UNIVERSIDADE FEDERAL FLUMINENSE
GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

CIRO MAGALHÃES DA ROSA

APLICAÇÃO DE *MACHINE LEARNING* NA IDENTIFICAÇÃO DAS CAUSAS DE FALHAS EM MÁQUINAS PARA AUXÍLIO NA GESTÃO DA MANUTENÇÃO

NITERÓI - RJ

2022

UNIVERSIDADE FEDERAL FLUMINENSE

CIRO MAGALHÃES DA ROSA

APLICAÇÃO DE MACHINE LEARNING NA IDENTIFICAÇÃO DAS CAUSAS DE FALHAS EM MÁQUINAS PARA AUXÍLIO NA GESTÃO DA MANUTENÇÃO

Trabalho de Conclusão de Curso apresentado ao Corpo Docente do Departamento de Engenharia de Produção da Escola de Engenharia da Universidade Federal Fluminense, como parte dos requisitos necessários à obtenção do título de Engenheira(o) de Produção.

Orientador(a):
Prof(a). Dr(a) Valdecy Pereira

NITERÓI - RJ

2022

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

FICHA CATALOGRÁFICA GERADA EM:
http://www.bibliotecas.uff.br/bee/ficha-catalografica

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

R788a Rosa, Ciro Magalhães da
Aplicação de machine learning na identificação das causas de falhas em máquinas para auxílio na gestão da manutenção / Ciro Magalhães da Rosa. - 2022.
65 f.

Orientador: Valdecy Pereira.
Trabalho de Conclusão de Curso (graduação)-Universidade Federal Fluminense, Escola de Engenharia, Niterói, 2022.

1. Gestão da manutenção. 2. Aprendizado de máquina. 3. Falha de equipamentos. 4. Algoritmo c4.5. 5. Produção intelectual. I. Pereira, Valdecy, orientador. II. Universidade Federal Fluminense. Escola de Engenharia. III. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

UNIVERSIDADE FEDERAL FLUMINENSE

CIRO MAGALHÃES DA ROSA

APLICAÇÃO DE MACHINE LEARNING NA IDENTIFICAÇÃO DAS CAUSAS DE FALHAS EM MÁQUINAS PARA AUXÍLIO NA GESTÃO DA MANUTENÇÃO

Trabalho de Conclusão de Curso apresentado ao Corpo Docente do Departamento de Engenharia de Produção da Escola de Engenharia da Universidade Federal Fluminense, como parte dos requisitos necessários à obtenção do título de Engenheira(o) de Produção.

Aprovado em 16 de dezembro de 2022, com nota 8,00 (oito), pela banca examinadora.

BANCA EXAMINADORA

_____

Prof(a). Dr(a) Valdecy Pereira - Orientador
UFF

_____

Prof(a). Dr(a) Marcos Costa Roboredo – Membro Convidado
UFF

_____

Prof(a). Dr(a). José Kimio Ando – Membro Convidado
UFF

NITERÓI - RJ

2022

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Ana Beatriz e Antonio José, por todo esforço e energia que depositaram na minha criação, sempre priorizando minha educação e me ensinando o poder transformador que ela tem. O que se fez fundamental para a minha evolução como pessoa e me deu as capacidades necessárias par me graduar como engenheiro de produção. Sou grato também aos meus irmãos, Pedro, Paloma, Marina, e Tito, que aos tapas e beijos encheram a minha vida de alegria e boas memórias. Aos meus avós, Terezinha, João, Helena e José, que sempre me apoiaram e contribuíram para que eu fosse em direção aos meus sonhos.

A todos os meus grandes amigos que passaram por essa longa jornada de crescimento. Pude absorver um pedaço de cada um com quem criei laços e espero ter deixado um pouco de mim também. Foram inúmeros os aprendizados e momentos enriquecedores que tivemos e tenho a consciência de que eles foram pessoas indispensáveis para que eu trilhasse o meu caminho até o fim desta graduação, sendo assim, partes desta conquista. Em especial a minha companheira de vida e namorada, Teresa, que me incentivou a ir além, deu suporte nos momentos mais difíceis e dividiu a felicidade de cada passo neste processo. Foi de suma importância ter uma pessoa como ela ao meu lado.

Por fim, gostaria de demostrar a minha gratidão a todos os profissionais e educadores que me passaram seus ensinamentos e agregaram na minha formação como indivíduo e profissional. Sobretudo ao professor, Valdecy, que topou o desafio de orientar um aluno que buscava realizar o seu primeiro projeto de ciência de dados voltado para engenharia. A todo tempo ele se colocou a disposição e solicito para me ajudar, além de sempre me incentivar a dar o melhor de mim para demostrar todo o conhecimento que adquiri ao longo da graduação. Agradecer também, aos professores José Kimio e Marcos Roboredo por toparem participar da banca de avaliação do trabalho e fazerem parte deste momento final do curso.

RESUMO

O presente trabalho é um estudo sobre a capacidade que o *machine learning* possui para contribuir na gestão da manutenção de máquinas. É introduzido por um histórico da evolução de como ela vem sendo realizada na indústria até os dias atuais. Explica-se, nesse contexto, que tanto a identificação quanto a compreensão das causas que levam uma máquina a falhar, podem otimizar a produção no cenário industrial. Atualmente, com o avanço constante da inteligência artificial, o *machine learning* se tornou uma importante ferramenta nessa investigação. Com isso, o objetivo do estudo é entender como um modelo de *machine learning* pode auxiliar nas descobertas dos porquês de uma determinada falha acontecer. Para tanto, foi utilizada uma metodologia de projeto para *data science* na qual houve uma exploração de uma base de dados criada artificialmente para estudos de manutenção. A partir da investigação da base, foi aplicado o modelo de aprendizado de máquina C4.5, em que é gerada uma árvore de decisão com as condições para que ocorra uma determinada falha no equipamento. Em seguida, é feita uma análise da árvore obtida para explicar como ela é capaz de contribuir para a definição das razões que levam ao defeito das máquinas. Dessa forma, a discussão tem como base a comparação das condições descritas na árvore junto aos critérios previamente conhecidos que geraram a base de dados de forma artificial. Assim, foi possível avaliar os resultados obtidos pelo modelo e entender as suas capacidades. Por fim, o trabalho se encerra concluindo as possibilidades que este estudo proporciona para auxílio na gestão da manutenção.

Palavras-Chave: gestão da manutenção, aprendizado de máquina, falha de equipamentos, algoritmo C4.5

ABSTRACT

The present work is a study on the capacity that machine learning has to contribute to the management of machine maintenance. It is introduced by a review of the evolution of how this has been carried out in the industry to nowadays. In this framework, it is explained that both the identification and understanding of the reasons that cause a machine to fail can help optimize the production in the industrial scenario. Currently, with the constant advance of artificial intelligence, machine learning has become an important tool in this investigation. Thus, the aim of the study is to better understand how a machine learning model can help in finding out the reason why a particular failure occurs. For this purpose, a design methodology for data science was applied in which there was an exploration of a database created artificially for maintenance studies. From the database investigation, the C4.5 machine learning algorithm was applied in which a decision tree is generated with the conditions for a certain equipment failure. Then, an analysis of the obtained tree is done to explain how it can contribute to the definition of the reasons that lead to the defect of the machines. Therefore, the discussion is based on the comparison of the conditions described in the tree with the previously known criteria that generated the database artificially. Hence, it was possible to evaluate the results obtained by the algorithm and understand its capabilities. Finally, the work ends by concluding the possibilities that this study provides to assist in the management of maintenance.

Keywords: maintenance management, machine learning, machine failure, C4.5 algorithm

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

HDF	*Heat Dissipation Failure*
OSF	*Overstrain Failure*
PWF	*Power Failure*
TWF	*Tool Wear Failure*
K	Kelvin
Rpm	Rotação por minuto
W	Watts
Min	minutos
Nm	Newton metro
RNF	*Randon Failuure*

SUMÁRIO

1.INTRODUÇÃO

1.1 Contexto

Quando podemos dizer que surgiu a manutenção? Afinal de contas, a necessidade humana de preservar e manter as "coisas" é algo que acompanha a nossa evolução desde sempre. No entanto, a manutenção ganha uma importância e racionalização diferentes com a chegada da revolução industrial.

A indústria cresceu de forma desenfreada e junto com ela as técnicas e métodos de trabalho também se desenvolveram. Máquinas e linhas de produção eram projetadas para atingirem o maior grau de produtividade possível. Para que isso fosse verdade, diversas áreas e segmentações surgiram dentro da indústria sendo cada vez mais especializadas. Uma delas foi a manutenção, cuja evolução pode ser dividida em três gerações (MOUBRAY, 1997)

A primeira delas começa em 1940, nesse momento o mais comum para evitar as falhas era o uso de máquinas robustas, simples e confiáveis. A manutenção era completamente reativa, sendo ela mais voltada para limpeza e uso de lubrificantes, não existiam métodos ou técnicas complexas. Porém, principalmente com Ford em suas fábricas, é quando começam a surgir as primeiras equipes voltadas apenas para o reparo rápido e específico de máquinas (KARDEC, 2001).

A segunda geração surge após a 2ª Guerra Mundial, um momento em que a produção industrial foi impulsionada. Esse crescimento fez com que as máquinas e os processos produtivos na indústria se tornassem mais complexos. Nesse contexto, as falhas e defeitos se tornaram cada vez mais críticos, e a importância de manter a confiabilidade e eficiência aumentaram. Portanto, é nesse momento em que surge a manutenção preventiva e os controles de perdas no processo produtivo (KARDEC, 2001).

A terceira geração se inicia nos anos 80 a partir de uma nova fase de busca por performance, eficiência e resultados. Com a utilização de indicadores de qualidade, confiabilidade e técnicas estatísticas, o objetivo agora é otimizar a produção. A manutenção deixa de ser uma necessidade operacional, mas também uma necessidade estratégica (KARDEC, 2001).

Hoje, vivemos um processo de desenvolvimento da indústria 4.0, na qual as novas tecnologias trazem inúmeras oportunidades para a agregação de valor aos clientes e aumento de produtividade de processos (FLÁVIO, 2018). Uma das ferramentas que vem ganhando espaço dentro desse cenário de alta tecnologia é o *machine learning.* Uma subcategoria da inteligência artificial que possui diversas aplicações e uma grande contribuição nas tomadas de decisão. Tornando-se peça fundamental em muitos processos *(*Operacionalizar o *machine learning,* 2020*)*. E para a manutenção não é diferente, as indústrias nos dias de hoje aplicam a ferramenta na gestão da manutenção em diversos desafios visando reduzir os custos e otimizar processos.

É nesse contexto, que esse trabalho visa contribuir para o desenvolvimento de análises voltadas para aumento de eficiência dos processos de manutenção e suas tomadas de decisão. Entender como a ferramenta que vem sendo amplamente utilizada na indústria, o *machine learning*, pode agregar valor na gestão da manutenção de máquinas é a motivação deste trabalho.

1.2 Objetivo geral

O diagnóstico da manutenção se tornou ponto estratégico nas indústrias, auxiliando os responsáveis pelos processos de reparo e prevenção a terem o entendimento mais geral das falhas ocorridas nos equipamentos da fábrica. Caso uma máquina se quebre é natural assumir que gostaríamos de saber a sua causa para que o defeito não ocorra novamente no futuro. Esse é um passo conhecido como eliminação de recorrência. Além disso, as análises realizadas contribuem também para que as novas máquinas a serem projetadas sejam construídas sem as falhas estruturais. Portanto, um bom diagnóstico da relação de causalidade das falhas é fundamental para a criação de um ambiente de produtividade (O impacto do diagnóstico de manutenção na produtividade, 2022).

O objetivo do presente trabalho é investigar a validade do uso de técnicas de *machine learning* para diagnosticar as relações de causa e efeitos da falha de uma máquina. A pergunta que buscamos responder é: o aprendizado de máquina é capaz de colaborar para a interpretação das causas de defeitos em equipamentos?

1.3 Objetivo específico

A partir da aplicação de um modelo de *machine learning*, conhecido como C4.5 (QUINLAN, 1993), um modelo de árvores de decisão, será feita uma análise dos seus resultados. O modelo, com base nos dados fornecidos, nos promove um conjunto de regras para que a falha ocorra. Dessa forma, é possível estudar as regras geradas para entender as causas da falha.

Será usado um *dataset* com dados de uma máquina em que algum momento ocorreu algum tipo de falha. Este *dataset* foi criado artificialmente e as regras para que ocorra a falha são conhecidas previamente. Sendo assim, após aplicarmos o modelo de aprendizado, será feita uma análise das regras geradas e uma comparação com as originais para avaliarmos o potencial que essa técnica possui para auxiliar nas análises de causa e efeitos de falhas.

1.4 Limitações

Vale ressaltar que os modelos de *machine learning* são comumente usados para realizar algum tipo de predição ou classificação. Porém, esse não é o intuito deste estudo. Nosso objetivo não é prever a falha ou o tipo dela, mas sim descobrir as suas causas específicas para cada tipo de falha, e assim, possibilitar a geração de planos estratégicos de manutenção mais rápidos e eficientes.

1.5 Estrutura do trabalho

O trabalho consta de 5 capítulos dos quais são apresentadas as bases teóricas usadas no seu desenvolvimento e que geram as conclusões. Ele se inicia com uma introdução, na qual é feita uma contextualização a respeito do histórico da gestão da manutenção. Além disso, é descrito as motivações para realização do trabalho e quais são os objetivos que queremos alcançar ao final, bem como as suas limitações.

Em seguida partimos para o segundo capítulo onde apresentamos o referencial teórico usado neste trabalho. Os conceitos e definições necessárias para o total compreendimento da dissertação.

No terceiro capítulo é feita uma descrição e análise exploratória dos dados utilizados. É informada a sua procedência, como foram obtidos e o seu pré-processamento. Assim como a aplicação de ferramentas estatísticas com o intuito de obter informações sobre os dados.

Já no quarto capítulo apresentamos os resultados obtidos a partir da implementação do modelo utilizado. Também é feita uma discussão das possibilidades de uso dos resultados para a implantação na gestão da manutenção e geração de valor para o negócio.

Por fim, no quinto capítulo, é realizada a conclusão de todo o trabalho, bem como são apresentadas sugestões de melhorias e suas limitações.

2. REFERENCIAL TEÓRICO

2.1 Os principais tipos de manutenção

A manutenção das máquinas dentro da indústria é um mecanismo usado já há bastante tempo para otimizar a produção e os lucros. Ao longo do tempo ela passou por alguns estágios e foi evoluindo, hoje, temos 3 formas mais utilizadas para a manutenção (KARDEC, 2001).

1 - Manutenção corretiva: esse modelo de manutenção atua após ocorrer a falha do equipamento. (KARDEC, 2001).

2 - Manutenção programada: através de análises, é calculado a manutenção para ser realizada em intervalos de tempos, normalmente iguais, já pré-programadas. Com o objetivo de atuar antes que ocorra a falha e assim não é preciso interromper a produção. (KARDEC, 2001).

3 - Manutenção preditiva: esse modelo visa prever quando a falha irá ocorrer e realizar a manutenção um pouco antes dela de fato acontecer. A ideia é que só será realizada a manutenção quando realmente for necessário. (KARDEC, 2001).

As 3 formas de se realizar a manutenção possuem seus pontos fortes e fracos, e a verdade é que não existe um modelo melhor do que o outro. Cabe aos responsáveis, analisar para cada caso, cada máquina e parte da produção, qual forma é a mais eficiente para a situação em questão.

No entanto, do ponto de vista financeiro e de produtividade, a manutenção preditiva se destaca. Ela é a que tem o menor índice de manutenções desnecessárias, e a que perde menos tempo para ser realizada. Assim, ela diminui os custos da produção, ao mesmo tempo que otimiza a produtividade já que ocorrem menos interrupções na linha de produção.

2.2 *Machine learning*

Machine Learning é um campo de estudo da inteligência artificial que gera para máquinas a capacidade de aprenderem sem serem explicitamente programadas (ARTHUR SAMUEL, 1959). O que significa uma quebra de paradigma da programação tradicional, na qual as instruções são explicitamente passadas ao computador, para uma programação indireta, em que o computador aprende as “regras ocultas” nos dados fornecidos. Tom Mitchell, um cientista da computação conhecido pelas suas contribuições no avanço de *machine learning*, formalizou o problema de aprendizado:

> *“Um programa de computador "aprende" a partir da experiência E, em relação à uma tarefa T e um indicador de performance P, se seu desempenho em T, medido por P, melhora com a experiência E.”*

2.3 Árvores de decisão

Uma árvore de decisão é uma estrutura similar a uma árvore que auxilia no processo de decisão. Esse método cria uma hierarquia de regras que levam a decisão final, de forma que gere o menor conjunto de regras possíveis (SHALEV-SCHWARTZ,2014). Dessa forma, a árvore é criada com nós de decisão e nós folha. Um nó de decisão é aquele que possui uma ou mais ramificações que levam a outra regra. Já o nó folha representa uma classificação ou decisão, é o nó final de uma ramificação (SHALEV-SCHWARTZ,2014). Podemos ver um exemplo na figura 1:

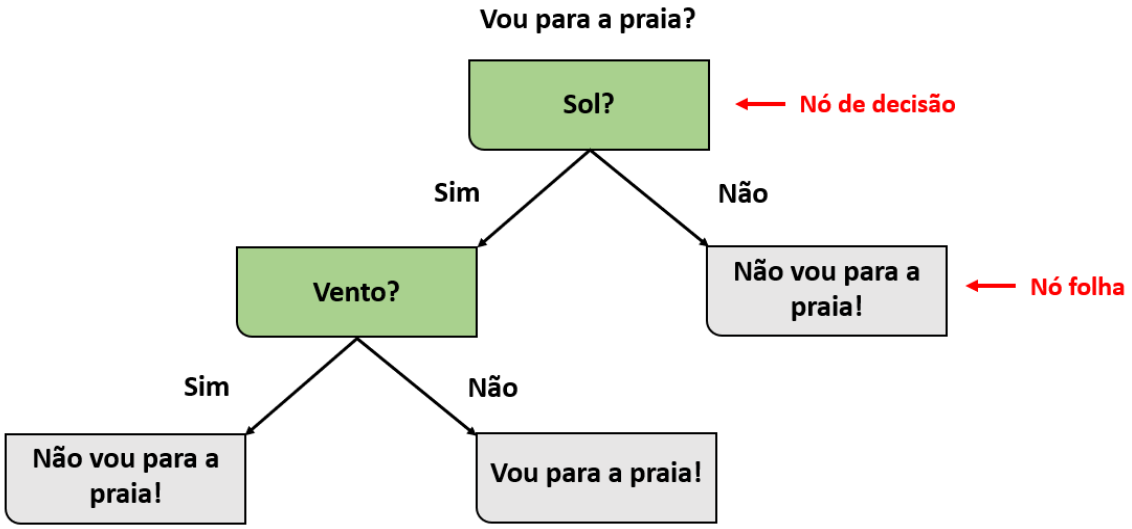

Figura 2.1: Exemplo de uma árvore de decisão

Uma árvore de decisão então é um modelo de aprendizado de máquina supervisionado, quando é fornecida uma variável resposta, usada para classificação ou regressão. O modelo cria as ramificações com base nos atributos e respostas das quais se obtém o melhor ganho de informação. Começando a partir de uma regra inicial, conhecida como nó-raiz, ela vai criando as regras que melhor separa os dados até chegar no nó folha, que representa a classificação ou decisão.

No presente trabalho, ao invés de analisarmos as classificações geradas, estamos interessados em estudar as regras de decisões da árvore. Devido ao fato de que nosso objetivo é entender as causas para que tenha ocorrido a falha. E as regras podem ser entendidas como as condições para que uma falha específica ocorra.

2.4 Algoritmo C4.5

O algoritmo C4.5 é um modelo de árvore de decisão feito por Quinlan (QUINLAN, 1993) e muito usado para classificação. Ele é uma extensão do algoritmo ID3 (QUINLAN, 1993) também criado por ele.

Seu funcionamento se dá a partir de um conjunto de dados, no qual cada amostra possui atributos e uma variável alvo, aquela que classifica a amostra. Para cada nó da árvore de decisão o C4.5 escolhe o atributo que melhor separa o conjunto de dados em subconjuntos que tendem para uma categoria ou outra. O critério para divisão é um teste de ganho de informação baseado no conceito de entropia. O ganho de informação é calculado pela diferença de entropia antes da divisão e a soma ponderada das entropias após a divisão. O cálculo é feito para cada atributo, e assim, o atributo que apresentar o maior ganho de informação normalizado, será o escolhido para tomar a decisão. Esse processo é feito até chegar em uma folha, ou seja, uma classificação (PEREIRA, 2020).

Suas principais características que o diferenciam do algoritmo que o antecede (ID3) são:

- Capaz de lidar com atributos contínuos e discretos.
- Capaz de lidar com dados faltantes.
- Usa o método de poda, no qual o algoritmo percorre a árvore após ela ser criada substituindo os ramos que não contribuem para classificação por folhas.
- Lida com atributos de diferentes custos.

Na figura 2.2, o seu pseudocódigo:

```
Se todas as labels = C então
        retorna o único nó da árvore com C
Se não
        a ← atributo que melhor clássica S
        para v ∈ a fazer
                adicionar novo ramo da árvore a = v
        fim
        Sv ← amostra de exemplos em S em que a = v
        Se Sv está vazio então
                adicionar nó folha com o valor mais comum da label em S
        se não
                adicionar uma ramificação (Sv,A – [a],C)
```

Figura 2.2: Pseudocódigo do algoritmo C4.5

2.5 Regras de decisão

Como foi explicado, as árvores de decisão são formadas a partir de nós e ramificações. Na ligação entre nós, temos regras de se-então, que são as regras de decisão. É feita uma pergunta acerca de uma condição do tipo: A temperatura é maior que 30°C?, e as ramificações são criadas para os casos em que a temperatura seja menor ou maior do que 30°C. Importante ressaltar, que essas regras são geradas a partir do aprendizado do modelo baseado no ganho de informação e entropia, e que representam as condições das quais o modelo julga necessário para que certo evento/classificação ocorra.

No exemplo fictício abaixo, temos uma árvore de decisão baseada em 3 atributos e que visa classificar o tipo de falha que ocorreu em uma máquina. Este exemplo é similar ao que será encontrado no desenvolvimento do trabalho.

Atributos:

- Torque
- Velocidade de Rotação
- Temperatura

Possíveis falhas:

- Falha de força
- Super Aquecimento
- Falha na Peça

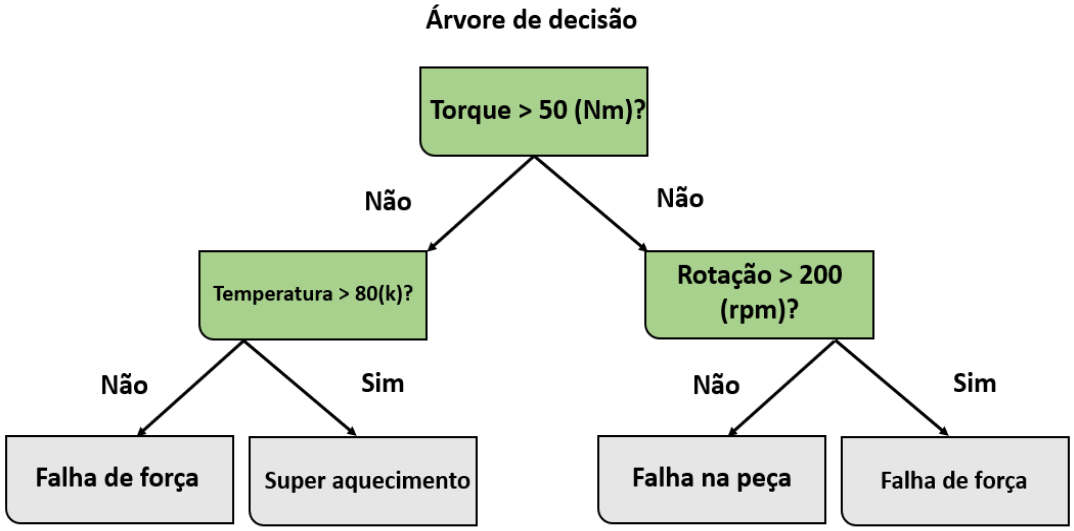

Figura 2.3: Árvore de decisão das falhas de uma máquina

Fazendo uma análise das regras que levam ao superaquecimento, temos:

- Torque < 50
- Temperatura > 80°C

Portanto, segundo o modelo, essas são as condições para que ocorra uma falha por super aquecimento.

3. METODOLOGIA

3.1 Metodologia

A metodologia do trabalho foi baseada em um fluxo conhecido como processo CRISP-DM, muito usada em projetos de ciência de dados, e apresentada na figura 3.1 (SHEARER, 2020):

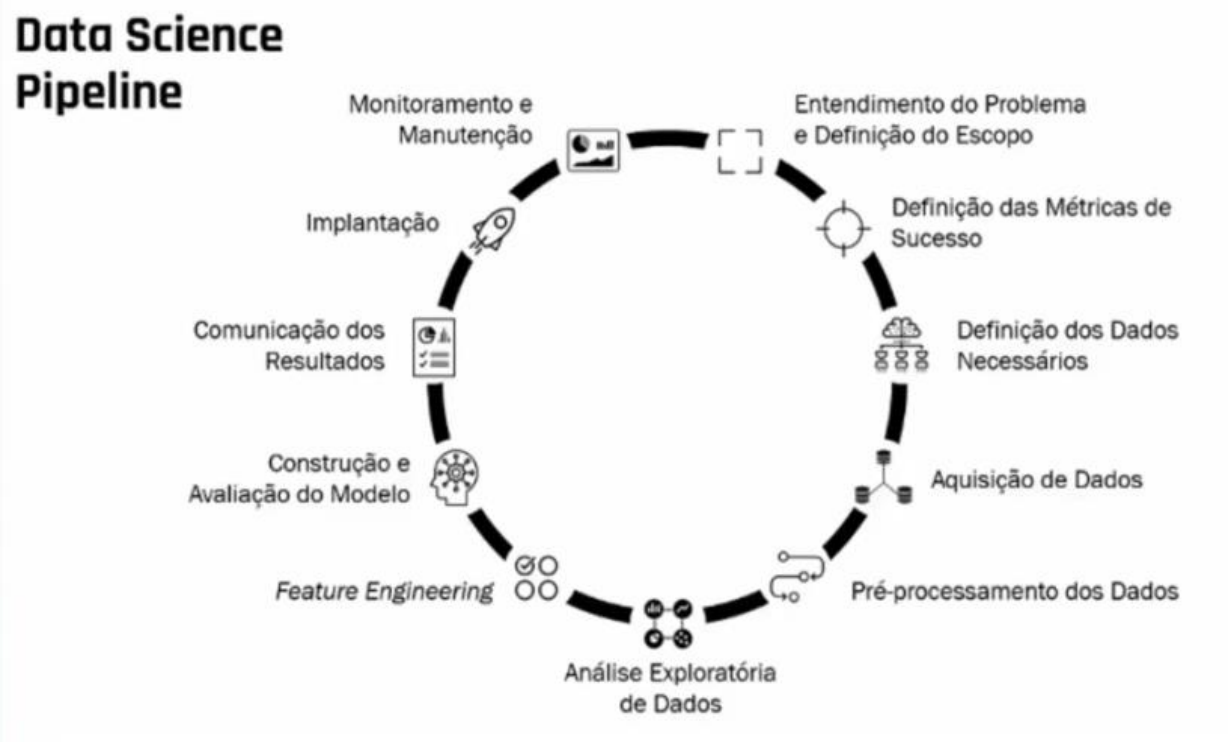

Figura 3.1: Pipeline de um projeto de Ciência de Dados (VAI *Academy*)

No começo é feito o entendimento do contexto e do problema que se visa atuar, então é passado o histórico da gestão da manutenção no mundo para nos contextualizar a respeito da situação em que o tema se encontra atualmente. E em cima disso, é apresentada a motivação para a construção do presente trabalho, explicitando as razões e os motivos que geraram o interesse no estudo. Assim, foram estabelecidos os objetivos que se almejam atingir ao término do trabalho, como também as suas limitações.

Partimos então para a confecção de um referencial teórico, no qual são apresentados os conceitos básicos para o entendimento geral. Esses conceitos são: os 3 tipos de manutenção, *machine learning*, árvores de decisão, algoritmo c4.5 e regras de decisão. Todos eles são essenciais e sua compreensão fundamenta o restante da pesquisa. São conceitos da engenharia e da ciência de dados.

Em seguida, é iniciado o desenvolvimento do estudo do problema, que consiste na coleta de dados, uma análise exploratória dos mesmos, a aplicação de um algoritmo de *machine learning* e por fim uma análise dos resultados e seus benefícios para a gestão da manutenção.

A respeito dos dados, sua coleta foi feita dentro de um repositório online no qual pessoas de todo o mundo compartilham *datasets* com o intuito de contribuir para os estudos de outras pessoas. O *dataset* em questão foi gerado artificialmente por Stephan Matzka (School of Engineering - Technology and Life) devido a dificuldade de encontrar dados de manutenção.

Eles representam atributos de uma máquina e indicam se ocorreu falha na mesma, assim como qual é o tipo de falha caso tenha acontecido. Esse *dataset* então passa por uma análise exploratória, no qual é destrinchado com a intenção de promover *insights* e descobertas que nos ajudem a atingir o objetivo final do trabalho. Além disso, são tratados e remodelados, o que é conhecido como engenharia de variáveis, parte do processo que é feita para gerar melhores resultados no modelo de *machine learning.*

O próximo passo então é a aplicação do algoritmo c4.5, um modelo de árvore de decisão que aprende com os dados fornecidos as regras para que se ocorram as falhas nas máquinas deste *dataset*. As regras geradas são usadas para entender as causas e as razões que levam a máquina até a falha. Assim, como *insights* que levem a ações que contribuam para a gestão da manutenção.

Conclui-se com uma avaliação da capacidade das regras geradas explicarem o motivo das falhas. Na qual é feita uma comparação entre os resultados obtidos após a aplicação do modelo c4.5 e as regras previamente conhecidas que geraram a base de dados. Dessa forma, será possível responder a pergunta feita no objetivo deste trabalho: o aprendizado de máquina é capaz de colaborar para a interpretação das causas dos defeitos feitos de máquinas?

4. BASE DE DADOS

4.1 Origem dos dados

Para realização do estudo sobre a validação do uso de *machine learning* no auxílio à gestão da manutenção foi usado um *dataset* público, que pode ser encontrado no site UCI "*Machine Learning Repository*". Ele foi disponibilizado no *site* e usado por Stephan Matzka em seu artigo "'*Explainable Artificial Intelligence for Predictive Maintenance Applications*", apresentado na 3ª conferência internacional de Inteligência Artificial para Industriais (AI4I 2020). Devido à dificuldade de se encontrar bases de dados reais sobre manutenção, esse *dataset* foi criado usando dados artificiais que refletem da melhor forma possível a realidade encontrada na indústria.

4.2 Descrição da base de dados

Cada linha do *dataset* representa uma máquina e as colunas apresentam dados referentes ao seu funcionamento. Além disso, existem colunas que indicam se aquele produto sofreu falha durante o processo e caso sim, nos diz também qual tipo de falha ocorreu, sendo delas 5 possíveis. O *dataset* então possui um total de 10.000 casos (linhas) x 14 Atributos (colunas). Dos 14 atributos, podemos dividir em 3 categorias, primeiros aqueles que servem como elementos identificadores das máquinas. Em segundo, os atributos que nos fornecem informações sobre o funcionamento dos equipamentos. E por último as colunas trazem informações sobre a ocorrência de falha e o seu tipo. Abaixo segue a descrição de cada um dos atributos:

<u>Atributos de identificação:</u>

- **UDI:** É o número identificador de cada linha que varia de 1 a 10.000

- **Product ID:** É o número identificador de cada produto. Assim, todas as linhas são diferentes e cada ID é único. É composto pelas letras L, M e H seguido de um número serial específico.

	UDI	Product ID
0	1	M14860
1	2	L47181
2	3	L47182
3	4	L47183
4	5	L47184

Figura 4.1: exemplo de como são os dados dos atributos de identificação

Atributos sobre o funcionamento da máquina:

- ***Type*:** Informa qual é a qualidade do produto. São 3 categorias, L (baixa), M (Média) e H (alta). No *dataset* são observados que 60% dos produtos são de categoria L, 30% de M e 10% de H.

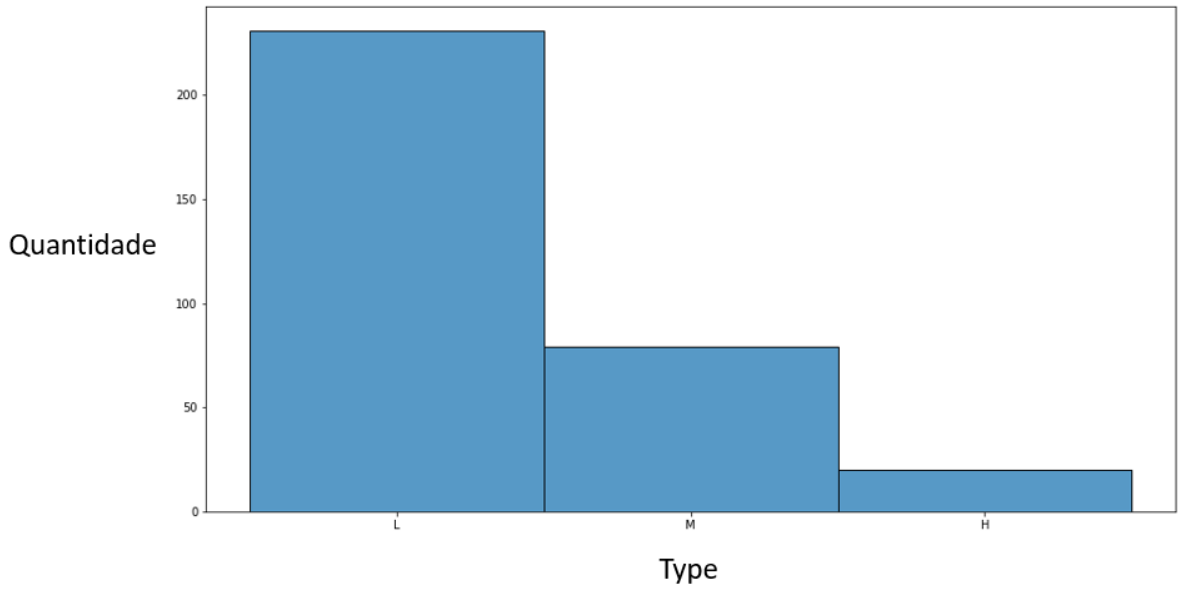

Figura 4.2: gráfico da quantidade por tipo de das peças na base de dados

- ***Air Temperature (K):*** É a temperatura do ar em Kelvin. Foi gerada usando *random walk process* e normalizadas posteriormente com um desvio padrão de 2 K em torno de 300 K. Possui 93 valores distintos.

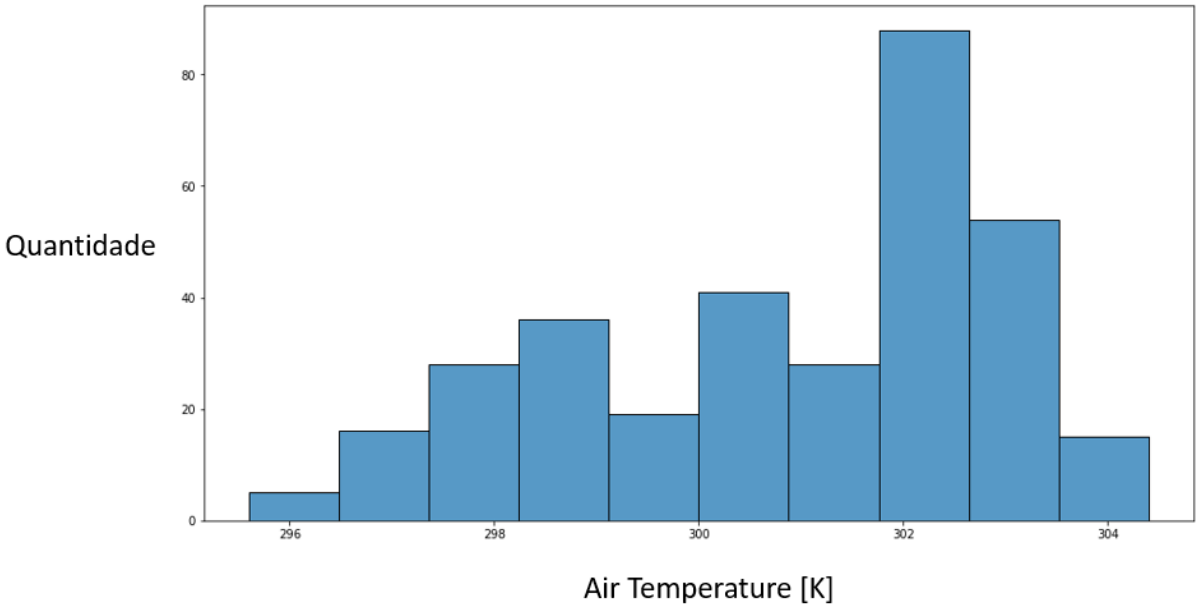

Figura 4.3: gráfico da distribuição da temperatura do ar na base da dados

- ***Process Temperature (K):*** É a temperatura do processo em Kelvin. Foi gerada usando *random walk process* e normalizadas posteriormente com um desvio padrão de 1 K adicionado ao *air temperature* + 10 K. Possui 82 valores distintos.

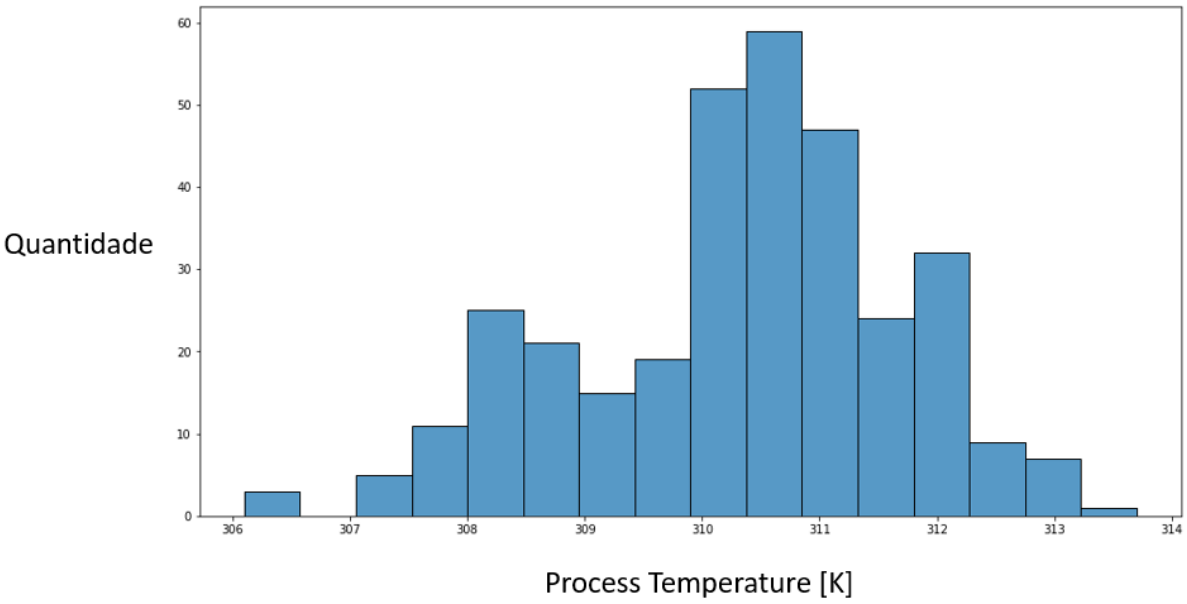

Figura 4.4: gráfico da distribuição da temperatura do processo na base da dados

- **Rotational Speed (rpm):** É a velocidade de rotação do produto em rotações por minuto. Foi calculado a partir de uma potência de 2860 W, sobreposto a um ruído normalmente distribuído. Possui 941 valores distintos.

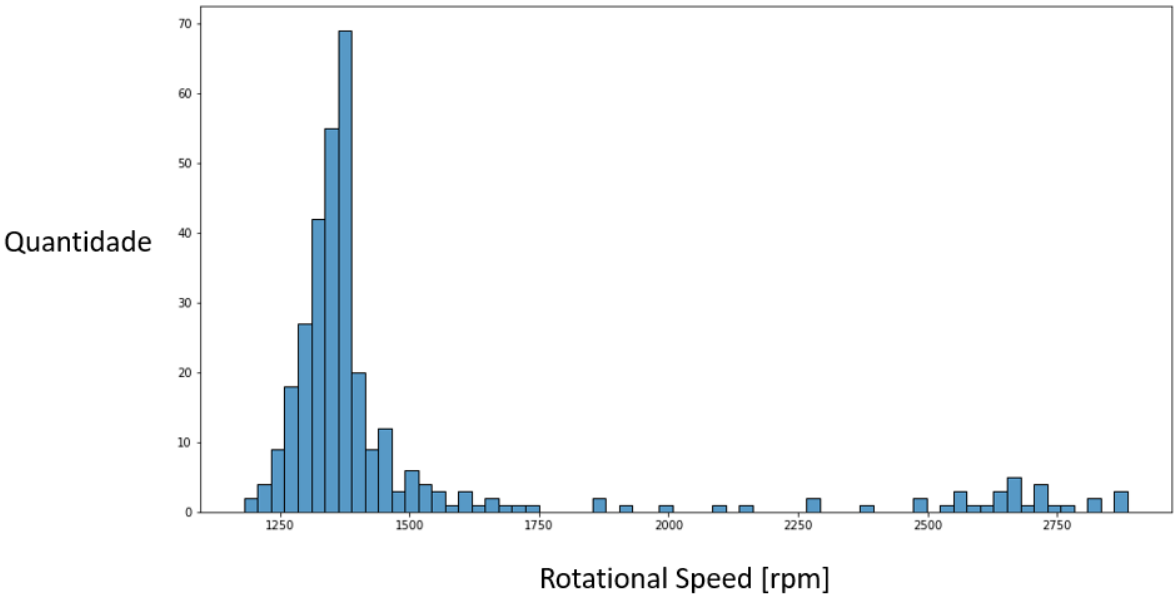

Figura 4.5: gráfico da distribuição da velocidade de rotação na base da dados

- **Torque (Nm):** É o torque exercido pelo produto em Nm. Seus valores vem de um distribuição normal em torno de 40 Nm e sem valores negativos. Possui 577 valores distintos.

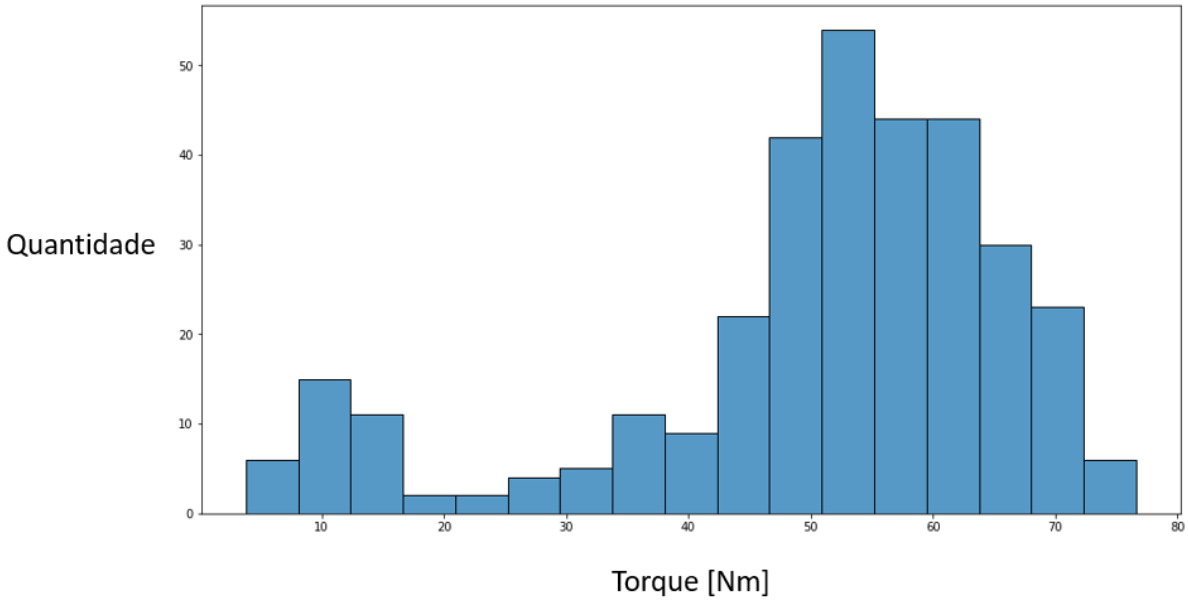

Figura 4.6: gráfico da distribuição do torque na base da dados

- **Tool Wear (min):** É o desgaste da ferramenta em minutos. Dependendo da qualidade do produto L/M/H são adicionados 5/3/2 minutos ao Tool Wear. Possui 246 valores distintos.

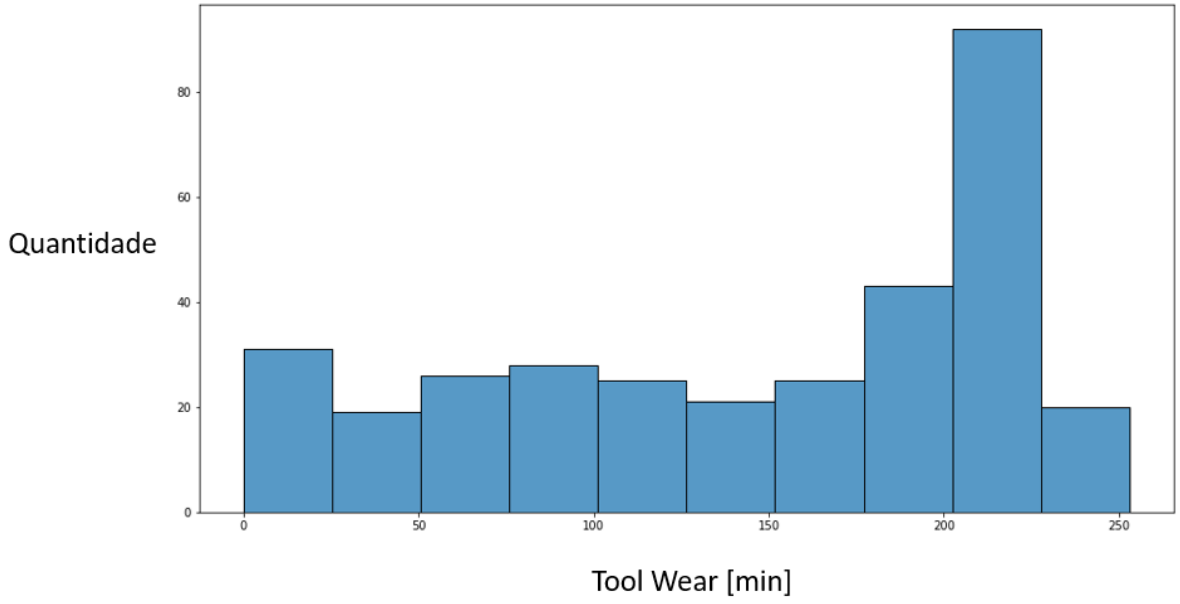

Figura 4.7: gráfico da distribuição desgaste da peça na base da dados

Atributos sobre as falhas:

- ***Machine Failure:*** Indica se houve falha na máquina quando seu valor é "1" e quando não houve sendo "0".

- ***Tool Wear Failure (TWF):*** Falha por desgaste da ferramenta. A ferramenta será substituída por falha em um tempo de desgaste da ferramenta selecionado aleatoriamente entre 200 – 240 minutos (120 vezes em nosso conjunto de dados). Neste momento, a ferramenta é substituída 69 vezes e falha 51 vezes (atribuída aleatoriamente).

- ***Heat Dissipation Failure (HDF):*** Falha no processo devido a dissipação de calor. A dissipação de calor causa uma falha no processo, se a diferença entre a temperatura do ar e do processo estiver abaixo de 8,6 K e a velocidade de rotação da ferramenta estiver abaixo de 1380 rpm. Este é o caso de 115 pontos de dados.

- ***Power Failure (PWF):*** Falha no processo devido à potência estar baixa ou alta demais, sendo potência o produto do torque e de da velocidade de rotação. O produto do torque e da velocidade de rotação (em rad/s) é igual à potência necessária para o processo. Se esta potência estiver abaixo de 3500 W ou acima de 9000 W, o processo falha, o que ocorre 95 vezes em nosso conjunto de dados.

- ***Overstrain Failure (OSF):*** Falha por sobretensão quando o desgaste e o torque são muito altos. Se o produto do desgaste da ferramenta e do torque exceder 11.000 minNm para a variante de produto L (12.000 M, 13.000 H), o processo falha devido a sobretensão. Isso é verdade para 98 pontos de dados.

- ***Random Failures (RNF):*** Falha aleatória. Cada processo tem uma chance de 0,1% de falhar independentemente de seus parâmetros de processo. Este é o caso de apenas 5 pontos de dados, menos do que o esperado para 10.000 pontos de dados em nosso conjunto de dados.

4.3 Pré-processamento dos dados

Para aplicarmos o modelo é preciso realizar um pré-processamento da base, uma vez que nem todos os dados fornecidos contribuem para o nosso propósito. Sendo assim, algumas modificações foram realizadas.

O *dataset* possui um total de 330 instâncias no qual ocorre falha, isso representa uma porcentagem de 3,30% do total dos equipamentos. Apesar de parecer uma amostra pequena, esse valor seria uma taxa alta de falha em uma produção em massa (MATZKA, 2020). No entanto, esse valor representa uma base de dados desbalanceada quando se trata de equipamentos que ocorreram falha. Como objetivo do nosso estudo é realizar uma análise das causas e efeitos das falhas, para fins de treinamento do algoritmo, foram consideradas apenas

as 330 instâncias em que ocorrem falhas, uma vez que as instâncias onde não ocorrem não contribuíram para o treinamento do modelo.

Vale ressaltar, que pelo fato da base ter sido gerada artificialmente o *dataset* não possui dados faltantes, algo comum de se encontrar em uma base de dados mas que não foi preciso lidar neste caso.

Outro ponto foi a retirada de algumas colunas que não contribuem para a aplicação do algoritmo c4.5. Foram elas as duas primeiras que servem apenas para identificação, a UDI e a de Product ID, ou seja, não fornecem dados sobre o funcionamento da máquina. Os atributos que apresentam dados sobre o funcionamento da máquina foram mantidos normalmente pois são valiosos para a classificação a ser realizada pelo modelo, sendo eles: *Type*, *Air Temperature*, *Process Temperature*, Torque, *Rotational Speed* e *Tool Wear*. Além disso, os 6 atributos que dizem respeito às falhas foram concatenados em apenas uma coluna que nos indica qual tipo de falha ocorreu, uma vez que agora temos apenas linhas em que o produto falhou.

Portanto a base de dados ao final do pré-processamento contém 330 linhas e 7 colunas. A figura 4.8 exemplifica algumas das linhas.

	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Machine failure type
0	L	298.9	309.1	2861	4.6	143	pwf
1	L	298.9	309.0	1410	65.7	191	pwf_osf
2	L	298.8	308.9	1455	41.3	208	twf
3	L	298.4	308.2	1282	60.7	216	osf
4	L	298.3	308.1	1412	52.3	218	osf
...	...	...	...	...	...	...	...

Figura 4.8: exemplo das linhas da base de dados

4.4 Análise exploratória

O objetivo desta parte do trabalho é realizar uma análise dos dados e obter *insights* que contribuam para o objetivo geral de entender as relações de causa e efeito das falhas. Realizando uma análise exploratória, é possível identificar padrões e correlações entres os dados que podem ser valiosos para o entendimento do problema. No contexto do trabalho, vamos começar por uma exploração das falhas, e posteriormente dos atributos e como elas estão relacionadas.

4.4.1 Análise dos tipos de falhas

No gráfico abaixo temos uma distribuição das falhas e a quantidade de vezes em que elas ocorrem. O primeiro fato observado é que as falhas explicadas anteriormente podem acontecer também de forma paralela, ou seja, ao mesmo tempo. Elas representam 24 (7%) das falhas totais. Devido a sua baixa representatividade dentro da base de dados, e pelo motivo de que nosso objetivo é entender melhor como cada falha ocorre separadamente, foi decidido não usar essas instâncias no processamento do modelo. Além disso, uma base de dados desbalanceada dificultado o processamento realizado pelo modelo.

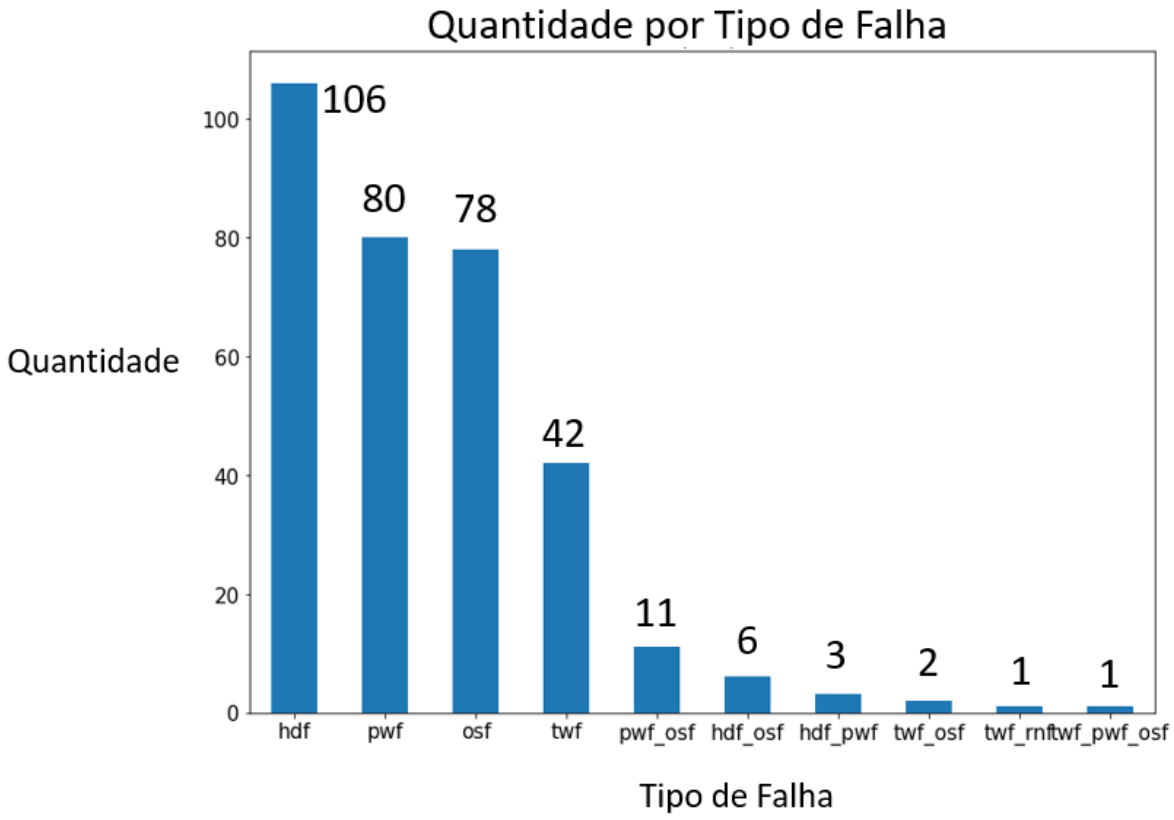

Figura 4.9: gráfico da quantidade de cada tipo de falha

Com a eliminação dessas instâncias, temos uma base balanceada como é possível observar no gráfico abaixo. Também verificamos que a falha do tipo HDF é a falha mais comum, seguida do PWF, OSF e TWF. É interessante perceber, que é mais comum a máquina falhar por algum motivo antes do que pelo desgaste da peça (TWF).

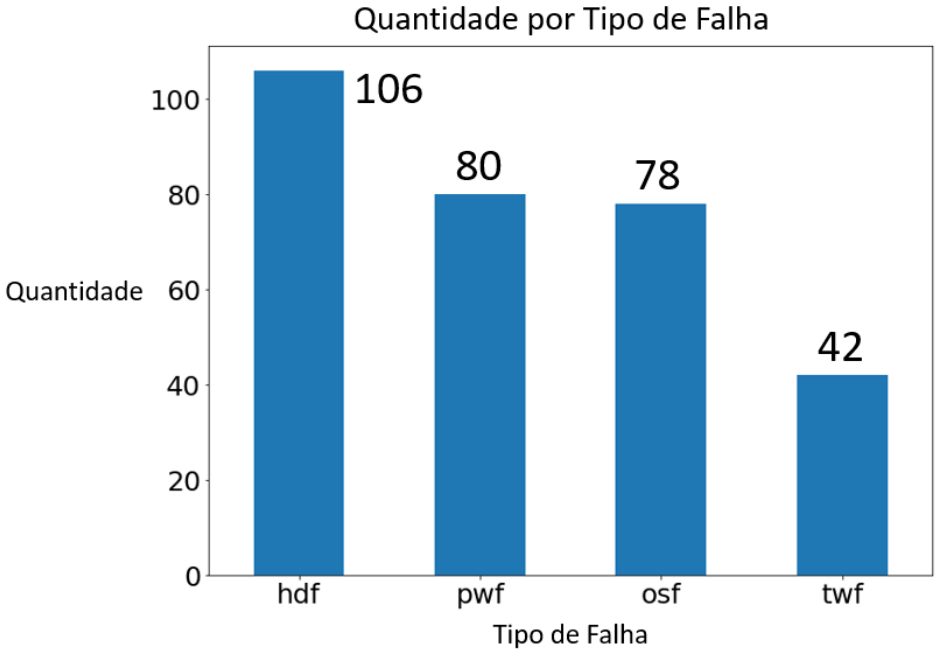

Figura 4.10: gráfico da quantidade de cada tipo de falha após a limpeza de dados

4.4.2 Análise da correlação entre as falhas e os atributos

O objetivo deste momento é entender como os atributos de funcionamento das máquinas estão relacionados entre si e com cada tipo de falha.

Iniciamos realizando um *heatmap*, um gráfico que serve para calcular a correlação entre variáveis, que vai de -1 a 1, sendo -1 a máxima correlação negativa e 1 a máxima positiva. Caso o valor esteja próximo de 0, isso significa que as variáveis possuem baixa correlação (WILKINSON, 2009). No nosso caso, queremos entender se os atributos apresentam alguma correlação entre si. Podemos observar na figura 4.11, uma alta correlação positiva entre o *Air Temperature* e o *Process Temperature*. Observamos também uma alta correlação negativa entre o Torque e o *Rotational Speed*.

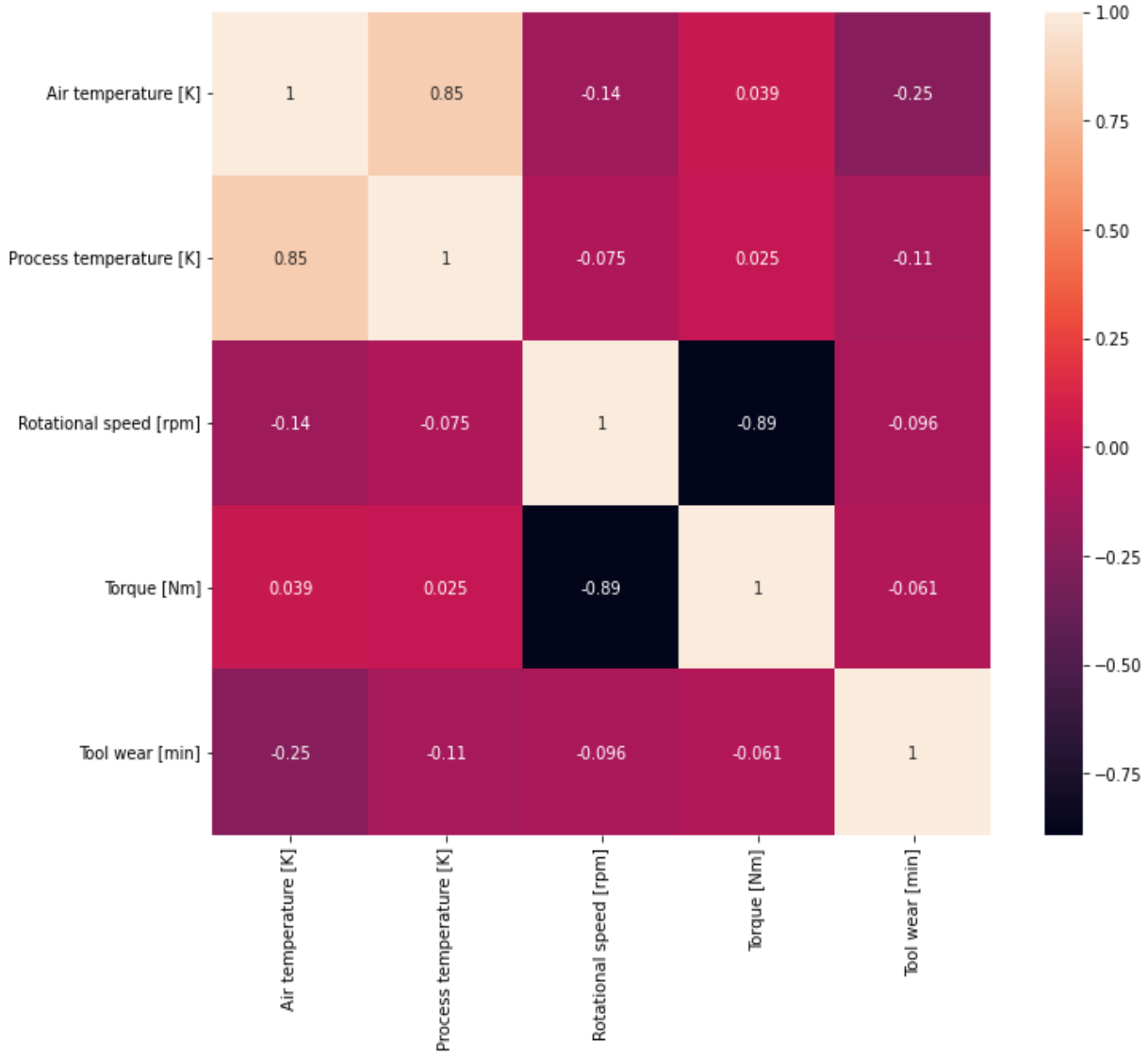

Figura 4.11: gráfico de calor dos atributos da base de dados

Fizemos também, o uso de um gráfico conhecido como *box plot*, nele são apresentadas 5 estatísticas: o mínimo, o primeiro quartil (Q1), a mediana, o terceiro quartil (Q3) e o máximo (ROSS, 2020). A fim de explorar como as variações nos valores dos atributos se comportam para cada falha separadamente, geramos um *box plot* por falha em relação a cada um dos atributos. A ideia aqui é analisar o comportamento do atributo dentro de uma falha em comparação às outras falhas. O fato do atributo se comportar de forma similar entre as falhas pode indicar que ele não tem influência sobre as mesmas. Agora, uma vez que ele apresenta um comportamento diferente para uma falha específica, isso pode ser um indicativo de que aquele atributo exerça influência para que a falha ocorra. Importante ressaltar que essa análise não determina a causalidade, mas sim nos fornece uma informação a mais em nossa pesquisa. Seguindo então este raciocínio, a análise nos indica que os atributos que possivelmente exercem influência em cada falha são.

Air Temperature e o *Process Temperature* estão possivelmente relacionados à falha HDF. Observamos que o *Air Temperature* e o *Process Temperature* no caso do HDF acontecem em intervalos de valores bem determinados, enquanto nas demais falhas aparentemente podem ocorrer em qualquer temperatura.

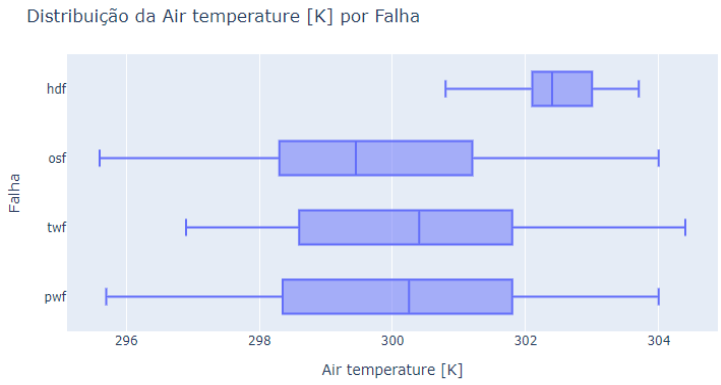

Figura 4.12: gráfico da distribuição da temperatura do ar para cada falha

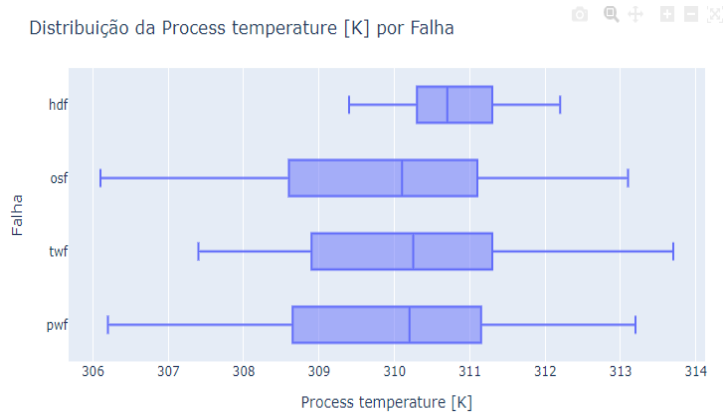

Figura 4.13: gráfico da distribuição da temperatura do processo para cada falha

Rotational Speed e o Torque estão possivelmente relacionados com a falha PWF. Para o *Rotational Speed*, o atributo tem comportamento diferente dos demais, no qual parece acontecer em um intervalo grande de valores enquanto os demais em apenas alguns bem definidos. Já para o Torque, o atributo tem comportamento diferente dos demais, no qual parece acontecer em um intervalo grande de valores enquanto os demais em apenas alguns bem definidos.

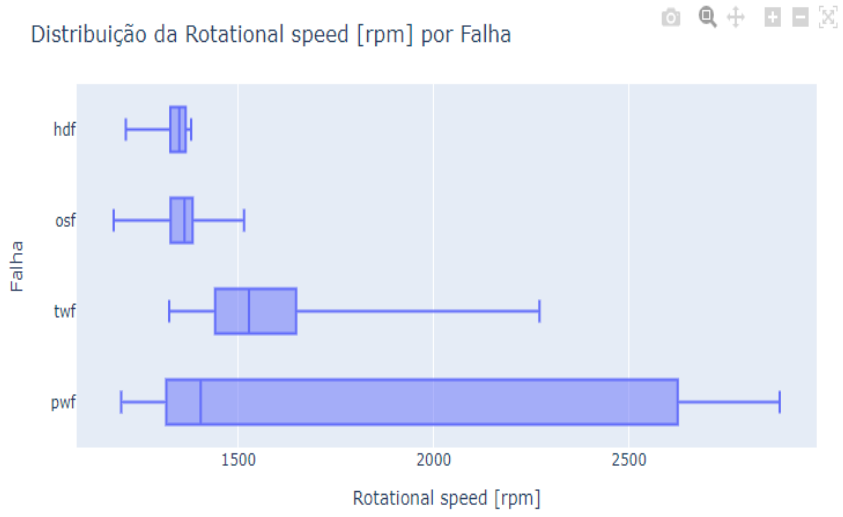

Figura 4.14: gráfico da distribuição da velocidade de rotação para cada falha

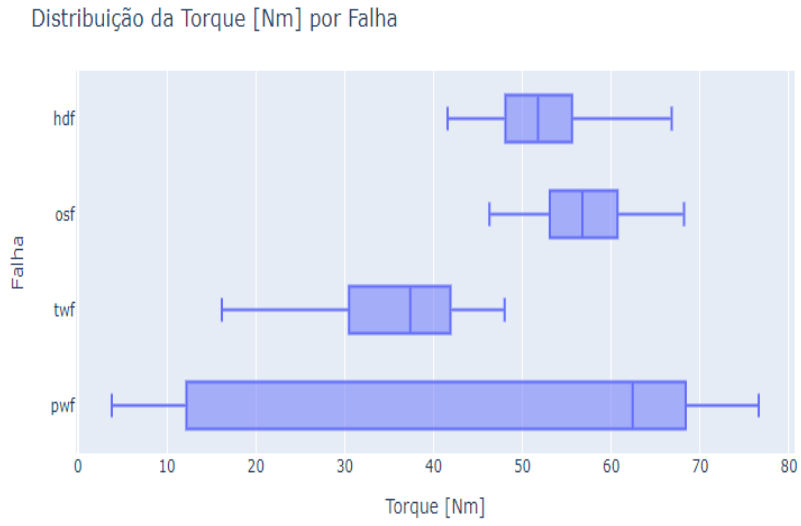

Figura 4.15: gráfico da distribuição do torque para cada falha

Tool Wear está possivelmente relacionado às falhas de OSF e TWF. O atributo parece ter um comportamento específico para essas falhas, eles apresentam valores altos de *Tool Wear* quando ocorrem, enquanto para as demais falhas pode ocorrer em qualquer valor deste atributo.

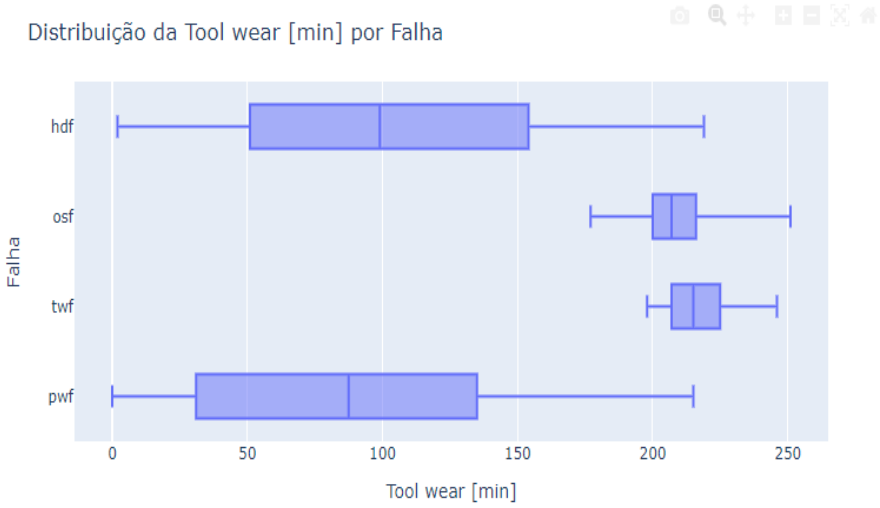

Figura 4.16: gráfico da distribuição do desgaste da peça para cada falha

Para ir mais a fundo na análise, realizamos também um gráfico conhecido como *scatter plot* que é usado para verificar se existe uma relação entre causa e efeito entre duas variáveis numéricas (CHAMBERS, 2018). Isso não significa que uma variável causa efeito na outra, mas apenas se existe uma relação e qual intensidade entre essa relação. Fizemos então um gráfico para cada combinação de atributos e colorimos os pontos de acordo com o tipo de falha. Apresentamos aqui apenas os gráficos que agregam com *insights* para a nossa investigação.

Quando observamos a figura 4.15 gerada entre o Torque e o *Rotational Speed*, identificamos uma relação negativa para as falhas do tipo TWF e PWF. Também é interessante reparar que a falha por PWF acontece quando os valores de *Rotational Speed* são mais altos ou mais baixos, ou seja, nas suas extremidades.

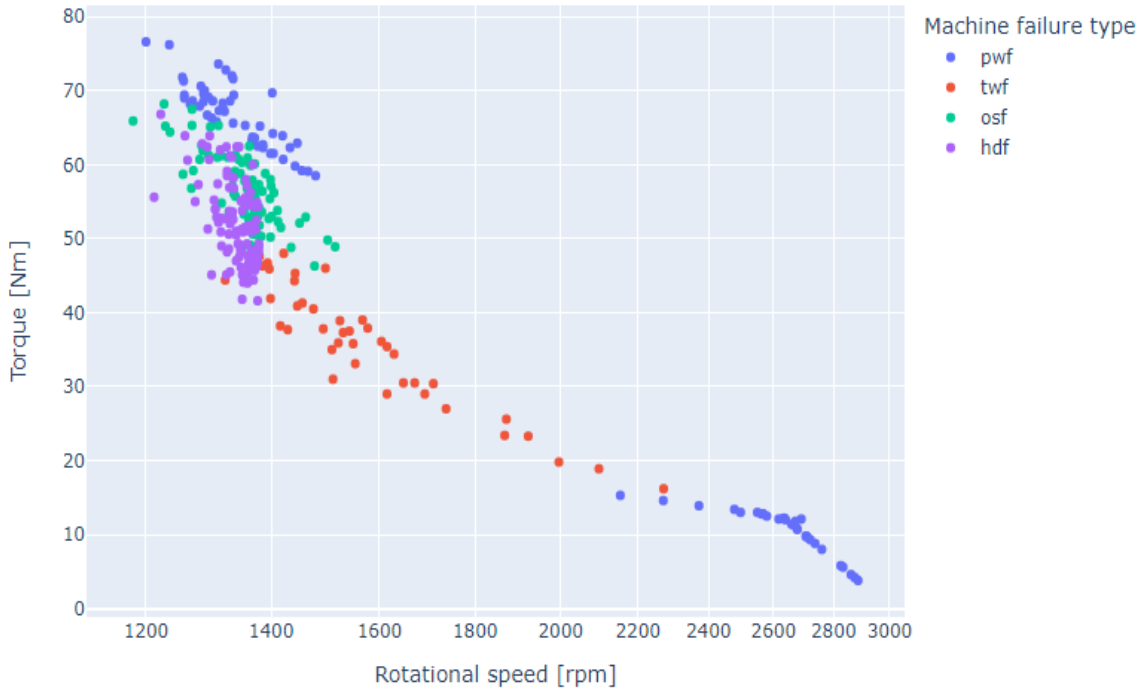

Figura 4.17: gráfico de relação entre o torque e velocidade de rotação

No gráfico gerado entre o *Air Temperature* e o *Process temperature*, identificamos uma relação positiva entre o todas as falhas. No entanto, o interessante de se observar, é que para o HDF acontecer as duas temperaturas são sempre os maiores valores. O que torna mais uma evidência que esses atributos estão relacionados com a falha de HDF.

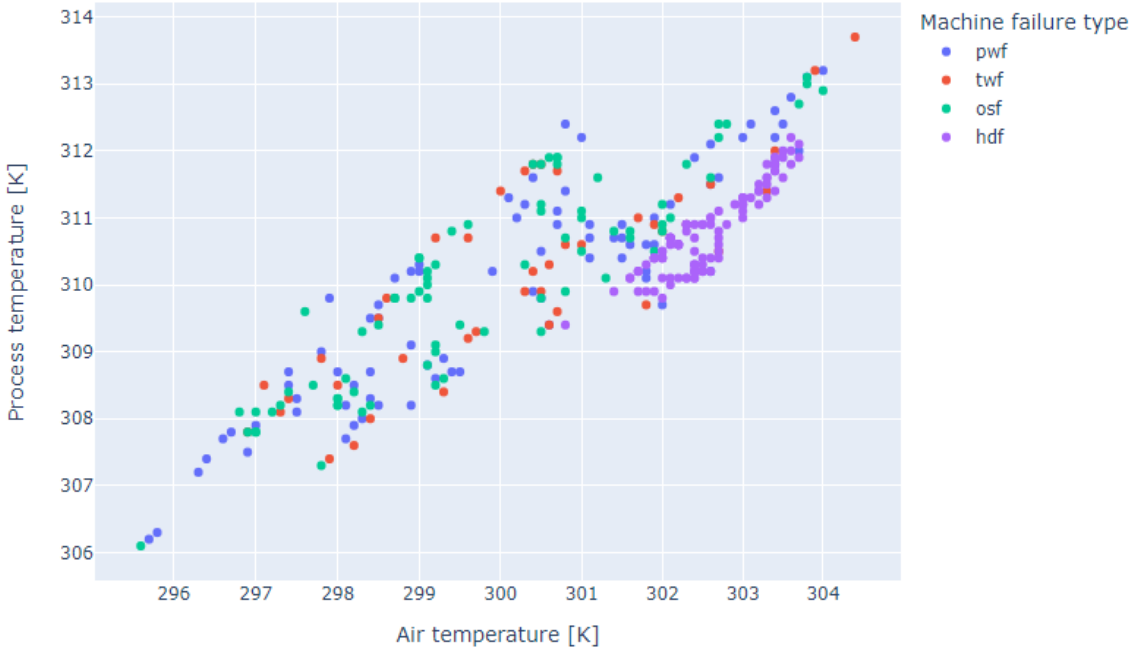

Figura 4.18: gráfico de relação entre a temperatura do ar e do processo

Quando colocamos o *Tool Wear* no eixo X para ver a sua relação com as outras variáveis, encontramos o mesmo padrão em todos os gráficos. As falhas de OSF e TWF acontecem sempre com valores elevados de *Tool Wear*, enquanto que as demais parecem poder ocorrer em qualquer valor. Além disso, a variável Torque foi a única a apresentar uma diferença, quando os valores de Torque são altos, ocorre o OSF e na metade mais baixa ocorre o TWF.

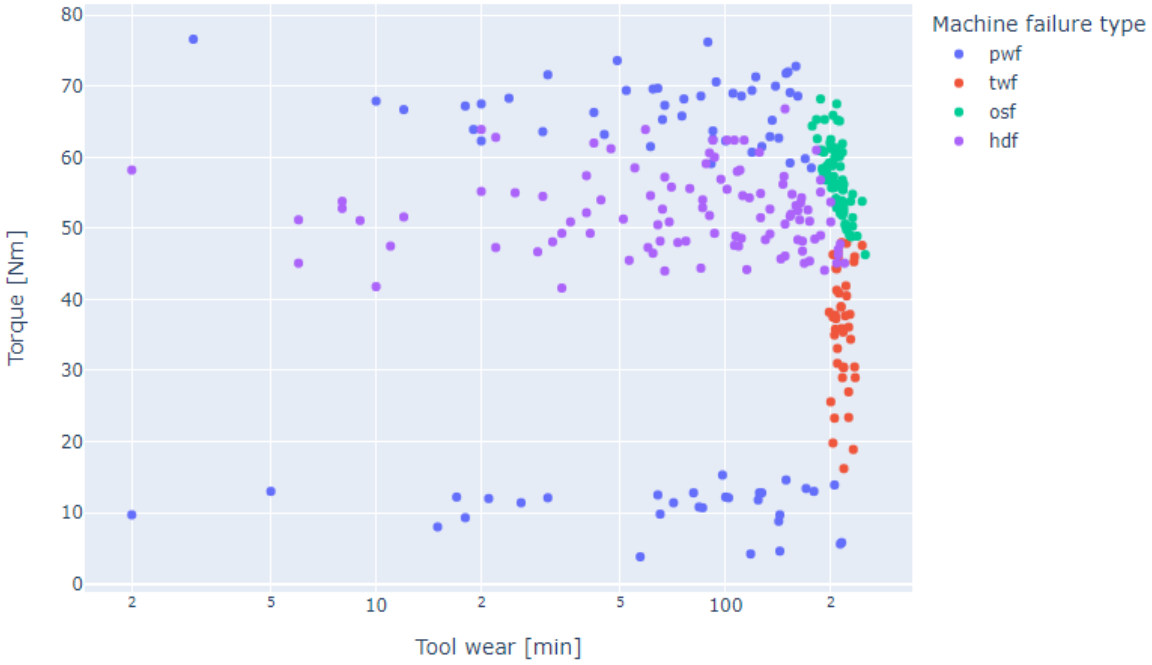

Figura 4.19: gráfico de relação entre o troque e o desgaste da peça

4.5 Engenharia das variáveis

É comum que antes de colocarmos os dados para serem processados pelo algoritmo, seja feita uma avaliação dos atributos usados, com o intuito de obtermos o melhor resultado. No caso do trabalho, decidimos substituir as variáveis *Air temperature* e *Process Temperature* pela diferença entre elas. Pois elas apresentam uma alta correlação para todas as falhas, e quando isso acontece, significa que colocar os dois dados para o algoritmo implicaria em estar fornecendo quase a mesma informação. Então, para simplificar, resolvemos colocar a diferença entre elas.

	Type	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Machine failure type	Dif_Temp
0	L	2861	4.6	143	pwf	10.2
2	L	1455	41.3	208	twf	10.1
3	L	1282	60.7	216	osf	9.8
4	L	1412	52.3	218	osf	9.8
5	L	1433	62.3	20	pwf	9.9
...	...	...	...	...	...	...

Figura 4.20: exemplo da base de dados após a engenharia de variáveis

4.6 Aplicando o algoritmo C4.5

Após realizado o tratamento dos dados, uma análise exploratória e a engenharia de variáveis, chega o momento de aplicarmos o algoritmo. Rodamos o algoritmo e obtemos as seguintes regras para cada tipo de falha:

HDF

Árvore_HDF_1: Diferença de Temperatura <= 8.59 e *Rotational Speed* <= 1379

PWF

Árvore_PWF_1: Diferença de Temperatura <= 8.59 e *Tool Wear* <= 176

Árvore_PWF_2: Diferença de Temperatura <= 8.59 e *Tool Wear* > 176 e *Rotational Speed* > 2271

Árvore_PWF_3: Diferença de Temperatura <= 8.59 e *Rotational Speed* > 1379 e *Tool Wear* <= 208 e *Type* = M

Árvore_PWF_4: Diferença de Temperatura <= 8.59 e *Rotational Speed* > 1379 e *Tool Wear* > 208 e *Rotational Speed* > 1497 e o *Type* = L

OSF

Árvore_OSF_1: Diferença de Temperatura <= 8.59 e *Rotational Speed* > 1379 e *Tool Wear* <=208 e o *Type* = L

Árvore_OSF_2: Diferença de Temperatura > 8.59 e *Tool Wear* > 176 e *Rotational Speed* <= 2271 e Torque > 48

Árvore_OSF_3: Diferença de Temperatura > 8.59 e *Tool Wear* > 176 e *Rotational Speed* <= 2271 e Torque <= 48 e *Tool Wear* > 235 e *Type* = L

TWF

Árvore_TWF_1: Diferença de Temperatura <= 8,59 e *Rotational Speed* > 1379 e *Tool Wear* > 208 e *Rotational Speed* <= 1497

Árvore_TWF_2: Diferença de Temperatura <= 8,59 e *Rotational Speed* > 1379 e *Tool Wear* > 208 e *Rotational Speed* > 1497 e *Type* = M

Árvore_TWF_3: Diferença de Temperatura > 8,59 e *Tool Wear* > 176 e *Rotational Speed* <= 2271 e Torque < 48 e *Tool Wear* <= 235

Árvore_TWF_4: Diferença de Temperatura > 8,59 e *Tool Wear* > 176 e *Rotational Speed* <= 2271 e Torque < 48 e *Tool Wear* > 235 e o *Type* = H

Assim, a árvore de decisão geral é representada na figura 4.21:

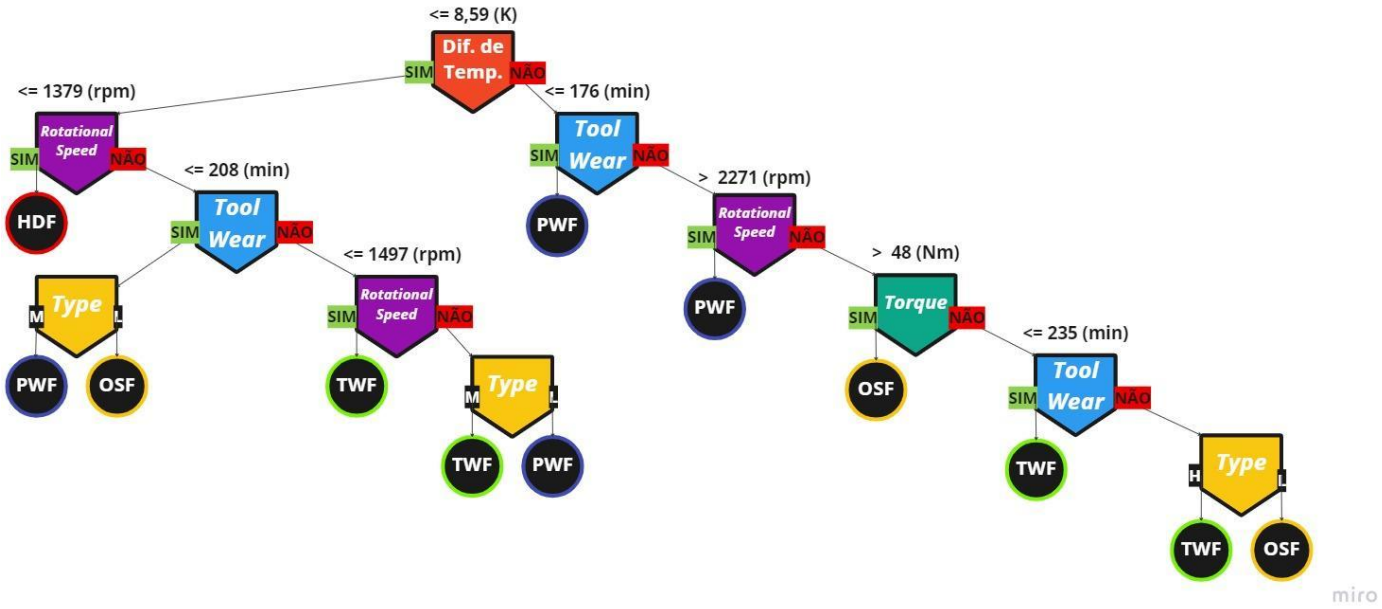

Figura 4.21: Árvore de decisão geral

5. RESULTADOS

5.1 Análise das regras

O intuito deste momento é fazer uma análise qualitativa dos resultados gerados a partir da aplicação do modelo de classificação c4.5. Para isso, será realizada uma comparação das regras obtidas e as regras já conhecidas previamente das quais geraram os dados de forma artificial. Dessa forma, será possível avaliar a eficiência dos resultados. Seguimos então para uma análise de cada tipo de falha separadamente.

Heat Dissipation Failure (HDF)

As regras impostas para gerar os dados artificialmente que levavam a falha por HDF são:

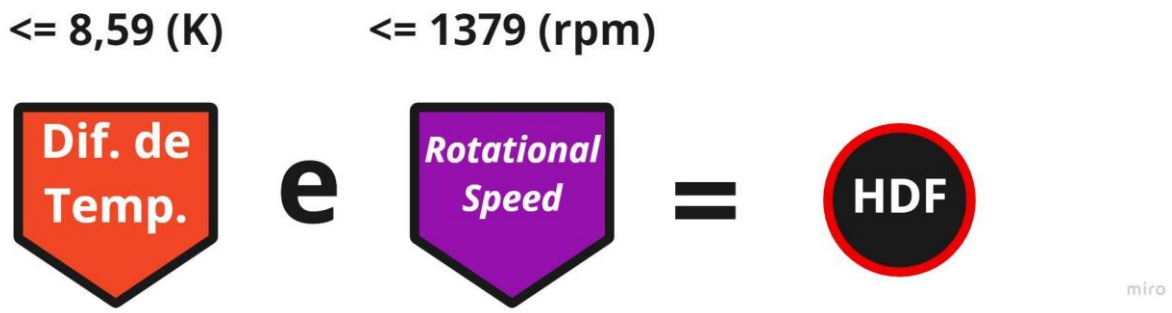

Figura 5.1: regras para gerar as falhas por HDF artificialmente

Para esse tipo de falha foi gerada apenas uma árvore de decisão, e as regras foram:

Árvore_HDF_1: Diferença de Temperatura <= 8.59 e *Rotational Speed* <= 1379

Esse foi o melhor resultado obtido, ele definiu exatamente as regras necessárias para que ocorra a falha. Isso se deve muito pela engenharia de variáveis feita, uma vez que a análise exploratória nos indicou uma forte relação entre os atributos *Air Temperature* e *Process Temperature*, pudemos criar a variável diferença de temperatura. A decisão de criar uma nova variável foi feita para simplificar as regras geradas pelo modelo, já a decisão de substituir as variáveis pela sua diferença, se deve ao fato de conhecermos previamente as regras que geraram os dados. Em uma aplicação em que não fossem conhecidas, outras variáveis poderiam ter sido criadas, como o produto entre elas ou até a soma. Para facilitar o estudo, foi feita a diferença, e isso deve ser levado em conta nos resultados.

O modelo cria a árvore de decisão pelas regras que melhor separam os dados através do ganho de informação como foi explicado no referencial teórico. E dois fatores contribuíram para que essas regras tivessem o maior ganho de informação. Primeiro se deve a amostragem de falhas por HDF ser a mais representativa dentro da base de dados e segundo, por que os atributos de temperatura aparentam ter uma relação apenas com essa falha, enquanto que outros atributos aparentam estar relacionados com mais de uma falha, o que diminui o ganho de informação. Assim, essas regras geram o maior ganho de informação e iniciam a árvore para separar as falhas por HDF. O fato dessas regras estarem no começo da árvore contribuiu para que ficassem mais bem definidas.

Portanto, o resultado obtido foi muito satisfatório para explicar a causa que leva a ocorrência da falha por *Heat Dissipation* em um produto. Nos entregando quase que perfeitamente as mesmas condições impostas para gerar os dados artificialmente.

***Power Failure* (PWF)**

As regras impostas para gerar os dados artificialmente que levavam a falha por PWF são:

Figura 5.2: regras para gerar as falhas por PWF artificialmente

Para esse tipo de falha foram geradas 4 árvores de decisão, e as regras foram:

Árvore_PWF_1: Diferença de Temperatura <= 8.59 e *Tool Wear* <= 176

Árvore_PWF_2: Diferença de Temperatura <= 8.59 e *Tool Wear* > 176 e *Rotational Speed* > 2271

Árvore_PWF_3: Diferença de Temperatura <= 8.59 e *Rotational Speed* > 1379 e *Tool Wear* <= 208 e *Type* = M
Árvore_PWF_4: Diferença de Temperatura <= 8.59 e *Rotational Speed* > 1379 e *Tool Wear* > 208 e *Rotational Speed* > 1497 e o *Type* = L

Em primeiro lugar cabe destacar que as primeiras regras de cada árvore são fruto da separação feita inicialmente que distingue as falhas por HDF. Na sequência, era esperado encontrarmos as regras que definissem as causas do PWF, no entanto, é curioso notar que o atributo Torque, que faz parte das condições que geraram os dados artificialmente, não aparece em nenhuma das árvores. Aparentemente, esse atributo que demonstrou uma relação com a falha na investigação dos dados, não foi necessário para a sua separação.

O modelo parece ter escolhido as regras de forma a eliminar as outras possibilidades de falha do que definir as que são necessárias para a sua ocorrência. Por exemplo, na árvore_PWF_1 ela começa por eliminar a possibilidade de ser um HDF já que a diferença de temperatura é maior que 8,59 (K) e depois elimina a possibilidade de ser um OSF e TWF, com a regra de que o *Tool Wear* deve ser menor do que 176 (min), sendo que sabemos pela análise dos dados que OSF e TWF não ocorreram nenhuma vez em que o valor de *Tool Wear* foi menor que 176 (min). Assim, as regras não nos indicaram diretamente as causas que levam a um PWF, e sim, criou regras que descartam a possibilidade de outras falhas.

A segunda árvore tem uma lógica diferente, ela também elimina as chances de ser falha por HDF, mas considera que o *Tool Wear* seja maior que 176 (min), ou seja, uma região que também pode ocorrer OSF e TWF. Mas a regra 3 dessa árvore nos diz que o *rotational speed* deve ser maior que 2271, um valor alto em relação às demais falhas. Essa regra nos indica uma característica própria das falhas de PWF, na qual o alto *rotational speed* faz com que a potência esteja alta demais para o funcionamento do equipamento.

As árvores 3 e 4 são mais difíceis de interpretar, mas aparentam seguir uma lógica mais parecida com a de número 1, na qual é feita uma segmentação dos atributos de forma a descartar a possibilidade de outras falhas.

Uma possível explicação para que o algoritmo não tenha conseguido definir as regras através dos atributos que de fato causam a falha, é a existência de duas formas distintas para que ela ocorra, quando a potência é baixa ou alta. Isso deve ter dificultado o aprendizado do modelo já que as ocorrências dessa falha apresentavam valores muito distintos e não com uma lógica única. A divisão da falha PWF em dois tipos de falhas diferentes, as com potência alta, e as de potência baixa, contribuiria para uma melhor definição das regras.

***Overstrain Failure* (OSF)**

As regras impostas para gerar os dados artificialmente que levavam a falha por OSF são:

Figura 5.3: regras para gerar as falhas por OSF artificialmente

Para esse tipo de falha foram geradas 3 árvores de decisão, e as regras foram:

Árvore_OSF_1: Diferença de Temperatura <= 8.59 e *Rotational Speed* > 1379 e *Tool Wear* <=208 e o *Type* = L

Árvore_OSF_2: Diferença de Temperatura > 8.59 e *Tool Wear* > 176 e *Rotational Speed* <= 2271 e Torque > 48

Árvore_OSF_3: Diferença de Temperatura > 8.59 e *Tool Wear* > 176 e *Rotational Speed* <= 2271 e Torque <= 48 e *Tool Wear* > 235 e *Type* = L

Sabemos pelas regras que geraram o *dataset* que a falha por OSF é a única que leva em consideração o tipo; L, M ou H. Por esse motivo, era esperado que isso fosse mais evidente nas regras, mas não foi o que o modelo apresentou.

A primeira regra começa por eliminar a chance de HDF e em seguida, apresenta duas regras com atributos que são causas das falhas: *Tool Wear* e o *Type*. As regras nos indicam que quando o *Tool Wear* é menor do que 208 (min) é necessário que o *Type* da peça seja L, o *Type* que apresenta o menor valor da multiplicação entre Torque e *Tool Wear* para ocorrer a falha. Essas regras fazem bastante sentido, pois as outras falhas por OSF que apresentam *Tool Wear* mais altos possuem outro *Type* da peça. Evidenciando nas regras que o *Type* influencia nas falhas.

Na árvore número 2 também elimina a chance de HDF no início, e em seguida impõe que o *Tool Wear* precisa ser maior do que 176(min). Essa segunda regra está bastante ligada às causas do OSF, que como vimos ocorrem em valores mais altos de *Tool Wear*. A regra seguinte, sobre o *Rotational Speed*, parece eliminar a possibilidade de ser um PWF, que necessita de valores maiores que 2271 para acontecer. E por fim, ela se utiliza do atributo Torque para separar as duas opções que sobraram, TWF e OSF. Como vimos na nossa análise exploratória, os dois possuem uma forte causalidade com o *Tool Wear*, mas o OSF ocorre para valores altos de Torque, e esse valor que divide as duas falhas é a última regra, com Torque > 48. Assim, essa é uma regra que nos fornece uma informação valiosa quanto às causas da falha.

A árvore 3 se difere da 2 pois acontece em momentos em que o torque é menor ou igual a 48. E nesses casos, para que ocorra a OSF, são necessárias outras duas regras, *Tool Wear* maior do que 235 e o *Type* ser L. Observando a base de dados, essa parece ter sido uma regra para explicar casos muito específicos, muito provavelmente um *outlier*. Dessa forma, ela não nos traz informações de valor para o nosso objetivo de entender as causas da falha.

***Tool Wear Failure* (TWF)**

As regras impostas para gerar os dados artificialmente que levavam a falha por TWF são:

Figura 5.4: regras para gerar as falhas por TWF artificialmente

Para esse tipo de falha foram geradas 4 árvores de decisão, e as regras foram:

Árvore_TWF_1: Diferença de Temperatura <= 8,59 e *Rotational Speed* > 1379 e *Tool Wear* > 208 e *Rotational Speed* <= 1497

Árvore_TWF_2: Diferença de Temperatura <= 8,59 e *Rotational Speed* > 1379 e *Tool Wear* > 208 e *Rotational Speed* > 1497 e *Type* = M

Árvore_TWF_3: Diferença de Temperatura > 8,59 e *Tool Wear* > 176 e *Rotational Speed* <= 2271 e Torque < 48 e *Tool Wear* <= 235

Árvore_TWF_4: Diferença de Temperatura > 8,59 e Tool Wear > 176 e *Rotational Speed* <= 2271 e Torque < 48 e *Tool Wear* > 235 e o *Type* = H

A falha por TWF é a única que depende de apenas uma das variáveis, o *Tool Wear*. E ela acontece em algum momento entre 200 (min) e 240 (min). As regras geradas foram as que apresentaram a maior quantidade de condições, isso mostra que as falhas por TWF estão nos últimos níveis da árvore de decisão, o que torna mais difícil a interpretação de causalidade das regras. Aparentemente, o modelo definiu as regras por eliminação das outras falhas para chegar em TWF. Apesar de todas as árvores apresentarem o atributo *Tool Wear* elevado como condição, as outras não estão correlacionadas com a falha TWF, e sim, servem para dividir as outras falhas. Portanto, pouco se pode aproveitar dessas árvores para a identificação das causas de defeito no equipamento por TWF.

5.2 Discussão Geral da Análise das Árvores criadas pelo algoritmo C4.5

Nesse momento, retornamos ao objetivo específico proposto no capítulo de introdução deste trabalho, analisar a capacidade do algoritmo C4.5 de identificar as causas dos defeitos das máquinas dentro da base de dados usada. Após ser realizada uma análise individual para cada tipo de falha, e de cada árvore que foi gerada, concluímos que o modelo, sozinho, não é capaz de explicar os atributos que causam as falhas.

É importante ressaltar, que ele não é capaz de explicar sozinho as causas, porém, com um bom estudo prévio da base de dados é possível obter informações valiosas. A interpretação das árvores geradas só foi possível devido à exploração da base que foi realizada anteriormente, sem esse conhecimento prévio, não seria possível dizer quais regras das árvores estavam bem relacionadas com a falha.

Foi observado que nas árvores geradas algumas das regras explicam bem as causas das falhas enquanto que outras servem para descartar a possibilidade de outras falhas. Isso gera uma dificuldade de interpretação das árvores pois é preciso voltar para os *insight* obtidos na exploração e aos poucos ir determinando a que falha a regra está tentando definir ou eliminar. Dessa forma, na árvore de regras e uma falha pode conter informações valiosas sobre outras falhas também.

Outro ponto interessante é que as árvores menores, aquelas que estão na raiz da árvore principal, são as que apresentam mais diretamente e definem melhor as regras para que ocorram a falha. Enquanto as árvores que estão no final da árvore principal e são mais longas apresentam muito ruídos de regras que serviram mais para descartar outras possibilidades. Essa informação contribui para uma interpretação mais precisa das árvores.

Além disso, é importante dizer também que algumas árvores não contribuíram para a descoberta das causas das falhas por explicaram casos específicos demais para a base de dados. Isso ocorre devido a existência de *outliers*, ocorrências que destoam do geral. Por apresentarem critérios muito específicos, essas árvores não são interessantes para explicar a relação de causa e efeito de um modo geral.

Portanto, por mais que o algoritmo sozinho não foi capaz de definir as causas para que ocorram as falhas, através da interpretação das árvores geradas, baseada na análise exploratória realizada previamente, foi possível colher informações valiosas e que explicam as relações de causa e efeito dos defeitos das máquinas.

5.3 Discussão sobre a aplicabilidade para o uso na gestão da manutenção

A gestão da manutenção consiste em um complexo sistema que visa reduzir os impactos dos desgastes e quebras de máquinas na produção. Como foi exposto no capítulo de introdução, um bom conhecimento das causas que levam uma máquina a falhar serve de grande ajuda para a criação de planos estratégicos para manutenção. Nesse contexto, voltamos para a pergunta levantada nos objetivos gerais do trabalho: o aprendizado de máquina é capaz de colaborar para a interpretação das causas de defeitos em equipamentos?

Diante dos resultados obtidos e expostos neste capítulo chegamos à conclusão de que o aprendizado de máquina fornece informações valiosas para o entendimento das causas de falhas de máquinas. Através de uma metodologia já reconhecida para projetos de ciência de dados, usamos um modelo específico de classificação para realizar o estudo. E mesmo que tenha sido usado apenas um modelo para realizar essa avaliação, já foi possível chegar a conclusão de que o *machine learning* tem muito para acrescentar na gestão da manutenção.

Para exemplificar os potenciais usos das informações obtidas na base de dados vamos apresentar uma proposta de ação para cada um dos 3 principais tipos de manutenção expostos no referencial teórico:

Manutenção Corretiva:

Neste caso, como a falha já ocorreu, é possível analisar os dados de funcionamento da máquina no momento da quebra e percorrer a árvore de decisão geral. Dessa forma, uma resposta mais rápida e eficaz seria feita para a identificação do problema e reparação do defeito.

Manutenção Planejada:

Para evitar os casos em que ocorre falha por TWF, seria possível estipular um tempo de *Tool Wear* específico para realizar a substituição da peça. A sugestão de acordo com as árvores de decisão geradas, seria o valor de *Tool Wear* de 176 (min).

Manutenção Preditiva:

Para a manutenção preditiva seria possível realizar um monitoramento dos dados de funcionamento da máquina e criar zonas de perigo antes dos valores obtidos. Essas zonas seriam com valores próximos dos que encontramos nas árvores para as falhas. Quando os dados indicavam uma falha em eminência, a devida ação seria tomada para evitar a falha.

6. CONCLUSÕES

A gestão da manutenção foi se tornando mais complexa na medida que a indústria foi crescendo. Os processos e técnicas que surgiram visavam otimizar as linhas de produção para aumentar a produtividade e gerar margens de lucros maiores. Nesse contexto, muitas empresas começaram a utilizar o *machine learning* no universo da manutenção como ferramenta de apoio para análises e tomada de decisão.

No presente trabalho, foi realizado um estudo acerca das capacidades que esse instrumento oferece para resolver um dos grandes desafios que a gerência da manutenção encontra: a identificação das causas que levam uma máquina a falhar. Para isso, foi aplicado um algoritmo de classificação conhecido como c4.5 para gerar uma árvore de decisão das falhas, e a partir de uma análise dela identificar as causas dos defeitos.

Foi observado que esse método é de grande valor para a identificação das razões para os defeitos. No entanto, uma análise simples da árvore não serve de grande ajuda. É preciso que seja feita uma análise exploratória dos dados previamente para que seja possível interpretar á árvore gerada da melhor forma possível. Entender as relações que existem entre os dados de funcionamento da máquina, como eles se comportam em cada tipo de falha e como se diferenciam entre as falhas foram noções fundamentais para a compreensão da árvore de decisão. Além disso, as condições impostas na árvore serviam para caracterizar uma falha, ou para excluir a possibilidade de ser alguma outra. Nesse sentido, entender essa dinâmica fornecia informações valiosas para distinguir para qual defeito aquela condição estava relacionada. Ao fim, foi possível chegar muito próximo das mesmas regras para que ocorram as falhas usadas na geração da base de dados. Evidenciando assim, a capacidade que o algoritmo c4.5 mostrou para explicar as razões dos defeitos das máquinas do *dataset*.

O aprendizado de máquina então, demonstrou um grande potencial para auxiliar na gestão da manutenção. Identificando qual tipo de manutenção que se deseja utilizar a ferramenta possibilita diversas formas de uso que agregam valor para a linha de produção. Seria interessante que em trabalhos futuros fossem investigadas mais formas de se utilizar esta ferramenta. Dessa forma, trabalhos como este poderiam continuar estendendo os conhecimentos acerca da manutenção e novos métodos e processos seriam criados visando aumentar a produtividade nas indústrias.

7. REFERÊNCIAS

BLK SISTEMAS. O impacto do diagnóstico de manutenção na produtividade. www.blsistemas.com.br. Disponível em: http://www.blsistemas.com.br/o-impacto-do-diagnostico-de-manutencao-na-produtividade/. Acesso em: nov/2022

CHAMBERS, John M. et al. Graphical methods for data analysis. Chapman and Hall/CRC, 2018.

FLAVIO AUGUSTO, Brasil, Lean Institute. «Coluna - Manufatura avançada e lean: tecnologia com propósito claro e tamanho certo». www.lean.org.br. Consultado em 6 de agosto de 2018 ((Artigo publicado na Revista Máquinas e Equipamentos, da ABIMAQ, Mar/Abr 2018, Edição 07, Ano 02, pgs. 70-71))

KARDEC, Alan; NASCIF, Júlio. Manutenção: FunçãoEstratégica. 2.ªed. Riode Janeiro: Qualitymark Ed., 2001.

MITCHELL, Tom M.; MITCHELL, Tom M. Machine learning. New York: McGraw-hill, 1997.

PEREIRA, V. DS-Classification-04-C4.5. Research Gate, 2020. Disponível em: https://www.researchgate.net/publication/344389246_DS-Classification-04-C45. Acesso em: nove/2022

MATZKA, Stephan. Explainable artificial intelligence for predictive maintenance applications. In: 2020 Third International Conference on Artificial Intelligence for Industries (AI4I). IEEE, 2020. p. 69-74.

MOUBRAY, J. Reliability-centered maintenance: second edition. 2ª. ed. New York: Industrial Press Inc., 1997.

QUINLAN, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

SAMUEL, Arthur L. Machine learning. The Technology Review, v. 62, n. 1, p. 42-45, 1959.

ROSS, Sheldon M. Introduction to probability and statistics for engineers and scientists. Academic press, 2020.

SHAI, S.Shwartz; SHAI, B. David. Understanding Machine Learning: From Theory to Algorithms. Local de publicação: Cambridge Press, 2014.

SHEARER, C. The CRISP-DM model: the new blueprint for data mining, J Data Warehousing, 2000; 5:13—22.

SHIMMIN, B. Operacionalizar o machine learning. www.hpe.com, 2020. Disponível em: https://www.hpe.com/br/pt/collaterals/collateral.a00108395.The-Industrialization-of-AI-business-white-paper.html?rpv=cpf&parentPage=/br/pt/what-is/machine-learning . Acesso em: nov/2022.

WILKINSON, Leland; FRIENDLY, Michael. The history of the cluster heat map. The American Statistician, v. 63, n. 2, p. 179-184, 2009.

ANEXOS

Código python

```
# UFF - Universidade Federal Fluminense (Brazil)
# Created by: Prof. Valdecy Pereira, D.Sc.
# Lesson: Association Rules

# email:  valdecy.pereira@gmail.com
# GitHub Repository: <https://github.com/Valdecy>

# Required Libraries
import numpy  as np
import pandas as pd

from copy import deepcopy
from random import randint
from sklearn.metrics import classification_report
from sklearn.model_selection import KFold
from scipy import stats

# Function: Returns True, if a Column is Numeric
def is_number(string):
    for i in range(0, len(string)):
        if pd.isnull(string[i]) == False:
            try:
                float(string[i])
                return True
            except ValueError:
                return False

# Function: Returns True, if a Value is Numeric
def is_number_value(value):
    if pd.isnull(value) == False:
        try:
            float(value)
            return True
        except ValueError:
            return False

# Function: Performs a Chi_Squared Test or Fisher Exact Test
def chi_squared_test(label_df, feature_df):
    label_df.reset_index(drop=True, inplace=True)
    feature_df.reset_index(drop=True, inplace=True)
    data = pd.concat([pd.DataFrame(label_df.values.reshape((label_df.shape[0], 1))),
feature_df], axis = 1)
```

```
    data.columns=["label", "feature"]
    contigency_table = pd.crosstab(data.iloc[:,0], data.iloc[:,1], margins = False)
    m = contigency_table.values.sum()
    if m <= 10000 and contigency_table.shape == (2,2):
        p_value = stats.fisher_exact(contigency_table)
    else:
        p_value = stats.chi2_contingency(contigency_table, correction = False) # (No Yates'
Correction)
    return p_value[1]

# Function: Prediction
def prediction_dt_c45(model, Xdata):
    Xdata = Xdata.reset_index(drop=True)
    ydata = pd.DataFrame(index=range(0, Xdata.shape[0]), columns=["Prediction"])
    data  = pd.concat([ydata, Xdata], axis = 1)
    rule = []

    # Preprocessing - Boolean Values
    for j in range(0, data.shape[1]):
        if data.iloc[:,j].dtype == "bool":
            data.iloc[:,j] = data.iloc[:, j].astype(str)

    dt_model = deepcopy(model)

    for i in range(0, len(dt_model)):
        dt_model[i] = dt_model[i].replace("{", "")
        dt_model[i] = dt_model[i].replace("}", "")
        dt_model[i] = dt_model[i].replace(";", "")
        dt_model[i] = dt_model[i].replace("IF ", "")
        dt_model[i] = dt_model[i].replace("AND", "")
        dt_model[i] = dt_model[i].replace("THEN", "")
        dt_model[i] = dt_model[i].replace("=", "")
        dt_model[i] = dt_model[i].replace("<", "<=")

    for i in range(0, len(dt_model) -2):
        splited_rule = [x for x in dt_model[i].split(" ") if x]
        rule.append(splited_rule)

    for i in range(0, Xdata.shape[0]):
        for j in range(0, len(rule)):
            rule_confirmation = len(rule[j])/2 - 1
            rule_count = 0
            for k in range(0, len(rule[j]) - 2, 2):
                if is_number_value(data[rule[j][k]][i]) == False:
```

```
                if (data[rule[j][k]][i] in rule[j][k+1]):
                    rule_count = rule_count + 1
                    if (rule_count == rule_confirmation):
                        data.iloc[i,0] = rule[j][len(rule[j]) - 1]
                else:
                    k = len(rule[j])
            elif is_number_value(data[rule[j][k]][i]) == True:
                if rule[j][k+1].find("<=") == 0:
                    if data[rule[j][k]][i] <= float(rule[j][k+1].replace("<=", "")):
                        rule_count = rule_count + 1
                        if (rule_count == rule_confirmation):
                            data.iloc[i,0] = rule[j][len(rule[j]) - 1]
                    else:
                        k = len(rule[j])
                elif rule[j][k+1].find(">") == 0:
                    if data[rule[j][k]][i] > float(rule[j][k+1].replace(">", "")):
                        rule_count = rule_count + 1
                        if (rule_count == rule_confirmation):
                            data.iloc[i,0] = rule[j][len(rule[j]) - 1]
                    else:
                        k = len(rule[j])

    for i in range(0, Xdata.shape[0]):
        if pd.isnull(data.iloc[i,0]):
            data.iloc[i,0] = dt_model[len(dt_model)-1]

    return data

# Function: Calculates the Information Gain Ratio
def info_gain_ratio(target, feature = [], uniques = []):
    entropy = 0
    denominator_1 = feature.count()
    data = pd.concat([pd.DataFrame(target.values.reshape((target.shape[0], 1))), feature], axis
= 1)
    for entp in range(0, len(np.unique(target))):
        numerator_1 = data.iloc[:,0][(data.iloc[:,0] == np.unique(target)[entp])].count()
        if numerator_1 > 0:
            entropy = entropy - (numerator_1/denominator_1)*
np.log2((numerator_1/denominator_1))
    info_gain = float(entropy)
    info_gain_r = 0
    intrinsic_v = 0
    for word in range(0, len(uniques)):
        denominator_2 = feature[(feature == uniques[word])].count()
```

```
        if denominator_2[0] > 0:
            intrinsic_v = intrinsic_v - (denominator_2/denominator_1)*
np.log2((denominator_2/denominator_1))
        for lbl in range(0, len(np.unique(target))):
            numerator_2 = data.iloc[:,0][(data.iloc[:,0] == np.unique(target)[lbl]) & (data.iloc[:,1]
== uniques[word])].count()
            if numerator_2 > 0:
                info_gain = info_gain +
(denominator_2/denominator_1)*(numerator_2/denominator_2)*
np.log2((numerator_2/denominator_2))
    if intrinsic_v[0] > 0:
        info_gain_r = info_gain/intrinsic_v
    return float(info_gain_r)

# Function: Binary Split on Continuous Variables
def split_me(feature, split):
    result = pd.DataFrame(feature.values.reshape((feature.shape[0], 1)))
    for fill in range(0, len(feature)):
        result.iloc[fill,0] = feature.iloc[fill]
    lower = "<=" + str(split)
    upper = ">" + str(split)
    for convert in range(0, len(feature)):
        if float(feature.iloc[convert]) <= float(split):
            result.iloc[convert,0] = lower
        else:
            result.iloc[convert,0] = upper
    binary_split = []
    binary_split = [lower, upper]
    return result, binary_split

# Function: C4.5 Algorithm
def dt_c45(Xdata, ydata, cat_missing = "none", num_missing = "none", pre_pruning =
"none", chi_lim = 0.1, min_lim = 5):

    ################     Part 1 - Preprocessing    ##############################
    # Preprocessing - Creating Dataframe
    name = ydata.name
    ydata = pd.DataFrame(ydata.values.reshape((ydata.shape[0], 1)))
    for j in range(0, ydata.shape[1]):
        if ydata.iloc[:,j].dropna().value_counts().index.isin([0,1]).all():
            for i in range(0, ydata.shape[0]):
              if ydata.iloc[i,j] == 0:
                  ydata.iloc[i,j] = "zero"
              else:
```

```
            ydata.iloc[i,j] = "one"
dataset = pd.concat([ydata, Xdata], axis = 1)

 # Preprocessing - Boolean Values
for j in range(0, dataset.shape[1]):
    if dataset.iloc[:,j].dtype == "bool":
        dataset.iloc[:,j] = dataset.iloc[:, j].astype(str)

# Preprocessing - Missing Values
if cat_missing != "none":
    for j in range(1, dataset.shape[1]):
        if is_number(dataset.iloc[:, j]) == False:
            for i in range(0, dataset.shape[0]):
                if pd.isnull(dataset.iloc[i,j]) == True:
                    if cat_missing == "missing":
                        dataset.iloc[i,j] = "Unknow"
                    elif cat_missing == "most":
                        dataset.iloc[i,j] = dataset.iloc[:,j].value_counts().idxmax()
                    elif cat_missing == "remove":
                        dataset = dataset.drop(dataset.index[i], axis = 0)
                    elif cat_missing == "probability":
                        while pd.isnull(dataset.iloc[i,j]) == True:
                            dataset.iloc[i,j] = dataset.iloc[randint(0, dataset.shape[0] - 1), j]
elif num_missing != "none":
        if is_number(dataset.iloc[:, j]) == True:
            for i in range(0, dataset.shape[0]):
                if pd.isnull(dataset.iloc[i,j]) == True:
                    if num_missing == "mean":
                        dataset.iloc[i,j] = dataset.iloc[:,j].mean()
                    elif num_missing == "median":
                        dataset.iloc[i,j] = dataset.iloc[:,j].median()
                    elif num_missing == "most":
                        dataset.iloc[i,j] = dataset.iloc[:,j].value_counts().idxmax()
                    elif cat_missing == "remove":
                        dataset = dataset.drop(dataset.index[i], axis = 0)
                    elif num_missing == "probability":
                        while pd.isnull(dataset.iloc[i,j]) == True:
                            dataset.iloc[i,j] = dataset.iloc[randint(0, dataset.shape[0] - 1), j]

# Preprocessing - Unique Words List
unique = []
uniqueWords = []
for j in range(0, dataset.shape[1]):
    for i in range(0, dataset.shape[0]):
```

```
            token = dataset.iloc[i, j]
            if not token in unique:
                unique.append(token)
        uniqueWords.append(unique)
        unique = []

    # Preprocessing - Label Matrix
    label = np.array(uniqueWords[0])
    label = label.reshape(1, len(uniqueWords[0]))

    ################    Part 2 - Initialization    ##############################
    # C4.5 - Initializing Variables
    i = 0
    impurity = 0
    branch = [None]*1
    branch[0] = dataset
    gain_ratio = np.empty([1, branch[i].shape[1]])
    lower = "0"
    root_index = 0
    rule = [None]*1
    rule[0] = "IF "
    skip_update = False
    stop = 2
    upper = "1"

    ################     Part 3 - C4.5 Algorithm    ##############################
    # C4.5 - Algorithm
    while (i < stop):
        impurity = np.amax(gain_ratio)
        gain_ratio.fill(0)
        for element in range(1, branch[i].shape[1]):
            if len(branch[i]) == 0:
                skip_update = True
                break
            if len(np.unique(branch[i][0])) == 1 or len(branch[i]) == 1:
                if ";" not in rule[i]:
                    rule[i] = rule[i] + " THEN " + name + " = " + branch[i].iloc[0, 0] + ";"
                    rule[i] = rule[i].replace(" AND  THEN ", " THEN ")
                skip_update = True
                break
            if i > 0 and is_number(dataset.iloc[:, element]) == False and pre_pruning == "chi_2"
and chi_squared_test(branch[i].iloc[:, 0], branch[i].iloc[:, element]) > chi_lim:
                if ";" not in rule[i]:
```

```
                rule[i] = rule[i] + " THEN " + name + " = " + branch[i].agg(lambda
x:x.value_counts().index[0])[0] + ";"
                rule[i] = rule[i].replace(" AND  THEN ", " THEN ")
             skip_update = True
             continue
          if is_number(dataset.iloc[:, element]) == True:
             gain_ratio[0, element] = 0.0
             value = np.sort(branch[i].iloc[:, element].unique())
             skip_update = False
             if branch[i][(branch[i].iloc[:, element] == value[0])].count()[0] > 1:
                start = 0
                finish = len(branch[i].iloc[:, element].unique()) - 2
             else:
                start = 1
                finish = len(branch[i].iloc[:, element].unique()) - 2
             if len(branch[i]) == 2 or len(value) == 1 or len(value) == 2:
                start = 0
                finish = 1
             if len(value) == 3:
                start = 0
                finish = 2
             for bin_split in range(start, finish):
                bin_sample = split_me(feature = branch[i].iloc[:, element], split =
value[bin_split])
                if i > 0 and pre_pruning == "chi_2" and chi_squared_test(branch[i].iloc[:, 0],
bin_sample[0]) > chi_lim:
                   if ";" not in rule[i]:
                      rule[i] = rule[i] + " THEN " + name + " = " + branch[i].agg(lambda
x:x.value_counts().index[0])[0] + ";"
                      rule[i] = rule[i].replace(" AND  THEN ", " THEN ")
                   skip_update = True
                   continue
                igr = info_gain_ratio(target = branch[i].iloc[:, 0], feature = bin_sample[0],
uniques = bin_sample[1])
                if igr > float(gain_ratio[0, element]):
                   gain_ratio[0, element] = igr
                   uniqueWords[element] = bin_sample[1]
          if is_number(dataset.iloc[:, element]) == False:
             gain_ratio[0, element] = 0.0
             skip_update = False
             igr = info_gain_ratio(target = branch[i].iloc[:, 0], feature =
pd.DataFrame(branch[i].iloc[:, element].values.reshape((branch[i].iloc[:, element].shape[0],
1))), uniques = uniqueWords[element])
             gain_ratio[0, element] = igr
```

```
        if i > 0 and pre_pruning == "min" and len(branch[i]) <= min_lim:
            if ";" not in rule[i]:
                rule[i] = rule[i] + " THEN " + name + " = " + branch[i].agg(lambda
x:x.value_counts().index[0])[0] + ";"
                rule[i] = rule[i].replace(" AND  THEN ", " THEN ")
            skip_update = True
            continue

    if i > 0 and pre_pruning == "impur" and np.amax(gain_ratio) < impurity and
np.amax(gain_ratio) > 0:
        if ";" not in rule[i]:
            rule[i] = rule[i] + " THEN " + name + " = " + branch[i].agg(lambda
x:x.value_counts().index[0])[0] + ";"
            rule[i] = rule[i].replace(" AND  THEN ", " THEN ")
        skip_update = True
        continue

    if skip_update == False:
        root_index = np.argmax(gain_ratio)
        rule[i] = rule[i] + str(list(branch[i])[root_index])

        for word in range(0, len(uniqueWords[root_index])):
            uw = uniqueWords[root_index][word].replace("<=", "")
            uw = uw.replace(">", "")
            lower = "<=" + uw
            upper = ">" + uw
            if uniqueWords[root_index][word] == lower:
                branch.append(branch[i][branch[i].iloc[:, root_index] <= float(uw)])
            elif uniqueWords[root_index][word] == upper:
                branch.append(branch[i][branch[i].iloc[:, root_index]  > float(uw)])
            else:
                branch.append(branch[i][branch[i].iloc[:, root_index] ==
uniqueWords[root_index][word]])

            rule.append(rule[i] + " = " + "{" + uniqueWords[root_index][word] + "}")

        for logic_connection in range(1, len(rule)):
            if len(np.unique(branch[i][0])) != 1 and rule[logic_connection].endswith(" AND ")
== False  and rule[logic_connection].endswith("}") == True:
                rule[logic_connection] = rule[logic_connection] + " AND "
    skip_update = False
    i = i + 1
    print("iteration: ", i)
    stop = len(rule)
```

```
for i in range(len(rule) - 1, -1, -1):
    if rule[i].endswith(";") == False:
        del rule[i]

rule.append("Total Number of Rules: " + str(len(rule)))
rule.append(dataset.agg(lambda x:x.value_counts().index[0])[0])
print("End of Iterations")

return rule
```

APÊNDICE – A

```
# Loading Dataset
dataset = pd.read_csv('https://github.com/Valdecy/Datasets/raw/master/Data%20Science/AI-36-Machine%20Failure.csv', sep = ';')
dataset

# MF Dataset
dataset_mf = dataset[dataset['Machine failure'] == 1]
dataset_mf

# Separation (X, y) = (Independent Variables, Dependent Variable)
X = dataset_mf.iloc[:,2:8]
X = X.reset_index(drop = True)
y = dataset_mf.iloc[:,-1]
y = y.apply(str) # The Target Variable must be a String
y = y.reset_index(drop = True)

# MF X
X

# MF y
y

# MF X + y
df_mf = X.join(y)
df_mf

# Criando os dataset de cada tipo de falha
df_hdf = df_mf[df_mf['Machine failure type']=='hdf']
df_pwf = df_mf[df_mf['Machine failure type']=='pwf']
df_osf = df_mf[df_mf['Machine failure type']=='osf']
df_twf = df_mf[df_mf['Machine failure type']=='twf']

#Inicio da análise exploratória

#importando bibliotecas para contrução dos gráficos
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

#Analisando a distribuição das falhas
df_mf['Machine failure
type'].value_counts().plot(kind="bar",figsize=(12,9),title='Distribuição dos tipos de
```

```
falha',xlabel='Tipo de
falha',ylabel='Quantidade',rot='horizontal',fontsize=15,sort_columns=True)

plt.show()
df_mf['Machine failure type'].value_counts()

#Algumas das falhas acontecem em paralelo e poucas vezes em relação ao todo. Portanto vamos ignorar essas falhas e investigar como cada uma delas acontece individualmente.
df_mf2 = df_mf.loc[df_mf['Machine failure type']!='pwf_osf']
df_mf3 = df_mf2.loc[df_mf2['Machine failure type']!='hdf_osf']
df_mf4 = df_mf3.loc[df_mf3['Machine failure type']!='twf_osf']
df_mf5 = df_mf4.loc[df_mf4['Machine failure type']!='twf_mf']
df_mf6 = df_mf5.loc[df_mf5['Machine failure type']!='twf_pwf_osf']
df_mf7 = df_mf6.loc[df_mf6['Machine failure type']!='twf_rnf']
df_mf_f = df_mf7.loc[df_mf7['Machine failure type']!='hdf_pwf']
df_mf_f.info()

df_mf_f['Machine failure
type'].value_counts().plot(kind=''bar'',figsize=(12,9),title='Distribuição dos tipos de
falha',xlabel='Tipo de
falha',ylabel='Quantidade',rot='horizontal',fontsize=15,sort_columns=True)

plt.show()
df_mf_f['Machine failure type'].value_counts()

# Analises das variáveis  em função da falha

# Variáveis utilizadas
var_usadas = df_mf_f.set_index(['Machine failure type']).columns.tolist()

for var in var_usadas:
  fig = px.box(df_mf_f, x=var, y='Machine failure type', points=False)
  fig.update_traces(orientation='h')
  fig.update_layout(
      title=f'Distribuição da {var} por Falha',
      xaxis={'title': var},
      yaxis={'title': 'Falha', 'showline': False, 'ticks': ''}
  )

  fig.update_layout(
      width=800,
      height=400,
      showlegend=False)
fig.show()
```

```
#Fazendo um gráfico radial das variáveis
fig = px.line_polar(df_twf, r='Machine failure type', theta='Rotational speed [rpm]',
line_close=True)
fig.show()

#entendo a correlação entre os atributos
corrmat = df_mf_f.corr()
ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, annot=True, square=True);

#Investigando mais profundamente a relação entre as variáveis usando o scater plot
scar = px.scatter(df_mf_f, x = 'Rotational speed [rpm]', y = 'Torque [Nm]', color = 'Machine
failure type', log_x = True, width = 800)
scar.show()
scar2 = px.scatter(df_mf_f, x = 'Rotational speed [rpm]', y = 'Air temperature [K]', color =
'Machine failure type', log_x = True, width = 800)
scar2.show()
scar3 = px.scatter(df_mf_f, x = 'Rotational speed [rpm]', y = 'Process temperature [K]', color
= 'Machine failure type', log_x = True, width = 800)
scar3.show()
scar4 = px.scatter(df_mf_f, x = 'Rotational speed [rpm]', y = 'Tool wear [min]', color =
'Machine failure type', log_x = True, width = 800)
scar4.show()

#Rotational speed [rpm]
#Torque [Nm]
#Air temperature [K]
#Process temperature [K]
#Tool wear [min]


#Investigando mais profundamente a relação entre as variáveis usando o scater plot
scar = px.scatter(df_mf_f, x = 'Torque [Nm]', y = 'Rotational speed [rpm]', color = 'Machine
failure type', log_x = True, width = 800)
scar.show()
scar2 = px.scatter(df_mf_f, x = 'Torque [Nm]', y = 'Air temperature [K]', color = 'Machine
failure type', log_x = True, width = 800)
scar2.show()
scar3 = px.scatter(df_mf_f, x = 'Torque [Nm]', y = 'Process temperature [K]', color =
'Machine failure type', log_x = True, width = 800)
scar3.show()
scar4 = px.scatter(df_mf_f, x = 'Torque [Nm]', y = 'Tool wear [min]', color = 'Machine failure
type', log_x = True, width = 800)
```

```
scar4.show()

#Rotational speed [rpm]
#Torque [Nm]
#Air temperature [K]
#Process temperature [K]
#Tool wear [min]

#Investigando mais profundamente a relação entre as variáveis usando o scater plot
scar = px.scatter(df_mf_f, x = 'Air temperature [K]', y = 'Rotational speed [rpm]', color =
'Machine failure type', log_x = True, width = 800)
scar.show()
scar2 = px.scatter(df_mf_f, x = 'Air temperature [K]', y = 'Torque [Nm]', color = 'Machine
failure type', log_x = True, width = 800)
scar2.show()
scar3 = px.scatter(df_mf_f, x = 'Air temperature [K]', y = 'Process temperature [K]', color =
'Machine failure type', log_x = True, width = 800)
scar3.show()
scar4 = px.scatter(df_mf_f, x = 'Air temperature [K]', y = 'Tool wear [min]', color = 'Machine
failure type', log_x = True, width = 800)
scar4.show()

#Rotational speed [rpm]
#Torque [Nm]
#Air temperature [K]
#Process temperature [K]
#Tool wear [min]

#Investigando mais profundamente a relação entre as variáveis usando o scater plot
scar = px.scatter(df_mf_f, x = 'Process temperature [K]', y = 'Rotational speed [rpm]', color =
'Machine failure type', log_x = True, width = 800)
scar.show()
scar2 = px.scatter(df_mf_f, x = 'Process temperature [K]', y = 'Torque [Nm]', color =
'Machine failure type', log_x = True, width = 800)
scar2.show()
scar3 = px.scatter(df_mf_f, x = 'Process temperature [K]', y = 'Air temperature [K]', color =
'Machine failure type', log_x = True, width = 800)
scar3.show()
scar4 = px.scatter(df_mf_f, x = 'Process temperature [K]', y = 'Tool wear [min]', color =
'Machine failure type', log_x = True, width = 800)
scar4.show()

#Rotational speed [rpm]
#Torque [Nm]
```

```
#Air temperature [K]
#Process temperature [K]
#Tool wear [min]

#Investigando mais profundamente a relação entre as variáveis usando o scater plot
scar = px.scatter(df_mf_f, x = 'Tool wear [min]', y = 'Rotational speed [rpm]', color =
'Machine failure type', log_x = True, width = 800)
scar.show()
scar2 = px.scatter(df_mf_f, x = 'Tool wear [min]', y = 'Torque [Nm]', color = 'Machine failure
type', log_x = True, width = 800)
scar2.show()
scar3 = px.scatter(df_mf_f, x = 'Tool wear [min]', y = 'Air temperature [K]', color = 'Machine
failure type', log_x = True, width = 800)
scar3.show()
scar4 = px.scatter(df_mf_f, x = 'Tool wear [min]', y = 'Process temperature [K]', color =
'Machine failure type', log_x = True, width = 800)
scar4.show()

#Rotational speed [rpm]
#Torque [Nm]
#Air temperature [K]
#Process temperature [K]
#Tool wear [min]

#Criando um gráfico de coordenadas paralelas

#fig = px.parallel_coordinates(df_mf, color = 'Machine failure type')

fig = px.parallel_coordinates(df_mf_f)
fig.update_layout(
   title={
      'text': "Cluster analysis",
      'y':1
      },
   height=500,
   width=1000)
fig.show()

#HDF: observamos que o HDF tem relação com os atributos de AIR TEMP e PROCESS
TEMP
#PWF: observamos que o PWF tem relação com os atributos de ROT SPEED e TORQUE
#OSF: observamos que o OSF tem relação com os atributos TOOL WEAR
#TWF: observamos que o TWF tem relação com os atributos TOOL WEAR
```

#Para simplificarmos então a árvore de decisão que vai ser gerada, decidimos criar novos atributos que juntem essas variáveis

#Criando a variável Diferença de temperatura entre o processo e o ar

```
df_mf_f['Dif_Tempo'] = df_mf_f['Process temperature [K]'] - df_mf_f['Air temperature [K]']
df_mf_f=df_mf_f.drop(columns=['Process temperature [K]', 'Air temperature [K]'])
df_mf_f
```

#Criando a variável potência que é a Rot Speed multiplicado pelo Torque

```
df_mf_f['Potencia'] = df_mf_f['Torque [Nm]']*df_mf_f['Rotational speed [rpm]']
df_mf_f=df_mf_f.drop(columns=['Torque [Nm]', 'Rotational speed [rpm]'])
df_mf_f
```

```
# Separation (X, y) = (Independent Variables, Dependent Variable)
#X = df_mf_f.iloc[:,3:5].join(df_mf_f.iloc[:,:2])
#X = X.reset_index(drop = True)
#y = df_mf_f.iloc[:,2:3]
#y = y.apply(str) # The Target Variable must be a String
#y = y.reset_index(drop = True)
```

```
df_mf_f = df_mf_f[['Type','Rotational speed [rpm]','Torque [Nm]','Dif_Tempo','Tool wear [min]','Machine failure type']]
```

```
X = df_mf_f.iloc[:,:5]
X = X.reset_index(drop = True)
y = df_mf_f.iloc[:,-1]
y = y.apply(str) # The Target Variable must be a String
y = y.reset_index(drop = True)
```

```
y
```

```
# Run C4.5
dt_model = dt_c45(Xdata = X, ydata = y)
```

```
# Rules
dt_model
```

IF Dif_Tempo = {<=8.599999999999966} AND Rotational speed [rpm] = {<=1379} THEN Machine failure type = hdf

IF Dif_Tempo = {>8.599999999999966} AND Tool wear [min] = {<=176} THEN Machine failure type = pwf

IF Dif_Tempo = {>8.599999999999966} AND Tool wear [min] = {>176} AND Rotational speed [rpm] = {>2271} THEN Machine failure type = pwf
IF Dif_Tempo = {<=8.599999999999966} AND Rotational speed [rpm] = {>1379} AND Tool wear [min] = {<=208} AND Type = {M} THEN Machine failure type = pwf
IF Dif_Tempo = {<=8.599999999999966} AND Rotational speed [rpm] = {>1379} AND Tool wear [min] = {>208} AND Rotational speed [rpm] = {>1497} AND Type = {L} THEN Machine failure type = pwf

IF Dif_Tempo = {<=8.599999999999966} AND Rotational speed [rpm] = {>1379} AND Tool wear [min] = {<=208} AND Type = {L} THEN Machine failure type = osf
IF Dif_Tempo = {>8.599999999999966} AND Tool wear [min] = {>176} AND Rotational speed [rpm] = {<=2271} AND Torque [Nm] = {>48.0} THEN Machine failure type = osf
IF Dif_Tempo = {>8.599999999999966} AND Tool wear [min] = {>176} AND Rotational speed [rpm] = {<=2271} AND Torque [Nm] = {<=48.0} AND Tool wear [min] = {>235} AND Type = {L} THEN Machine failure type = osf

IF Dif_Tempo = {<=8.599999999999966} AND Rotational speed [rpm] = {>1379} AND Tool wear [min] = {>208} AND Rotational speed [rpm] = {<=1497} THEN Machine failure type = twf
IF Dif_Tempo = {<=8.599999999999966} AND Rotational speed [rpm] = {>1379} AND Tool wear [min] = {>208} AND Rotational speed [rpm] = {>1497} AND Type = {M} THEN Machine failure type = twf
IF Dif_Tempo = {>8.599999999999966} AND Tool wear [min] = {>176} AND Rotational speed [rpm] = {<=2271} AND Torque [Nm] = {<=48.0} AND Tool wear [min] = {<=235} THEN Machine failure type = twf
IF Dif_Tempo = {>8.599999999999966} AND Tool wear [min] = {>176} AND Rotational speed [rpm] = {<=2271} AND Torque [Nm] = {<=48.0} AND Tool wear [min] = {>235} AND Type = {H} THEN Machine failure type = twf;