Eduardo Acrani Ruivo

Ciência de Dados Aplicada para Manutenção Preditiva

UBERLÂNDIA – MG

2023

Eduardo Acrani Ruivo

Ciência de Dados Aplicada para Manutenção Preditiva

Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Câmpus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Elétrica

Orientador: Prof. Dr. Josué Silva de Morais

UBERLÂNDIA – MG

2023

Acrani Ruivo, Eduardo.

Ciência de Dados Aplicada para Manutenção Preditiva/ **Eduardo Acrani Ruivo**. **UBERLÂNDIA**, **2023** - 28p: il. (algumas color.); 30 cm.

Orientador: Prof. Dr. Josué Silva de Morais

Trabalho de Conclusão de Curso – Universidade Federal de Uberlândia – UFU Faculdade de Engenharia Elétrica. **2023**.

Inclui bibliografia.

1. Análise Exploratória de Dados 1. 2. Aprendizagem de Máquinas 2. 3. Manutenção Preditiva 3. I. Orientador Prof. Dr. Josué Silva de Morais. II. Universidade Federal de Uberlândia. III. Faculdade de Engenharia Elétrica. IV. Engenharia de Controle e Automação.

DEDICATÓRIA

Este trabalho é dedicado aos meus pais Sérgio Gallo Ruivo e Adriana Varanda Acrani Ruivo, aos meus irmãos Renato Acrani Ruivo e Victor Acrani Ruivo e aos meus amigos por sempre me apoiar, ajudar e motivar em todos os momentos da minha vida me possibilitando chegar aonde estou.

AGRADECIMENTO

À Universidade Federal de Uberlândia e a Faculdade de Engenharia Elétrica por todo aprendizado e qualidade de ensino.

Ao meu orientador Prof. Dr. Josué Silva de Morais pelo apoio, ajuda e direcionamento durante todos os períodos de graduação e na realização deste trabalho.

Por fim, àqueles que compartilharam momentos comigo durante toda essa trajetória. Amigos, profissionais e professores, agradeço-lhes pelos momentos de descontração, conversas, ensinamentos e experiências que levarei pelo resto da vida comigo.

“Nossa maior fraqueza está em desistir. O caminho mais certo de vencer é tentar mais uma vez”,

(Thomas Edison)

RESUMO

Com a introdução da Indústria 4.0 e o aumento das demandas de eficiência operacional, a aplicação de Análises Exploratórias de Dados (AED) e predições utilizando Aprendizagem de Máquina (ML na sigla em inglês) para manutenção preditiva tem se tornado uma decisão proativa e estratégica. Tais técnicas visam evitar falhas e maximizar o desempenho de equipamentos industriais.

Nesse contexto, esse trabalho busca utilizar da AED e da criação de modelos utilizando ML para estudar um conjunto de dados sintéticos gerados para reproduzir dados de todos os equipamentos de uma empresa. O objetivo principal é estudar a criação de modelos de predição capazes de identificar falhas em equipamentos, dar uma perspectiva nova para profissionais na área e servir de um ponto de referência para futuros estudos com dados adquiridos em situações reais.

Palavras-chave: Análise Exploratória de Dados, Aprendizagem de Máquinas, Manutenção Preditiva.

ABSTRACT

With the introduction of Industry 4.0 and the increased demands for operational efficiency, the application of Exploratory Data Analysis (AED) and predictions using Machine Learning (ML) for predictive maintenance has become a proactive and strategic decision. Such techniques aim to avoid failures and maximize the performance of industrial equipment.

In this context, this work seeks to use AED and the creation of models using ML to study a synthetic dataset generated to reproduce data from all equipment in a company. The main objective is to study the creation of prediction models capable of identifying equipment failures, giving a new perspective to professionals in the field and serving as a reference point for future studies with data acquired in real situations.

Keywords: Exploratory Data Analysis, Machine Learning, Predictive Maintenance.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIAÇÕES E SIGLAS

AED	Análise Exploratória de Dados
AWS	*Amazon Web Services*
CNP	Curva Normal Padrão
EDA	*Exploratory Data Analisys*
FN	*False Negative,* no original em inglês, ou Falso Negativo
FP	*False Positive,* no original em inglês, ou Falso Positivo
IA	Inteligência Artificial
IDE	*Integrated Development Environment,* no original em inglês, ou Ambiente de Desenvolvimento Integrado
IQR	*Interquartil Range,* no original em inglês, ou Intervalo Interquartil
KNN	*K-nearest Neighbors,* no original em inglês
ML	*Machine Learning,* no original em inglês, ou Aprendizado de Máquina
TCC	Trabalho de Conclusão de Curso
TN	*True Negative,* no original em inglês, ou Verdadeiro Negativo
TP	*True Positive,* no original em inglês, ou Verdadeiro Positivo

SUMÁRIO

1. INTRODUÇÃO

"O assunto manutenção preditiva é de extrema importância, por tratar-se de uma medida que visa acompanhar o ciclo de vida dos equipamentos." (KARDEC; NASCIF, 2013). A integração de sistemas inteligentes e sensores de larga escala possibilitados pela Indústria 4.0 permitiu a coleta contínua de dados em tempo real e, com isso, o surgimento da manutenção preditiva.

"A Manutenção Preditiva promove uma série de melhorias que induzem uma mudança de cultura na Operação e na Manutenção, passando pela melhoria da capacidade de análise e diagnóstico e resultando em maior disponibilidade, confiabilidade e segurança operacional, além de menores custos." (KARDEC; NASCIF, 2013).

A manutenção preditiva tem demonstrado ser uma estratégia altamente eficaz para evitar falhas inesperadas e aumentar a disponibilidade de máquinas e equipamentos em diversos setores industriais. A necessidade de minimizar o tempo de inatividade, maximizando a eficiência operacional, e reduzir os custos com manutenções corretivas, tem gerado um interesse crescente na aplicação de técnicas mais avançadas para otimizar os processos de manutenção.

Esse crescente volume de informações e a necessidade de otimização tem estimulado a adoção de abordagens analíticas mais sofisticadas, como a AED e a criação de modelos de previsão, utilizando da tecnologia de ML para dar uma utilidade a quantidade massiva de dados, que são armazenadas e, muitas vezes, não utilizadas para nada.

Neste contexto, utilizar de meios computacionais para analisar e diagnosticar defeitos nos equipamentos trará um grande avanço para a área da manutenção, uma vez que descarta a necessidade de ter uma pessoa monitorando constantemente os milhares de dados gerados em busca de algum sintoma de erro nos equipamentos.

Portanto, este trabalho procura trazer um estudo de um conjunto de dados sintético, que representa uma indústria inteiramente fictícia, criado com a intenção de simular uma indústria real a fim de gerar um norte para futuros projetos e possibilitar melhorias e dar uma perspectiva importante para aqueles que desejam otimizar os trabalhos da manutenção em sua empresa.

1.1. Motivação

A motivação para este TCC é explorar as capacidades da aplicação ciência de dados para aprimorar a manutenção preditiva manipulando a quantidade imensa de dados que são gerados diariamente e criar modelos de predição com a finalidade de classificar quais possíveis futuros erros os equipamentos apresentarão. Essa abordagem permite que máquinas e sistemas aprendam a partir de históricos de dados, identificando padrões complexos e sutis que podem não ser facilmente detectados por abordagens tradicionais de análise.

"O acesso e a interpretação desses dados tornaram-se um diferencial estratégico para empresas e também para governos, e podem dar suporte a estudos em várias áreas." (NETTO; MACIEL, 2021). O conjunto de dados que foi utilizado como objeto de estudo deste trabalho foi adquirido do *site* Kaggle e oferece uma oportunidade de iniciar os estudos para o cenário de manutenção, proporcionando uma base sólida para o

desenvolvimento de modelos preditivos. Ao compreender as relações entre as variáveis envolvidas nos processos de manutenção e os padrões das falhas que podem surgir, é possível construir modelos mais robustos e adaptá-los às particularidades de cada contexto industrial.

Por fim, a motivação deste estudo está profundamente relacionada à busca por soluções eficientes capazes de reduzir os desafios enfrentados pela indústria no que diz respeito ao planejamento da manutenção dos equipamentos. A aplicação do ML para a manutenção preditiva possui uma ideia promissora e, por meio deste TCC, almeja-se estudá-la e desenvolvê-la a fim de contribuir para o desenvolvimento de uma abordagem mais inteligente e proativa na manutenção industrial, beneficiando empresas, profissionais e a sociedade como um todo.

1.2. Objetivo

Este TCC busca estudar maneiras de contribuir com o campo da manutenção preditiva, utilizando de ML para fornecer perspectivas importantes para profissionais interessados em otimizar as estratégias de manutenção em seus ambientes industriais. Essa otimização pode resultar em economias significativas de recursos e melhorias na confiabilidade e segurança das operações.

O objetivo mais específico deste trabalho é checar a possibilidade de se criar um modelo de predição capaz de analisar uma quantidade grande de dados e classificar corretamente os tipo de erros que os equipamentos de uma indústria podem ter.

Com isso, no Capítulo 3 deste documento, foi explicitado como as análises dos dados e os treinamentos dos modelos de predição foram feitos, bem como os resultados de todos os treinos e testes.

2. REFERENCIAL TEÓRICO

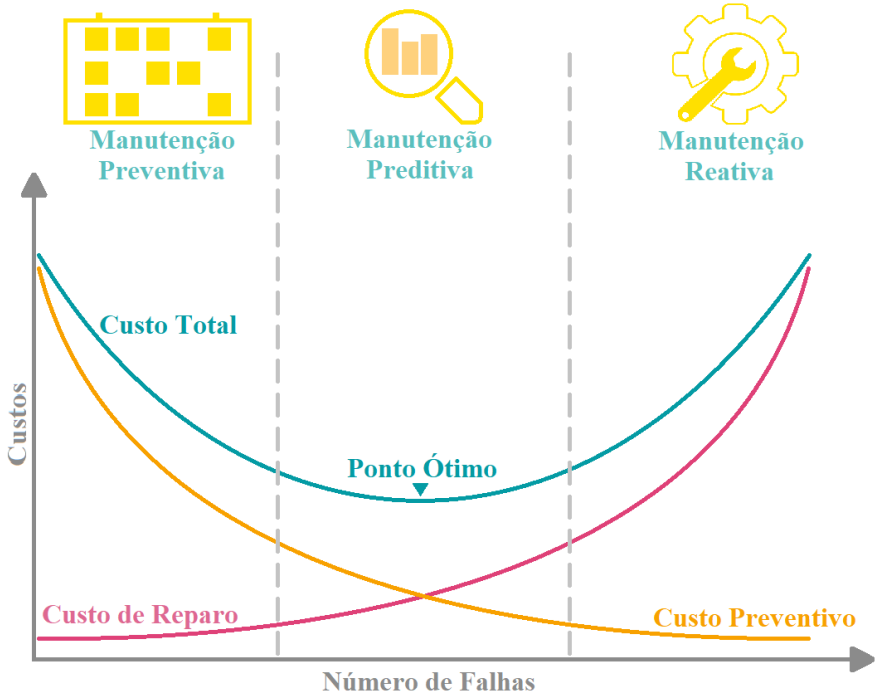
2.1. Manutenção Preditiva

Diferentemente da manutenção corretiva, que age após a ocorrência de falhas, e da manutenção preventiva, realizada em intervalos programados, a manutenção preditiva tem como objetivo antecipar problemas por meio da análise de dados e informações em tempo real, permitindo ações corretivas e preventivas antes que falhas ocorram.

"Manutenção Preditiva pode ser definida como método de monitoramento para antecipar falhas. Assim, prediz o tempo de vida útil de uma máquina, sistema ou componente. Ou seja, permite identificar sintomas iniciais de um problema, antes que se tornem potenciais falhas, e agir preventivamente." (ABECOM, 2021). Porém, para tal, são necessários aparelhos para monitoramento dos equipamentos durante suas operações.

Apesar de seu custo inicial com tais aparelhos, as vantagens tragas por eles compensam o seu gasto. Dentre essas, vale destacar o menor estoque de peças de reposição, menos intervenções nas máquinas, melhora no planejamento de compras, permite componentes com maior vida útil, aumento do tempo global de eficiência do equipamento e diminui as manutenções por paradas não planejadas (ABECOM, 2023).

Figura 1 – Gráfico de Comparação de Custos por Número de Falhas



Autoria Própria

Embora tal abordagem possua muitas vantagens, ainda sim possui algumas desvantagens em relação aos outros tipos de manutenção, sendo as principais a alta dependência de tecnologias e a necessidade de operadores capacitados para entendimento

dos sensores. Sendo assim, são apenas recomendados para equipamentos de extrema importância operacional da indústria.

2.2. Linguagem de Programação Python

"O Python é uma linguagem de programação amplamente usada em aplicações da Web, desenvolvimento de software, ciência de dados e *machine learning* (ML). Os desenvolvedores usam o Python porque é eficiente e fácil de aprender e pode ser executada em muitas plataformas diferentes[...]" (AWS, [s.d.]a). Atualmente na sua versão Python 3, essa linguagem de programação possui uma comunidade bastante ativa que desenvolve e atualiza bibliotecas para expandir suas opções de uso.

Os principais benefícios da utilização da linguagem, segundo a *Amazon Web Services* (AWS), podem ser descritos como:

- Os desenvolvedores podem ler e entender facilmente um programa Python, por sua semelhança com o Inglês;
- Aumento de produtividade pela quantidade reduzida de linhas em comparação com outras linguagem;
- Possui uma grande biblioteca-padrão com códigos reutilizáveis para quase toda tarefa;
- Pode ser usado facilmente com outras linguagens de programação populares, como Java, C e C++;
- Suporte rápido da comunidade devido a imensa comunidade ativa;
- Fácil aprendizado devido a quantidade de conteúdos úteis disponíveis na internet;
- É possível fazer a portabilidade do código em diferentes sistemas operacionais.

2.2.1. *DataFrame*

"*DataFrame* é uma estrutura de dados que organiza os dados em uma tabela bidimensional de linhas e colunas, como uma planilha. Os *DataFrames* são uma das estruturas de dados mais comuns na análise de dados moderna, pois são uma maneira flexível e intuitiva de armazenar e trabalhar com dados." (DATABRICKS, [s.d.]). Apesar de se paracer uma planilha comum, o *DataFrame* é possui diversas vantagens, dentre elas, a capacidade de ser formado pela junção de várias planilhas e essas não precisam estar na mesma máquina.

"Os DataFrames são o principal tipo de dados usado em pandas, a popular biblioteca de análise de dados Python [...]" (DATABRICKS, [s.d.]). Quando utilizados em conjunto com outras bibliotecas, os Dataframes possuem uma alta capacidade para AED e para treinamento de modelos de ML, tornando-se, assim, a combinação perfeita para os cientistas de dados. Os *DataFrames* não são limitados somente à esses profissionais, tendo uma grande abrangência pela sua intuitividade e flexibilidade, assim como foi dito pela DataBricks.

2.3. Análise Exploratória de Dados

A aquisição e armazenamento de dados são partes importantes do processo de predição utilizado na manutenção preditiva, porém, existem muitos problemas para se enfrentar. Dentre eles, uma das principais dificuldades é a garantia de que os dados estão sendo adquiridos de maneira correta para as posteriores análises.

"A análise de dados é um processo crítico na manutenção preditiva. Ferramentas necessárias para o tratamento de dados foram desenvolvidas, de forma a retornar conhecimento relevante a partir desse volume imenso de informações." (NETTO; MACIEL, 2021). Para conseguir retirar as melhores informações e mais verídicas, existe a necessidade de fazer um tratamento no volume imenso de dados adquiridos com o que é chamado de pré-processamento. Esse processo consiste em garantir que não tenha dados incoerentes ou faltantes interferindo na análise final de todo o conjunto.

Sendo assim, as etapas consistem em:

- **Integração dos dados**: É comum que dados venham de fontes diferentes. Integrá-los para que possam ser analisados como um conjunto é essencial.
- **Limpeza dos dados**: Verificar se há dados redundantes, faltantes ou incoerentes, verificar pontos fora da curva e dar o devido tratamento de acordo com a necessidade do projeto e garantir resultados coesos.
- **Normalização e Padronização**: As diversas variáveis possuem escalas próprias, normalizá-las e padronizá-las facilita a análise cruzada de variáveis e traz uma melhor análise. Importante ser feito pós limpeza de dados para que os pontos fora da curva não atrapalhem e na normalização dos dados.

Com isso, a análise pode ser realizada para melhor obtenção de resultados, gerando os melhores gráficos, retornando conhecimento relevante e escolhendo quais são as variáveis mais influentes para encontrar os padrões de cada valor alvo. Após a realização de todas as etapas, é possível afirmar que a AED do seu conjunto de dados está completa e este está pronto para a criação de modelos utilizando de ML.

2.3.1. Valores Discrepantes (*Outliers)*

Valores Discrepantes ou *outliers* são valores drasticamente discrepantes de todos os outros. "Em outras palavras, um *outlier* é um valor que foge da normalidade e que pode (e provavelmente irá) causar anomalias nos resultados obtidos por meio de algoritmos e sistemas de análise." (HOPPEN; PRATES, 2017). O entendimento destes é uma parte importante para a AED, uma vez que eles podem prejudicar a normalização dos dados, enviesar o aprendizado dos modelos criados com ML ou podem ser exatamente o comportamento que se está procurando e estudando.

Sendo um ponto crucial a ser estudados, são necessárias técnicas para detectá-los para que possam ser estudados e para que sejam tomadas as devidas decisões. Duas metodologias comuns utilizados na estatística são o método do Intervalo Interquartil (IQR na sigla em inglês), que possibilita a utilização visual por meio de *Box plot*, e Z-Score.

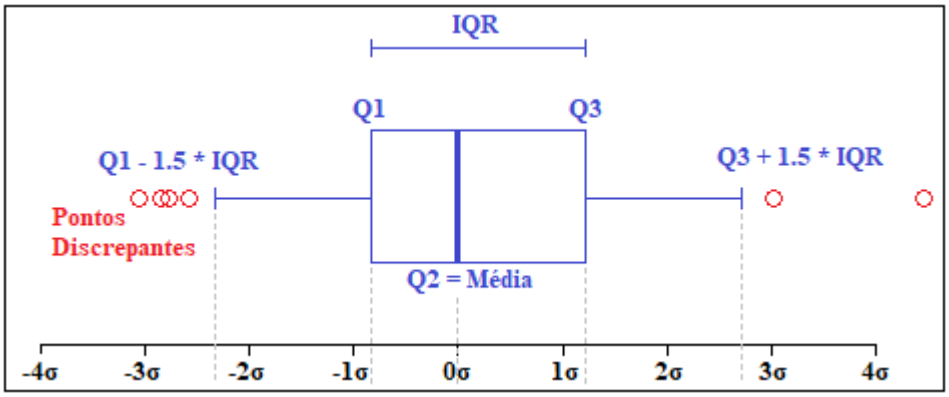
2.3.1.1. IQR e *Box plot*

O método de detecção de *outliers* do IQR foi inventado por John Tukey e, em algumas literaturas, é referido como Método de Tukey se baseia em cinco medidas importantes, que são conhecidos como "resumo dos cinco números", sendo essas o Limite Inferior, Primeiro Quartil, Mediana ou Segundo Quartil, Terceiro Quartil e Limite Superior.

"O intervalo entre quartis (IQR) é uma medida de variação que se baseia em dividir um conjunto de dados em quartis. Os quartis dividem um conjunto de dados ordenados por classificação em quatro partes iguais. Q1, Q2 e Q3. O IQR é definido como Q3 - Q1 e qualquer dado que estiver fora de Q3+1.5*IQR ou Q1-1.5*IQR será considerado um outlier." (ORACLE, [s.d.]a).

O *Box plot* pode ser definido como a visualização gráfica do método IQR que, atualmente, pode ser gerado com facilidade utilizando de códigos simples de programação.

Figura 2 – Exemplo de *Box Plot*



Autoria Própria

2.3.1.2. Z-Score

O método do Z-Score, ou pontuação padrão, é um modo de descrever um ponto relacionando-o com a média e o desvio padrão do grupo que pertence. Para adquirir um Z-score é necessário escolher um ponto que deseja saber em seu conjunto de números, subtraí-lo da média e dividir pelo desvio padrão, sendo assim, obtemos a seguinte equação:

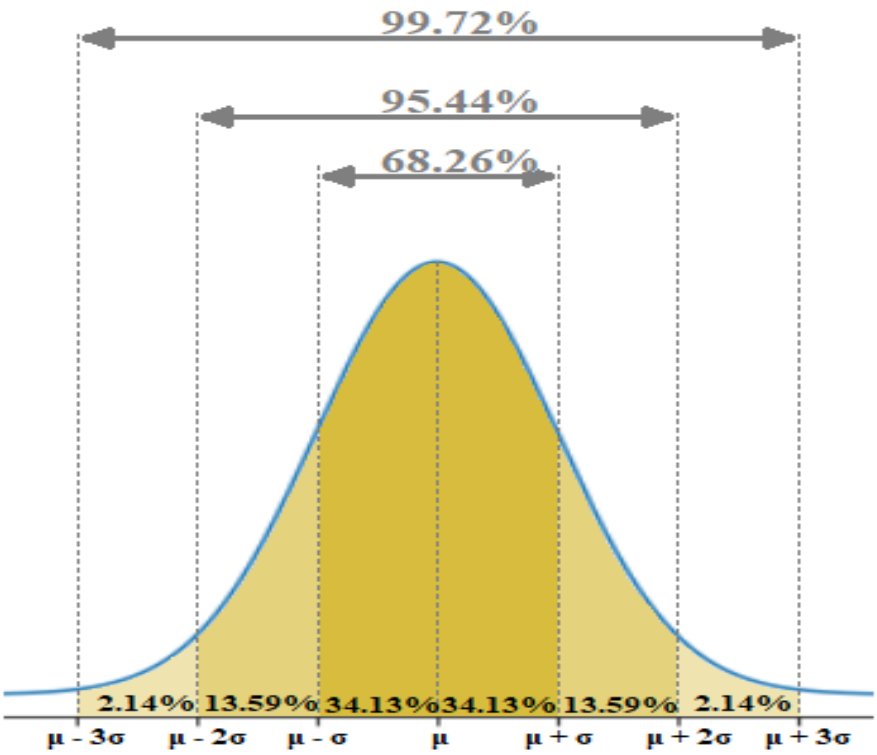
$$z = \frac{X - \mu}{\sigma} \qquad (1)$$

Onde X, μ e σ representam o ponto em que se deseja saber o Z-score, a média da curva normal e o desvio padrão, respectivamente.

Obtendo o Z-Score, é possível utilizar da Curva Normal Padrão (CNP) para definir uma área que indica quantos porcento do seu conjunto de dados está acima ou abaixo

deste valor utilizando a integral da área da CNP. Os cálculos já foram todos feitos e descritos em uma tabela facilitando todo o trabalho de calcular as áreas da CNP.

Figura 3 – Exemplo de Curva Normal



Autoria Própria

"O objetivo de obter Z-scores é remover os efeitos da localização e da escala dos dados, permitindo que diferentes conjuntos de dados sejam comparados diretamente. A intuição por trás do método Z-Score de detecção de outliers é que, uma vez que tenhamos centralizado e redimensionado os dados, qualquer valor que esteja muito longe de zero (o limite é geralmente um Z-Score de 3 ou -3) deve ser considerado um outlier." (ORACLE, [s.d]b).

2.3.2. Coeficientes de Correlação

Coeficientes de correlação são métodos utilizados na estatística para medir relação entre duas variáveis, em outras palavras, é um método utilizado para checar a influência que uma variável tem sobre outra. Para medir o nível dessa relação, temos o coeficiente de correlação de Pearson e o coeficiente de correlação de Spearman.

2.3.2.1. *Pearson*

O Coeficiente de Correlação de Pearson é uma técnica utilizada para checar a linearidade entre a relação de duas variáveis. Quando variáveis possuem uma relação

linear, significa que quando uma varia, a outra varia proporcionalmente, dependendo da relação. Sendo assim, o Coeficiente de Correlação de Pearson checa essa proporção de variação linear, podendo ser positiva ou negativa.

2.3.2.2. *Spearman*

O Coeficiente de Correlação de Spearman é uma técnica utilizada para checar a relação entre duas variáveis através de uma função monotética. De maneira simples, ele checa se quando uma variável aumenta, a outra não diminui e vice-versa. Por se tratar de uma técnica não paramétrica, ela não assume nenhuma distribuição para as variáveis do conjunto. Assim como o Coeficiente de Correlação de Pearson, esta correlação pode ser positiva ou negativa.

2.4. *Machine Learning*

"O *machine learning* (ML) é o subconjunto da inteligência artificial (IA) que se concentra na construção de sistemas que aprendem, ou melhoram o desempenho, com base nos dados que consomem." (ORACLE, [s.d.]c) Atualmente, os termos ML e IA são confundidos por muitos e são citados como se fossem a mesma coisa. É importante ressaltar que embora todo ML seja IA, nem toda IA é ML.

Existem três tipos de abordagens para o aprendizado em ML, o supervisionado, o não-supervisionado e o por reforço. No ML supervisionado "[...] um cientista de dados age como um guia e ensina ao algoritmo quais conclusões ele deve ter." (ORACLE, [s.d.]c). Já no ML não-supervisionado "[...] usa uma abordagem mais independente, na qual um computador aprende a identificar processos e padrões complexos sem que um ser humano forneça uma orientação próxima e constante." (ORACLE, [s.d.]c). Por sua vez, no ML por reforço o algoritmo interage em um ambiente que lhe fornece uma retroalimentação sobre suas decisões tomadas e ele, com essas informações, determina um comportamento ideal para aquele contexto específico.

2.4.1. Métricas de Avaliação de Modelo

Com a finalidade de saber se o modelo esta performando bem, é necessária uma métrica para de avaliação para que possa ser possível compara os modelos e decidir se ele está pronto ou não para implementação. Sendo assim, existem quatro métricas de avaliação para ser escolhido de acordo com o que é mais importante para seu trabalho.

Para entender as métricas de avaliação, primeiro é preciso entender o que é uma matriz de confusão. "Em problemas de classificação binária é utilizada uma matriz de tabulação cruzada dos resultados preditos com as classes originais observadas, conhecida como matriz de confusão. Contudo, esta matriz busca entender a relação entre acertos e erros que o modelo apresenta." (NOGARE, 2020).

Figura 4 – Exemplo de Matriz de Confusão

		Valor Predito	
		Sim	**Não**
Real	**Sim**	**Verdadeiro Positivo (TP)**	**Falso Negativo (FN)**
	Não	**Falso Positivo (FP)**	**Verdadeiro Negativo (TN)**

Autoria Própria

Para melhor entendimento, os valores da Matriz de Confusão são:

- **Verdadeiro Positivo** (TP na sigla em inglês) – O valor que é positivo e foi predito como positivo;
- **Falso Positivo** (FP na sigla em inglês) – O valor que é negativo e foi predito como positivo;
- **Verdadeiro Negativo** (TN na sigla em inglês) – O valor que é negativo e foi predito como negativo;
- **Falso Negativo** (FP na sigla em inglês) – O valor que é positivo e foi predito como negativo;

Com essas informações, agora podemos definir as métricas de avaliação:

- ***Accuracy*** – É a métrica que indica quantos dados foram preditos de maneira correta. A fórmula para cálculo pode ser escrita como $\mathbf{Accuracy} = \frac{\mathbf{TP+TN}}{\mathbf{TP+FP+TN+FN}}$;
- ***Precision*** – É a métrica que indica quantos valores Positivos ou Negativos foram classificados corretamente. Outra maneira de entender é que *Precision* se mede nada vertical. As fórmulas para cálculo podem ser escritas como $\boldsymbol{Precision\ para\ Positivo} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}}$ e $\mathbf{Precision\ para\ Negativo} = \frac{\mathbf{TN}}{\mathbf{TN+FN}}$;
- ***Recall*** – É a métrica que indica quantos valores Verdadeiros foram classificados corretamente. Outra maneira de entender é que *Recall* se mede na horizontal. As fórmulas para cálculo podem ser escritas como $\mathbf{Recall\ para\ TP} = \frac{\mathbf{TP}}{\mathbf{TP+FN}}$ e $\mathbf{Recall\ para\ TN} = \frac{\mathbf{TN}}{\mathbf{TN+FP}}$;
- ***F1-Score*** – É a métrica que consegue avaliar a *Precision* e o *Recall* ao mesmo tempo para uma avaliação geral da qualidade do modelo. A fórmula para cálculo pode ser escrita como $\mathbf{F1} = \mathbf{2} * \frac{\mathbf{Precision} * \mathbf{Recall}}{\mathbf{Precision} + \mathbf{Recall}}$.

2.4.2. Classificadores

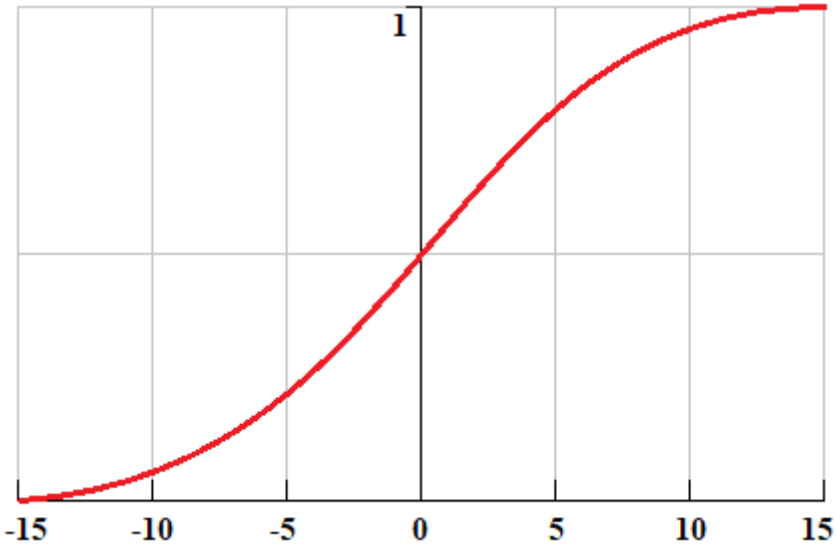
Em ML, um classificador é um algoritmo que, de maneira automática, organiza e classifica um ou mais tipos de classes. Um exemplo é um classificador que indica se uma transação bancária é fraude ou não. Após treinar um classificador com um conjunto de dados seguindo certos parâmetros escolhidos pelo programados, temos um modelo.

Os classificadores serão explicados de maneira simples, sem cálculos apenas para fins de entendimento dos resultados apresentados no capítulo 4.

2.4.2.1. Regressão Logística

"A regressão logística é uma técnica de análise de dados que usa matemática para encontrar as relações entre dois fatores de dados. Em seguida, essa relação é usada para prever o valor de um desses fatores com base no outro. A previsão geralmente tem um número finito de resultados, como sim ou não." (AWS, [s.d]b). Com o auxílio de bibliotecas prontas desenvolvidas para a linguagem Python, a Regressão Logística pode ser feita sem muitos problemas.

Figura 5 – Exemplo de Curva da Regressão Logística

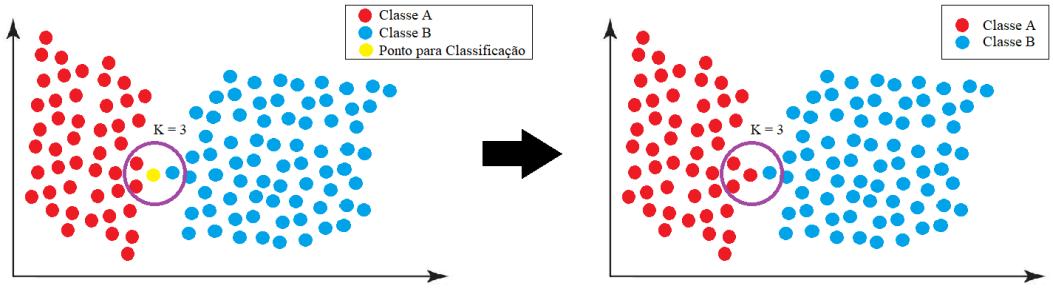


Autoria Própria

2.4.2.2. *K-Nearest Neighbors* (KNN)

O *K-Neares Neighbors* (KNN) ou K-ésimos Vizinhos Próximos é um classificador bastante utilizado em ML que baseia sua predição em analisar os K dados mais próximos, sendo K um valor definido pelo programador, observar a quantidade elementos de cada classe e predizer o dado como se sele fosse da classe mais presente. Em caso de empate, o algoritmo faz um cálculo de distância para decidir qual das duas classes o dado pertence.

Figura 6 – Exemplo de KNN

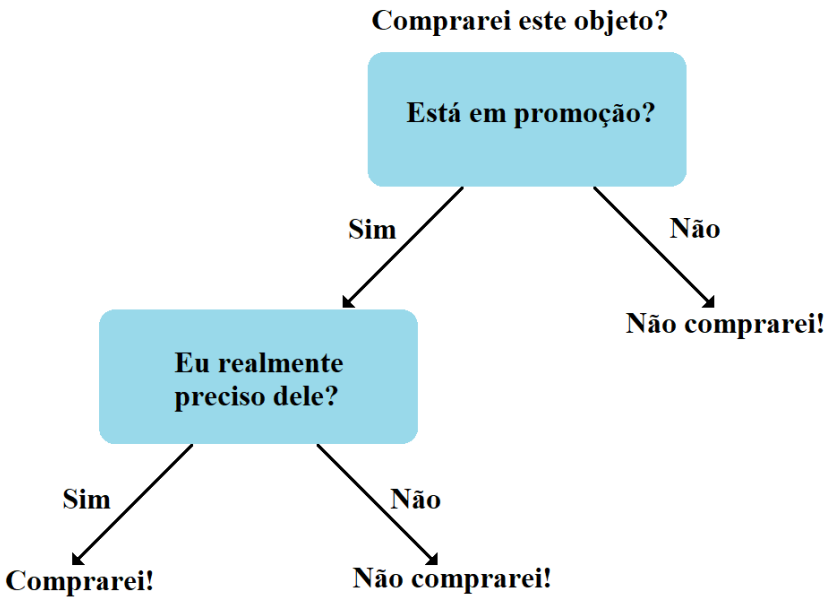


Autoria Própria

2.4.2.3. Árvore de Decisões

A Árvore de Decisões é um classificador que funciona de maneira automática assim como o seu nome sugere. Quando queremos separar dois ou mais grupos de classes começamos com uma pergunta inicial, a raiz, que pode ser dividida em outras perguntas, os nós, até chegar em respostas capazes de separar os pontos, as folhas. A quantidade de nós deste classificador deve ser limitada pelo programador, pois, caso contrário, existirá, possivelmente, uma folha para cada ponto, o que significa um esforço computacional elevado e, mesmo assim, pode não garantir os melhores resultados.

Figura 7 – Exemplo de Árvore de Decisões



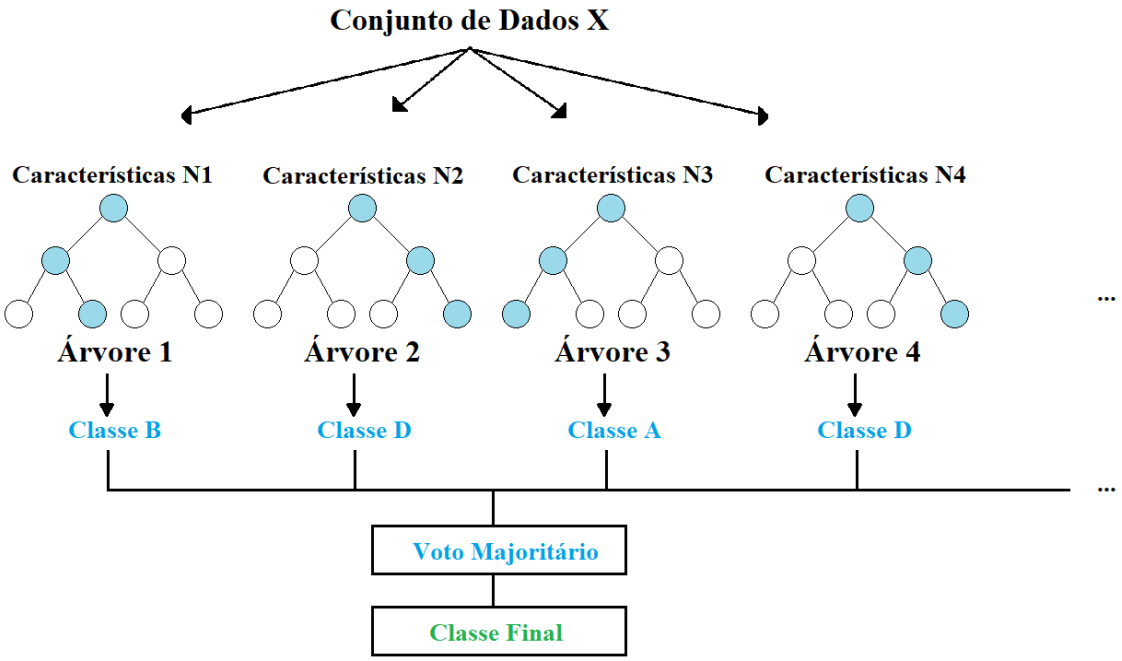
Autoria Própria

2.4.2.4. *Random Forest*

Random Forest é um algoritmo de aprendizado de máquina comumente usado, registrado por Leo Breiman e Adele Cutler, que combina a saída de múltiplas árvores de

decisão para chegar a um único resultado (IBM, [s.d.]). Para isso, o classificador utilizar de uma técnica de *Boostraping*, que é retirar dados aleatoriamente do conjunto de dados original, com reposição, para fazer diversos conjuntos de dados menores e, a partir disto, ele gera as múltiplas árvores de decisão, que, quando colocadas para teste, fazem suas predições uma a uma e o resultado final é o voto majoritário.

Figura 8 – Exemplo de *Random Forest*

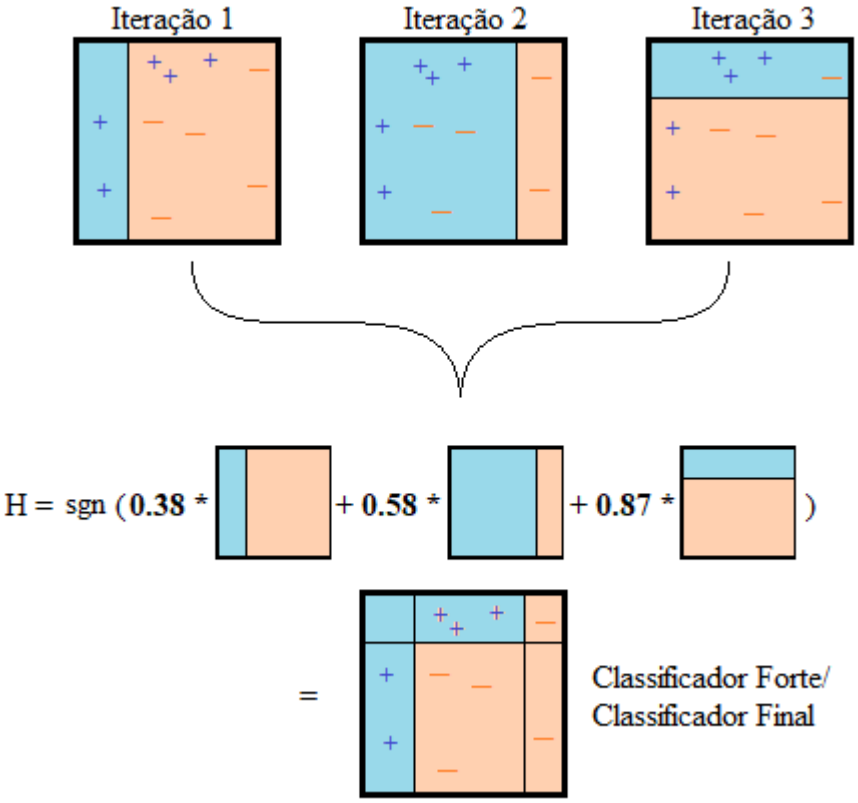


Autoria Própia

2.4.2.5. AdaBoost

"AdaBoost é um dos métodos mais populares de Boosting, conceito que se refere a combinação de diversos modelos de aprendizagem fracos para criar um modelo de aprendizagem forte. A idéia por trás consiste em treinar preditores sequencialmente, cada um tentado corrigir seu predecessor." (AZAMBUJA, 2020). Geralmente, esses modelos simples consistem em árvores de decisão de apenas um nó e utiliza do resultado desta para melhorar o resultado da próxima. As predições finais são feitas baseadas nos pesos de cada um desses modelos fracos para gerar um modelo forte.

Figura 9 – Exemplo de AdaBoost



Autoria Própria

2.4.3. Otimizadores de Hiperparâmetros

Hiperparâmetros são aqueles valores que são utilizados para controlar o processo de aprendizagem. Cada classificador possui uma série de hiperparâmetros que podem ser alterados de acordo com a necessidade, porém, testar um por um leva muito tempo. Para isso, foram inventados os Otimizadores de Hiperparâmetros, que serão abordados o *GridSearch* e o *RandomSearch*.

Ambos possuem uma estrutura parecida, onde o programador fornece o classificador que deseja utilizar, qual métrica de avaliação que ele deseja que o modelo baseie sua nota, os hiperparametros que deseja alterar e alguns outros parâmetros que não serão citados por não contribuírem para a explicação sugerida neste tópico.

Apesar de ambos possuírem uma estrutura semelhante, eles funcionam de maneira diferente. O *GridSearch* recebe todos os parâmetros citados anteriormente e faz uma varredura um a um treinando modelos e, no fim, retorna aquele que melhor performou seguindo a métrica de avaliação desejada. Já para o *RandomSearch,* o usuário fornece a quantidade de modelos que ele deseja que sejam testados e este escolhe aleatoriamente a quantidade especificada para fazer a varredura e, no fim, também retornar o modelo com a melhor performance.

3. METODOLOGIA

Para melhor entendimento e realização do projeto, este capítulo foi dividido em 4 partes principais:

- Visão Geral
- Pré-Processamento
- Análise Aprofundada dos Dados
- Criação dos Modelos

É importante ressaltar que o projeto final foi publicado em Outubro de 2023 no *Github* no repositório Trabalho-de-Conclusao-de-Curso-Eduardo-Acrani-Ruivo e contém o código e explicação de todas as etapas feitas para reprodução dos resultados.

3.1. Visão Geral

Esse projeto utilizou da linguagem de programação Python e foi inteiramente feito utilizando o Jupyter Notebook, "[...] um Ambiente de Desenvolvimento Integrado (IDE na sigla em inglês) que combina a edição e execução de código com visualização de resultados, sendo útil para desenvolvedores [...]" (LOCALWEB, 2023).

No auxílio do projeto, foram utilizadas as bibliotecas para Python:

- **Numpy:** Utilizada para se trabalhar com computação numérica;
- **Pandas:** Utilizada para visualizar e manipular dados relacionados;
- **Matplotlib e Seaborn:** Utilizadas para visualização gráfica;
- **Scipy:** Utilizada para execução do teste Z-Score de detecção de pontos fora da curva;
- **Scikit-Learn:** Utilizada para criação, treino e teste dos modelos de predição.

É importante ressaltar que os dados retirados do *site* Kaggle são dados sintéticos, uma vez que coletar esses dados é uma tarefa difícil e nenhuma empresa os disponibiliza para uso na *internet*. Sendo assim, os dados foram gerados para simular uma indústria completa e fictícia que reflete a realidade encontrada nas indústrias.

O conjunto de dados possui 10000 registros e 10 colunas, sendo:

- ***UDI*** - Identificador único na faixa de 1 até 10.000;
- ***Product ID*** - Consistindo em uma letra, sendo L para baixa (60% de todos os produtos), M para média (30%) e H para alta (10%) como variantes de qualidade do produto e um número de série específico da variante;
- ***Type*** - Consistindo em uma letra, sendo L para baixa (60% de todos os produtos), M para média (30%) e H para alta (10%) como variantes de qualidade do produto;
- ***Air temperature [K]*** - Temperatura do Ar em Kelvin. Gerado usando um processo de passo aleatório e posteriormente normalizado para um desvio padrão de 2.000 em torno de 300.000;

- ***Process temperature [K]*** - Temperatura do Processo em Kelvin. Gerado usando um processo de passo aleatório e posteriormente normalizado para um desvio padrão de 1.000, adicionado à temperatura do ar mais 10.000;
- ***Rotational speed [rpm]*** - Velocidade Rotacional em Rotações por Minuto. Calculado a partir de uma potência de 2860 W, sobreposta a um ruído normalmente distribuído;
- ***Torque [Nm]*** - Os valores de torque, em Newton-metros, são normalmente distribuídos em torno de 40 Nm com Ïf = 10 Nm e sem valores negativos;
- ***Tool wear [min]*** - Desgaste do Equipamento por minutos. As variantes de qualidade H/M/L adicionam 5/3/2 minutos de desgaste da ferramenta usada no processo;
- ***Target*** - Valor alvo. Indica se o equipamento falou ou não alguma vez;
- ***Failure Type*** - Tipo de falha. Indica qual tipo de falha o equipamento teve, caso teve. São eles *Heat Dissipation Failure* (falha na dissipação de calor), *Power Failure* (falha na energia), *Overstrain Failure* (falha por sobretensão), *Tool Wear Failure* (falha por desgaste do equipamento) e *Random Failures* (falha aleatória). Caso não possuir falhas, será notado como *No Failure* (sem falhas).

As informações acima foram retiradas da descrição dada pelo usuário que forneceu o conjunto de dados e foi traduzida e ajustada para melhor entendimento na língua portuguesa.

3.2. Pré-Processamento

3.2.1. Garantindo Coesão nos Dados

O primeiro passo, após baixar o conjunto de dados do Kaggle, foi importar as bibliotecas citadas na Sessão 3.1 para o ambiente de trabalho e, utilizando de funções da biblioteca Pandas, visualizar o conjunto de dados por meio de um *DataFrame*. Para evitar redundância e melhorar a visualização para pessoas que não são da área, a coluna da variável *UDI* foi transformada em índice por se tratar de uma coluna numérica sequencial que varia de 1 até 10.000.

Figura 10 – *DataFrame* Utilizando Coluna *UDI* Como Índice

UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
1	M14860	M	298.1	308.6	1551	42.8	0	0	No Failure
2	L47181	L	298.2	308.7	1408	46.3	3	0	No Failure
3	L47182	L	298.1	308.5	1498	49.4	5	0	No Failure
4	L47183	L	298.2	308.6	1433	39.5	7	0	No Failure
5	L47184	L	298.2	308.7	1408	40.0	9	0	No Failure
...	...	...	...	...	...	...	...	...	...
9996	M24855	M	298.8	308.4	1604	29.5	14	0	No Failure
9997	H39410	H	298.9	308.4	1632	31.8	17	0	No Failure
9998	M24857	M	299.0	308.6	1645	33.4	22	0	No Failure
9999	H39412	H	299.0	308.7	1408	48.5	25	0	No Failure
10000	M24859	M	299.0	308.7	1500	40.2	30	0	No Failure

10000 rows × 9 columns

Autoria Própria

A seguir, foi feita a verificação da caracterização e confiabilidade dos dados. Nesse processo foi notado que o conjunto de dados apresenta 0 dados nulos e um desbalanceamento na proporção, aproximada, de 96,5% para 3,5%, o que significa que os resultados podem não ser tão bons quanto o esperado. Além disso, 27 dados possuíam erro na atribuição da variável *Target* e foram corrigidos.

Figura 11 – Caracterização dos Dados do Conjunto

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 1 to 10000
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Product ID              10000 non-null  object
 1   Type                    10000 non-null  object
 2   Air temperature [K]     10000 non-null  float64
 3   Process temperature [K] 10000 non-null  float64
 4   Rotational speed [rpm]  10000 non-null  int64
 5   Torque [Nm]             10000 non-null  float64
 6   Tool wear [min]         10000 non-null  int64
 7   Target                  10000 non-null  int64
 8   Failure Type            10000 non-null  object
dtypes: float64(3), int64(3), object(3)
memory usage: 781.2+ KB
```

Autoria Própria

Figura 12 – Detecção de Inconsistência nos Dados do Conjunto

```
Coluna: Target
0    96.61
1     3.39
Name: Target, dtype: float64

Coluna: Failure Type  para Target = 0
No Failure         9643
Random Failures      18
Name: Failure Type, dtype: int64

Coluna: Failure Type
No Failure                  96.52
Heat Dissipation Failure     1.12
Power Failure                0.95
Overstrain Failure           0.78
Tool Wear Failure            0.45
Random Failures              0.18
Name: Failure Type, dtype: float64

Coluna: Failure Type para Target = 1
Heat Dissipation Failure    112
Power Failure                95
Overstrain Failure           78
Tool Wear Failure            45
No Failure                    9
Name: Failure Type, dtype: int64
```

Autoria Própria

Como dito na Sessão 3.1, na descrição das variáveis, caso a variável *Target* possuir um valor igual à 0, sua coluna *Failure Type* deve possuir o valor categórico *No Failure* e pode ser observado na Figura 12, que existem 18 dados com *Target* igual a 0 que possuem o valor de *Failure Type* como *Random Failures,* assim como têm 9 dados com *Target* igual a 1 e possuem o valor de *Failure Type* como *No Failures.*

Nesse caso, a inconsistência tornou-se fácil de ser arrumada apenas trocando seus valores errados do *Target* para seus devidos valores, como pode ser observado na Figura 13.

Figura 13 – Correção da Inconsistência dos Valores da Coluna *Target*

```
Coluna: Failure Type para Target = 1
Heat Dissipation Failure    112
Power Failure                95
Overstrain Failure           78
Tool Wear Failure            45
Random Failures              18
Name: Failure Type, dtype: int64

Coluna: Failure Type para Target = 0
No Failure    9652
Name: Failure Type, dtype: int64
```

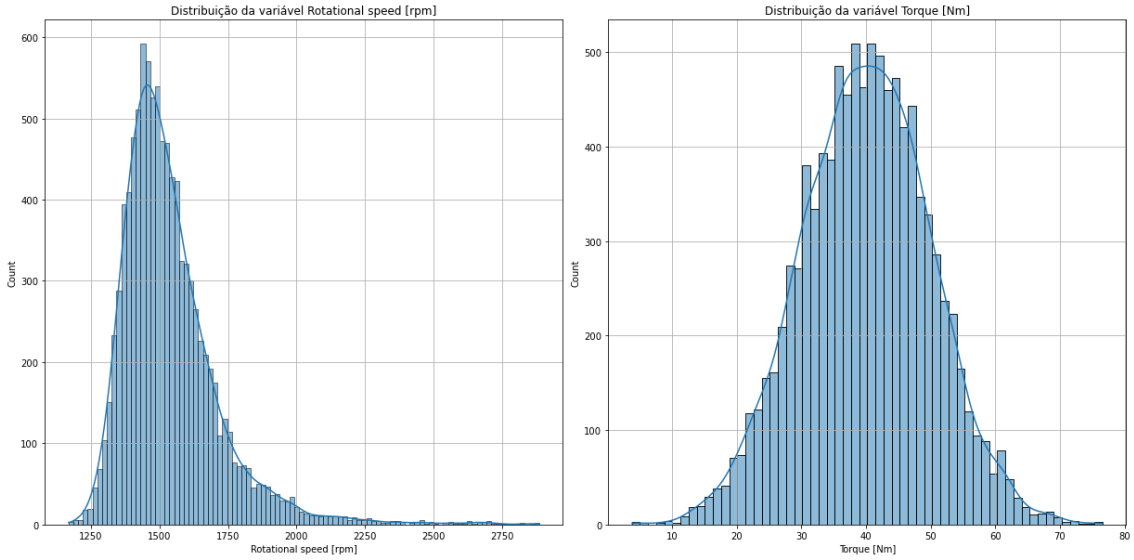
Autoria Própria

Após essa correção, checou-se a presença de dados duplicados que pudessem causar alguma predição enviesada e não foi constatado a presença desses. Também se notou que não existem duplicatas para os valores da coluna *Product ID*, com isso, podemos afirmar que existe somente 1 dado para cada, assim chamado, equipamento. Sendo assim foi removida esta coluna, pois se trata apenas da identificação dos equipamentos e essa não trará informações úteis para análises e predições.

3.2.2. Análise de *Outliers*

Para o início da análise de *outliers* foram feitos os histogramas das colunas *Air Temperature [K], Process Temperature [K], Rotational Speed [rpm], Torque [Nm]* e *Tool Wear [min]* para ver como se encontrava a distribuição de valores de cada uma das colunas. As que mais chamaram atenção e que, possivelmente, seriam as colunas com *outliers* foram a *Rotational Speed [rpm]* com sua distribuição assimétrica à direita e *Torque [Nm]* que segue uma curva normal.

Figura 14 – Distribuição das Colunas *Rotatinal Speed [rpm]* e *Torque [Nm]*



Autoria Própria

Para certificar de que eram apenas essas duas colunas que possuiriam *outliers*, foram feitos os testes do IQR e do Z-Score, descritos na sessão 2.3.1.1 e sessão 2.3.1.2.

3.2.2.1. Teste do IQR e Criação dos *Box plot*

Com auxílio de uma função pronta da biblioteca Pandas para *Dataframes*, a função *describe,* e algumas alterações feitas com linhas de código foi possível obter a média, desvio padrão, valor mínimo, primeiro quartil, a mediana, o terceiro quartil, valor máximo, IQR e limite superior e inferior de cada coluna, como pode ser observado na Figura 15.

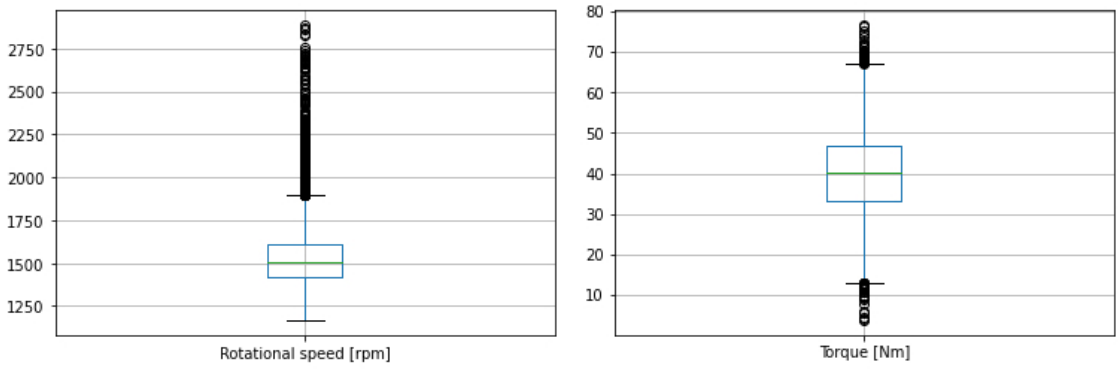
Figura 15 – Descrição das Colunas

	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	300.004930	310.005560	1538.776100	39.986910	107.951000	0.034800
std	2.000259	1.483734	179.284096	9.968934	63.654147	0.183282
min	295.300000	305.700000	1168.000000	3.800000	0.000000	0.000000
25%	298.300000	308.800000	1423.000000	33.200000	53.000000	0.000000
50%	300.100000	310.100000	1503.000000	40.100000	108.000000	0.000000
75%	301.500000	311.100000	1612.000000	46.800000	162.000000	0.000000
max	304.500000	313.800000	2886.000000	76.600000	253.000000	1.000000
IQR	3.200000	2.300000	189.000000	13.600000	109.000000	0.000000
Limite superior	306.300000	314.550000	1895.500000	67.200000	325.500000	0.000000
Limite inferior	293.500000	305.350000	1139.500000	12.800000	-110.500000	0.000000

Autoria Própria

Analisando a Figura 15, foi possível perceber que somente as colunas *Torque [Nm]* e *Rotational Speed [rpm]* possuem valores além dos limites, sendo assim, comprovando que elas seriam as colunas com *outliers* no projeto. Para melhor visualização, foram gerados os *Box plot* de ambas as colunas, a contagem de quantos *outliers* cada uma possui e a caracterização dos *outliers* quanto ao *Failure Type.*

Figura 16 – *Boxplot* das Colunas *Rotational Speed [rpm]* e *Torque [Nm]*



Autoria Própria

Figura 17 – Caracterização dos *Outliers* da Coluna *Rotational Speed [rpm]*

```
Quantidade de outliers da Colunas Rotational Speed: 418

Distruibuição dos outliers em relação ao seu tipo de falha:
No Failure           383
Power Failure         31
Tool Wear Failure      4
Name: Failure Type, dtype: int64
```

Autoria Própria

Figura 18 - Caracterização dos *Outliers* da Coluna *Torque [Nm]*

```
Quantidade de outliers da Colunas Torque: 64

Distruibuição dos outliers inferiores em relação ao seu tipo de falha:
Power Failure    22
No Failure        2
Name: Failure Type, dtype: int64

Distruibuição dos outliers superiores em relação ao seu tipo de falha:
Power Failure               33
No Failure                   4
Overstrain Failure           2
Heat Dissipation Failure     1
Name: Failure Type, dtype: int64
```

Autoria Própria

Observando a Figura 12, foi possível notar que existem apenas 95 dados que possuem o valor *Power Failure* na coluna *Failure Type* e, comparando com a Figura 19 e, na Figura 20, nota-se que para coluna *Rotational Speed [rpm],* 31 dos 95 dados são considerados *outliers* e para coluna *Torque [Nm],* 55 dos 95 são. Sendo assim, foi possível afirmar que não é razoável utilizar deste método para remover os *outliers* e, então, foi testado o método do Z-Score.

3.2.2.2. Teste do Z-Score

Utilizando de códigos, foi aplicado o teste do Z-Score, descrito na sessão 2.3.1.2, em todas as colunas para comparar a quantidade de outliers em relação ao teste do IQR e notou-se uma quantidade reduzida de destes.

Figura 19 – Resultado do Teste Z-Score para Identificação de *Outliers*

```
A coluna Rotational speed [rpm] tem 164 outliers!

Esses índices correspondem aos Tipos de Falhas:

No Failure          131
Power Failure        31
Tool Wear Failure     2
Name: Failure Type, dtype: int64
```

```
A coluna Torque [Nm] tem 25 outliers!

Esses índices correspondem aos Tipos de Falhas:

Power Failure    24
No Failure        1
Name: Failure Type, dtype: int64
```

Autoria Própria

Apesar da quantidade reduzida de *outliers*, percebeu-se que o número de dados que possuí o valor *Power Failure* na coluna *Failure Type* ainda é alto, sendo assim, será descartado a opção de remoção de *outliers.* Contudo, é possível retirar deste resultado a

hipótese de que tais *outliers* sejam de grande utilidade quando se trata de identificar esse tipo de erro.

3.3. Análise Aprofundada dos Dados

Com o pré-processamento pronto, foi a hora de separar quais testes seriam feitos para este projeto e, com isso, fazer o resto da análise exploratória para cada um deles. Para a realização dos testes, os valores da coluna *Type* foram alterados para valores numéricos seguindo uma ordem, pois se trata da qualidade do produto. Sendo assim, é possível alterar L para 0, M para 1 e H para 2 para que pudesse checar também se essa coluna é de importância para as predições.

O primeiro teste foi dividido em duas partes. A primeira parte consistiu em criar outro conjunto de dados, mas sem a coluna *Failure Type* para testar a separabilidade dos dados e ver a possibilidade de identificação de falhas, independente do seu tipo. A segunda parte se baseou em filtrar o conjunto de dados e deixar somente os dados que apresentam falha para saber se é possível diferenciar seu tipo, uma vez que o equipamento já foi identificado com uma falha.

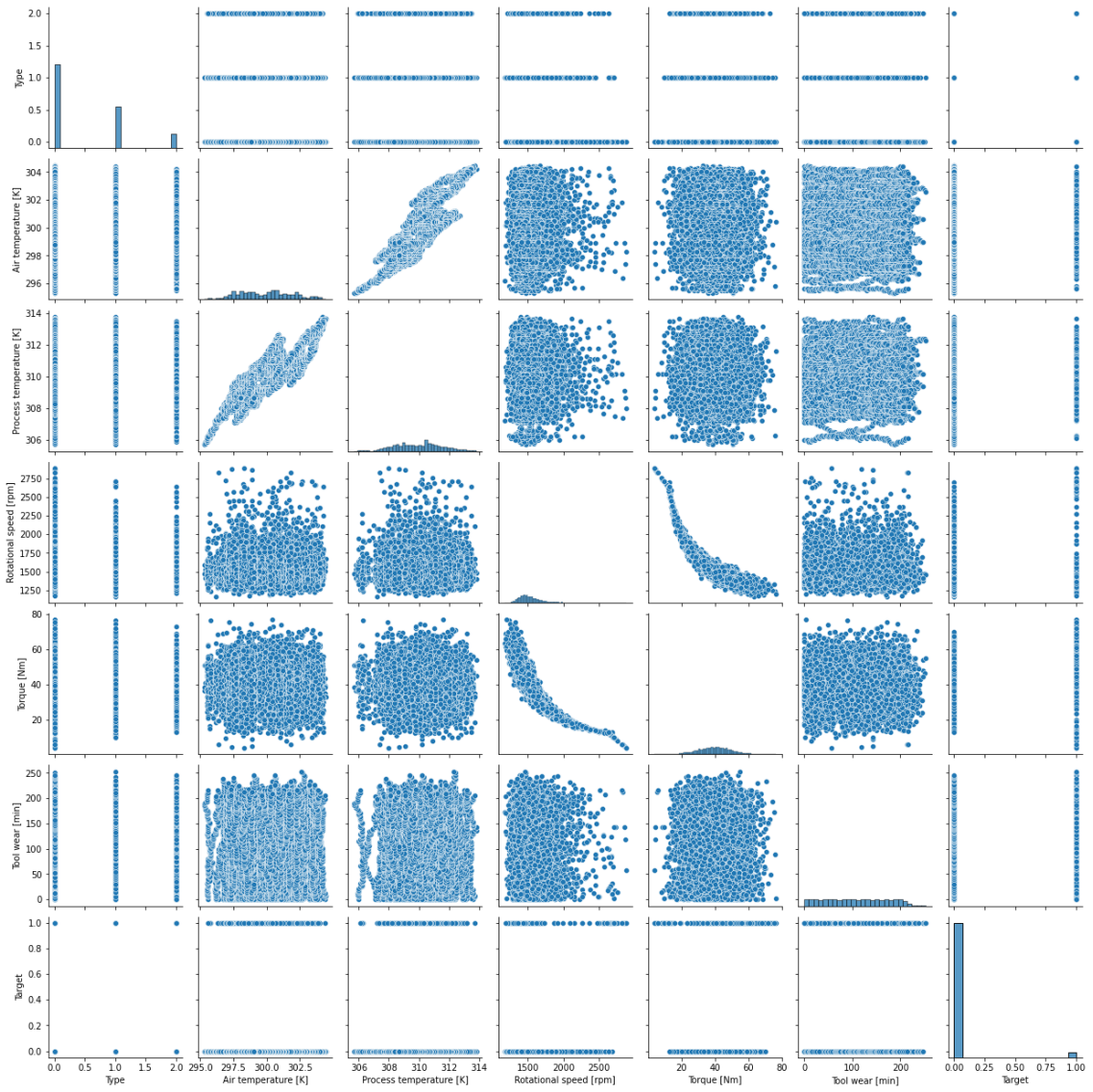
O segundo teste consistiu em criar mais um conjunto de dados, porém, dessa vez, sem a coluna *Target* com o objetivo de estudo para ver se os modelos de predição iriam ter a capacidade de diferenciar as 6 opções de tipo de falhas utilizando do conjunto de dados em sua totalidade. Os resultados de ambos os testes serão apresentados e discutidos no Capítulo 4.

3.3.1. Análise dos Dados para o Teste 1

3.3.1.1. Parte 1

Como dito na sessão 3.3, o primeiro passo para realização desse teste foi a remoção da coluna *Failure Type.* Em seguida, foi gerado um gráfico que cruza as colunas 2 a 2 para checar as correlações entre ela e, principalmente, com a coluna *Target*.

Figura 20 – Gráfico Cruzado 2 a 2 das Colunas da Parte 1 do Teste 1

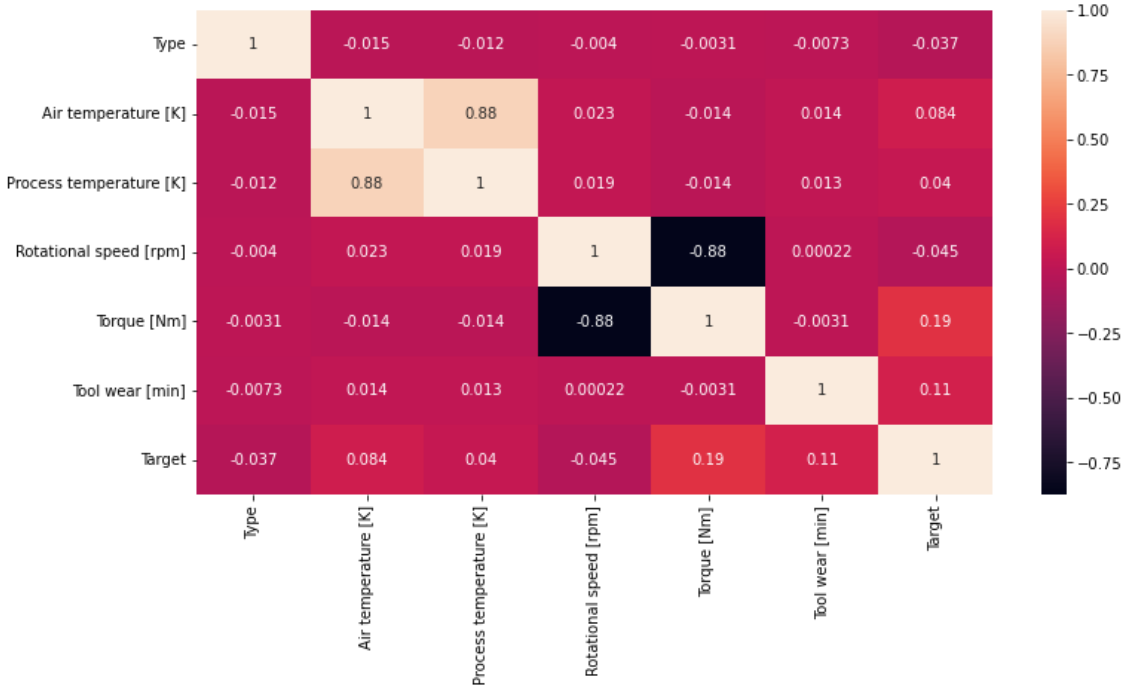


Autoria Própria

Ao analisar o gráfico, foi observado uma correlação próxima de linear entre as colunas *Process Temperature [K]* X *Air Temperature [K]* e *Rotational Speed [rpm]* X *Torque [Nm],* porém, nenhuma das colunas parecem ter uma correlação forte com a coluna *Target*.

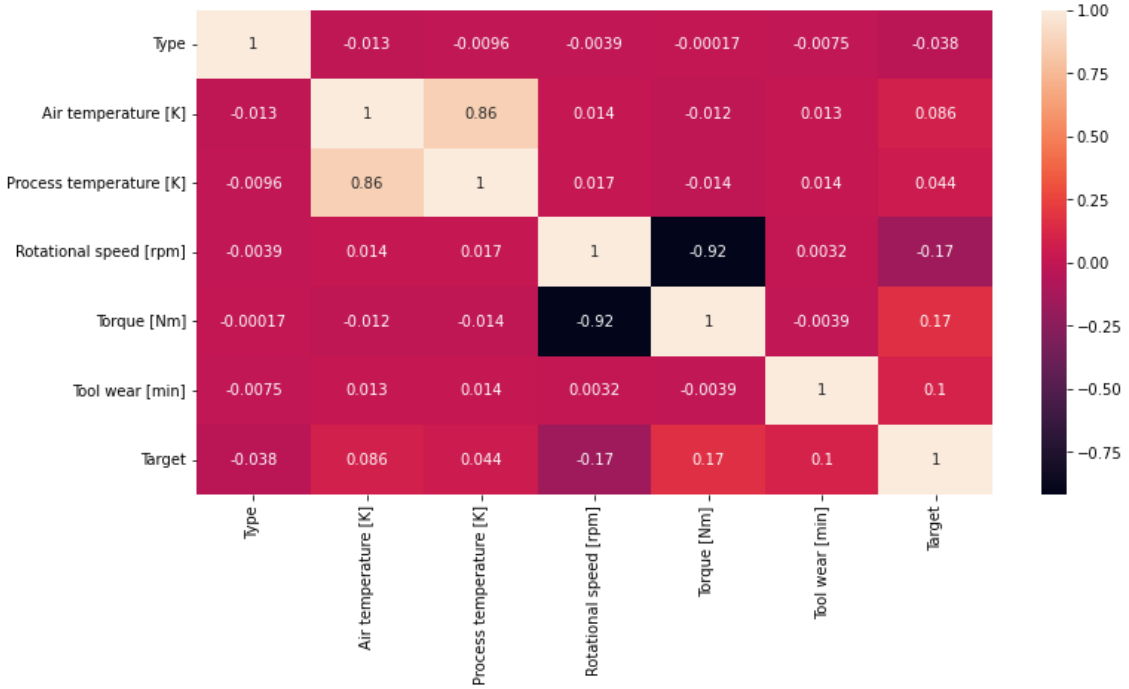
Para responder essa dúvida, foram feitos dois gráficos de correlação, conhecidos como mapa de calor, um com *Pearson* e outro com *Spearman*, utilizando de uma função pronta da biblioteca Seaborn, a *heatmap*, para checar o quão influente são as variáveis na coluna *Target*.

Figura 21 – Mapa de Calor Utilizando da Correlação de *Pearson* para a Parte 1 do Teste 1



Autoria Própria

Figura 22 - Mapa de Calor Utilizando da Correlação de *Spearman* para a Parte 1 do Teste 1



Autoria Própria

Analisando a Figura 21 e a Figura 22, foi possível confirmar as correlações citadas anteriormente e que, nenhuma das colunas tem uma correlação significativamente alta

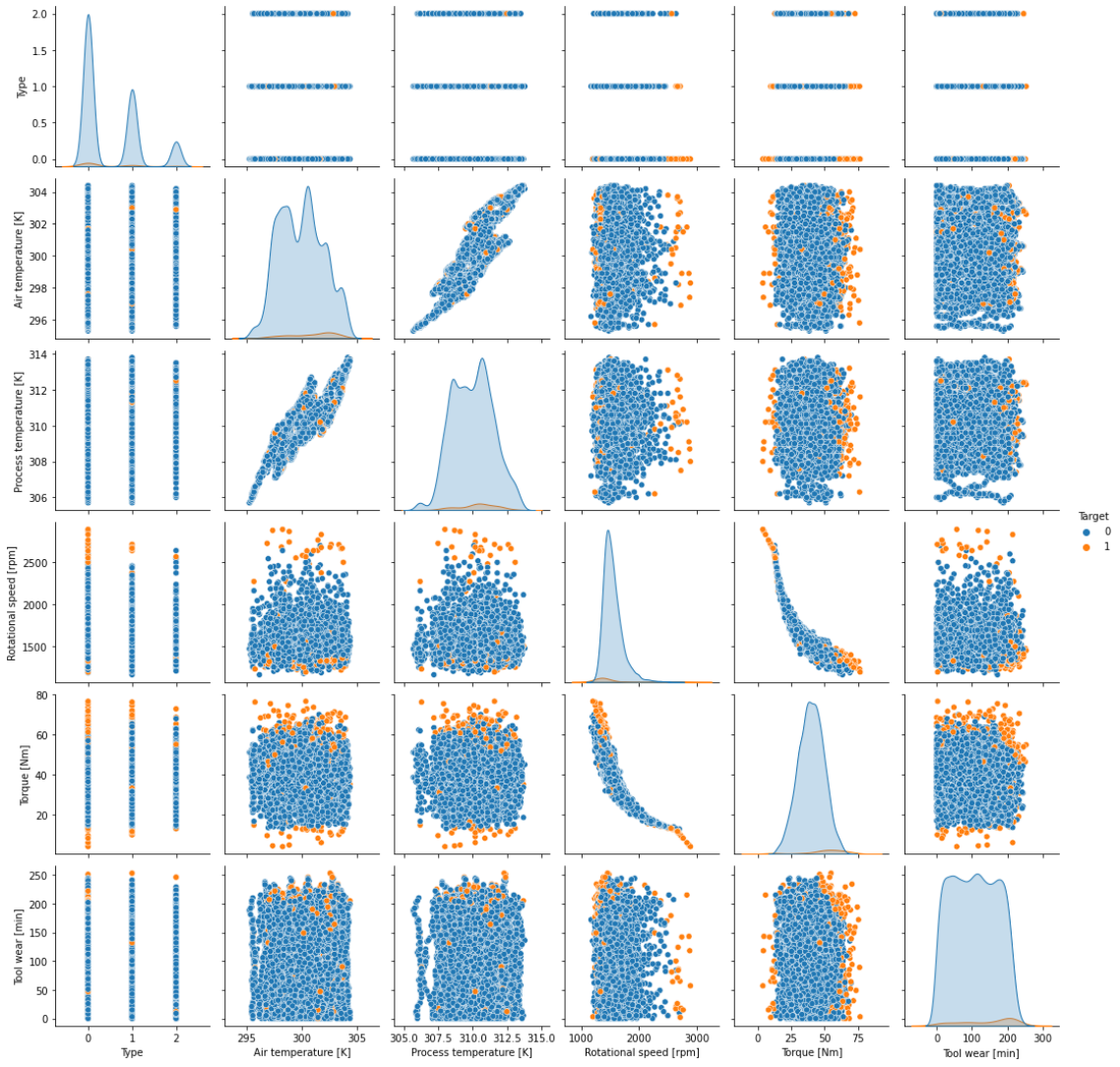
com a coluna *Target*. Sendo assim, testes mais profundos foram feitos para assegurar que os modelos serão capazes de efetuar a classificação.

3.3.1.1.1. Checando a Separabilidade do Alvo

Para execução desse passo, a primeira coisa feita foi separar o conjunto de dados em dois, um contendo 80% dos dados, que foi utilizado para treinar os modelos, e o outro com 20% dos dados, que serviu para validar e avaliar o desempenho deles. Para garantir que todos os dados com falha fiquem em um dos conjuntos, a separação foi feita com o estado de aleatoriedade fixado em 42, para propósitos de reprodução do trabalho, e de maneira estratificada, ou seja, mantendo a proporção de 96.5% de não falha e 3.5% de falha em ambos os conjuntos.

Após a separação, foi feito outro gráfico cruzado 2 a 2 das colunas, porém, dessa vez, com a separação de cor da coluna *Target* para checar a distribuição das falhas.

Figura 23 – Gráfico Cruzado 2 a 2 das Colunas com Separação de *Target*



Autoria Própria

Analisando a Figura 23, notou-se que a separabilidade do *Target* não é trivial como passar uma linha, sendo assim, métodos lineares não seriam de grande utilidade para esse problema. Com isso, ouve a necessidade de realizar um teste de hipótese que foi utilizado para validar se existe alguma diferença substancial para os elementos de cada classe e, portanto, podendo dizer se existia a possibilidade de fazer uma separação com algum modelo de ML.

3.3.1.1.2. Teste de Hipótese de Separabilidade de *Target*

O teste consistiu nas seguintes etapas:

- Agrupar os dados pelos níveis categóricos do Target e calcular a média de cada uma das colunas;
- Fazer um teste de hipótese para determinar se, a um nível de significância de 5%, há diferença na média de cada uma das sub-amostras de cada classe, para todas as variáveis.

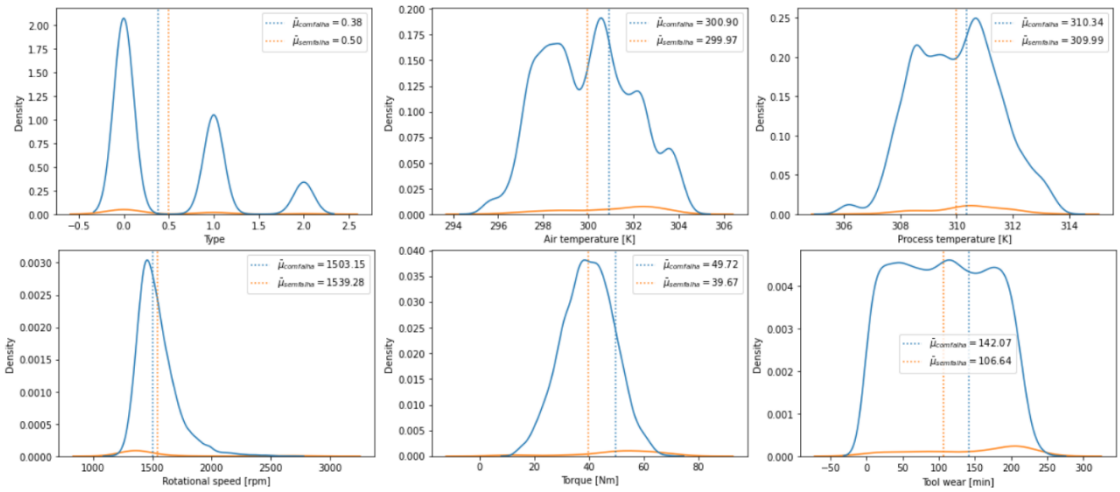
As hipóteses testadas foram:

- $H0$: $\mu 1 = \mu 2$ ou $H0$: $\mu 1 - \mu 2 = 0$
- $H1$: $\mu 1 \neq \mu 2$ ou $H1$: $\mu 1 - \mu 2 \neq 0$

Para tais hipóteses, as respostas obtidas para cada coluna foram:

- *Type*: Rejeição de H0;
- *Air Temperature [K]*: Rejeição de H0;
- *Process Temperature [K]*: Rejeição de H0;
- *Rotational Speed [rpm]*: Falha em rejeitar H0;
- *Torque [Nm]*: Rejeição de H0;
- *Tool Wear [min]*: Rejeição de H0.

Figura 24 – Visualização Gráfica do Teste de Hipótese de Separabilidade de *Target*



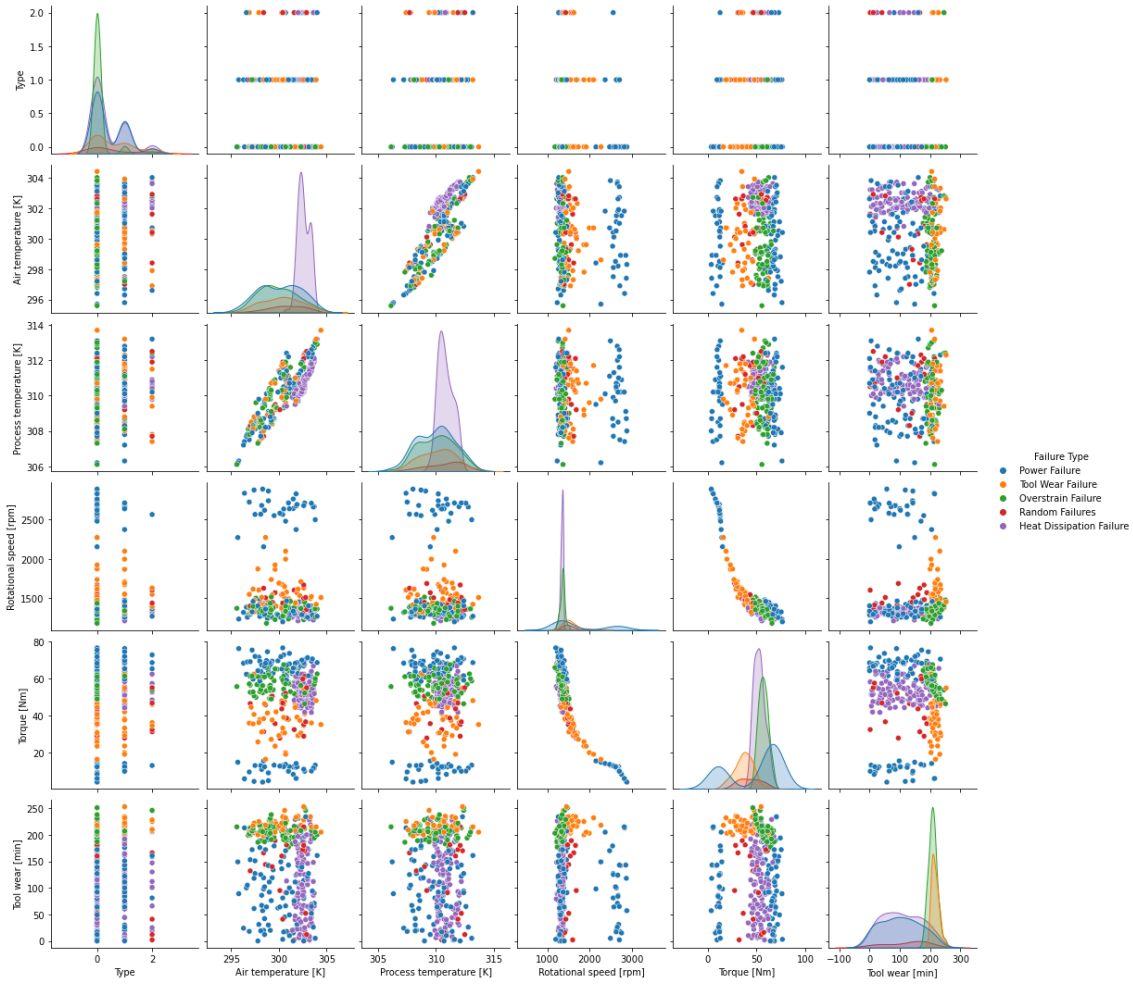
Autoria Própria

Com o teste de hipótese concluído, é possível afirmar que a AED está concluída e quase todas as variáveis conseguem gerar uma diferença entre a Falha e a Não Falha, indicando a capacidade de fazer modelos de predição, apesar da separabilidade não ser trivial. A criação destes será abordado na sessão 3.4 por questão de organização.

3.3.1.2. Parte 2

Para esta parte do trabalho, o conjunto de dados foi filtrado para conter apenas os dados cujo valor do *Target* é 1, ou seja, somente os dados que possuem falhas. O propósito deste é saber se, depois que o modelo previu uma falha, ele consegue nos dizer qual a falha específica. Sendo assim, foi feito novamente um gráfico cruzando 2 a 2 as colunas do conjunto de dados para entender a separabilidade de cada uma das classes.

Figura 25 – Gráfico Cruzado 2 a 2 das Colunas com Separação de *Failure Type* do Conjunto de Dados Filtrado pelo *Target*

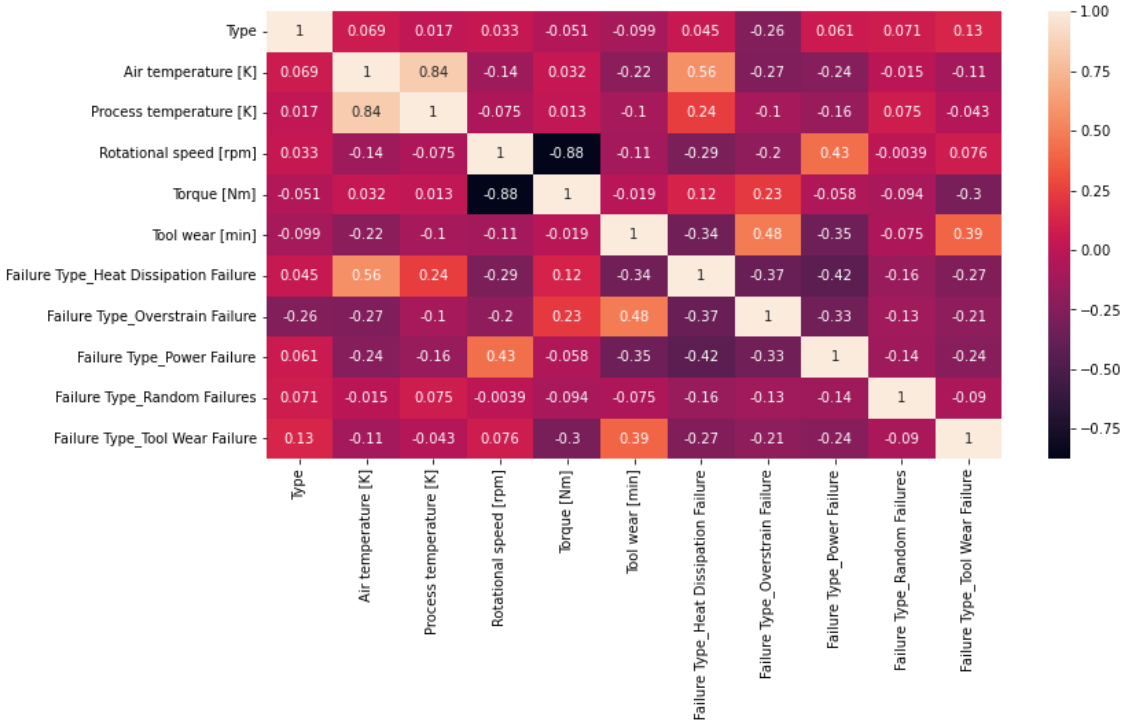


Autoria Própria

Observando a Figura 25, foi possível observar uma boa separabilidade das classes, principalmente da *Power Failure*. Para checar mais a fundo essa separabilidade, a coluna

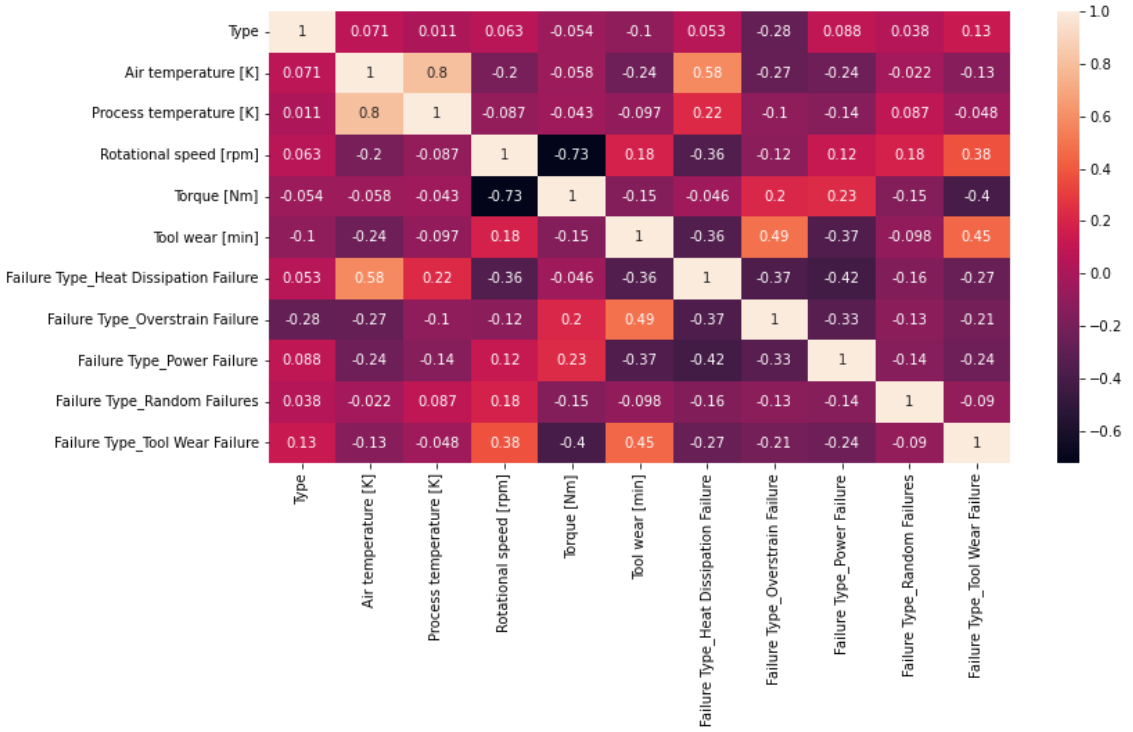
Failure Type foi separada em cinco colunas, cada uma para um tipo de falha¸ utilizando da função pronta *get_dummies* e, em seguida, foram feitos outros dois mapas de calor para checar as correlações entre cada coluna e cada classe de *Failure Type.*

Figura 26 – Mapa de Calor Utilizando da Correlação de *Pearson* para a Parte 2 do Teste 1



Autoria Própria

Figura 27 – Mapa de Calor Utilizando da Correlação de *Spearman* para a Parte 2 do Teste 1



Autoria Própria

Analisando os 2 mapas de calor, foi identificado algumas relações interessantes. Algumas delas são:

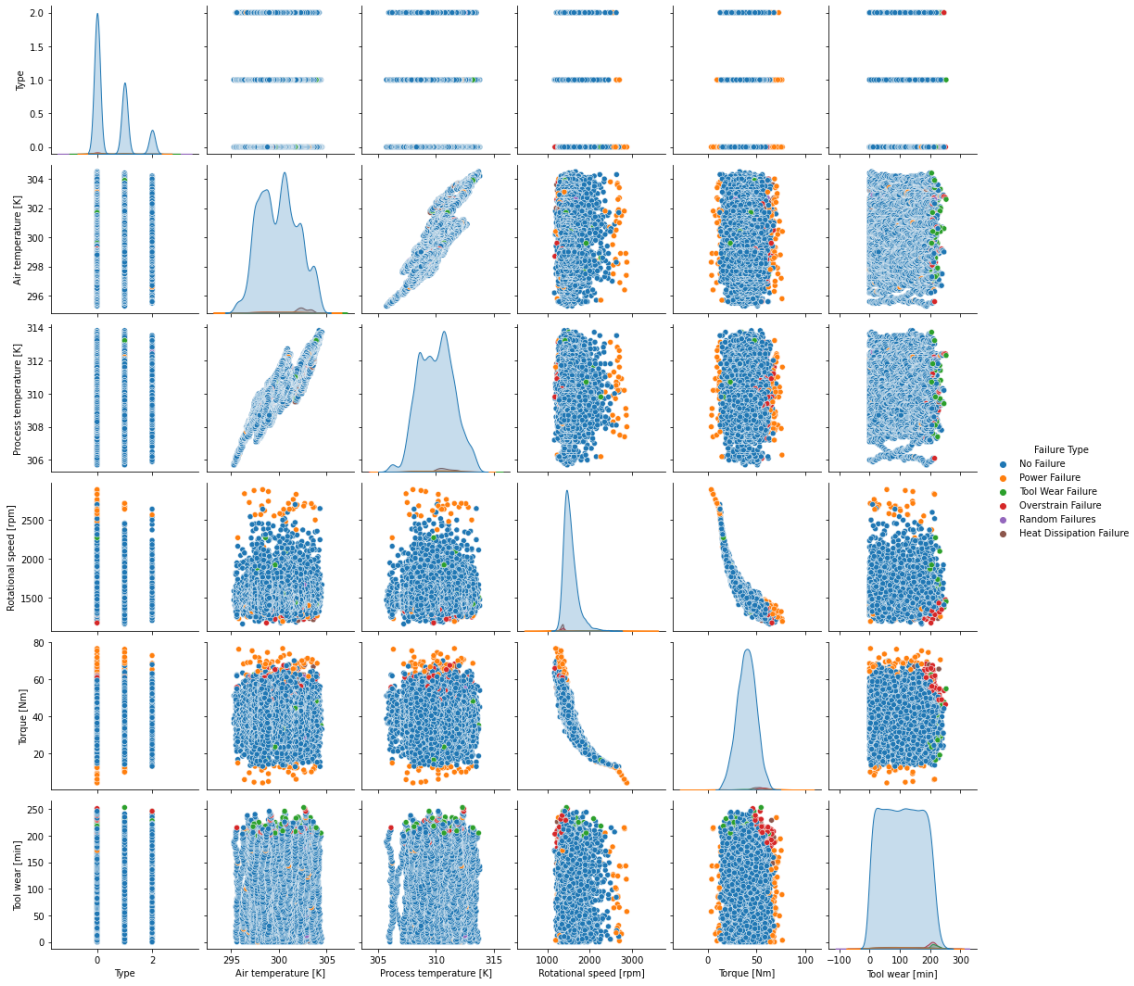
- Torque tem uma correlação média linear e monotética com as falhas por *Overstrain* e *Tool Wear*;
- A falha por *Tool Wear* possui uma correlação linear quase nula com a *Rotational Speed*, porém tem uma correlação monotética média com ela.
- A falha por *Heat Dissipation* tem uma correlação forte e positiva com a *Air Temperature* e possui correlações médias negativas com a *Rotational Speed* e com o *Tool Wear*.

Para essa parte do teste, não foi necessário a comprovação por teste de hipótese, uma vez que é possível comprovar uma separabilidade clara das classes através dos mapas de calor. Sendo assim, a AED para essa parte está concluída e a criação dos modelos para a classificação será abordado na sessão 3.4 por questão de organização.

3.3.2. Análise dos Dados para o Teste 2

Como dito na sessão 3.3, esse teste consistiu na remoção da coluna *Target* para ver como os modelos performariam em tentar classificar direto todas as classes de *Failure Type* utilizando do conjunto de dados em sua totalidade. Para esse teste, foi feito novamente um gráfico cruzado 2 a 2 das colunas separando as cores pelas classes de *Failure Type*, porém, agora, em sua totalidade.

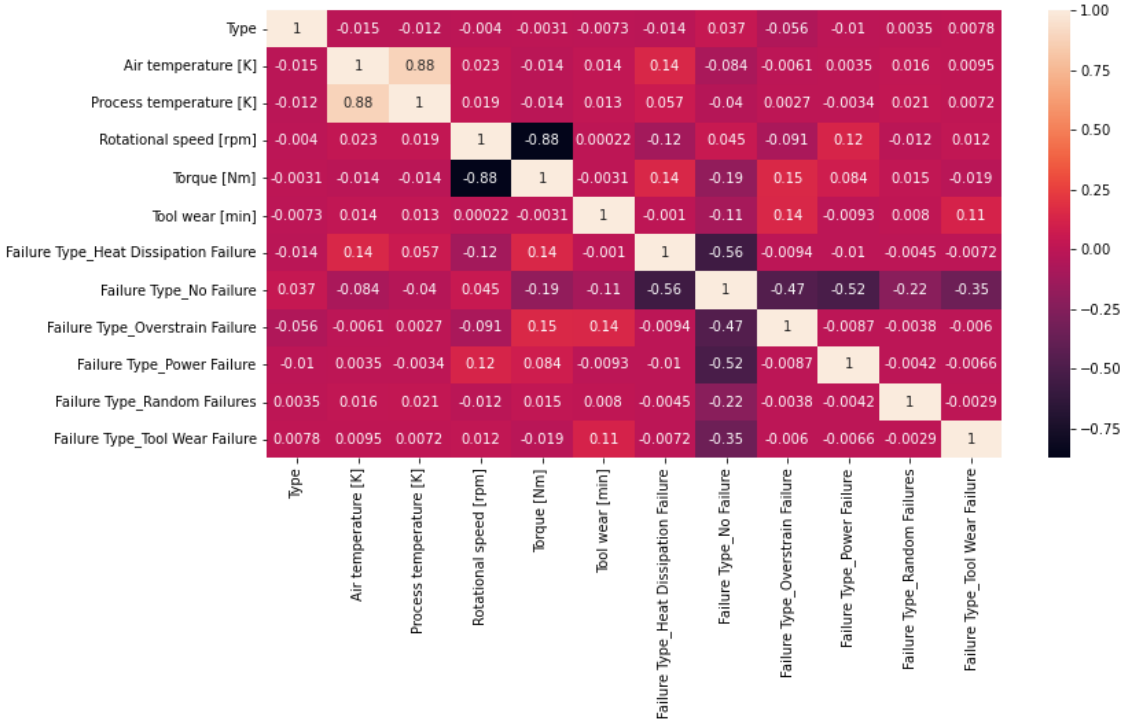
Figura 28 - Gráfico Cruzado 2 a 2 das Colunas com Separação de *Failure Type* do Conjunto de Dados em sua Totalidade



Autoria Própria

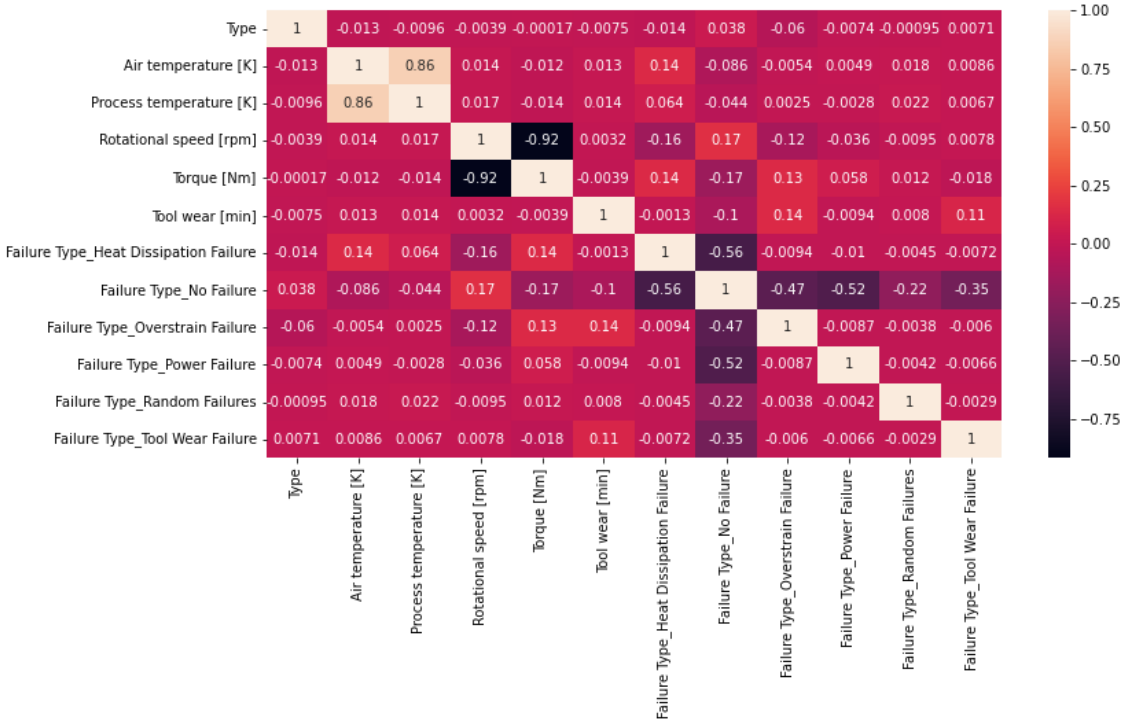
Após, foram feitos os mapas de calor do conjunto de dados, separando novamente as classes de *Failure Type* utilizando da função *get_dummies* para checar suas correlações.

Figura 29 – Mapa de Calor Utilizando da Correlação de *Pearson* para o Teste 2



Autoria Própria

Figura 30 – Mapa de Calor Utilizando da Correlação de *Spearman* para o Teste 2



Autoria Própria

Analisando a Figura 29 e a Figura 30, foi possível perceber a nítida diferença entre as correlações do teste proposto na sessão 3.3.1.2 e a as correlações para esse teste, o que indicou que, talvez, fosse mais difícil conseguir diferencial as classes. Como o teste de hipótese já foi realizado na sessão 3.3.1.1.2, não foi necessário repeti-lo para termos certeza da separabilidade. Neste ponto, foi concluída a AED para esse teste e os modelos de predição criados serão abordados na sessão 3.4 para melhor organização do trabalho.

3.4. Construindo os Modelos de Predição

Nessa sessão será apenas apresentado os parâmetros utilizados em cada modelo e quais métricas de avaliação eles devem priorizar uma vez que os resultados serão discutidos no Capítulo 4. Com exceção da métrica de avaliação, os parâmetros de busca utilizados para os três testes foram os mesmos para cada modelo.

3.4.1. Métricas Escolhidas

Como as métricas foram explicadas na sessão 2.4.1, nessa sessão será abordado apenas os motivos de casa uso em cada teste.

A métrica escolhida para cada foi:

- **Teste 1 Parte 1**: *Recall* para *Target* = 1. Nesse caso, foi de extrema importância dar prioridade para o *Recall* pois, caso a predição diga que o equipamento está com problema, uma equipe técnica irá averiguar se realmente tem um problema e se compensa parar a produção para realizar sua manutenção ou se agendará um dia específico para tal. Case a predição erre dizendo que o equipamento não vai ter um problema e ele, de fato, terá, ocasionará em uma falha não esperada no meio de uma produção e os danos podem ser muito maiores.
- **Teste 1 Parte 2**: *Accuracy*. Para este caso, como o conjunto de dados se trata dos dados que já foram constatados como falha, identificar qual a falha é a opção correta. Apesar de não ser de extrema importância igual à da parte anterior, acertar qual foi o dano sofrido pelo equipamento poupará tempo dos funcionários em tentar identificá-lo.
- **Teste 2**: *Precision* para classe *No Failure*. Assim como no caso do Teste 1 Parte 1, é menos danoso que as predições errem os tipos de falha que ela diga que uma delas será não falha. Sendo assim, como não é possível priorizar o *Recall* de cinco classes ao mesmo tempo, é de boa prática dar prioridade à *Precision* para classe *No Failure*.

3.4.2. Criação dos Modelos

Para escolher quais os melhores parâmetros de cada modelo, foi necessário utilizar das funções *GridSearch* e *RandomSearch* da biblioteca Scikit-Learn para fazer o treinamento dos modelos usando as combinações fornecidas, assim como explicado na sessão 2.4.3, para depois escolher aquela que melhor performou nos treinos para depois colocá-los para fazer os testes.

Os parâmetros utilizados para a criação dos modelos se encontram todos no trabalho publicado no *Github* no repositório de nome Trabalho-de-Conclusao-de-Curso-Eduardo-Acrani-Ruivo e pode ser utilizado para reprodução de tal. A explicação e apresentação dos parâmetros destes não possui relevância para os resultados do projeto.

4. RESULTADOS E DISCUSSÕES

Com todos os melhores parâmetros escolhidos, foi hora de submetê-los ao treino e em seguida ao teste para que fosse possível ter uma comparação. Logo após os modelos performarem, foi hora de comparar qual deles obteve a melhor performance. Para melhor organização, toda a solução se encontra, também, no trabalho publicado no *Github e* serão apresentados e discutidos aqueles que tiveram os melhores resultados.

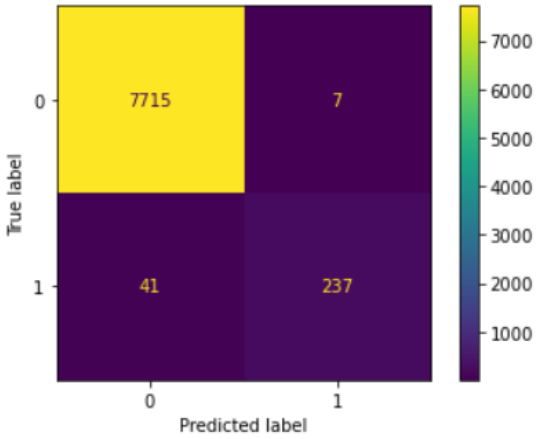
4.1. Resultado do Teste 1

4.1.1. Parte 1

Para essa parte do primeiro teste, a métrica de pontuação utilizada para foi o *Recall*, assim como explicado na sessão 3.4.1, e o classificador que melhor performou foi a Árvore de Decisões com uma pontuação média de 0.644 na escolha do modelo, obteve um resultado de 85% de *Recall* no treino e de 77% para o teste.

Figura 31 – Resultado do Treino do Modelo do Teste 1 Parte 1

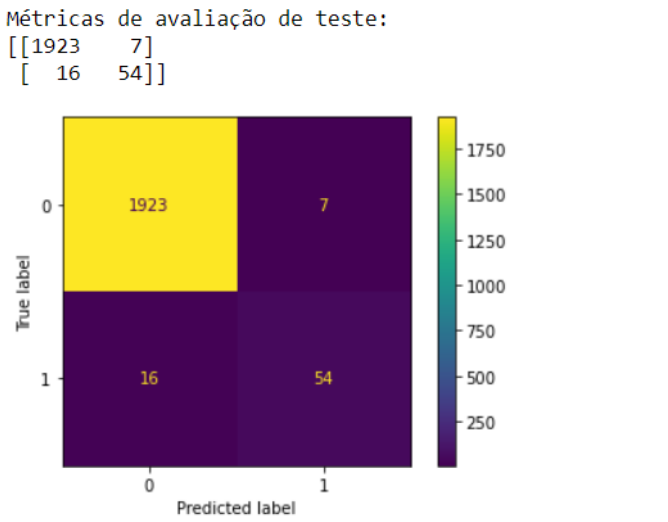
Métricas de avaliação de treino:



	precision	recall	f1-score	support
0	0.99	1.00	1.00	7722
1	0.97	0.85	0.91	278
accuracy			0.99	8000
macro avg	0.98	0.93	0.95	8000
weighted avg	0.99	0.99	0.99	8000

Autoria Própria

Figura 32 – Resultado do Teste do Modelo do Teste 1 Parte 1



	precision	recall	f1-score	support
0	0.99	1.00	0.99	1930
1	0.89	0.77	0.82	70
accuracy			0.99	2000
macro avg	0.94	0.88	0.91	2000
weighted avg	0.99	0.99	0.99	2000

Autoria Própria

Pelo nível de desbalanceamento do conjunto de dados, esse resultado pode ser considerado surpreendente, uma vez que a quantidade de dados que o conjunto tinha para treinar era mínima e, ainda sim, conseguiu uma performance de 77% para o *Recall*. É perceptível uma diferença de 8% entre o resultado do treino e do teste do modelo, que pode ter sido dado pela difícil separabilidade dos dados e pela distribuição que, na separação aleatória dos conjuntos de dados, pode ter alocado dados no conjunto de teste que não puderam ser observados no conjunto de treino.

Apesar do resultado, é possível afirmar que este modelo ainda não está pronto para implementação nas indústrias, uma vez que ele classificou como "não falha" 16 equipamentos que vão falhar. Mesmo que a porcentagem de acerto ser relativamente alta, esse número de equipamentos que vão sofrer falha, pode ser de extrema importância para o funcionamento da indústria e causaria um prejuízo maior que os 54 equipamentos classificados corretamente proporcionariam de lucro. Sendo assim, esse modelo preciso de mais estudos que serão sugeridos no Capítulo 5.

Para a primeira parte do primeiro teste, foi colocado em ordem de porcentagem de acerto do *Recall* os resultados dos testes de todos os modelos para quesitos de comparação:

- 1º - Árvore de decisões – 77%
- 2º - Regressão Logistica – 53%
- 3º - Ada Boost – 41%

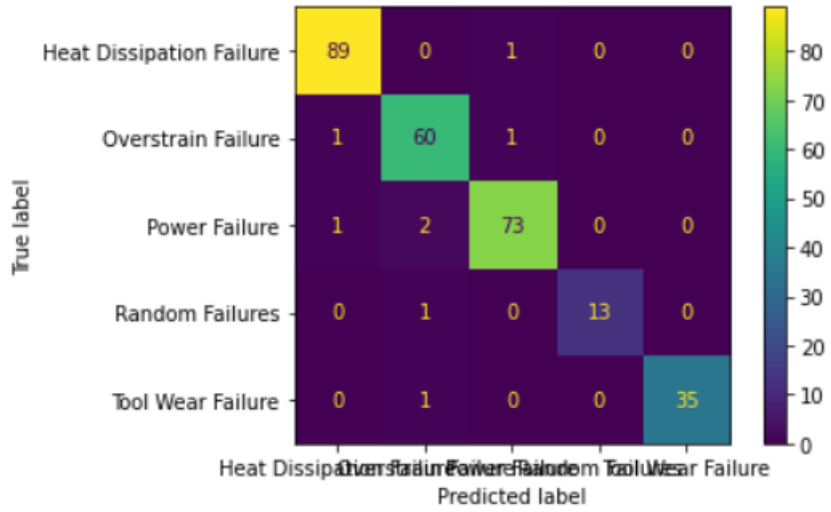
- 4º - Knn – 40%
- 5º - Random Forest – 21%

4.1.2. Parte 2

Já para essa parte do primeiro teste, a métrica de pontuação utilizada para foi a *Accuracy*, assim como explicado na sessão 3.4.1, e o classificador que melhor performou foi a Regressão Logística com uma pontuação média de 0.885 na escolha do modelo, obteve um resultado de 97% de *Accuracy* no treino e de 87% para o teste.

Figura 33 – Resultado do Treino do Modelo do Teste 1 Parte 2

Métricas de avaliação de treino:

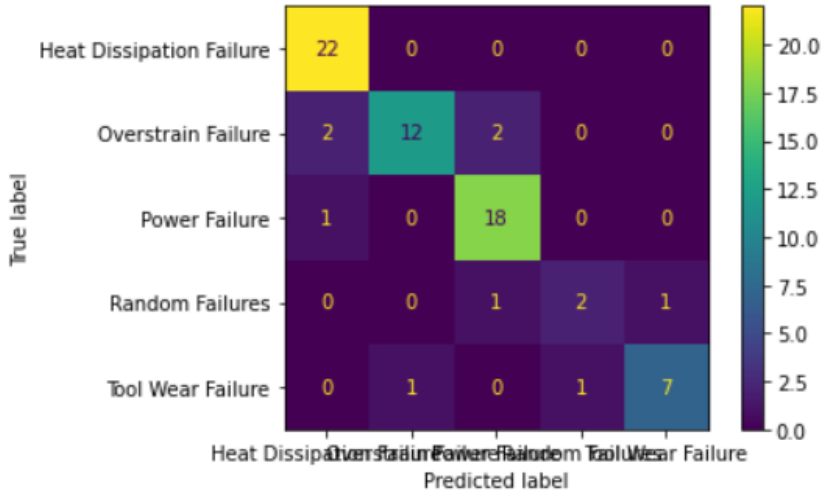


	precision	recall	f1-score	support
Heat Dissipation Failure	0.98	0.99	0.98	90
Overstrain Failure	0.94	0.97	0.95	62
Power Failure	0.97	0.96	0.97	76
Random Failures	1.00	0.93	0.96	14
Tool Wear Failure	1.00	0.97	0.99	36
accuracy			0.97	278
macro avg	0.98	0.96	0.97	278
weighted avg	0.97	0.97	0.97	278

Autoria Própria

Figura 34 – Resultado do Teste do Modelo do Teste 1 Parte 2

```
Métricas de avaliação de teste:
[[22  0  0  0  0]
 [ 2 12  2  0  0]
 [ 1  0 18  0  0]
 [ 0  0  1  2  1]
 [ 0  1  0  1  7]]
```



	precision	recall	f1-score	support
Heat Dissipation Failure	0.88	1.00	0.94	22
Overstrain Failure	0.92	0.75	0.83	16
Power Failure	0.86	0.95	0.90	19
Random Failures	0.67	0.50	0.57	4
Tool Wear Failure	0.88	0.78	0.82	9
accuracy			0.87	70
macro avg	0.84	0.80	0.81	70
weighted avg	0.87	0.87	0.87	70

Autoria Própria

Observando tanto o treino e o teste, é possível notar que, de fato, os dados com valores considerados *outliers* da coluna *Torque [Nm]* realmente foram de utilidade para a classificação correta da classe *Power Failure* com um *f1-score* de 90%. Apesar da diferença de 10% na *Accuracy* do treino e do teste do modelo, é possível dizer que este é confiável, principalmente para classificação do *Heat Dissipation Failure* e *Power Failure* e economizaria tempo dos funcionários ao tentar identificar o problema.

Apesar do resultado, também é possível afirmar que este modelo não está pronto para implementação nas indústrias, uma vez que este faz parte de um conjunto maior e tem a necessidade da classificação correta do *Target* para ser utilizado. Como sua primeira parte foi reprovada pela quantidade de equipamentos classificados errados, este também se torna inviável.

Porém, esse modelo não pode ser descartado completamente, já que sozinho, seu poder de classificação e de ajuda nas indústrias é alto e seus erros não são impactantes na manutenção, caso aplicado de maneira correta. Sendo assim, como modelo individual é possível ser implementado, caso tenha algo ou alguém que consiga identificar corretamente quaisquer indícios de falha nos equipamentos.

Para a segunda parte do primeiro teste, foi colocado em ordem de porcentagem de acerto da *Accuracy* os resultados dos testes de todos os modelos para quesito de comparação:

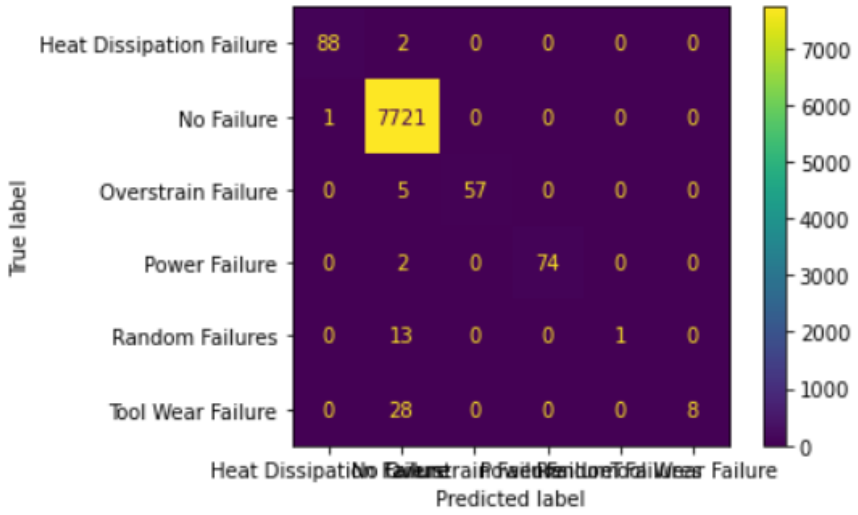
- 1º - Regressão Logistica – 87%
- 2º - Árvore de Decisões – 86%
- 3º - Random Forest – 86%
- 4º - Knn – 76%
- 5º - AdaBoost – 59%

4.2. Resultados do Teste 2

Para o segundo teste, a métrica de pontuação utilizada para foi a *Precision*, assim como explicado na sessão 3.4.1, e o classificador que melhor performou foi a Árvore de Decisões com uma pontuação média de 0.977 na escolha do modelo, obteve um resultado de 99% de *Precision* no treino e de 99% para o teste.

Figura 35 – Resultado do Treino do Modelo do Teste 2

Métricas de avaliação de treino:

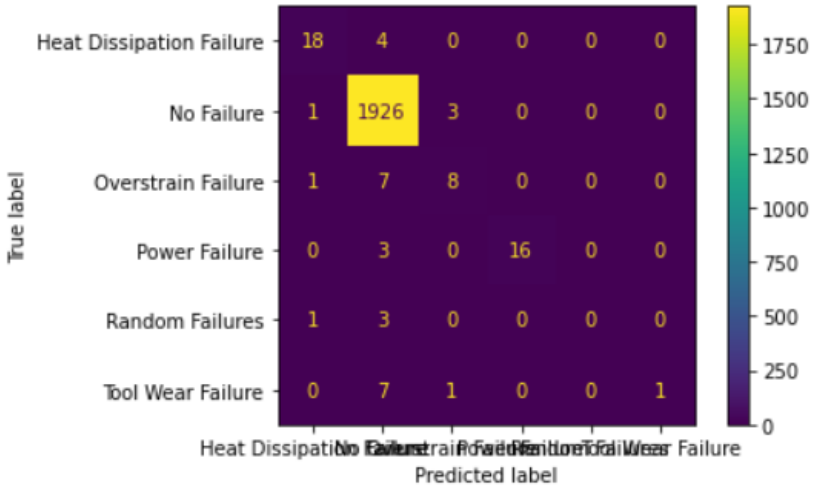


	precision	recall	f1-score	support
Heat Dissipation Failure	0.99	0.98	0.98	90
No Failure	0.99	1.00	1.00	7722
Overstrain Failure	1.00	0.92	0.96	62
Power Failure	1.00	0.97	0.99	76
Random Failures	1.00	0.07	0.13	14
Tool Wear Failure	1.00	0.22	0.36	36
accuracy			0.99	8000
macro avg	1.00	0.69	0.74	8000
weighted avg	0.99	0.99	0.99	8000

Autoria Própria

Figura 36 – Resultado do Teste do Modelo do Teste 2

```
Métricas de avaliação de teste:
[[  18    4    0    0    0    0]
 [   1 1926    3    0    0    0]
 [   1    7    8    0    0    0]
 [   0    3    0   16    0    0]
 [   1    3    0    0    0    0]
 [   0    7    1    0    0    1]]
```



	precision	recall	f1-score	support
Heat Dissipation Failure	0.86	0.82	0.84	22
No Failure	0.99	1.00	0.99	1930
Overstrain Failure	0.67	0.50	0.57	16
Power Failure	1.00	0.84	0.91	19
Random Failures	0.00	0.00	0.00	4
Tool Wear Failure	1.00	0.11	0.20	9
accuracy			0.98	2000
macro avg	0.75	0.54	0.59	2000
weighted avg	0.98	0.98	0.98	2000

Autoria Própria

Observando o resultado do treino e teste do modelo, a primeira impressão é de que o modelo está quase perfeito. Porém, como o conjunto de dados é desbalanceado e possui, aproximadamente, 96.5% dos dados com o *Failure Type* como *No Failure* e, caso o modelo preveja todos os dados como tal, ele já tem uma performance de 96.5% para *Accuracy* e para *Precision.* O que indica que a prioridade é analisar quantos dados que são considerados falhas que este prevê como não tendo falha.

Tendo essa observação em mente, é possível dizer que o modelo não está pronto para ser implementado na indústria, uma vez que a quantidade de erros cometidos pelo modelo ao classificar falhas como não falhas é maior que os erros cometidos na primeira parte do primeiro teste. Esse resultado pode ser sido causado tanto pela distribuição aleatória de dados entre o conjunto de dados de treino e de teste, quanto pela dificuldade

de identificação direta dos *Failure Type* devido a difícil separação constatada na sessão 3.3.1.1.2.

Para o segundo teste, foi colocado em ordem de porcentagem de acerto da *Precision* os resultados dos testes de todos os modelos para quesitos de comparação. Para critério de desempate foi utilizado a quantidade de dados errados para precisão:

- 1º - Árvore de Decisões – 99% - 24 Erros de precisão - 31 Erros totais
- 2º - Regressão Logistica – 99% - 26 Erros de precisão - 36 Erros totais
- 3º - Knn – 97% - 50 Erros de precisão - 57 Erros totais
- 4º - Random Forest – 97% - 58 Erros de precisão - 60 Erros totais
- 5º - AdaBoost – 91% - 24 Erros de precisão - 1738 Erros totais

5. CONCLUSÃO E TRABALHOS FUTUROS

Esse trabalho teve como objetivo estudar um conjunto de dados sintéticos gerados para simular uma indústria fictícia, que foram gerados para se assemelhar com a realidade das indústrias reais, uma vez que a obtenção desses dados é difícil, na tentativa de gerar uma melhoria para a área de manutenção quando se trata de manutenção preditiva. O monitoramento feito por uma máquina que consegue distinguir quando um equipamento será danificado por algum tipo de falha, tem a capacidade de melhorar a qualidade da manutenção, facilitar o trabalho importante que esse setor possui e reduzir as perdas da empresa.

Levando em consideração esses objetivos, o estudo foi executado tomando em conta seu desbalanceamento dos dados e a separabilidade dos alvos não trivial, sendo assim, tendo as expectativas sobre os resultados reduzidas, uma vez que a biblioteca Scikit-Learn não é recomendado para trabalhar com essas condições.

Sabendo disso, a separação do trabalho em dois testes diferentes e a execução da AED completa para ambos foi de grande importância para o entendimento do problema como um todo e as dificuldades a serem enfrentadas. Com os modelos criados, foi observado que, em ambos os testes, apresentaram dificuldade para classificar corretamente os dados, assim como esperado, e os classificadores que melhor se sobressaíram foram a Árvore de Decisões e a Regressão Logística. Apesar das dificuldades encontradas e da expectativa baixa, os resultados foram surpreendentes.

Embora as expectativas terem sido superadas, os modelos continuam precários e precisam passar por mais estudos para evitar ao máximo problemas nas indústrias e poder facilitar tanto o trabalho do setor de manutenção, quanto para reduzir o tempo em que a produção está parada, assim, aumentando o lucro das indústrias.

Sendo assim, concluímos que os modelos, do jeito que estão, não possuem competência o suficiente para serem postos em prática, pois, apesar da quantidade de acertos ser bem maior que a quantidade de erros, os erros têm uma capacidade prejudicial grande suficiente para que seja mais viável o monitoramento humano. Por fim, ainda existe muito estudo na área para serem feitos a fim de resolver esse problema de maneira que seja mais viável a modelagem de um modelo utilizando de ML do que delegando essa função a um funcionário.

5.1. Trabalhos Futuros

Como os dados utilizados são sintéticos, uma proposta seria adquirir dados de alguma empresa que os fornecesse para fins de estudos e tomando este trabalho como um ponto de comparação. Dessa forma, seria possível descobrir se o projeto conseguiu, de fato, representar um cenário real.

Outro trabalho que pode ser feito é a utilização de métodos de balanceamento artificial dos dados para que os classificadores utilizados para criação dos modelos tenham uma base melhor para identificação das classes. Esse balanceamento pode ser feito tanto nos dados sintéticos quanto em dados reais.

Por fim, estudar se existem bibliotecas de predição que consigam trabalhar com os dados desbalanceados. Caso existam, é possível refazer os modelos e utilizar esse trabalho com base de comparação para os novos modelos.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ACRANI, Eduardo. Trabalho-de-Conclusao-de-Curso-Eduardo-Acrani-Ruivo, 2023. Disponível em: < https://github.com/EduardoAcrani/Trabalho-de-Conclusao-de-Curso-Eduardo-Acrani-Ruivo >. Acesso em 9 de out. de 2023

AZAMBUJA, Pedro. AdaBoost (Adaptive Boost), 2020. Disponível em: < https://pedroazambuja.medium.com/adaboost-adaptive-boosting-dbbec150fced >. Acesso em 16 de out. de 2023

BANSAL, Shivam. Machine Predictive Maintenance Classification, 2021. Disponível em: < https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification >. Acesso em 4 de jul. de 2023

BELMONTE, Paulo. Manutenção Preditiva com Machine Learning: Previsões e Otimização. **AWARI**, 2023. Disponível em: <https://awari.com.br/manutencao-preditiva-com-machine-learning-previsoes-e-otimizacao/?utm_source=blog&utm_campaign=projeto+blog&utm_medium=Manuten%C3%A7%C3%A3o%20Preditiva%20com%20Machine%20Learning:%20Previs%C3%B5es%20e%20Otimiza%C3%A7%C3%A3o>. Acesso em 20 de jul. de 2023

Entenda o que é análise exploratória de dados e sua importância para a Data Science. **ONCASE**, 2021. Disponível em: <https://www.oncase.com.br/blog/data-driven/analise-exploratoria-de-dados/>. Acesso em 16 de jul. de 2023

HOPPEN, Joni; PRATES, Wlademir. Outliers, o que são e como tratá-los em uma análise de dado?. **AQUARELA,** 2017. Disponível em: < https://www.aquare.la/o-que-sao-outliers-e-como-trata-los-em-uma-analise-de-dados/> . Acesso em 15 de out. de 2023

KARDEC, ALAN ; NASCIF, JÚLIO **Manutenção Preditiva: Fator de Sucessão na Gestão Empresarial**. 1. Ed - Rio de Janeiro: Qualitymark Editora, 2013.

LOCALWEB, Equipe. Jupyter Notebook: saiba o que é e como utilizar a ferramenta. **LOCALWEB**, 2023. Disponível em: < https://www.locaweb.com.br/blog/temas/codigo-aberto/jupyter-notebook-o-que-e/>. Acesso em 09 de out. de 2023

Manutenção Preditiva: o que é? Quais técnicas e vantagens? **ABECOM**, 2021. Disponível em: <https://www.abecom.com.br/o-que-e-manutencao-preditiva/>. Acesso em 16 de jul. de 2023

NETTO, AMÍLCAR; MACIEL, FRANCISCO **Python Pata Data Science: E Machine Learning Descompilado**. 1. Ed - Rio de Janeiro: Alta Books Editora, 2021.

NOGARE, Diego. Performance de Machine Learning – Matriz de Confusão. **DIEGONOGARE,** 2020. Disponível em: < https://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/ >. Acesso em 16 de out. de 2023

O que é Machine Learning?. **ORACLE** [s.d.]c. Disponível em: < https://www.oracle.com/br/artificial-intelligence/machine-learning/what-is-machine-learning/ >. Acesso em 16 de out. de 2023

O que é Python?. **AWS**, [s.d.]a. Disponível em: < https://aws.amazon.com/pt/what-is/python/>. Acesso em 15 de out. de 2023

O que é regressão logística?. **AWS**, [s.d.]b. Disponível em: < https://aws.amazon.com/pt/what-is/logistic-regression/ >. Acesso em 16 de out. de 2023

O que é um DataFrame?. **DATABRICKS** [s.d.]. Disponível em: < https://www.databricks.com/br/glossary/what-are-dataframes>. Acesso em 9 de out. de 2023

Oracle® Fusion Cloud EPM Como Trabalhar com o Planning – IQR (Intervalo entre Quartis). **ORACLE** [s.d.]a. Disponível em: < https://docs.oracle.com/cloud/help/pt_BR/pbcs_common/PFUSU/insights_metrics_IQR.htm#PFUSU-GUID-CF37CAEA-730B-4346-801E-64612719FF6B >. Acesso em 28 de set. de 2023

Oracle® Fusion Cloud EPM Como Trabalhar com o Planning – Z-Score. **ORACLE** [s.d.]b. Disponível em: < https://docs.oracle.com/cloud/help/pt_BR/pbcs_common/PFUSU/insights_metrics_Z-Score.htm#PFUSU-GUID-640CEBD1-33A2-4B3C-BD81-EB283F82D879 >. Acesso em 28 de set. de 2023

PERES, Luca. Como detectar e tratar outliers com Python. **MEDIUM**, 2021. Disponível em: <https://medium.com/@lucapqg/como-detectar-e-tratar-outliers-com-python-ca2cf088c160>. Acesso em 21 de set. de 2023

Quais as vantagens e desvantagens da manutenção preditiva? **ABECOM**, 2023. Disponível em: <https://www.abecom.com.br/manutencao-preditiva-vantagens-e-desvantagens/>. Acesso em 09 de out. de 2023

What is random forest?. **IBM**, [s.d.]. Disponível em: < https://www.ibm.com/topics/random-forest>. Acesso em 16 de out. de 2023