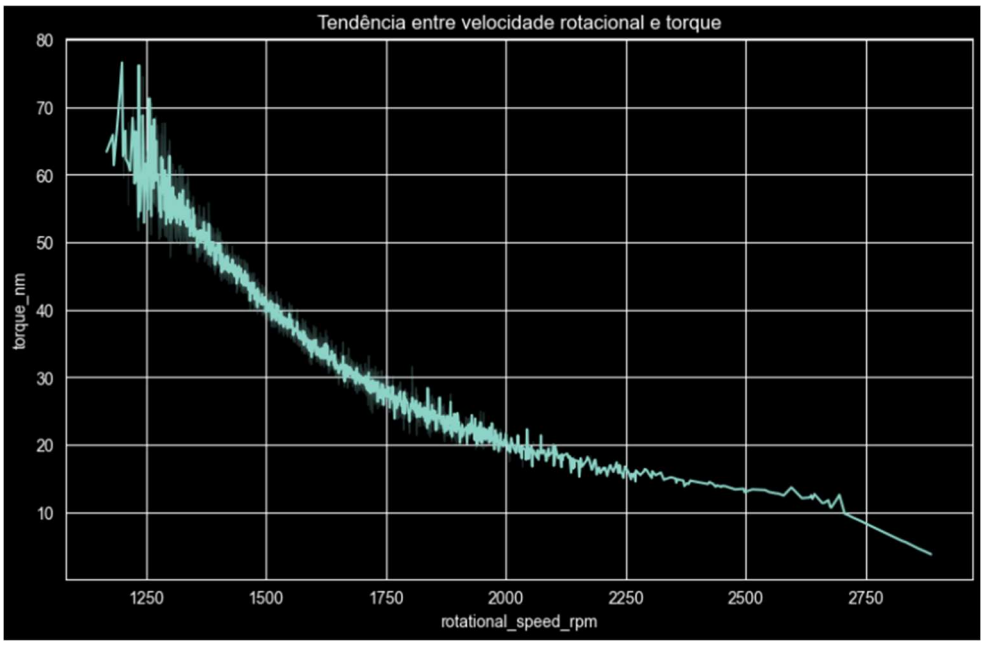
Ao analisar mais detalhadamente as variáveis mencionadas, através dos gráficos 6 e 7, fica evidente a correlação entre elas.

Gráfico 6 - Gráfico de correlação entre Temperatura do Ar e Temperatura de Processo



Fonte: autoria própria.

Gráfico 7 - Gráfico de correlação entre Torque e Velocidade Rotacional



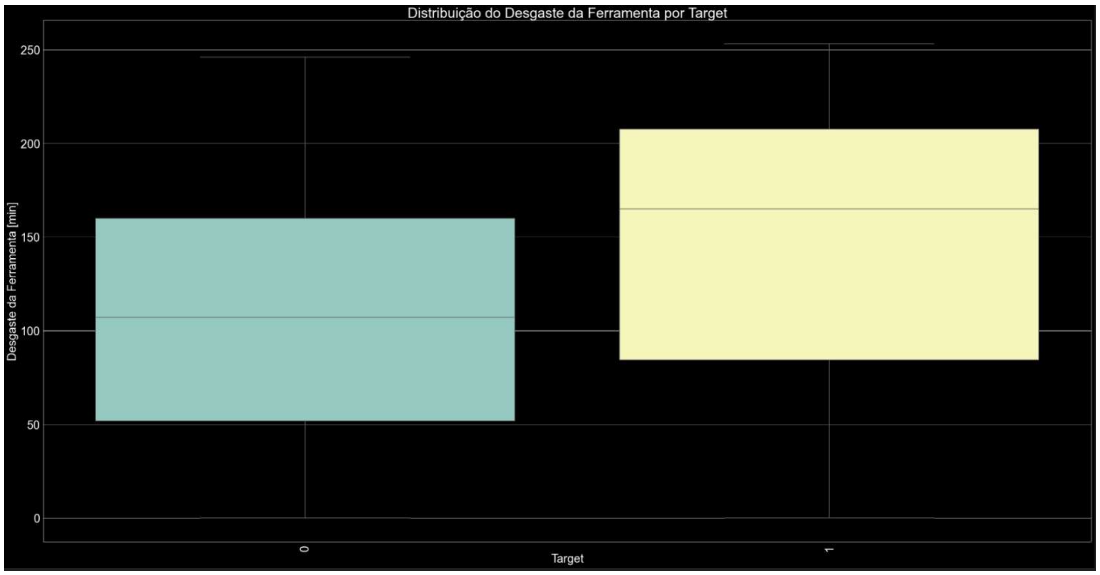
Fonte: autoria própria.

As correlações entre as demais variáveis analisadas não foram significatias e, portanto, pode-se dizer que não há correlação forte entre elas.

iv. Análise da distribuição dos dados por Variável Target

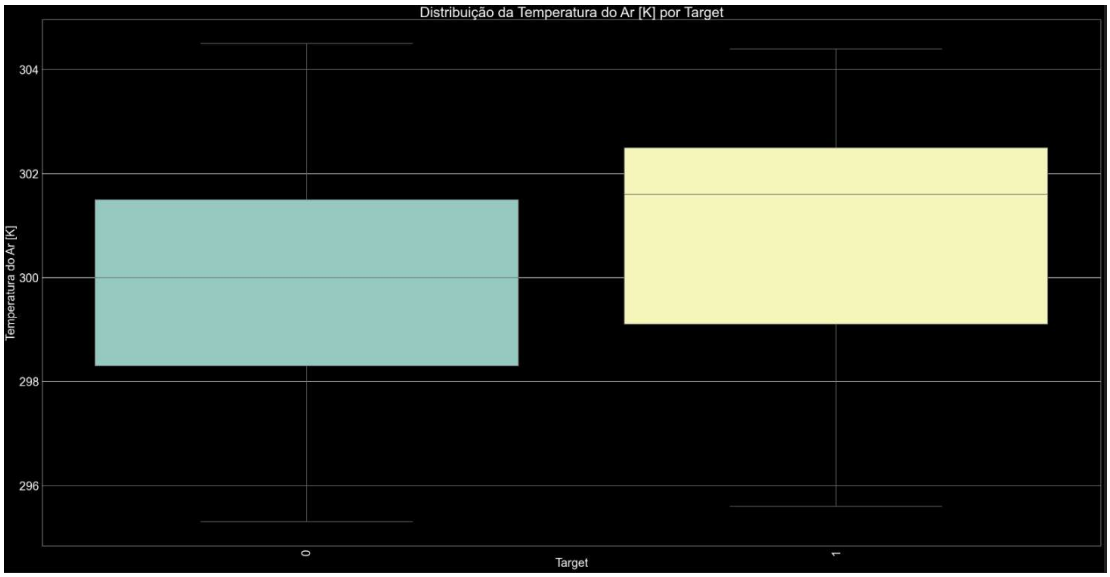
Abaixo, nos gráfico 8, 9, 10, 11 e 12, pode-se encontrar a distribuição dos dados das variáveis por Variável Target (Não falha ou Falha).

Gráfico 8 - Distribuição do Desgaste de Ferramenta por Variável Target



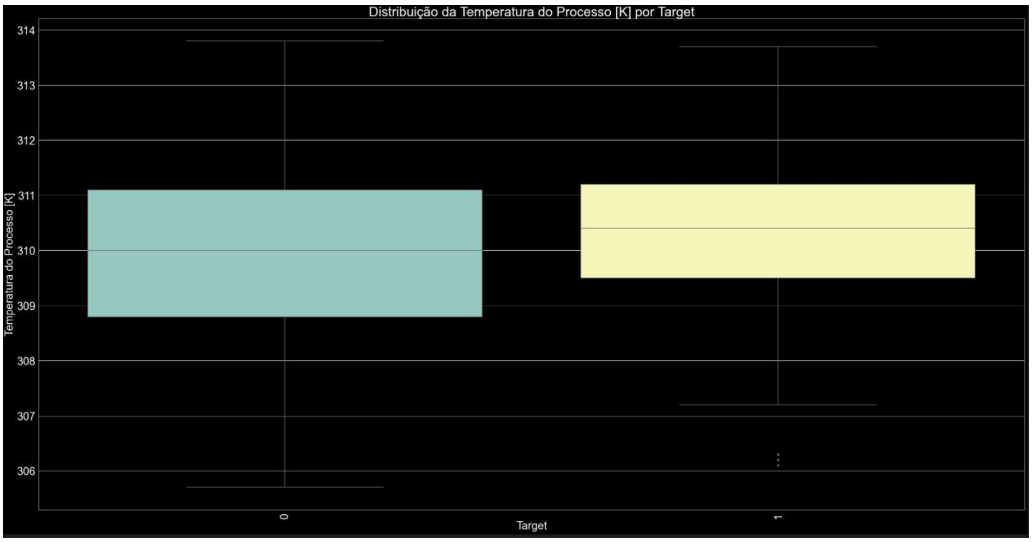
Fonte: autoria própria.

Gráfico 9 - Distribuição da Temperatura do Ar por Variável Target



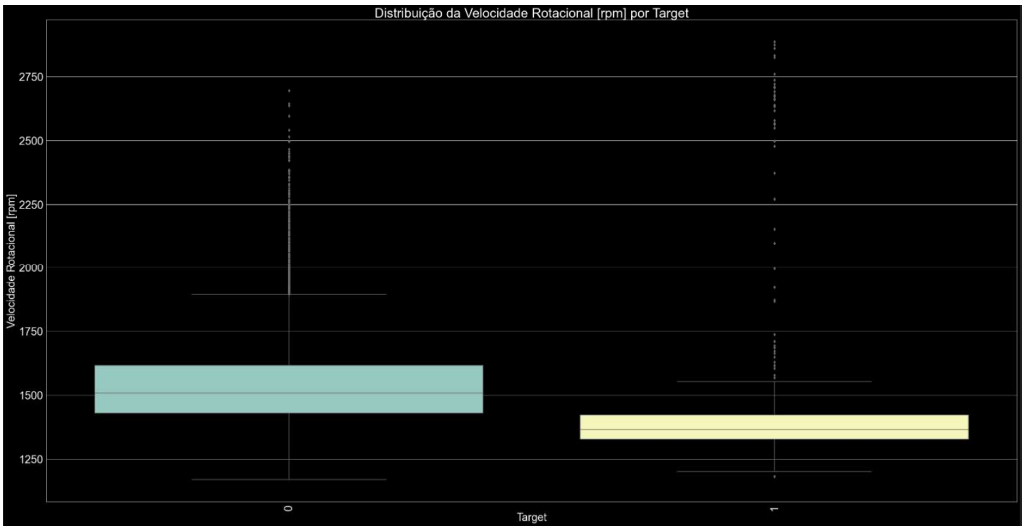
Fonte: autoria própria.

Gráfico 10: Distribuição da Temperatura do Processo por Variável Target



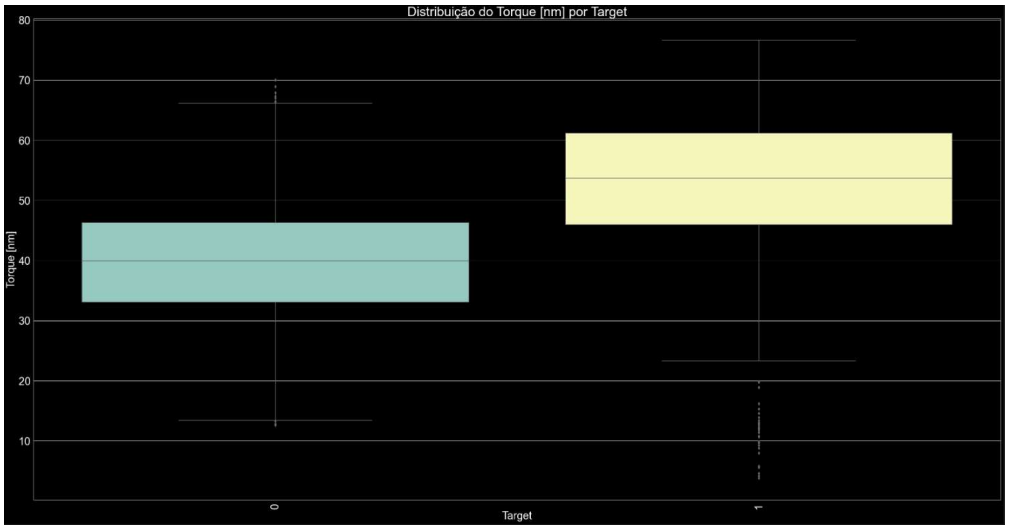
Fonte: autoria própria.

Gráfico 11: Distribuição da Velocidade Rotacional por Variável Target



Fonte: autoria própria.

Gráfico 12: Distribuição do Torque por Variável Target



Fonte: autoria própria.

É evidente um comportamento distinto entre as variáveis quando segmentadas entre ‘Não Falha’ (0) e ‘Falha’ (1). Isso se torna particularmente nítido, principalmente, ao analisar as variáveis ‘Torque’, ‘Desgaste de Ferramenta’ e ‘Temperatura do Ar’, onde há um notável deslocamento da mediana.

4.2 Construção e treinamento dos modelos de Machine Learning

i. Random Forest

O código da Figura 5 foi utilizado para gerar e treinar o Modelo de Machine Learning *Random Forest*.

Figura 5 – Código do modelo *Random Forest*

```
# Criar e treinar o modelo Random Forest
random_forest = RandomForestClassifier(n_estimators=100, random_state=21)
random_forest.fit(X_train, y_train)

y_pred_rf = random_forest.predict(X_test)

random_forest_train_accuracy = round(random_forest.score(X_train, y_train) * 100, 2)
random_forest_accuracy = round(accuracy_score(y_test, y_pred_rf) * 100, 2)

print("Acuracia:", random_forest_accuracy, "%")

disp = ConfusionMatrixDisplay.from_predictions(
    y_test,
    y_pred_rf,
    labels=np.unique(y_test),
    display_labels=np.unique(y_test),
    cmap=plt.cm.Blues,
)
disp.ax_.set_title('Confusion Matrix')
```

Fonte: autoria própria.

O parâmetro ‘n_estimators’ determina o número de árvores de decisão que serão criadas no conjunto. Para tanto, é escolhido o valor 100, frequentementemente utilizado para o treinamento deste tipo de modelo.

Enquanto o parâmetro ‘random_states’ é utilizado para controlar a aleatoriedade do conjunto de dados de treinamento do modelo. Para tanto, é escolhido o valor ‘21’ em todos os modelos a serem treinados, dessa forma, sendo possível reproduzir o treinamento dos demais modelos utilizando os mesmos dados.

ii. Support Vector Machine (SVM)

O código da Figura 6 foi utilizado para gerar e treinar o Modelo de Machine Learning *Support Vector Machine*.

Figura 6 – Código do modelo *Support Vector Machine*

```
# Criar e treinar o modelo SVM
svm_model = svm.SVC(kernel="linear", C=0.01)
svm_model.fit(X_train, y_train)

y_pred_svm = svm_model.predict(X_test)

accuracy = np.mean(y_pred_svm == y_test)
print("Acuracia:", round(accuracy * 100, 2), "%")

disp = ConfusionMatrixDisplay.from_predictions(
    y_test,
    y_pred_svm,
    labels=np.unique(y_test),  # Definindo todos os rotulos de classe possiveis
    display_labels=np.unique(y_test),  # Usando os rotulos verdadeiros do conjunto de teste
    cmap=plt.cm.Blues,
)
disp.ax_.set_title('Confusion Matrix')

plt.show()
```

Fonte: autoria própria.

O parâmetro de kernel utilizado neste caso é o ‘linear’, apropriado para os casos que podem ser separados linearmente.

Em relação ao hiperparâmetro C, é utilizado o valor 0,01. Esse hiperparâmetro afeta na regularização do modelo. Ou seja, quando o valor de C é pequeno, o modelo permite uma certa quantidade de erro de classificação no conjunto de treinamento, o que pode resultar em uma fronteira de decisão mais simples, como no caso do modelo treinado.

iii. Multilayer Perceptron (MLP)

O código da Figura 7 foi utilizado para gerar e treinar o Modelo de Machine Learning *Multilayer Perceptron*.

Figura 7 – Código do modelo *Multilayer Perceptron*

```
# Criar e treinar o modelo MLP
mlp_model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=1000, random_state=21)
mlp_model.fit(X_train, y_train)

y_pred_mlp = mlp_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred_mlp)
print("Acuracia:", round(accuracy * 100, 2), "%")

disp = ConfusionMatrixDisplay.from_estimator(
    mlp_model,
    X_test,
    y_test,
    labels=np.unique(y_test),
    display_labels=np.unique(y_test),
    cmap=plt.cm.Blues,
)
disp.ax_.set_title('Confusion Matrix')
plt.show()
```

Fonte: autoria própria.

Foi estabelecida uma arquitetura de modelo com duas camadas ocultas: a primeira camada contém 100 neurônios e a segunda camada tem 50 neurônios. A quantidade de neurônios em cada camada é especificada por meio do parâmetro 'hidden_layer_sizes'.

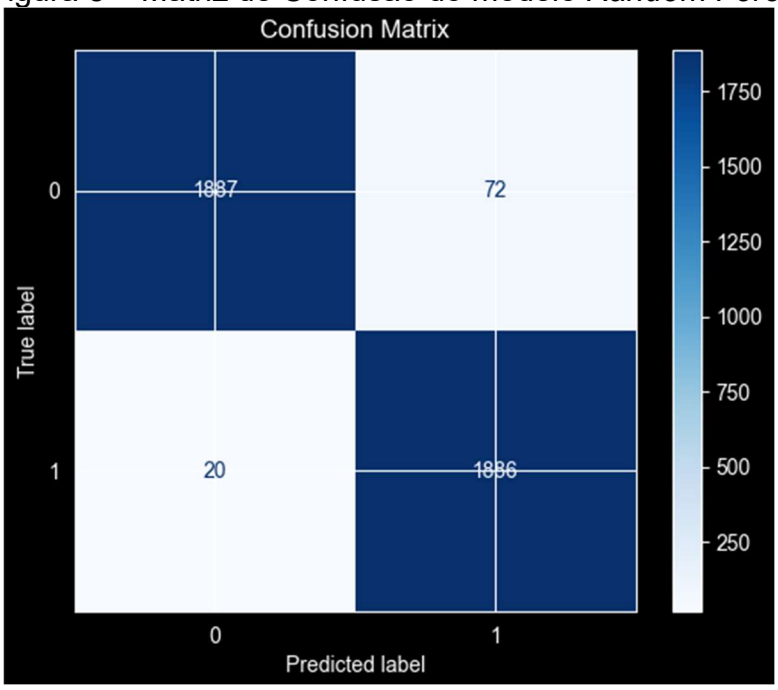
Além disso, para o modelo treinado, o número máximo de iterações é limitado a 1000, controlado pelo parâmetro 'max_iter'.

4.3 Avaliação do desempenho dos modelos de Machine Learning

i. Random Forest

Os resultados de acurácia do modelo *Random Forest* podem ser consultados na matrix de confusão na Figura 8:

Figura 8 – Matriz de Confusão do modelo *Random Forest*



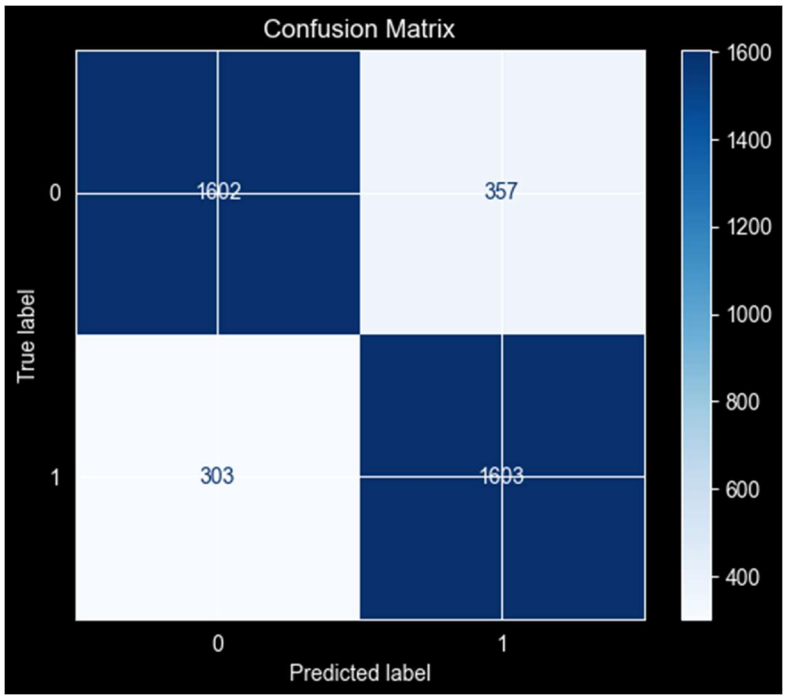
Fonte: autoria própria.

Os resultados da Figura 8 apresentam o desempenho do modelo em dois cenários distintos: casos de falha e casos de não falha. No que se refere aos casos de falha, o modelo conseguiu predizer corretamente 1886 resultados e falhou na classificação de 20 resultados. Quando se trata da situação sem falha, o modelo obteve 1887 previsões corretas e 72 previsões incorretas. Com isso, o modelo resultou em uma acurácia de 97,62% entre os dados de teste.

ii. Support Vector Machine (SVM)

Os resultados de acurácia do modelo *Support Vector Machine* podem ser consultados na matrix de confusão na Figura 9:

Figura 9 – Matriz de Confusão do modelo *Support Vector Machine*



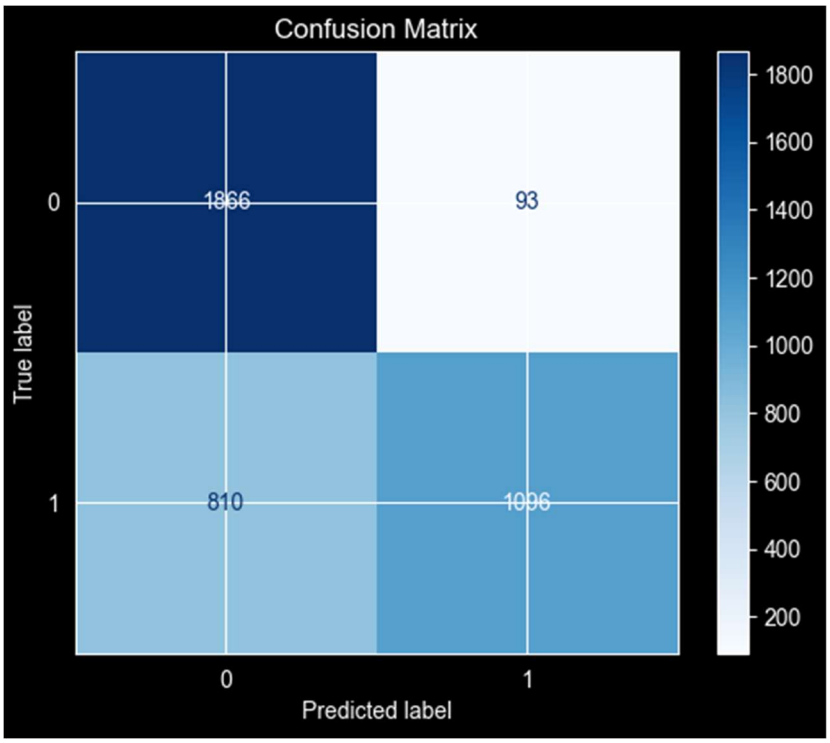
Fonte: autoria própria.

Os resultados da Figura 9 apresentam o desempenho do modelo em dois cenários distintos: casos de falha e casos de não falha. No que se refere aos casos de falha, o modelo conseguiu predizer corretamente 1603 resultados e falhou na classificação de 303 resultados. Quando se trata da situação sem falha, o modelo obteve 1602 previsões corretas e 357 previsões incorretas. Com isso, o modelo resultou em uma acurácia de 82,92% entre os dados de teste.

iii. Multilayer Perceptron (MLP)

Os resultados de acurácia do modelo *Multilayer Perceptron* podem ser consultados na matrix de confusão na Figura 10:

Figura 10 – Matriz de Confusão do modelo *Multilayer Perceptron*



Fonte: autoria própria.

Os resultados da Figura 10 apresentam o desempenho do modelo em dois cenários distintos: casos de falha e casos de não falha. No que se refere aos casos de falha, o modelo conseguiu predizer corretamente 1096 resultados e falhou na classificação de 810 resultados. Quando se trata da situação sem falha, o modelo obteve 1866 previsões corretas e 93 previsões incorretas. Com isso, o modelo resultou em uma acurácia de 76,64% entre os dados de teste.

5 CONCLUSÃO

O presente estudo evidenciou o impacto significativo da análise exploratória de dados na construção de um modelo de Machine Learning, proporcionando uma compreensão abrangente dos padrões e comportamentos dos equipamentos. A aplicação eficaz dessas técnicas revelou-se satisfatória para a identificação eficaz de parâmetros de falha, destacando a capacidade de classificação dos modelos desenvolvidos. O trabalho demonstrou a viabilidade das abordagens utilizadas, assim como a eficiência do sistema proposto, ressaltando seu potencial para otimizar estratégias de manutenção, reduzir custos operacionais e aumentar a confiabilidade dos equipamentos industriais.

O estudo concentrou-se na condução da análise exploratória dos dados do maquinário, visando compreender o comportamento das variáveis no processo. Além disso, foram selecionados e treinados algoritmos para a classificação, com a finalidade de eleger o modelo mais eficaz na classificação de falhas.

Na análise exploratória, identificaram-se as variáveis que impactavam a classificação de falhas e o seu comportamento em estado de operação normal e de falha, a qual se mostrou significativamente diferente. As variáveis sem influência foram excluídas dos dados de treinamento do modelo, visando simplificar e otimizar a construção do modelo de machine learning.

Já em relação ao treinamento, os modelos de Machine Learning escolhidos para o presente estudo foram: *Support Vector Machine, Random Forest e Multilayer Perceptron*, tendo a acurácia de suas previsões como o principal indicativo de qualidade dos modelos.

A acurácia do modelo *Support Vector Machine* atingiu 97,62%, sendo o modelo de maior acurácia entre os modelos testados. Em seguida, se encontra o modelo *Random Forest*, com 82,92% de acurácia. Por último, o modelo *Multilayer Perceptron*, que atingiu uma acurácia de 76,64%. Os resultados demonstram que a abordagem utilizada no primeiro método se mostrou muito eficaz para a classificação de falhas, levando em consideração os dados disponíveis.

Os resultados obtidos destacam o impacto significativo que a escolha do modelo de Machine Learning pode ter na assertividade da classificação de falhas. A variação de aproximadamente 21 pontos percentuais na acurácia entre o modelo de maior acurácia e o modelo de menor acurácia demonstra a importância dessa decisão.

Vale ressaltar que vários fatores podem afetar a assertividade dos modelos

de Machine Learning, tais como a qualidade dos dados disponíveis e os métodos utilizados.

Por fim, como sugestão para trabalhos futuros associados a esse tema de estudo, recomenda-se:

- Validação prática: Realizar experimentos práticos em um ambiente real para validar a eficácia dos modelos desenvolvidos e testar sua aplicabilidade em condições reais de operação, o que proporcionaria uma validação mais robusta dos resultados;
- Predição dos tipos de falha: Expandir a análise dos modelos para prever não apenas a ocorrência de falhas, mas também identificar o tipo de falha. Isso permitirá direcionar de maneira mais assertiva a manutenção do maquinário, otimizando recursos e tempo.
- Aplicativo de controle de maquinário fabril: Criar um aplicativo para monitoramento dos dados em tempo real a fim de controlar os parâmetros de máquinas sensíveis. Este aplicativo poderia ser projetado para fornecer alertas identificados pelos modelos de machine learning desenvolvidos neste trabalho. Aumentando a eficácia na identificação de possíveis falhas e permitindo uma resposta rápida, aumentando o tempo de disponibilidade do maquinário. Além disso, o aplicativo poderia incluir visualizações gráficas dos parâmetros críticos das máquinas, aprimorando a operação.
- Integração de sistema fabril: Em um contexto mais avançado, uma recomendação para trabalhos futuros seria conectar e consolidar dados em tempo real de todas as máquinas na planta, a fim de prever os horários ideais para paradas programadas para manutenção preventiva. Essa integração permitiria otimizar ainda mais o tempo de disponibilidade de maquinário na fabricação, evitando paradas não essenciais.

REFERÊNCIAS

SEEBO. **Why Predictive Maintenance is Driving Industry 4.0: The definite Guide.** In: Seebo. [S.I.], 20 out. 2018. Disponível em: https://www.seebo.com/predictive-maintenance/. Acesso em: 04 abr. 2023.

KARDEC, Alan; NASCIF, Júlio. **Manutenção: função estratégica.** 3.ed. Rio de Janeiro: Ed. Qualitymark, p. 384, 2009.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 5462**: Confiabilidade e mantenabilidade. Rio de Janeiro, 1994. 37 p

MAURA, Sandra. **Os caminhos para a transformação das empresas no futuro.** In: Indústria 4.0. [S.I.], 12 out. 2019. Disponível em: https://www.industria40.ind.br/artigo/18906-industria-40-os-caminhospara-a-transformacao-das-empresas-para-o-futuro. Acesso em: 13 abr. 2023

PIERSON, Lilian. **Data Science para leigos.** 2. ed. Rio de Janeiro: Alta Books, 2019.

MOBLEY, R. Keith. An Introduction to Predictive Maintenance. 2. ed. Butterworth-Heinemann, 2002.

RIBEIRO, Diogo Aranha. **Análise de vibração em motores elétricos com mouse óptico.** 2017. 69f. Dissertação de Mestrado – Universidade Federal de Lavras, Lavras, 2017.

SIEMANN, G. M. **Análise de vibração: estudo da técnica e aplicação prática em uma Indústria Siderúrgica,** 64 p. Disponível em: https://repositorio.unesp.br/bitstream/handle/11449/216863/siemann_gm_tcc_guara.pdf?sequence=5. Acesso em: 10/05/2023.

Emir Husni, Galuh Hertantyo, Daniel Wicaksono, Faisal Hasibuan, Andri Rahayu, and Muhamad Triawan. 2016. **Applied Internet of Things (IoT): Car monitoring system using IBM BlueMix.**

Mitchell, T. M. (1997). **Machine Learning.** McGraw-Hill.

McKinney, W. (2017). **Python para Análise de Dados: Manipulação de Dados com Pandas, NumPy e IPython.** Novatec Editora.

VDMA. (2019). **Machine Learning em Engenharia Mecânica e de Plantas.** VDMA.

Beghi, A., Garatti, S., & Piuri, V. (2015). **Machine Learning Techniques for Predictive Maintenance.** Wiley.

Breiman, L. (2001). **Two cultures of statistical modeling.** Statistical Science, 16(3), 199-231. https://doi.org/10.1214/ss/1009213726

Smith, J. D., Johnson, L. K., & Lee, K. M. (2021). **Evaluation from precision, recall and F-factor to ROC, informedness, markedness and correlation.** Journal of Data Science, 19(3), 456-478. https://doi.org/10.1080/10618600.2021.1901234

Shen, R., & Zhang, B. (2018). **The research of regression model in machine learning field.** MATEC Web of Conferences, 176, 01033.

Bengio, Y., Goodfellow, I., & Courville, A. (2016). **Deep Learning.** MIT Press. Filatro, A. (2020). Análise de Regressão: Teoria e Aplicações. Elsevier.

Piassa, A. R. (2019). **Aprendizado de Máquina: A Modelagem Estatística e Computacional do Aprendizado.** Blucher.

Sutton, R. S., & Barto, A. G. (2018). **Reinforcement Learning: An Introduction.** MIT Press.

Faceli, K., Lorena, A. C., Gama, J., & Carvalho, A. C. P. L. F. (2021**). Inteligência Artificial: uma abordagem de aprendizado de máquina.** LTC Editora.

Harrison, M. (2020). **Machine Learning: Guia de Referência Rápida.** Novatec Editora.

Myatt, G. J. (2007). **Making Sense of Data I: A Practical Guide to Exploratory Data Analysis and Data Mining.** Wiley.

FLUKE. **Introdução aos princípios da termografia.** Ed. ATP, 2009.

CARAMALHO, Alberto. **25 anos em termografia.** 1ª ed. Portugal: Bubok, 2012.

GAMA, Ana Luiza Ferraz. **O uso de Machine Learning na implementação de manutenção preditiva em usinas termelétricas.** Rio de Janeiro, 2017. Projeto de Graduação (Curso de Engenharia de Produção) - Escola Politécnica, Universidade Federal do Rio de Janeiro.

MACÊDO, Letícia Costa. **Manutenção Preditiva no Contexto da Indústria 4.0: Um Modelo Preditivo em uma Fábrica do Ramo Metalúrgico.** 2020. Monografia (Curso de Engenharia Metalúrgica) - Instituto Federal do Espírito Santo, Campus Vitória.

ÇINAR, Zeki Murat. **Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0.** 2020. Artigo - Departamento de Engenharia Mecânica, Universidade do Mediterrâneo Oriental.

NETO, Cesare Di Girolamo. **Potencial de técnicas de mineração de dados para o mapeamento de áreas cafeeiras.** INPE, São José dos Campos, 2014.

LORENA, Ana Carolina. **Uma introdução às Support Vector Machines.** FAPESP, 2007.

SHEYKHMOUSA, Mohammadreza. **Support Vector Machine Versus Random Forest for Remote Sensing Image Classification: A Meta-Analysis and Systematic Review.** 2020.

SCHONLAU, Matthias. **The random forest algorithm for statistical learning.** 2020.

APÊNDICE A – Código em Python completo da AED e Construção dos Modelos de Machine Learning

Machine Predictive Maintenance

Link Dataset: https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification

Importando Bibliotecas Úteis:

In [62]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import jinja2
import sklearn as skl
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, Conf
usionMatrixDisplay
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import datasets, svm
from sklearn.neural_network import MLPClassifier
```

Definindo estilo para dataframes:

In [63]:

```
def apply_background_color(value):
    background_color = 'background-color: #082032'
    return background_color


#adicionar fundo aos valores do dataframe:
#.style.applymap(apply_background_color)
```

Análise Exploratória dos Dados:

Entendendo e tratando a base de dados

- **Removendo duplicatas;**
- **Identificando valores nulos por colunas;**
- **Checando tipos de colunas;**
- **Verificando valores únicos por coluna;**
- **Removendo colunas desnecessárias.**

Trazendo o DataFrame para análise:

In [64]:

```
cursor.execute("SELECT * FROM tcc.tcc_dados")
df = pd.DataFrame(cursor.fetchall(), columns=['udi','product_id','type', 'air_temperatur
e_k', 'process_temperature_k', 'rotational_speed_rpm', 'torque_nm', 'tool_wear_min', 'tar
get', 'failure_type'])
df.head(8).style.applymap(apply_background_color)
```

Out[64]:

udi	product_id	type	air_temperature_k	process_temperature_k	rotational_speed_rpm	torque_nm	tool_wear_min	target	f

	udi	product_id	type	air_temperature_k	process_temperature_k	rotational_speed_rpm	torque_nm	tool_wear_min	target
1	2	L47181	L	298.2	308.7	1408.0	46.3	3.0	0
2	3	L47182	L	298.1	308.5	1498.0	49.4	5.0	0
3	4	L47183	L	298.2	308.6	1433.0	39.5	7.0	0
4	5	L47184	L	298.2	308.7	1408.0	40.0	9.0	0
5	6	M14865	M	298.1	308.6	1425.0	41.9	11.0	0
6	7	L47186	L	298.1	308.6	1558.0	42.4	14.0	0
7	8	L47187	L	298.1	308.6	1527.0	40.2	16.0	0

Identificando e removendo duplicatas do dataset:

In [65]:

```
# Checando duplicatas no dataset:

total_duplicatas = df.duplicated().sum()
print("Total de duplicatas no dataset: ",total_duplicatas)
```

```
Total de duplicatas no dataset:  0
```

In [66]:

```
# Checando se a coluna produto id (valores distintos por produto) possui duplicatas:

produto_duplicatas = df.duplicated(subset=['product_id']).sum()
print("Total de duplicatas no dataset por produto_id: ",produto_duplicatas)
```

```
Total de duplicatas no dataset por produto_id:  0
```

O dataset não possui duplicatas. Por isso, segue para o próximo passo de identificação de valores ausentes/nulos:

In [67]:

```
nulos = df.isnull().sum()
print("Total de nulos por coluna: ",nulos)
```

```
Total de nulos por coluna:  udi                        0
product_id                 0
type                       0
air_temperature_k          0
process_temperature_k      0
rotational_speed_rpm       0
torque_nm                  0
tool_wear_min              0
target                     0
failure_type               0
dtype: int64
```

Como não há valores nulos, não é necessário nenhum tratamento de remoção ou substituição de valores. Próximo passo será de checagem de tipos de colunas:

Explorando informações gerais sobre o dataframe:

In [68]:

```
informacoes = df.info()
print("Informações gerais sobre o dataset: ", informacoes)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
```

```
---  ------                  --------------  -----
 0   udi                     10000 non-null  int64
 1   product_id              10000 non-null  object
 2   type                    10000 non-null  object
 3   air_temperature_k       10000 non-null  object
 4   process_temperature_k   10000 non-null  object
 5   rotational_speed_rpm    10000 non-null  object
 6   torque_nm               10000 non-null  object
 7   tool_wear_min           10000 non-null  object
 8   target                  10000 non-null  object
 9   failure_type            10000 non-null  object
dtypes: int64(1), object(9)
memory usage: 781.4+ KB
Informações gerais sobre o dataset:  None
```

Várias colunas que são numéricas estão como objeto. Para dar prosseguimento nas análise é necessária a conversão de tipo das colunas:

- **air_temperature_k;**
- **process_temperature_k;**
- **rotational_speed_rpm;**
- **torque_nm;**
- **tool_wear_min.**

In [69]:

```
df['air_temperature_k'] = df['air_temperature_k'].astype(float)
df['process_temperature_k'] = df['process_temperature_k'].astype(float)
df['rotational_speed_rpm'] = df['rotational_speed_rpm'].astype(float)
df['torque_nm'] = df['torque_nm'].astype(float)
df['tool_wear_min'] = df['tool_wear_min'].astype(float)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   udi                     10000 non-null  int64
 1   product_id              10000 non-null  object
 2   type                    10000 non-null  object
 3   air_temperature_k       10000 non-null  float64
 4   process_temperature_k   10000 non-null  float64
 5   rotational_speed_rpm    10000 non-null  float64
 6   torque_nm               10000 non-null  float64
 7   tool_wear_min           10000 non-null  float64
 8   target                  10000 non-null  object
 9   failure_type            10000 non-null  object
dtypes: float64(5), int64(1), object(4)
memory usage: 781.4+ KB
```

Tipagem corriga, o próximo passo é a identificação da quantidade de valores únicos por colunas:

In [70]:

```
unicos = df.apply(lambda x: x.nunique())

print("Valores únicos por coluna:\n", unicos)
```

```
Valores únicos por coluna:
 udi                     10000
product_id               10000
type                         3
air_temperature_k           93
process_temperature_k       82
rotational_speed_rpm       941
torque_nm                  577
tool_wear_min              246
```

```
target                       2
failure_type                 6
dtype: int64
```

Identificação das colunas desnecessárias:

In [71]:

```
df.head(2)
```

Out[71]:

	udi	product_id	type	air_temperature_k	process_temperature_k	rotational_speed_rpm	torque_nm	tool_wear_min	target	f
0	1	M14860	M	298.1	308.6	1551.0	42.8	0.0	0	
1	2	L47181	L	298.2	308.7	1408.0	46.3	3.0	0	

As três primeiras colunas não serão necessárias para análise e construção do modelo pois não dizem respeito à variáveis de processo de fabricação.

In [72]:

```
df = df[['air_temperature_k','process_temperature_k','rotational_speed_rpm','torque_nm',
'tool_wear_min','target','failure_type']]

df.head(2)
```

Out[72]:

	air_temperature_k	process_temperature_k	rotational_speed_rpm	torque_nm	tool_wear_min	target	failure_type
0	298.1	308.6	1551.0	42.8	0.0	0	No Failure
1	298.2	308.7	1408.0	46.3	3.0	0	No Failure

Análise da distribuição das variáveis

Histplots para identificação das distribuição das variáveis em seu range.

Definição de tema:

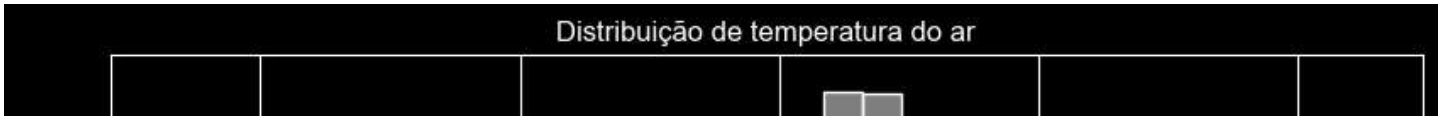
In [73]:

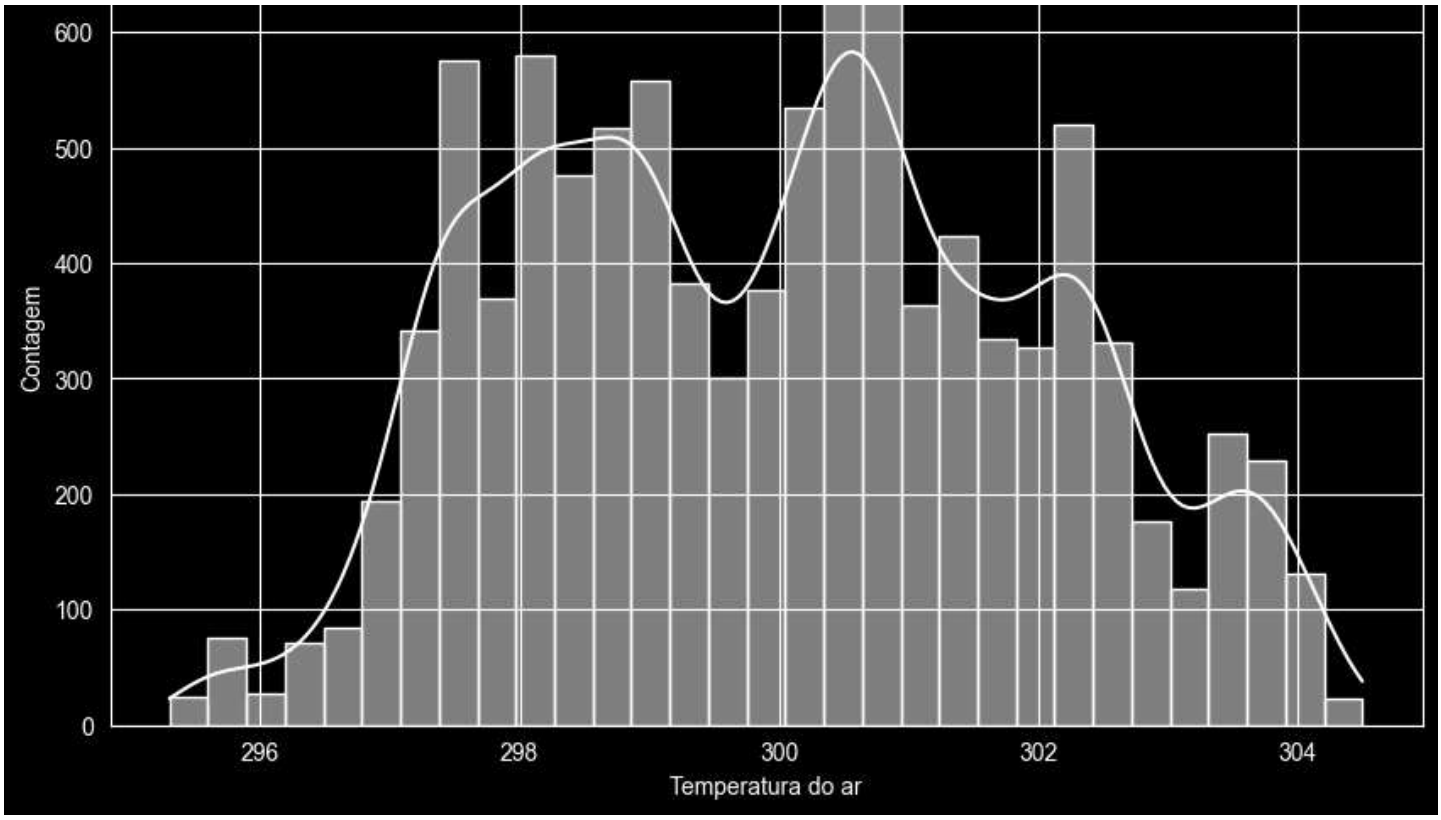
```
# Definir estilo do Seaborn
sns.set_style("darkgrid")

# Definir tema do Matplotlib
plt.style.use("dark_background")
```

Análise de distribuição de temperatura do ar:

In [74]:

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="air_temperature_k", kde=True, color="white")
plt.title("Distribuição de temperatura do ar", color="white")
plt.xlabel("Temperatura do ar", color="white")
plt.ylabel("Contagem", color="white")
plt.tick_params(colors="white")

plt.show()
```

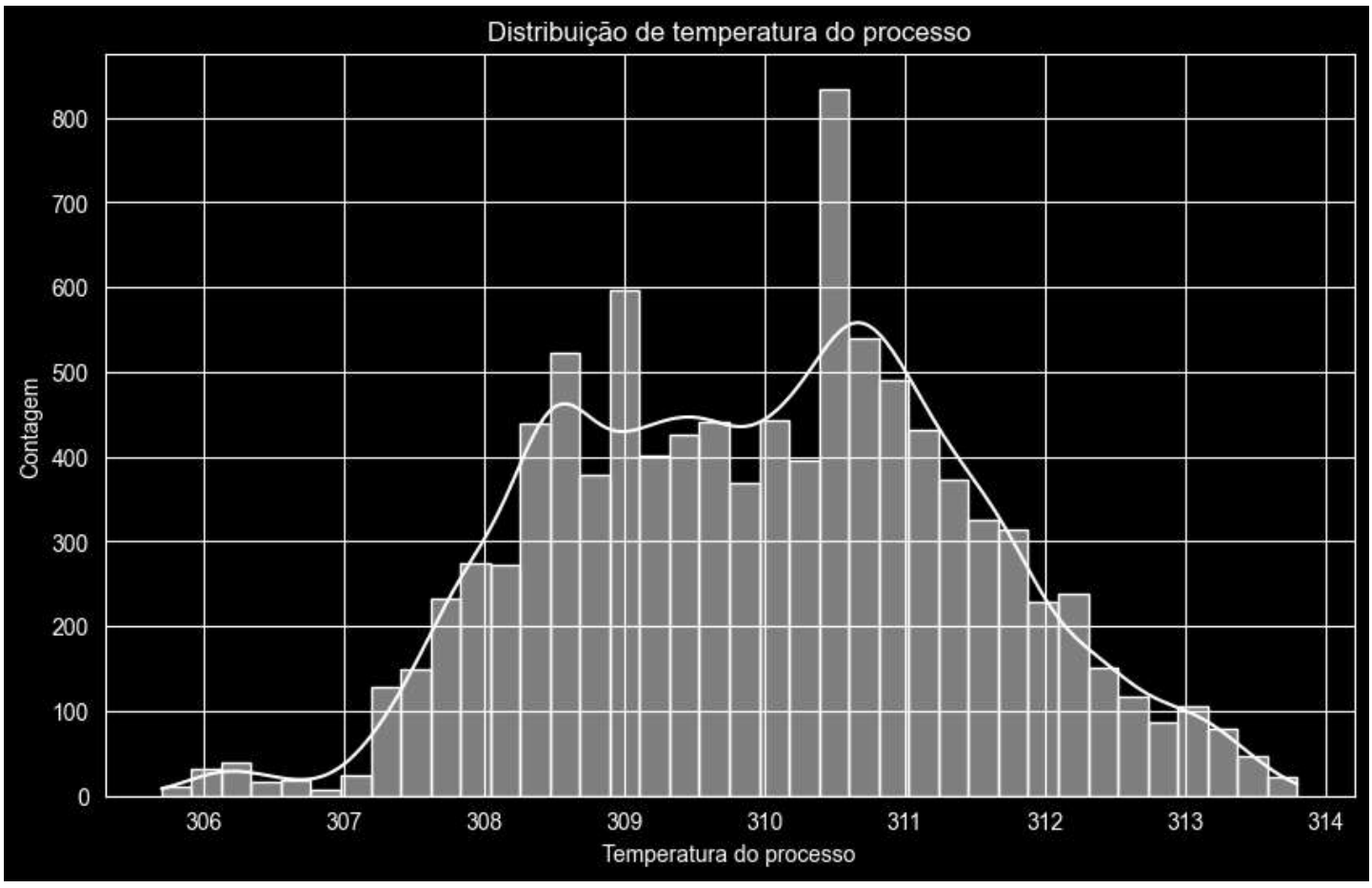




Análise de distribuição de temperatura do processo:

In [75]:

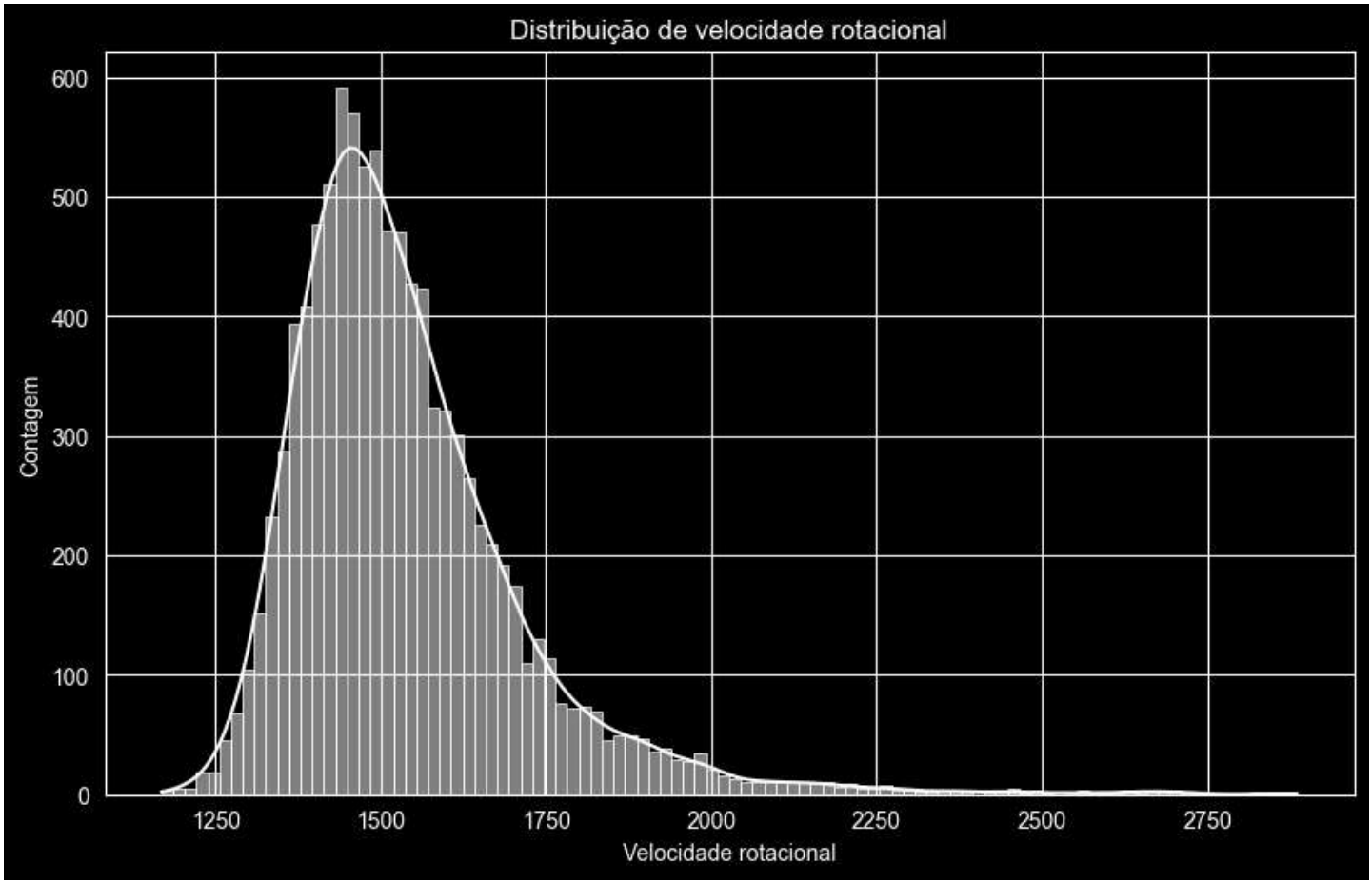
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="process_temperature_k", kde=True, color="white")
plt.title("Distribuição de temperatura do processo", color="white")
plt.xlabel("Temperatura do processo", color="white")
plt.ylabel("Contagem", color="white")
plt.tick_params(colors="white")

plt.show()
```



Análise de distribuição de velocidade rotacional:

In [76]:

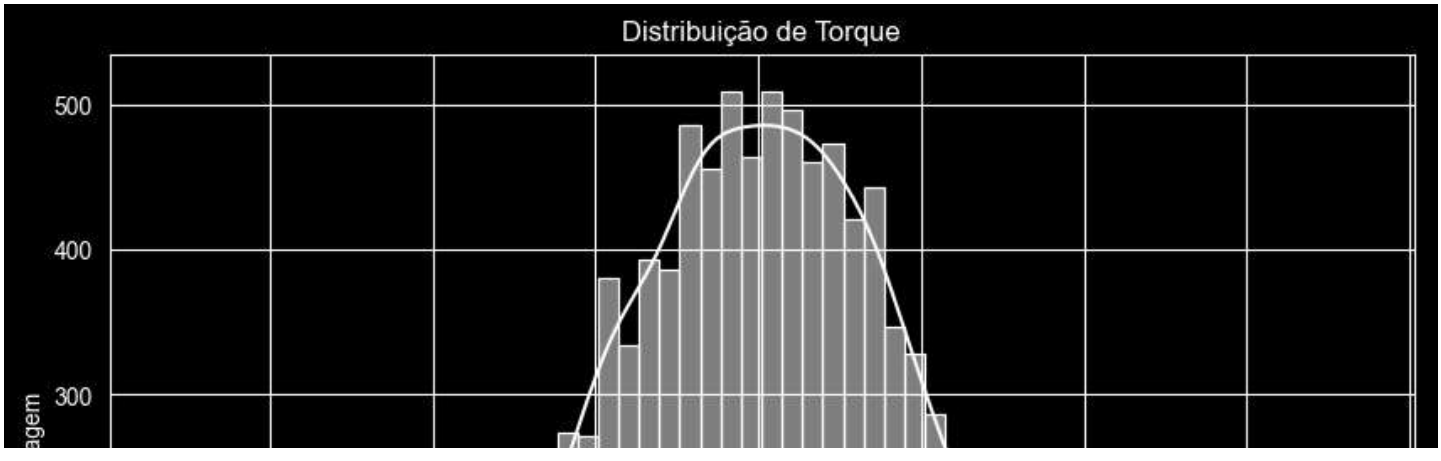
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="rotational_speed_rpm", kde=True, color="white")
plt.title("Distribuição de velocidade rotacional", color="white")
plt.xlabel("Velocidade rotacional", color="white")
plt.ylabel("Contagem", color="white")
plt.tick_params(colors="white")

plt.show()
```

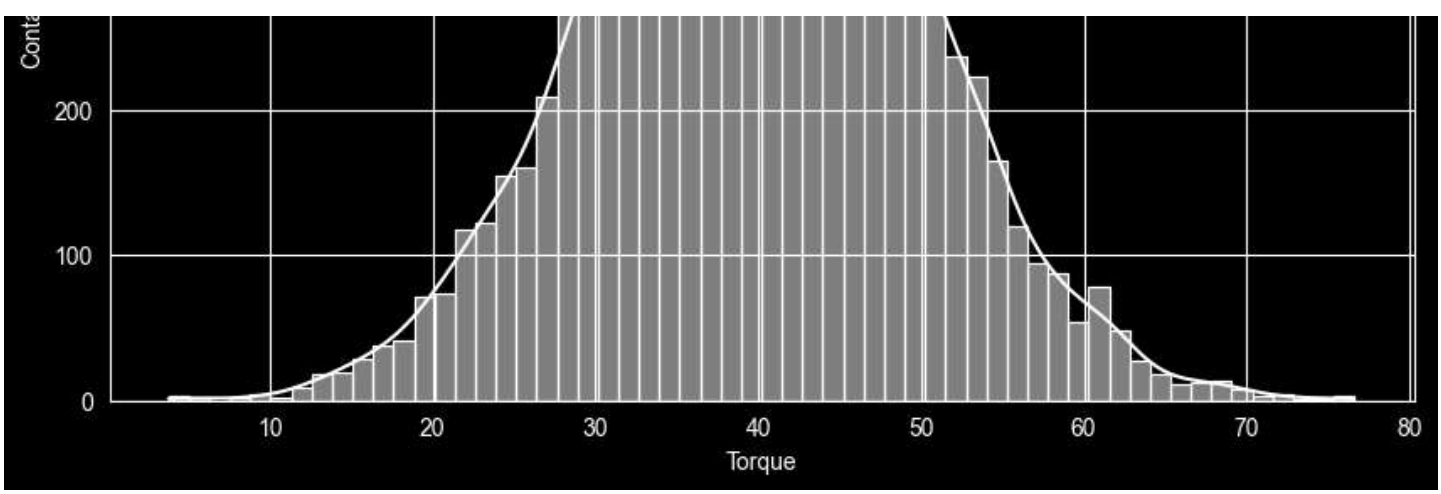


Análise de distribuição de torque:

In [77]:

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="torque_nm", kde=True, color="white")
plt.title("Distribuição de Torque", color="white")
plt.xlabel("Torque", color="white")
plt.ylabel("Contagem", color="white")
plt.tick_params(colors="white")

plt.show()
```

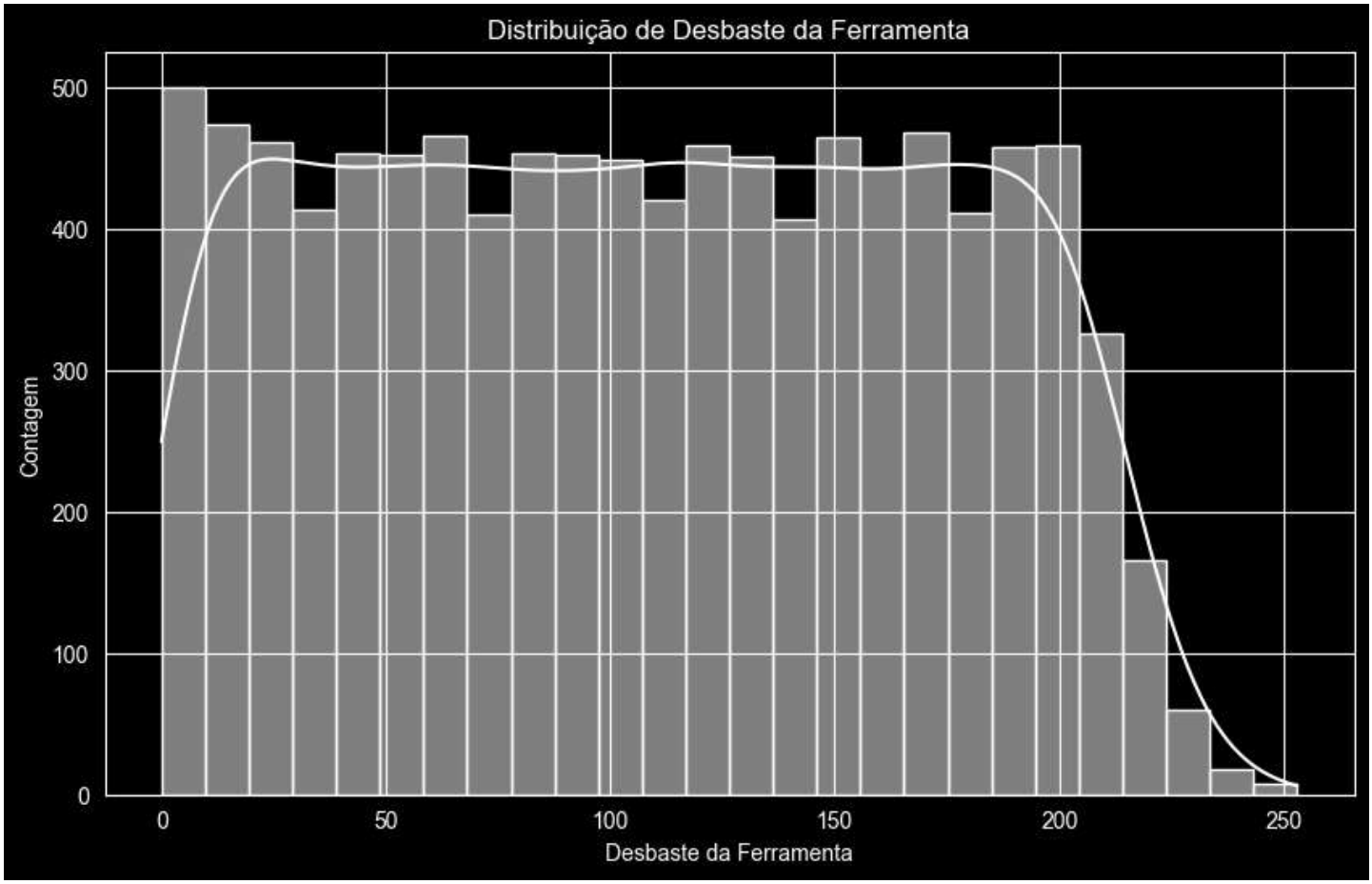




Análise de distribuição de Desbaste de ferramenta (tool wear):

In [78]:

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="tool_wear_min", kde=True, color="white")
plt.title("Distribuição de Desbaste da Ferramenta", color="white")
plt.xlabel("Desbaste da Ferramenta", color="white")
plt.ylabel("Contagem", color="white")
plt.tick_params(colors="white")

plt.show()
```



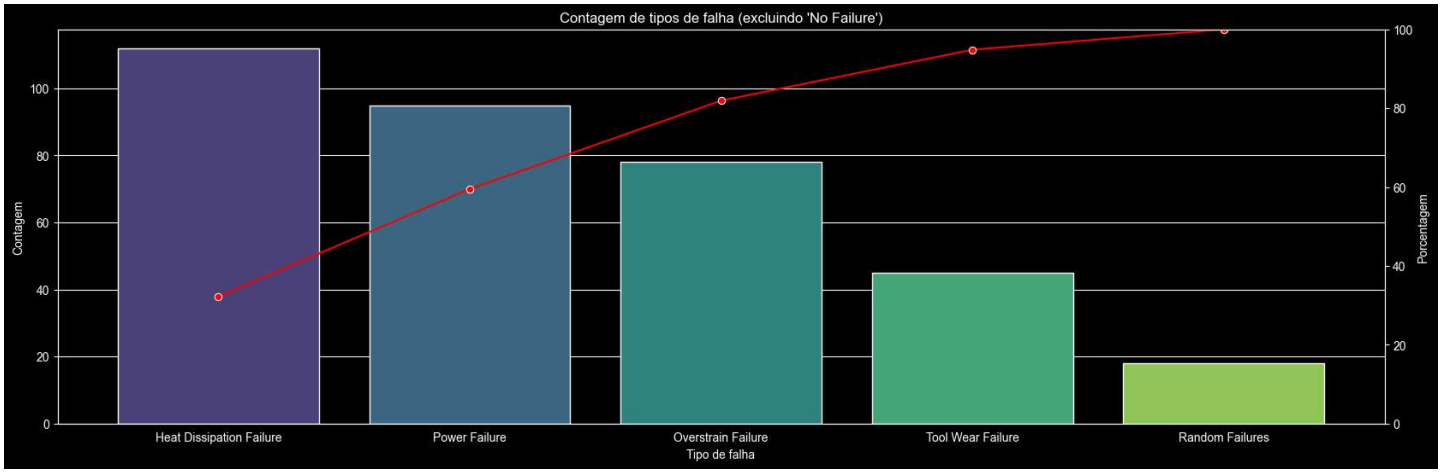
Identificação da ocorrência das falhas:

Distribuição do Tipo de Falha via diagrama de Pareto:

In [79]:

```
# Filtrar as linhas onde 'failure_type' não é igual a 'No Failure'
df_filtered = df[df['failure_type'] != 'No Failure']
```

```
failure_counts = df_filtered['failure_type'].value_counts().sort_values(ascending=False)
total_failures = failure_counts.sum()
failure_percentages = (failure_counts / total_failures) * 100
failure_cumulative_percentages = failure_percentages.cumsum()

# Gerar o gráfico de Pareto
plt.figure(figsize=(20, 6))
ax = sns.barplot(x=failure_counts.index, y=failure_counts.values, palette="viridis")
plt.title("Contagem de tipos de falha (excluindo 'No Failure')")
plt.xlabel("Tipo de falha")
plt.ylabel("Contagem")

# Adicionar eixo y secundário com as porcentagens acumuladas
ax2 = ax.twinx()
ax2.set_ylabel("Porcentagem")
ax2.set_ylim(0, 100)
ax2.grid(False)
sns.lineplot(x=failure_counts.index, y=failure_cumulative_percentages, marker="o", color="red", ax=ax2)

plt.xticks(rotation=90)
plt.show()
```

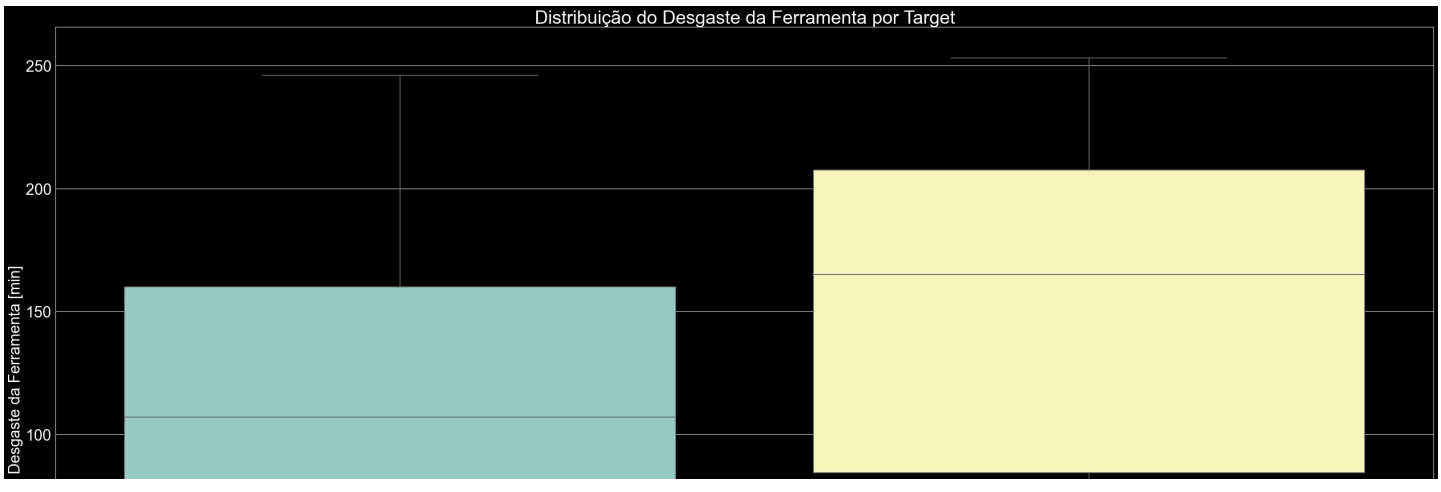


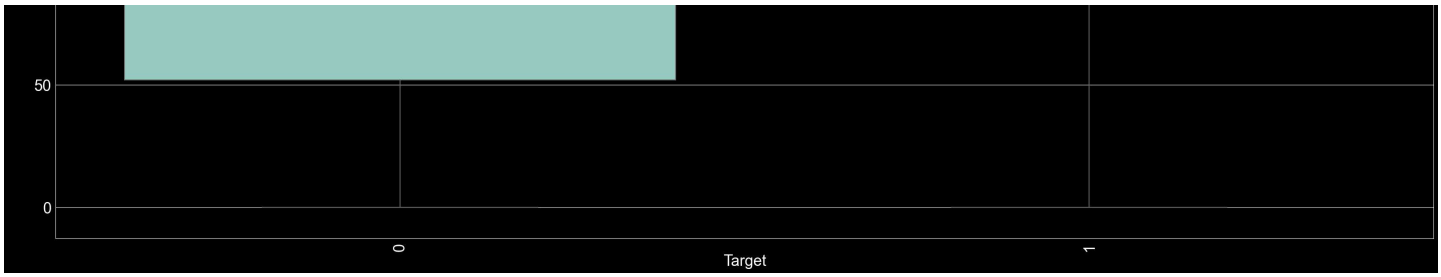
Análise de comportamento das variáveis por target.

Análise de variação por desgaste da ferramenta [min]:

In [80]:

```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='target', y='tool_wear_min')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição do Desgaste da Ferramenta por Target", fontsize=30)
plt.xlabel("Target", fontsize=25)
plt.ylabel("Desgaste da Ferramenta [min]", fontsize=25)
plt.show()
```

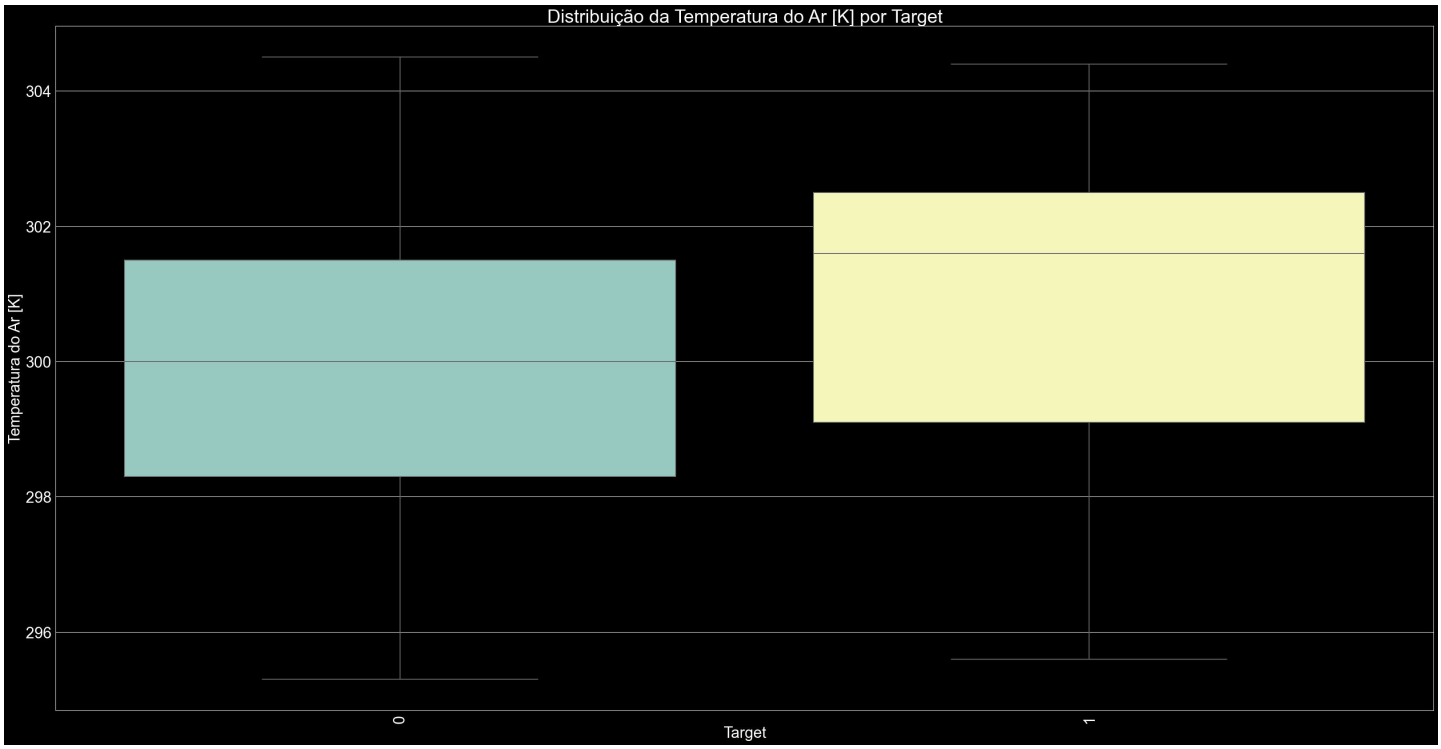




Análise de variação da temperatura do ar [K]:

In [81]:

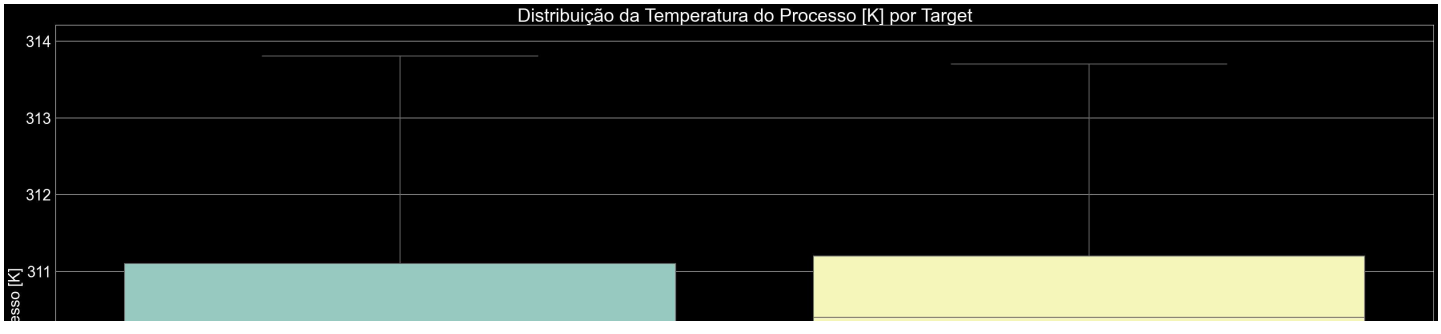
```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='target', y='air_temperature_k')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição da Temperatura do Ar [K] por Target", fontsize=30)
plt.xlabel("Target", fontsize=25)
plt.ylabel("Temperatura do Ar [K]", fontsize=25)
plt.show()
```

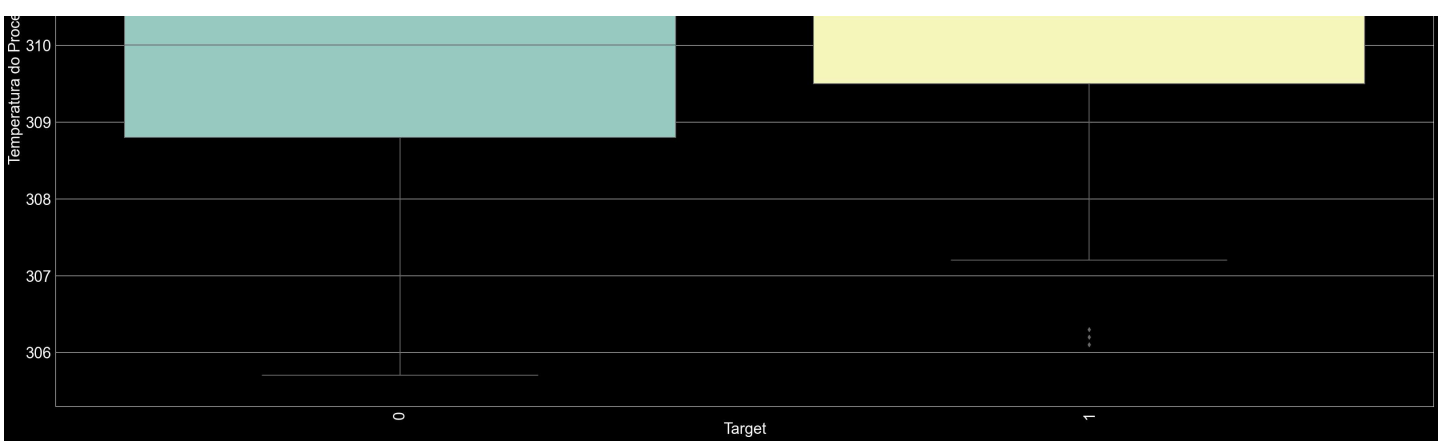


Análise de variação da temperatura do processo [K]:

In [82]:

```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='target', y='process_temperature_k')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição da Temperatura do Processo [K] por Target", fontsize=30)
plt.xlabel("Target", fontsize=25)
plt.ylabel("Temperatura do Processo [K]", fontsize=25)
plt.show()
```

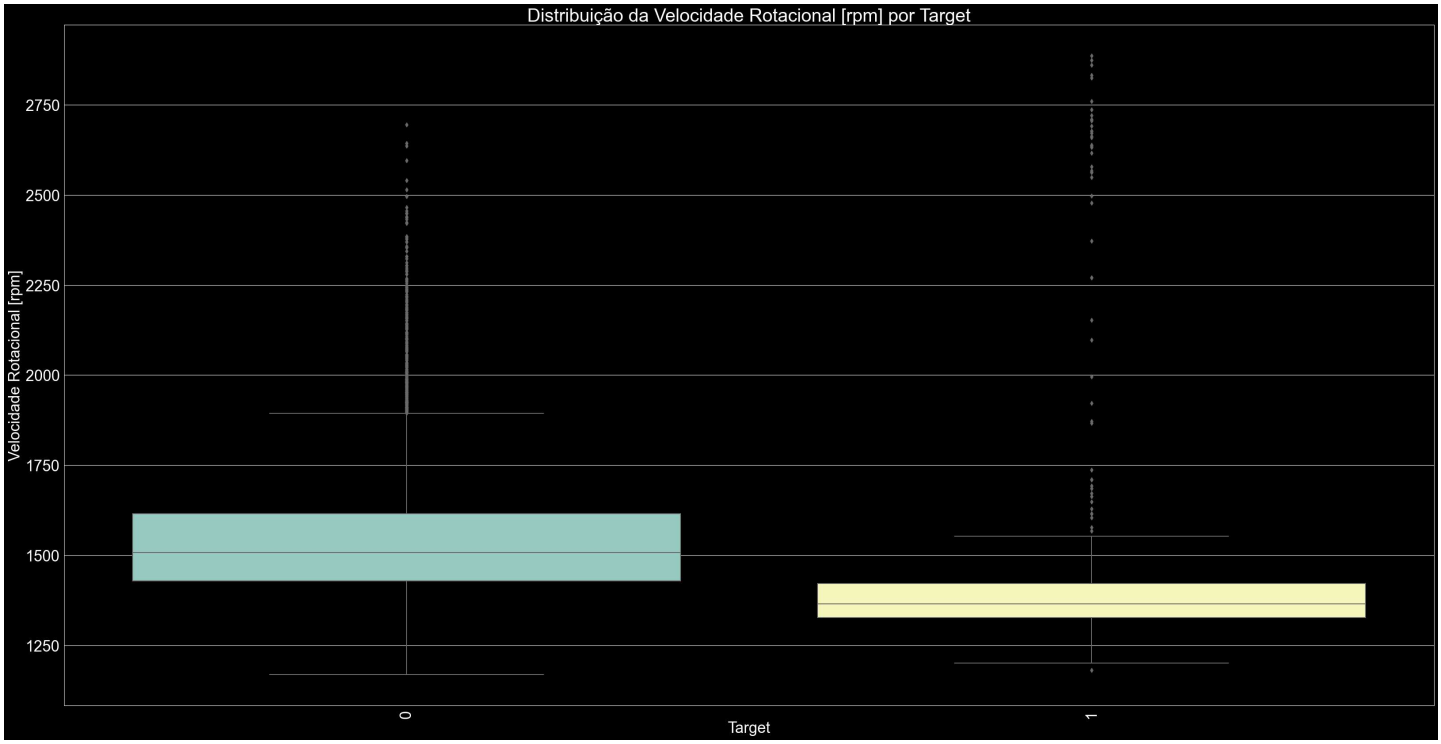




Análise de variação por velocidade rotacional [K]:

In [83]:

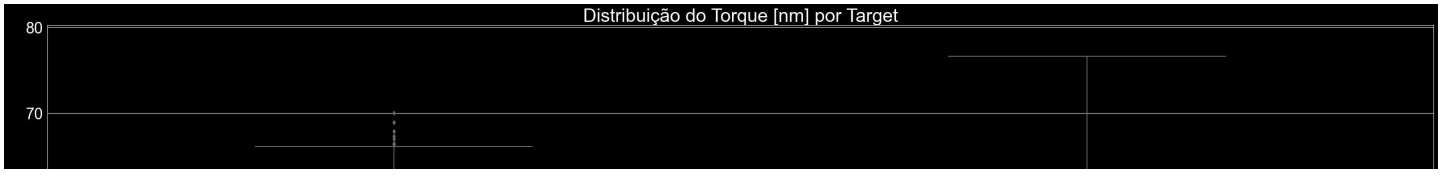
```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='target', y='rotational_speed_rpm')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição da Velocidade Rotacional [rpm] por Target", fontsize=30)
plt.xlabel("Target", fontsize=25)
plt.ylabel("Velocidade Rotacional [rpm]", fontsize=25)
plt.show()
```

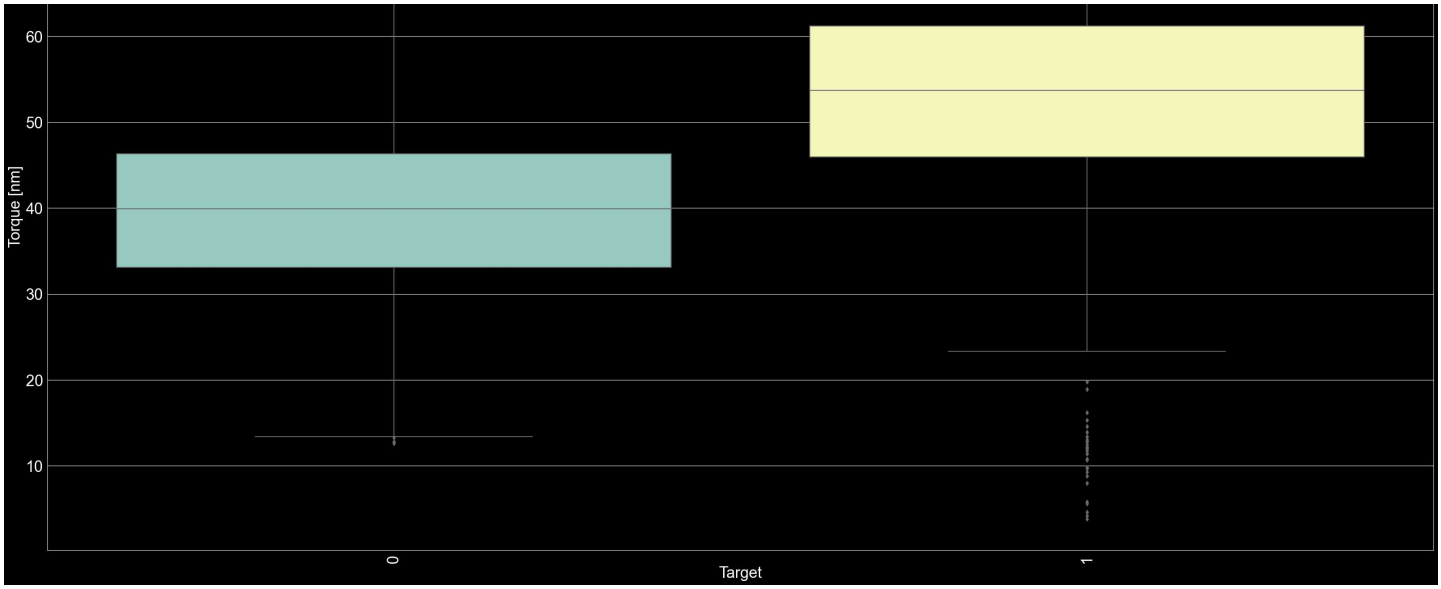


Análise de variação do torque [nm]:

In [84]:

```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='target', y='torque_nm')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição do Torque [nm] por Target", fontsize=30)
plt.xlabel("Target", fontsize=25)
plt.ylabel("Torque [nm]", fontsize=25)
plt.show()
```





Análise de comportamento das variáveis por tipo de falha

Análise de correlação geral incluindo todas as variáveis numéricas e, transformando as variáveis categóricas em numéricas para avaliar a correlação:

In [85]:

```
df.head(3)
```

Out[85]:

	air_temperature_k	process_temperature_k	rotational_speed_rpm	torque_nm	tool_wear_min	target	failure_type
0	298.1	308.6	1551.0	42.8	0.0	0	No Failure
1	298.2	308.7	1408.0	46.3	3.0	0	No Failure
2	298.1	308.5	1498.0	49.4	5.0	0	No Failure

In [86]:

```
encoded_dataframe = df.copy()

label_encoder = LabelEncoder()

encoded_dataframe = encoded_dataframe[encoded_dataframe['failure_type'] != 'No Failure']

encoded_dataframe['failure_type'] = label_encoder.fit_transform(encoded_dataframe['failur
e_type'])
encoded_dataframe['target'] = encoded_dataframe['target'].astype(float)

encoded_dataframe.head(5)
```

Out[86]:

	air_temperature_k	process_temperature_k	rotational_speed_rpm	torque_nm	tool_wear_min	target	failure_type
50	298.9	309.1	2861.0	4.6	143.0	1.0	2
69	298.9	309.0	1410.0	65.7	191.0	1.0	2
77	298.8	308.9	1455.0	41.3	208.0	1.0	4
160	298.4	308.2	1282.0	60.7	216.0	1.0	1
161	298.3	308.1	1412.0	52.3	218.0	1.0	1

In [87]:

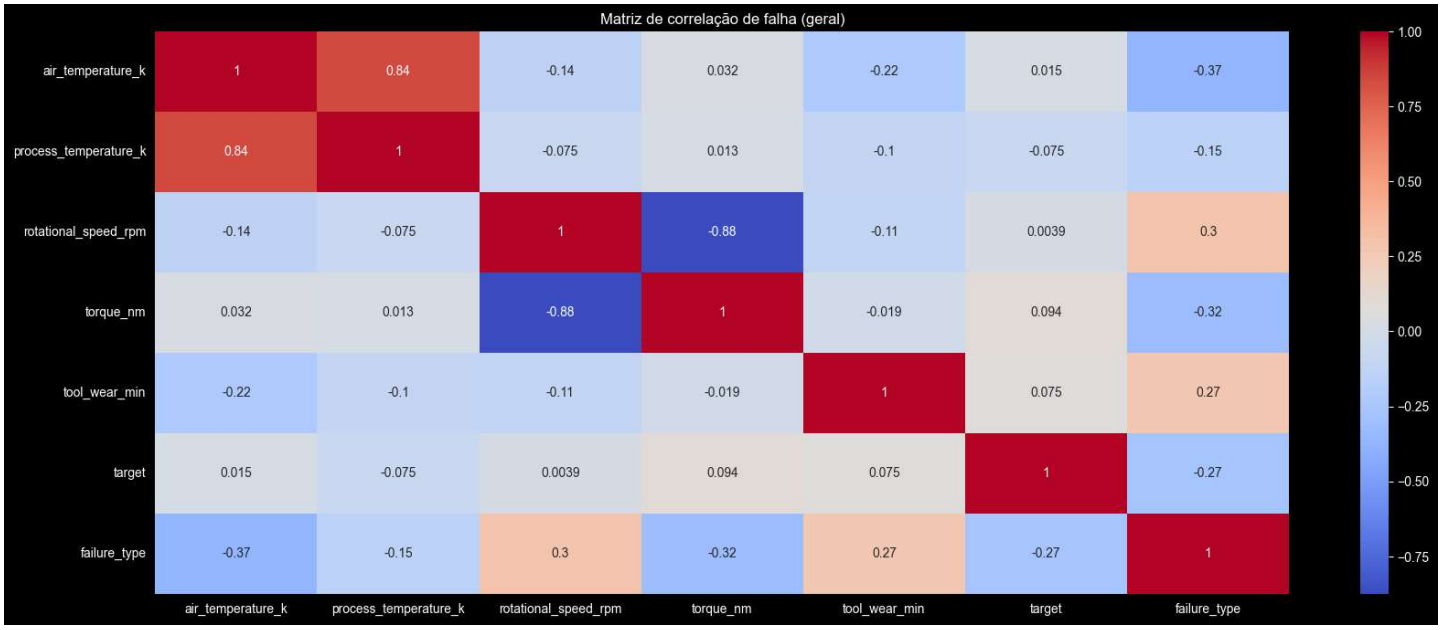
```
encoded_dataframe['failure_type'].value_counts()
```

Out[87]:

```
failure_type
0    112
2     95
1     78
4     45
3     18
Name: count, dtype: int64
```

In [88]:

```
# Calcular a matriz de correlação
corr_matrix = encoded_dataframe.corr()

# Plotar o mapa de calor da matriz de correlação
plt.figure(figsize=(20, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.title("Matriz de correlação de falha (geral)")
plt.show()
```

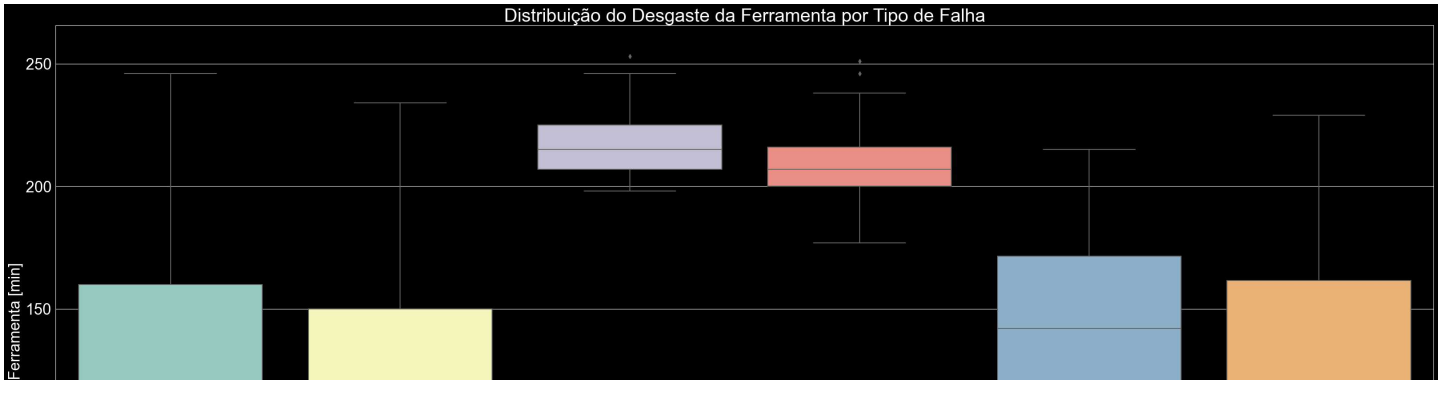


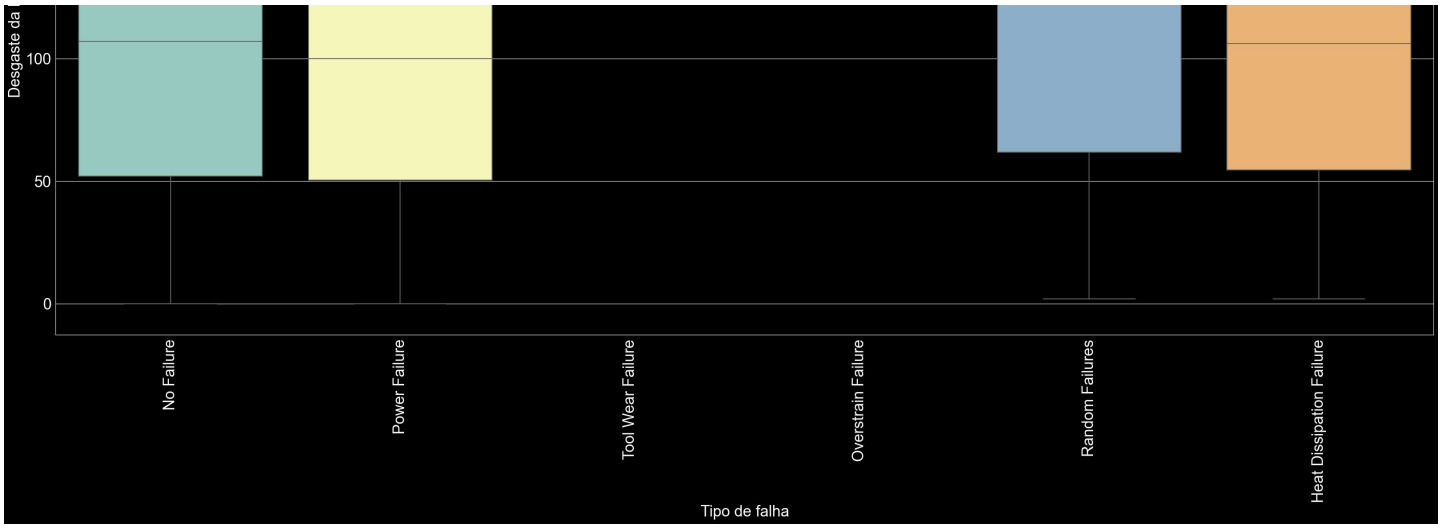
De maneira geral, as únicas correlações explícitas por falha são a da temperatura do ar pela temperatura do processo e do torque com a velocidade rotacional.. Para verificar quais variáveis impactam cada tipo de falha será analisado separadamente uma matriz de correlação por tipo de falha.

Análise de variação por Desgaste da Ferramenta [min]

In [89]:

```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='failure_type', y='tool_wear_min')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição do Desgaste da Ferramenta por Tipo de Falha", fontsize=30)
plt.xlabel("Tipo de falha", fontsize=25)
plt.ylabel("Desgaste da Ferramenta [min]", fontsize=25)
plt.show()
```

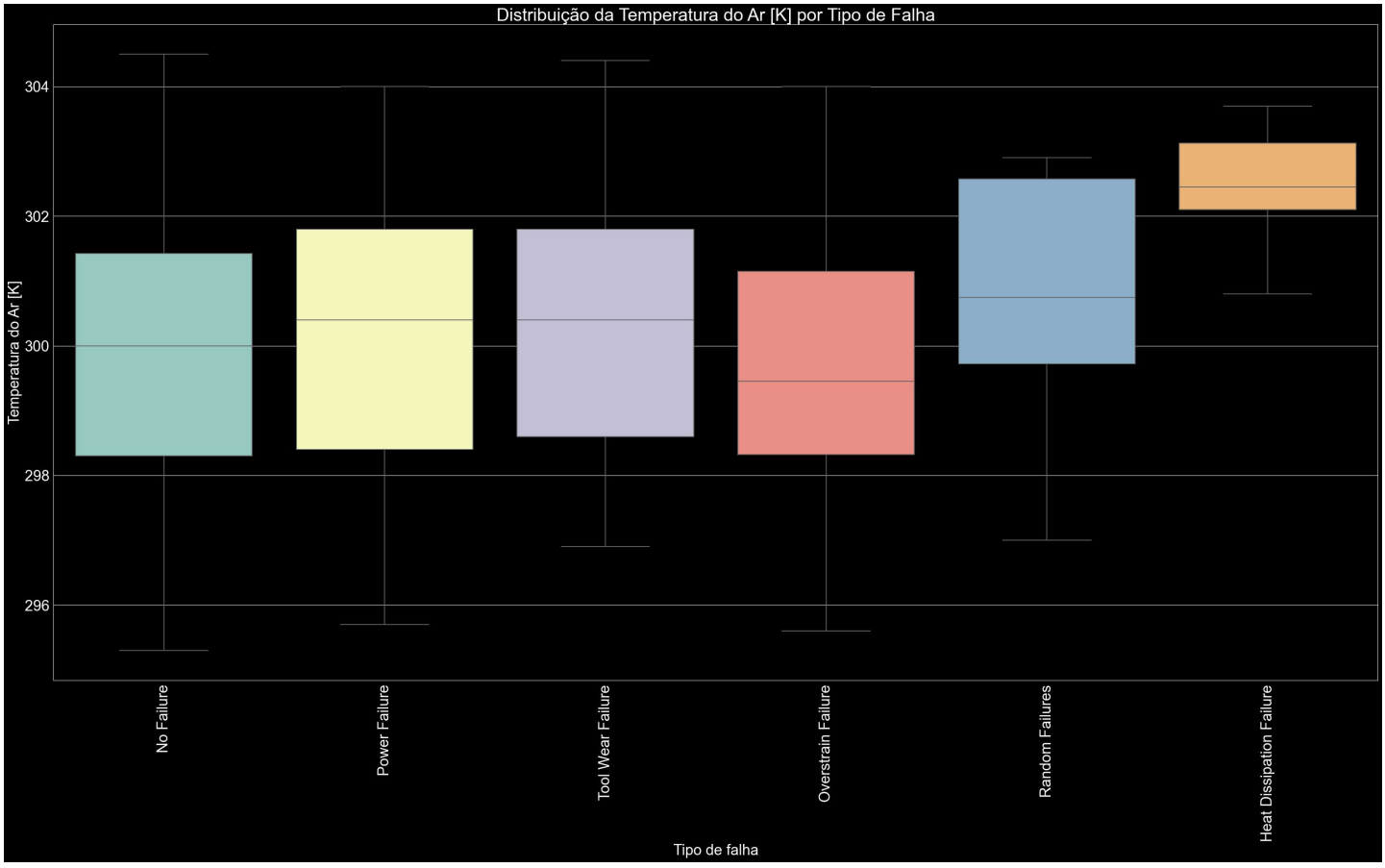




Análise de variação de temperatura do ar [k] por tipo de falha:

In [90]:

```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='failure_type', y='air_temperature_k')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição da Temperatura do Ar [K] por Tipo de Falha", fontsize=30)
plt.xlabel("Tipo de falha", fontsize=25)
plt.ylabel("Temperatura do Ar [K]", fontsize=25)
plt.show()
```

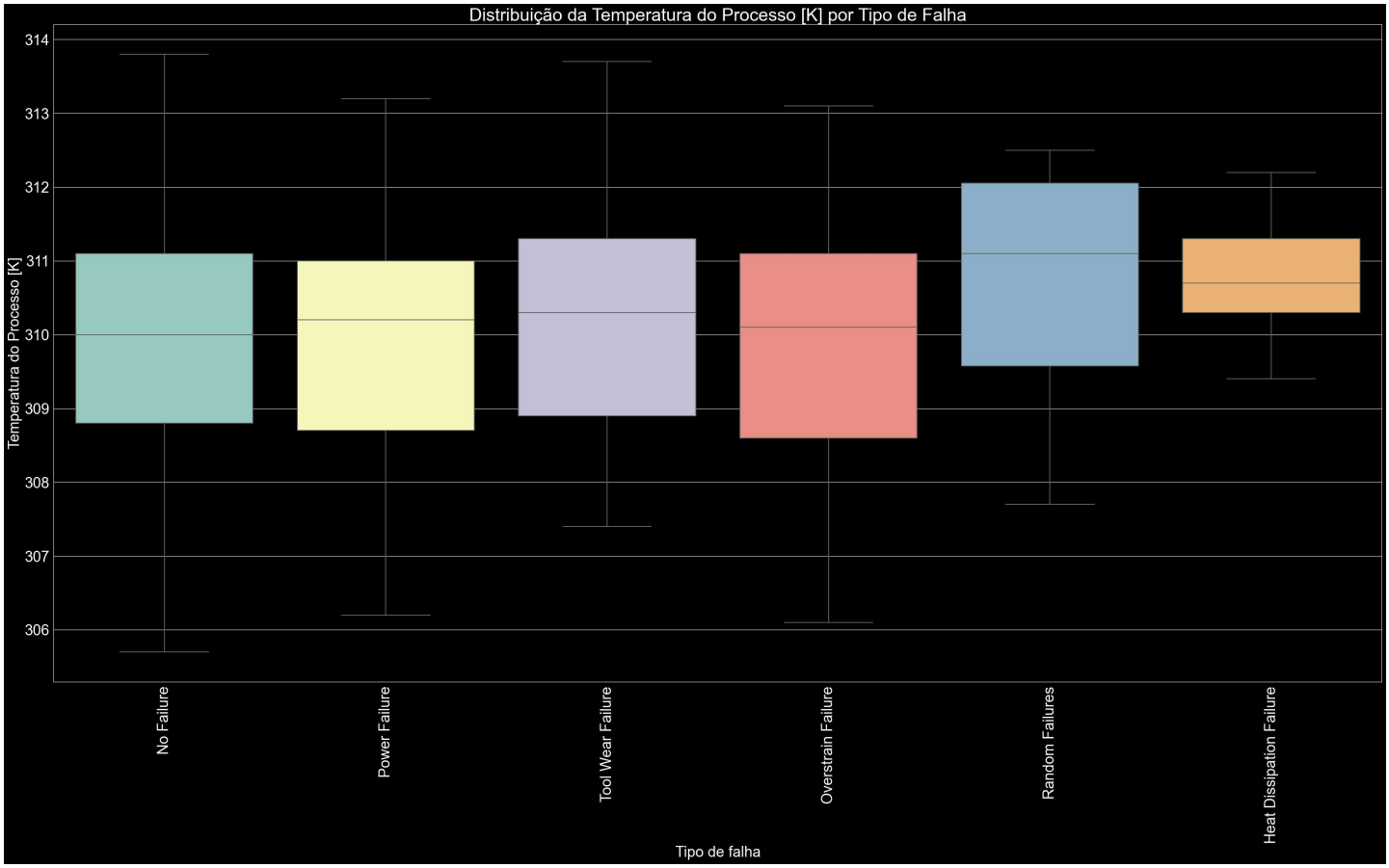


Análise de variação de temperatura do processo [k] por tipo de falha:

In [91]:

```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='failure_type', y='process_temperature_k')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição da Temperatura do Processo [K] por Tipo de Falha", fontsize=30)
```

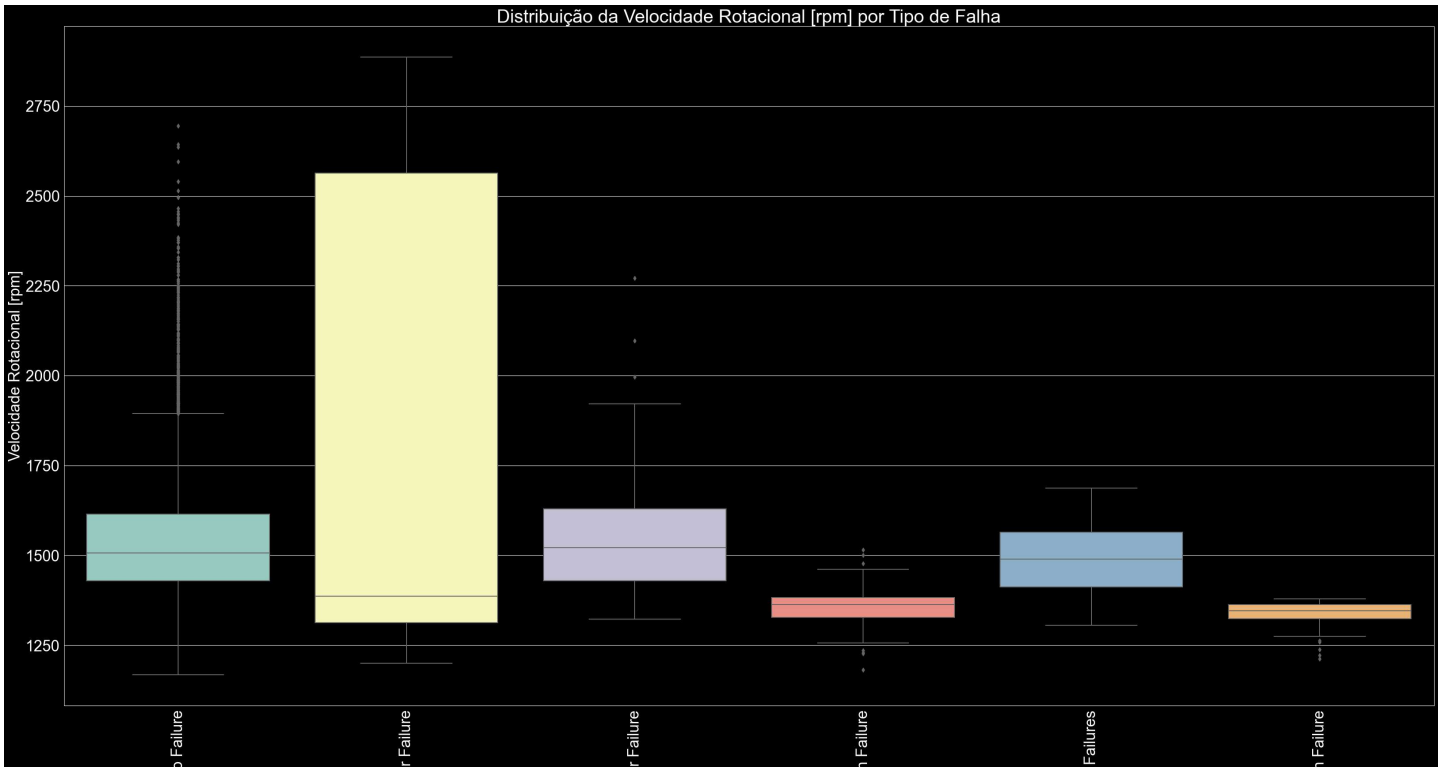
```
plt.xlabel("Tipo de falha", fontsize=25)
plt.ylabel("Temperatura do Processo [K]", fontsize=25)
plt.show()
```



Análise da variação da velocidade rotacional [rpm] por tipo de falha:

In [92]:

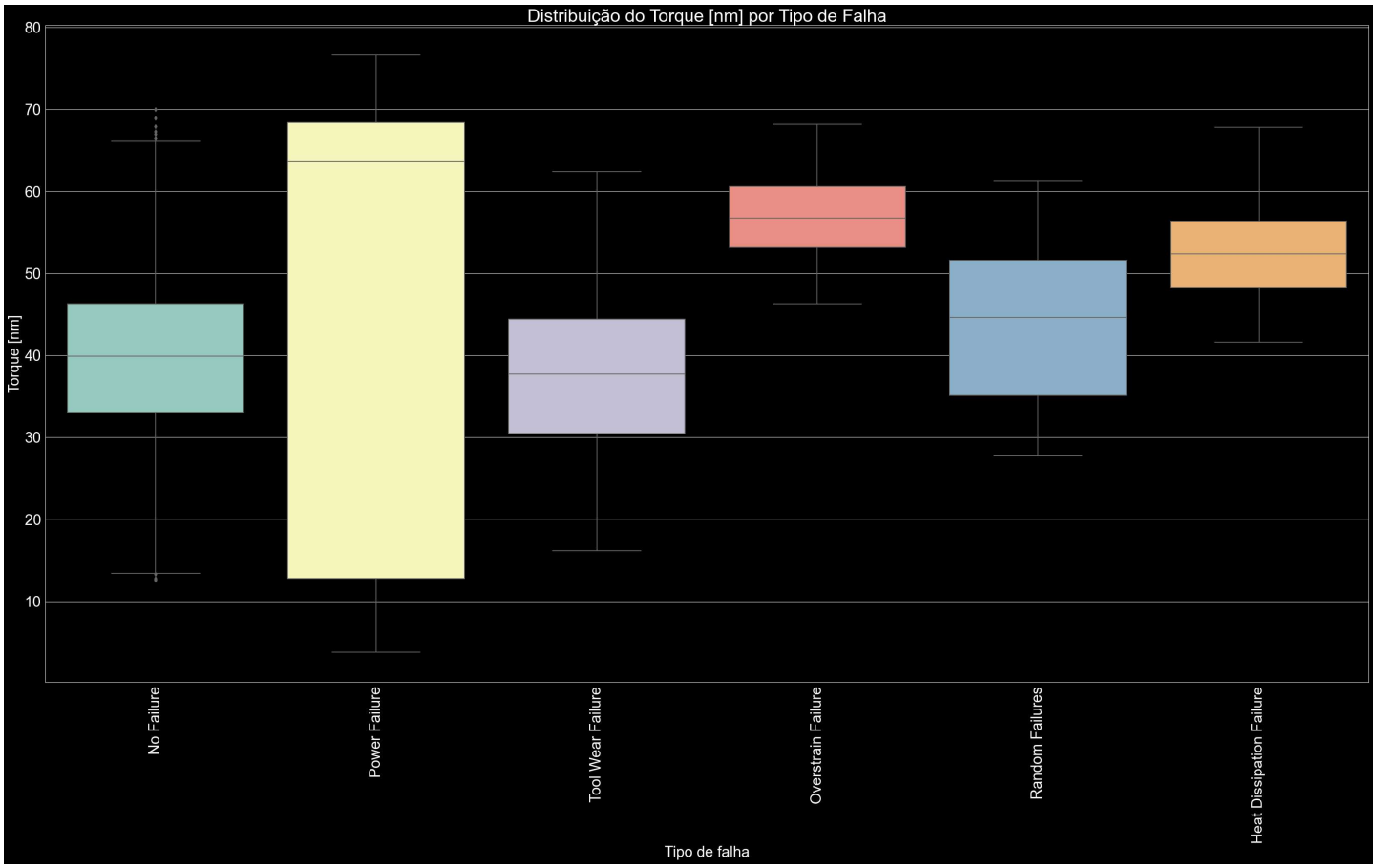
```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='failure_type', y='rotational_speed_rpm')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição da Velocidade Rotacional [rpm] por Tipo de Falha", fontsize=30)
plt.xlabel("Tipo de falha", fontsize=25)
plt.ylabel("Velocidade Rotacional [rpm]", fontsize=25)
plt.show()
```



Análise da variação do torque [nm] por tipo de falha:

In [93]:

```
plt.figure(figsize=(40, 20))
sns.boxplot(data=df, x='failure_type', y='torque_nm')
plt.xticks(rotation=90, fontsize=25)
plt.yticks(fontsize=25)
plt.title("Distribuição do Torque [nm] por Tipo de Falha", fontsize=30)
plt.xlabel("Tipo de falha", fontsize=25)
plt.ylabel("Torque [nm]", fontsize=25)
plt.show()
```



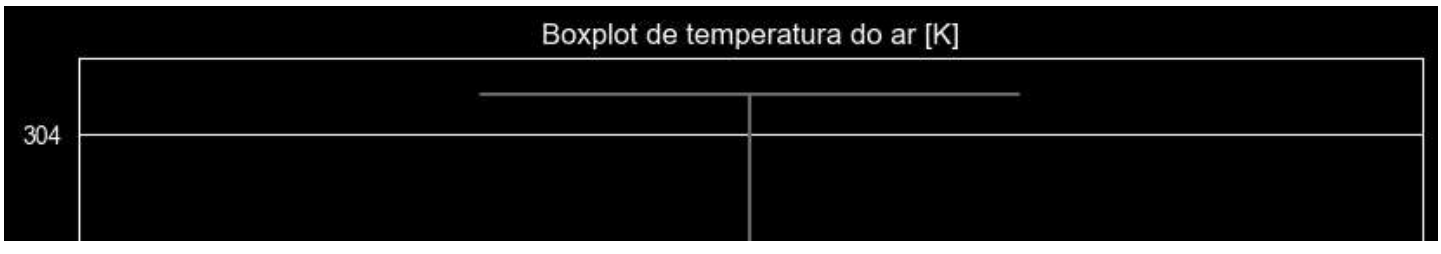
Com essas análises, se pode observar que todos as variáveis influenciam na falha da máquina.

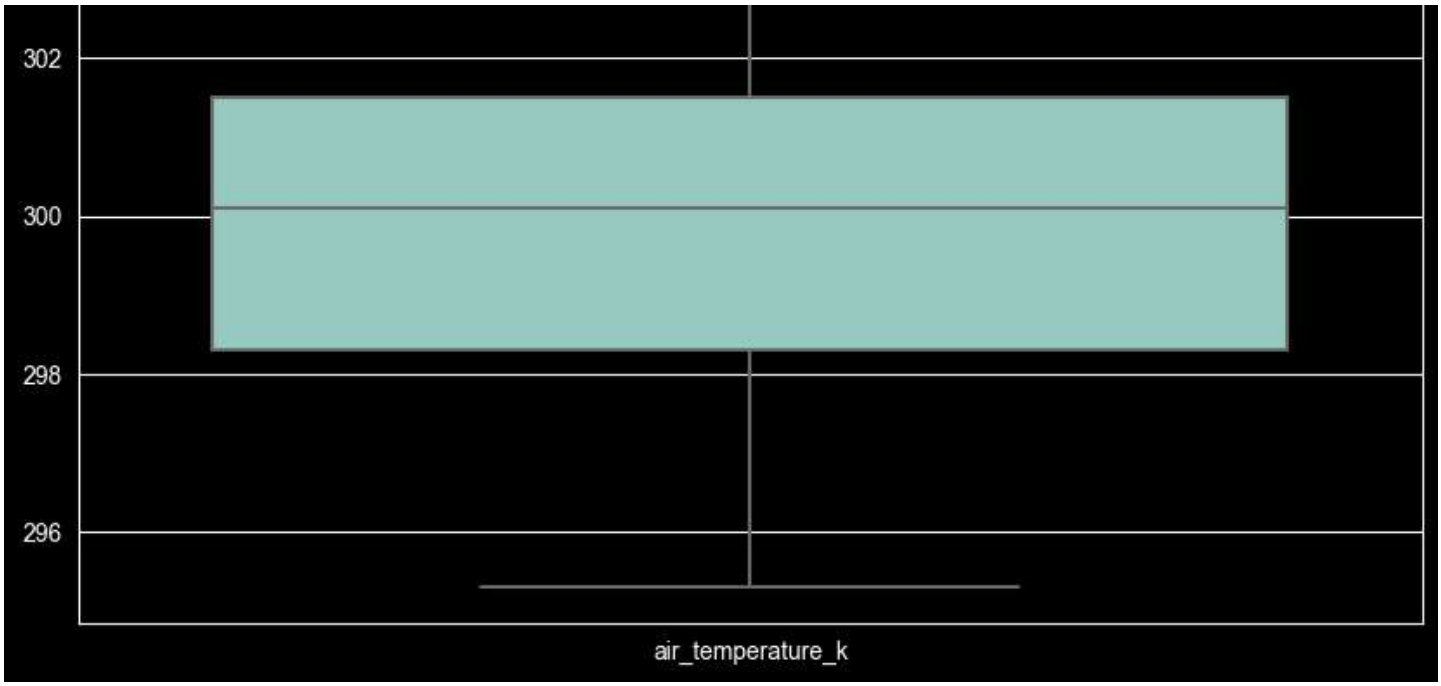
Análise de Outliers

Análise de Outliers de temperatura do ar

In [94]:

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[["air_temperature_k"]])
plt.title("Boxplot de temperatura do ar [K]")
plt.show()
```

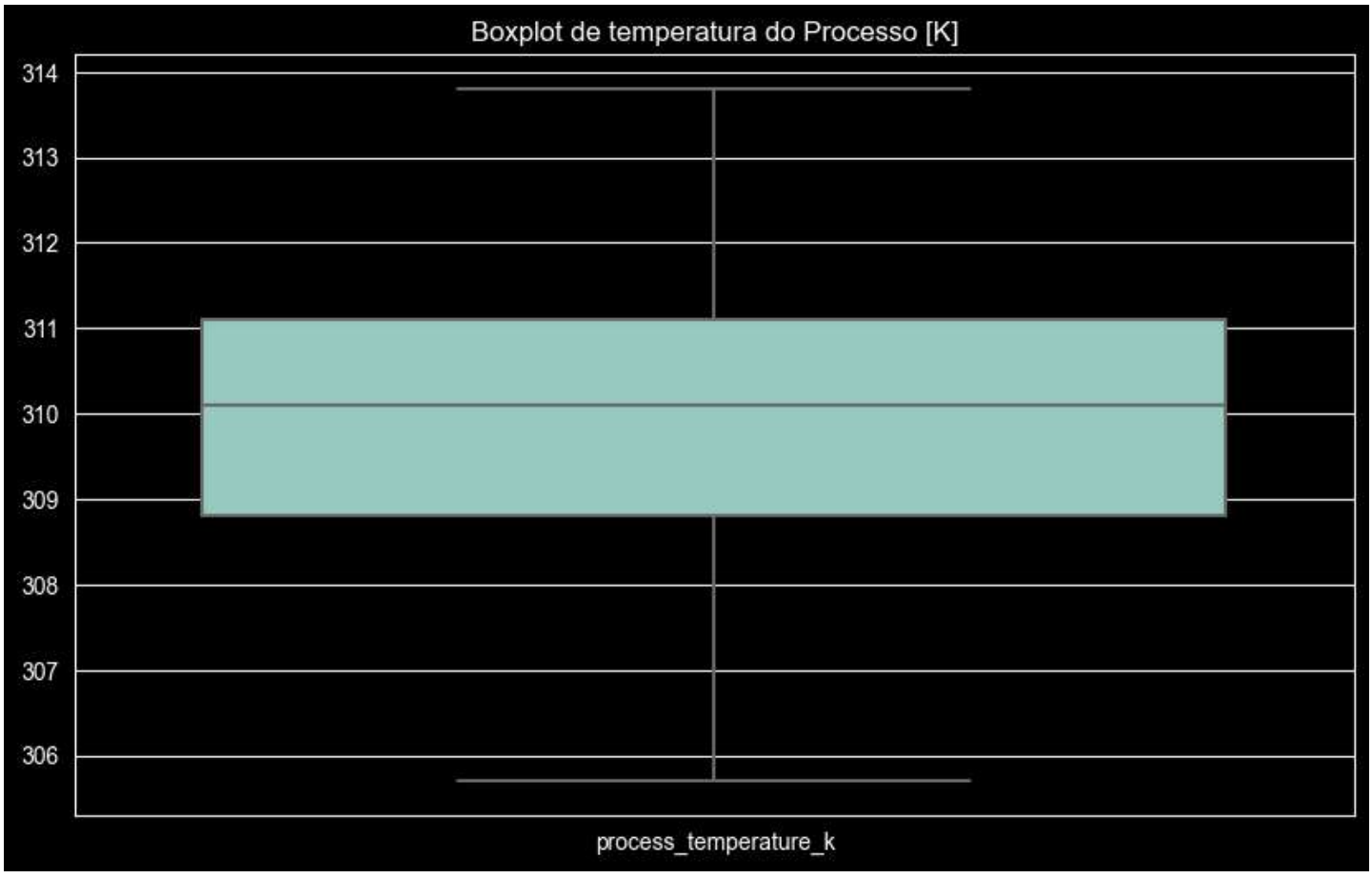




Análise de Outliers de temperatura do processo

In [95]:

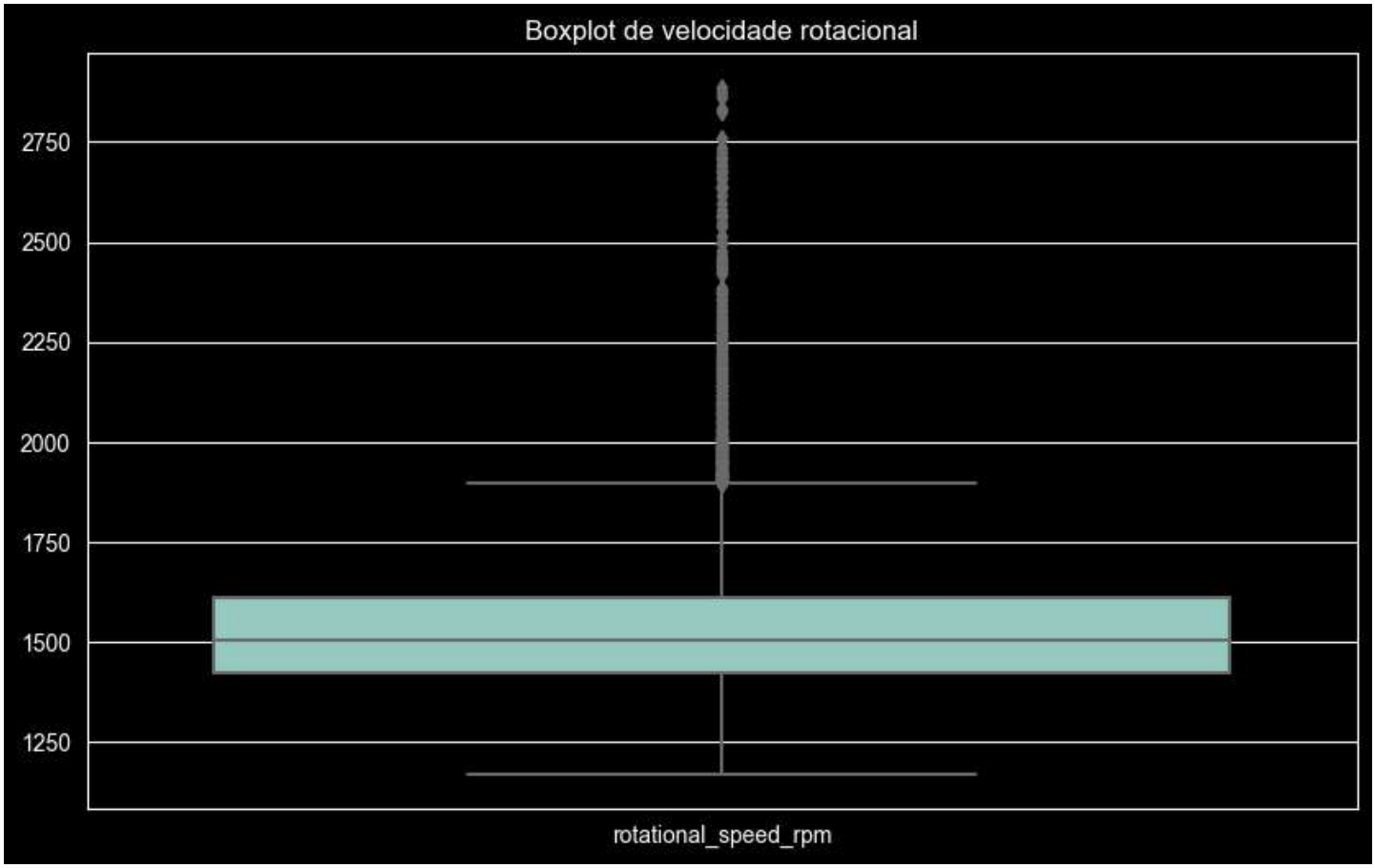
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[["process_temperature_k"]])
plt.title("Boxplot de temperatura do Processo [K]")
plt.show()
```



Análise de Outliers de velocidade rotacional

In [96]:

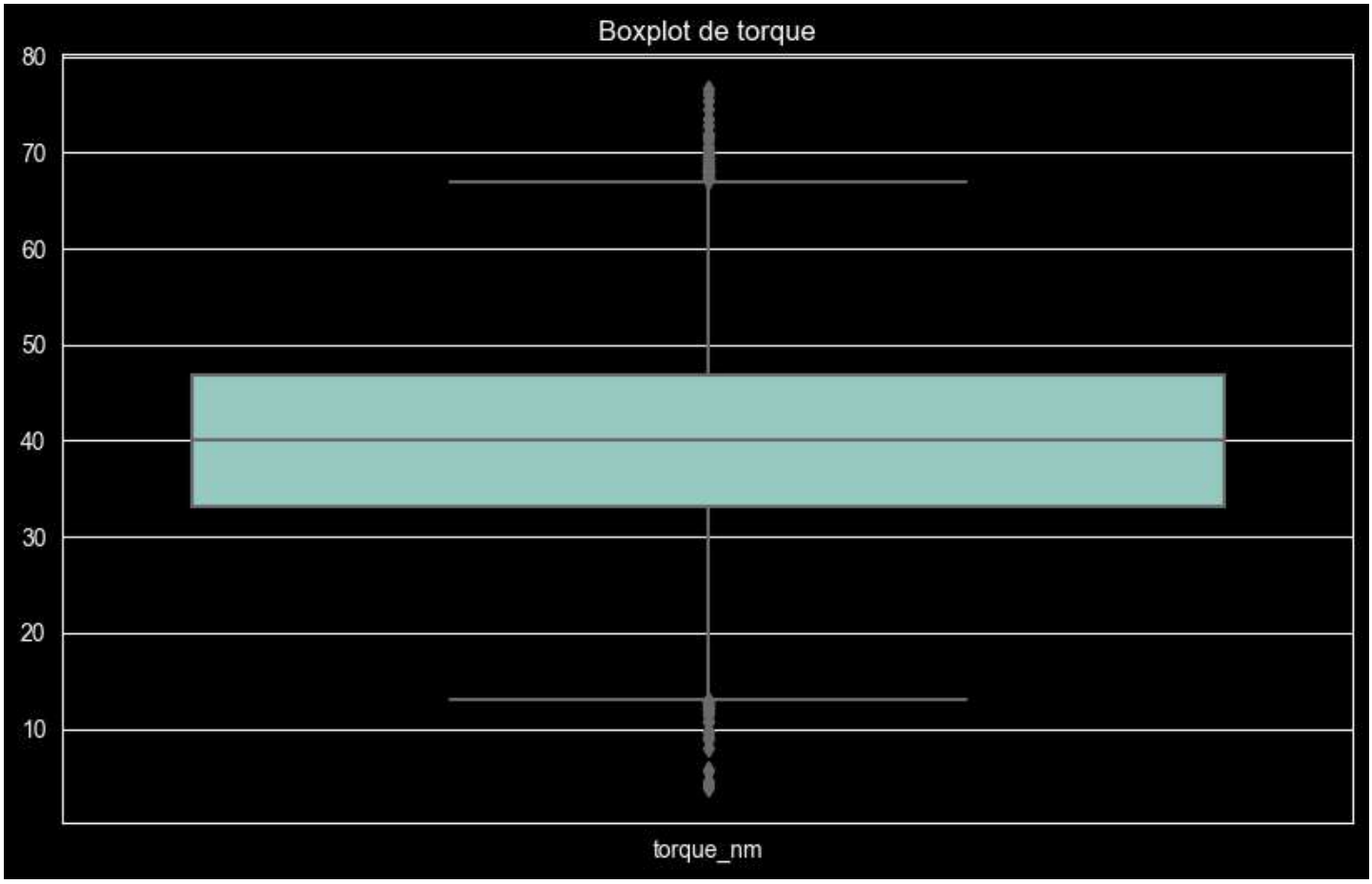
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[["rotational_speed_rpm"]])
plt.title("Boxplot de velocidade rotacional")
plt.show()
```



Análise de Outliers de torque

In [97]:

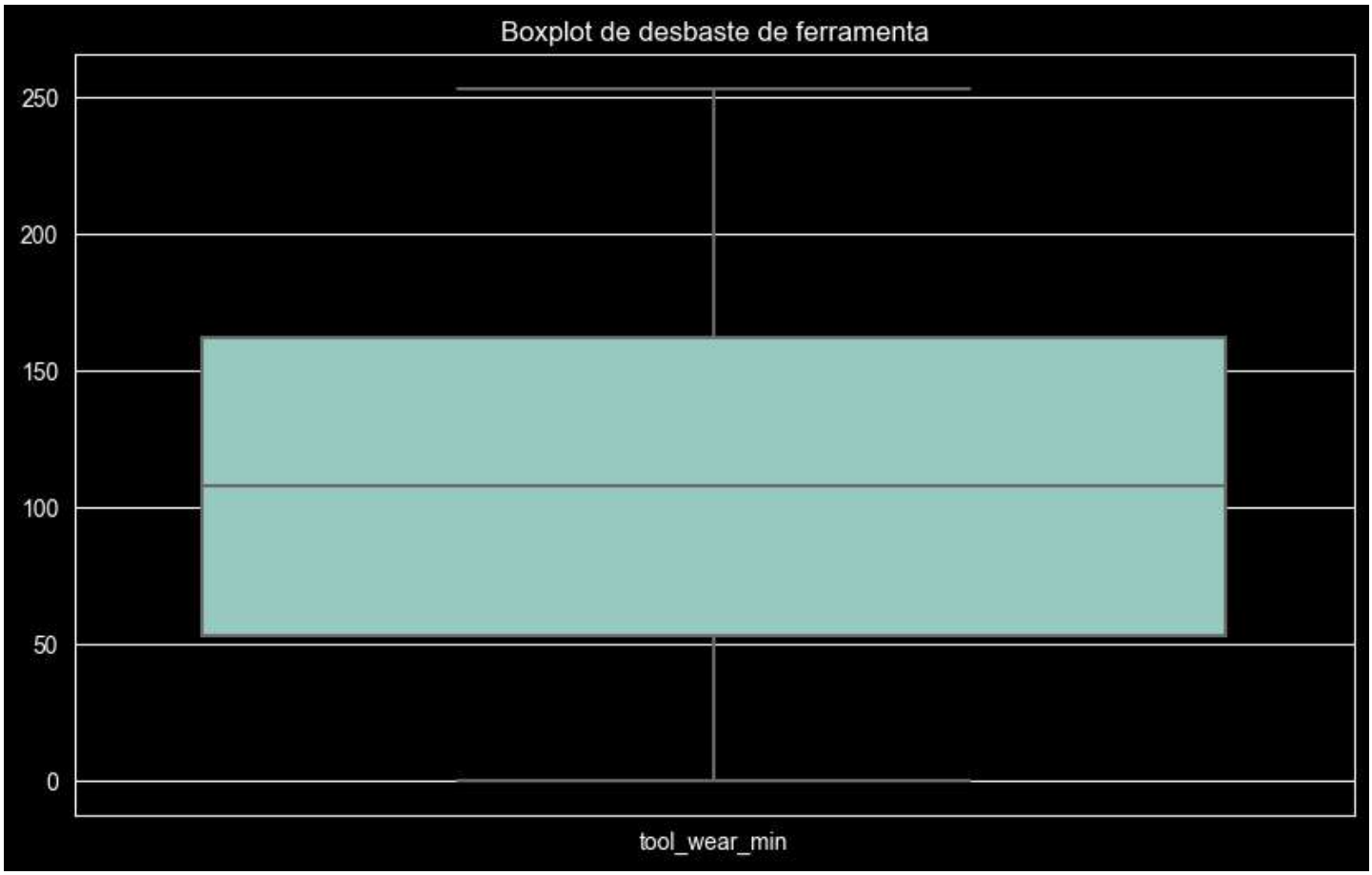
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[["torque_nm"]])
plt.title("Boxplot de torque")
plt.show()
```



Análise de Outliers de desbaste de ferramenta

In [98]:

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[["tool_wear_min"]])
plt.title("Boxplot de desbaste de ferramenta")
plt.show()
```

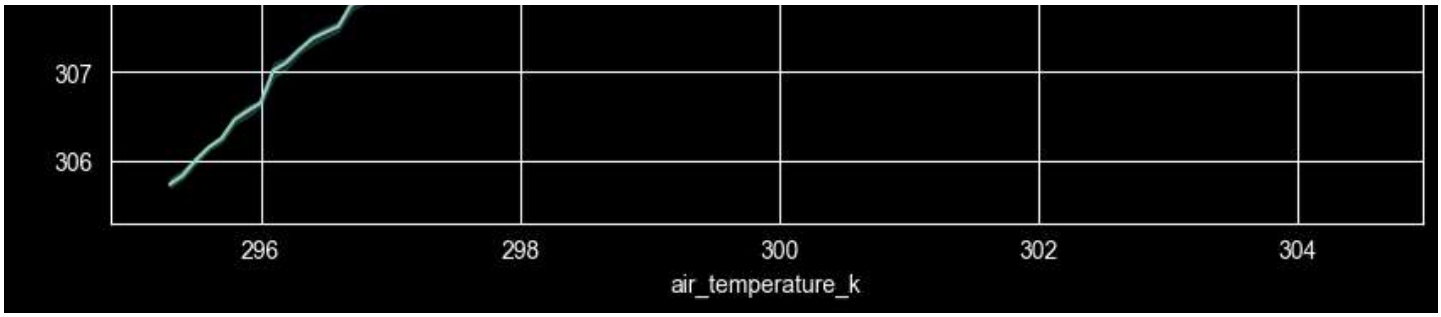


Visualização de padrões e tendências

Tendência entre temperatura do ar e do processo

In [99]:

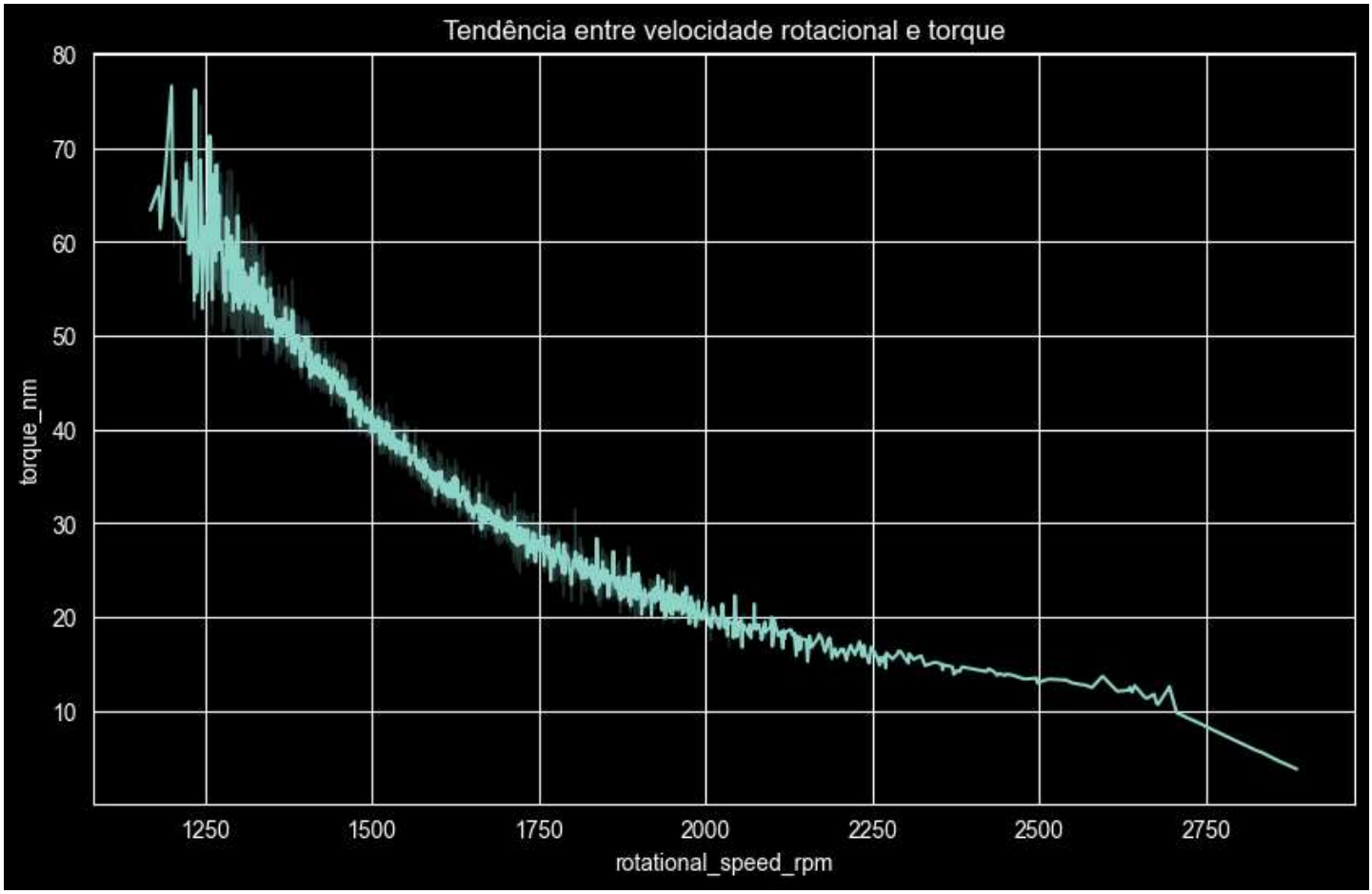
```
plt.figure(figsize=(10, 6))
sns.lineplot(data=df, x="air_temperature_k", y="process_temperature_k")
plt.title("Tendência entre temperatura do ar e do processo")
plt.show()
```



Tendência entre velocidade rotacional e torque

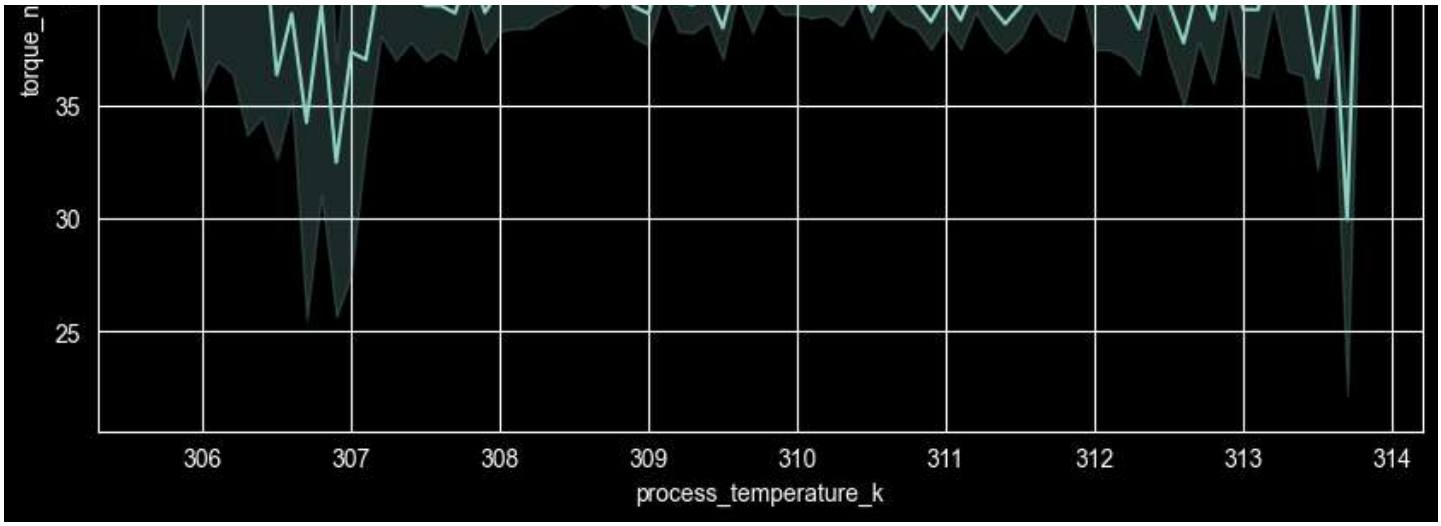
In [100]:

```
plt.figure(figsize=(10, 6))
sns.lineplot(data=df, x="rotational_speed_rpm", y="torque_nm")
plt.title("Tendência entre velocidade rotacional e torque")
plt.show()
```



In [101]:

```
plt.figure(figsize=(10, 6))
sns.lineplot(data=df, x="process_temperature_k", y="torque_nm")
plt.title("Tendência entre temperatura do processo e torque")
plt.show()
```



Análise por grupo:

In [102]:

```
# Selecionar apenas as colunas numéricas
numeric_cols = ['air_temperature_k', 'process_temperature_k', 'rotational_speed_rpm', 'to
rque_nm', 'tool_wear_min']

# Criar um novo DataFrame com as colunas numéricas
numeric_df = df[numeric_cols]

# Agrupar por "target" e calcular as médias
print("Médias por grupo:")
numeric_df.groupby(df["target"]).mean().style.applymap(apply_background_color)
```

Médias por grupo:

Out[102]:

target	air_temperature_k	process_temperature_k	rotational_speed_rpm	torque_nm	tool_wear_min
0	299.973999	309.995570	1540.260014	39.629655	106.693717
1	300.886431	310.290265	1496.486726	50.168142	143.781711

Método de augumentation

Como pode ser visto abaixo, a incidência de não falha é muito superior a de falha no dataset:

In [103]:

```
incidencia_nao_falha = (len(df[df['target'] == 0])/len(df))*100
incidencia_falha = (len(df[df['target'] == 1])/len(df))*100

print("O percentual de não falha no dataset é de: ",incidencia_nao_falha,"%, enquanto de
falha é apenas: ",incidencia_falha,"%.")
```

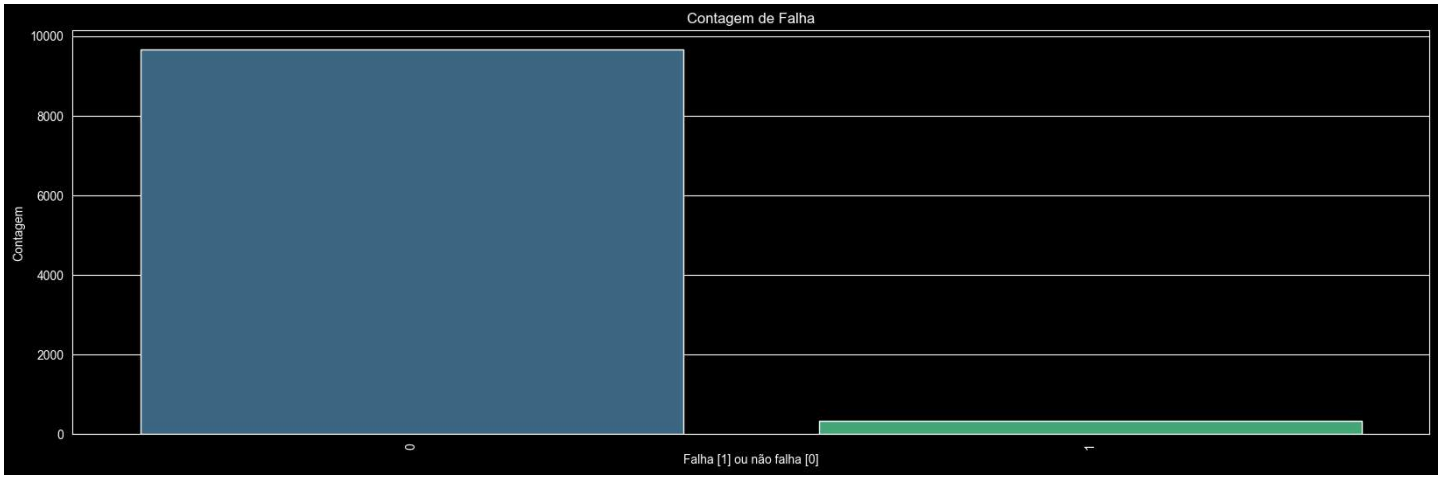
O percentual de não falha no dataset é de: 96.61 %, enquanto de falha é apenas: 3.39 %.

incidencia_falha = len(df['target'] == '0')

In [104]:

```
failure_counts = df['target'].value_counts()

plt.figure(figsize=(20, 6))
sns.barplot(x=failure_counts.index, y=failure_counts.values, palette='viridis')
plt.title('Contagem de Falha')
plt.xlabel('Falha [1] ou não falha [0]')
```

```
plt.ylabel('Contagem')
plt.xticks(rotation=90)
plt.show()
```



Para o método de augumentation será usada a geração de dados sintética SMOTE, utilizando a biblioteca imbalanced-learn.

Synthetic Data Generation (Geração de Dados Sintéticos): Utilizar técnicas de geração de dados sintéticos para criar novas amostras da classe minoritária. Um exemplo popular é o SMOTE (Synthetic Minority Over-sampling Technique), que cria novas amostras interpolando pontos entre amostras existentes da classe minoritária.

Para tal, a coluna categórica 'failure_type' será desconsiderada. Visto que será analisada a probabilidade de falha da máquina, e não o seu tipo de falha.

In [105]:

```
# Converter valores categóricos da coluna 'target' em valores numéricos (0 e 1)
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['target'])

# Separar as features (X)
X = df.drop(columns=['target', 'failure_type'])

# Aplicar SMOTE para gerar dados sintéticos
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

resampled_df = pd.DataFrame(X_resampled, columns=X.columns)
resampled_df['target'] = y_resampled
resampled_df['failure_type'] = df['failure_type']  # Adicionar a coluna failure_type nov
amente

resampled_df.head(5)
```

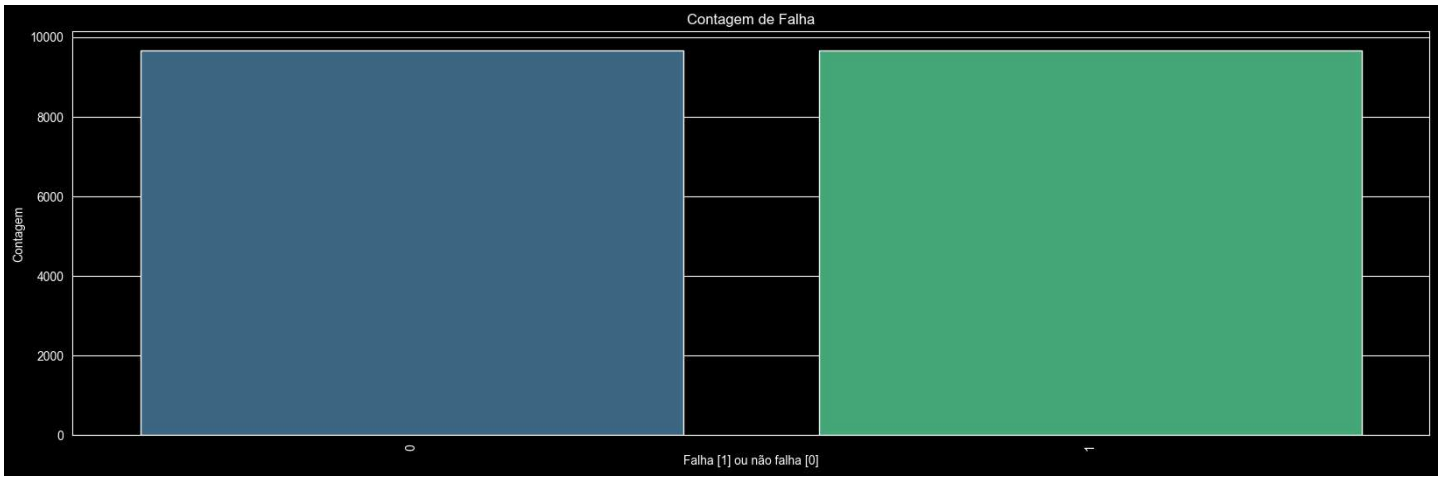
Out[105]:

	air_temperature_k	process_temperature_k	rotational_speed_rpm	torque_nm	tool_wear_min	target	failure_type
0	298.1	308.6	1551.0	42.8	0.0	0	No Failure
1	298.2	308.7	1408.0	46.3	3.0	0	No Failure
2	298.1	308.5	1498.0	49.4	5.0	0	No Failure
3	298.2	308.6	1433.0	39.5	7.0	0	No Failure
4	298.2	308.7	1408.0	40.0	9.0	0	No Failure

In [106]:

```
failure_counts = resampled_df['target'].value_counts()

plt.figure(figsize=(20, 6))
sns.barplot(x=failure_counts.index, y=failure_counts.values, palette='viridis')
plt.title('Contagem de Falha')
```

```
plt.xlabel('Falha [1] ou não falha [0]')
plt.ylabel('Contagem')
plt.xticks(rotation=90)
plt.show()
```



In [107]:

```
print('O tamanho do dataset original é: ',len(df),', já, o tamanho do dataset balanceado é: ',len(resampled_df))
```

```
O tamanho do dataset original é:  10000 , já, o tamanho do dataset balanceado é:  19322
```

Divisão do dataset em treino e teste para aplicação do modelo de machine learning

O dataset será divido em 20% de teste e 80% de treino

In [108]:

```
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=21)

print('O tamanho do conjunto de treino é: ',len(X_train))
print('O tamanho do conjunto de teste é: ',len(X_test))
```

```
O tamanho do conjunto de treino é:  15457
O tamanho do conjunto de teste é:  3865
```

Aplicação dos Modelos

Aplicação do modelo Random Forest

In [109]:

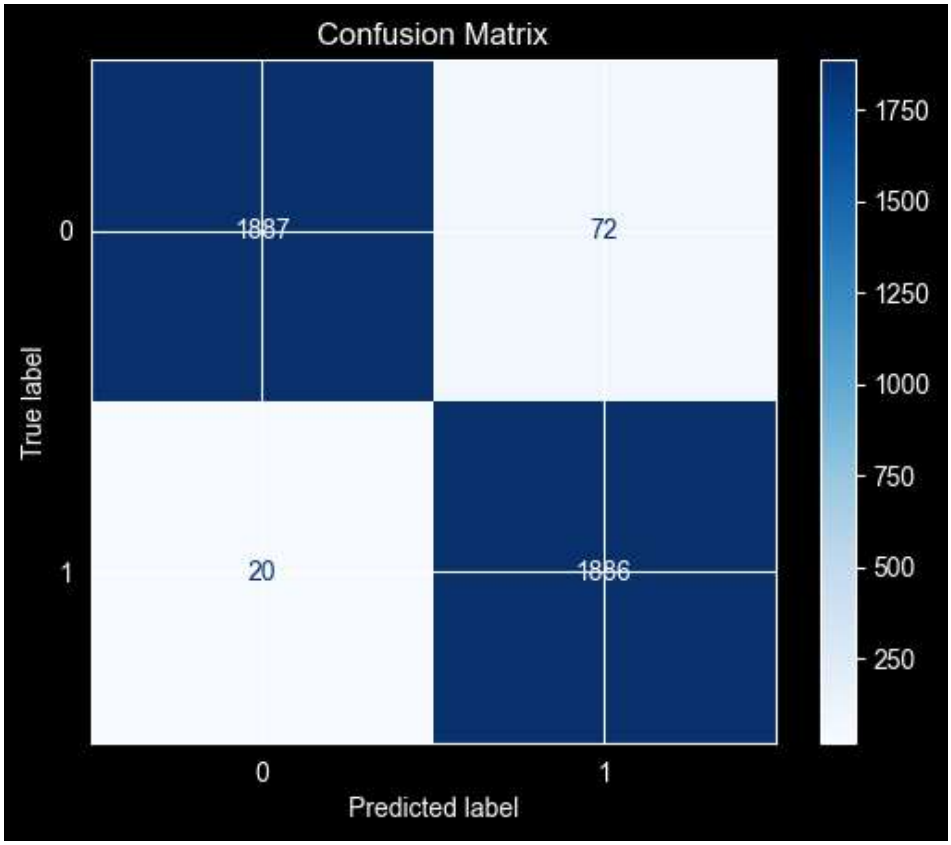
```
# Criar e treinar o modelo Random Forest
random_forest = RandomForestClassifier(n_estimators=100, random_state=21)
random_forest.fit(X_train, y_train)

y_pred_rf = random_forest.predict(X_test)

random_forest_train_accuracy = round(random_forest.score(X_train, y_train) * 100, 2)
random_forest_accuracy = round(accuracy_score(y_test, y_pred_rf) * 100, 2)

print("Acurácia:", random_forest_accuracy, "%")

disp = ConfusionMatrixDisplay.from_predictions(
    y_test,
    y_pred_rf,
    labels=np.unique(y_test),
    display_labels=np.unique(y_test),
```

```
    cmap=plt.cm.Blues,
)
disp.ax_.set_title('Confusion Matrix')
```

```
Acurácia: 97.62 %
```

Out[109]:

```
Text(0.5, 1.0, 'Confusion Matrix')
```



Como resultados gerais da aplicação do modelo Random Forest, tem-se a partir da matriz de confusão que:

- **Nos casos de falha o modelo conseguiu predizer corretamente 1886 resultados e, falhou na predição de 20 resultados;**
- **Nos casos de não falha o modelo conseguiu predizer corretamente 1887 resultados e, falhou na predição de 72 resultados.**

Já, a acurácia do modelo foi de 97,62%

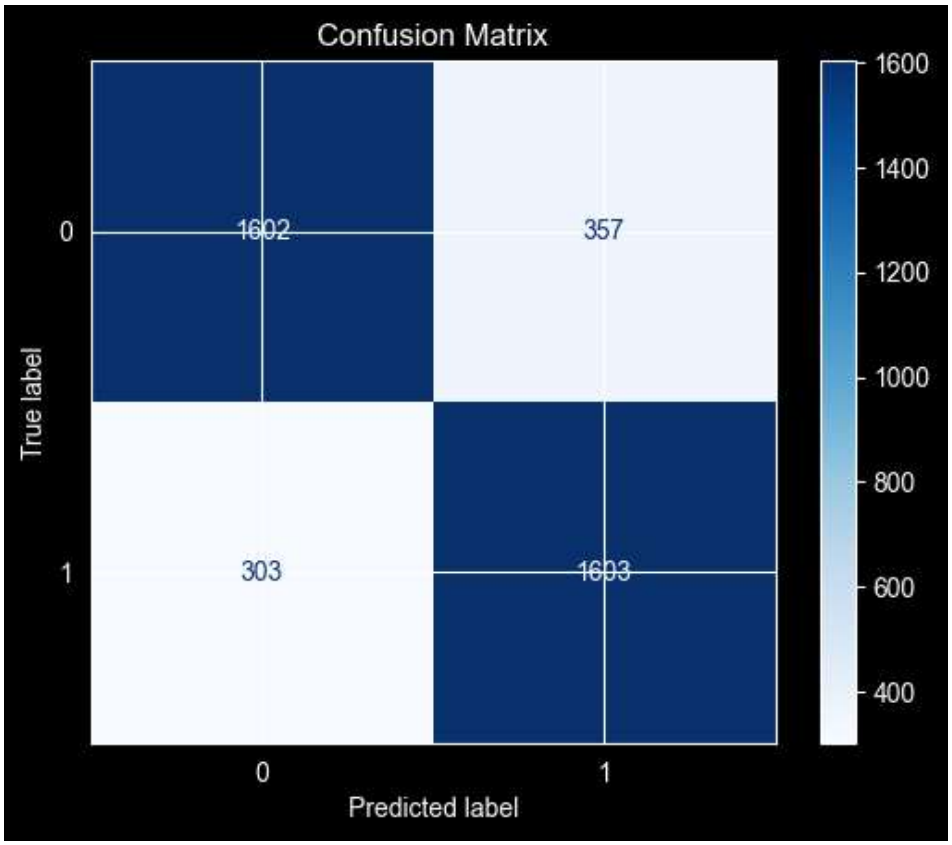
Aplicação do modelo SVM:

In [110]:

```
# Create and train the model SVM
svm_model = svm.SVC(kernel="linear", C=0.01)
svm_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_svm = svm_model.predict(X_test)

# Evaluate the model
accuracy = np.mean(y_pred_svm == y_test)
print("Acurácia:", round(accuracy * 100, 2), "%")

# Plot the confusion matrix with all class labels
disp = ConfusionMatrixDisplay.from_predictions(
    y_test,
    y_pred_svm,
    labels=np.unique(y_test),  # Definindo todos os rótulos de classe possíveis
    display_labels=np.unique(y_test),  # Usando os rótulos verdadeiros do conjunto de tes
te
    cmap=plt.cm.Blues,
```

```
)
disp.ax_.set_title('Confusion Matrix')
#print("Confusion Matrix:")
#print(disp.confusion_matrix)

plt.show()
```

```
Acurácia: 82.92 %
```



Como resultados gerais da aplicação do modelo Suport Vector Machine, tem-se a partir da matriz de confusão que:

- **Nos casos de falha o modelo conseguiu predizer corretamente 1603 resultados e, falhou na predição de 303 resultados;**
- **Nos casos de não falha o modelo conseguiu predizer corretamente 1602 resultados e, falhou na predição de 357 resultados.**

Já, a acurácia do modelo foi de 82,92%

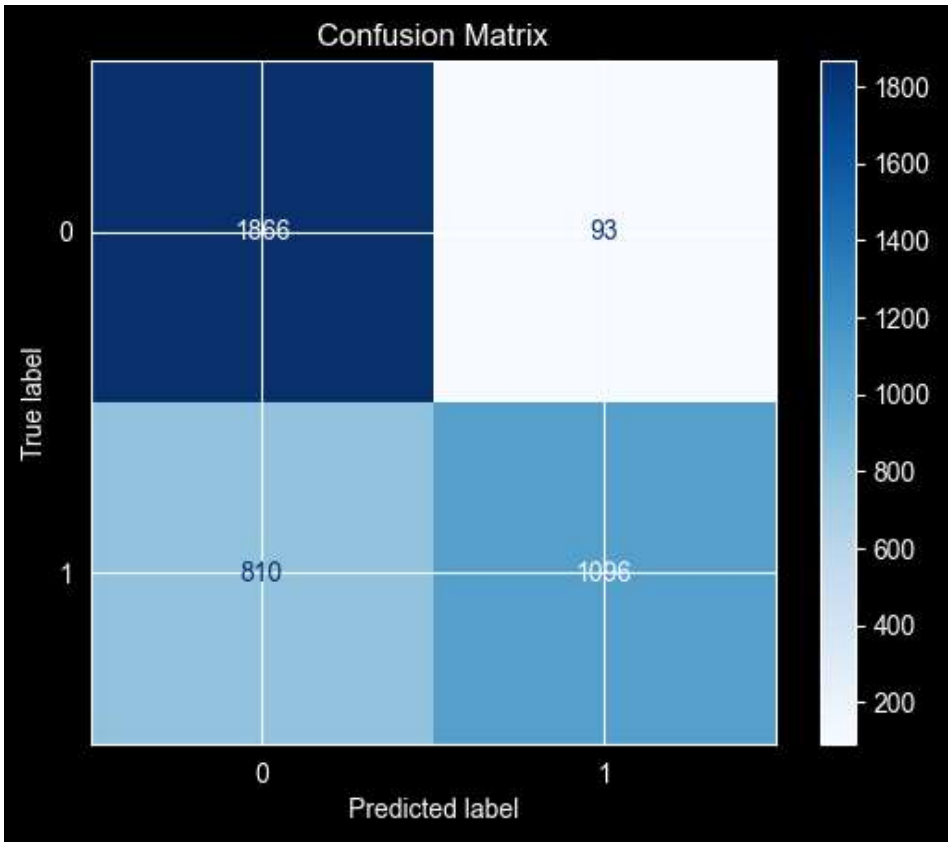
Aplicação do modelo MLP:

In [111]:

```
# Create and train the MLP model
mlp_model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=1000, random_state=21)
mlp_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_mlp = mlp_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred_mlp)
print("Acurácia:", round(accuracy * 100, 2), "%")

# Plot the confusion matrix with all class labels
disp = ConfusionMatrixDisplay.from_estimator(
    mlp_model,
    X_test,
    y_test,
    labels=np.unique(y_test),
    display_labels=np.unique(y_test),
```

```
    cmap=plt.cm.Blues,
)
disp.ax_.set_title('Confusion Matrix')
plt.show()
```

```
Acurácia: 76.64 %
```



Como resultados gerais da aplicação do modelo MLP, tem-se a partir da matriz de confusão que:

- **Nos casos de falha o modelo conseguiu predizer corretamente 1096 resultados e, falhou na predição de 810 resultados;**
- **Nos casos de não falha o modelo conseguiu predizer corretamente 1866 resultados e, falhou na predição de 93 resultados.**

Já, a acurácia do modelo foi de 76,64%