UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA

MURILO LEITE DE SOUSA

CIÊNCIA DE DADOS COMO FERRAMENTA PARA MANUTENÇÃO PREDITIVA PARA EQUIPAMENTOS INDUSTRIAIS SENSORIZADOS

ILHA SOLTEIRA
2023

MURILO LEITE DE SOUSA

CIÊNCIA DE DADOS COMO FERRAMENTA PARA MANUTENÇÃO PREDITIVA PARA EQUIPAMENTOS INDUSTRIAIS SENSORIZADOS

Trabalho de conclusão de curso apresentado como requisito parcial
para obtenção do título de Bacharel em Engenharia Elétrica
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Orientadora: Profª. Drª. Mara Lúcia Martins Lopes

ILHA SOLTEIRA
2023

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

S725c

Sousa, Murilo Leite de.
Ciência de dados como ferramenta para manutenção preditiva para equipamentos industriais sensorizados / Murilo Leite de Sousa. -- Ilha Solteira: [s.n.], 2023
80 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2023

Orientador: Mara Lúcia Martins Lopes

Inclui bibliografia

1. Ciência de dados. 2. Manutenção preditiva. 3. Aprendizado de máquina. 4. Equipamentos industriais.

Amanda Sertori dos Santos
Bibliotecária - CRB/8-9061
Seção Técnica de Referência, Atendimento ao Usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos dezoito do mês de dezembro do ano de dois mil e vinte e três, o discente ***Murilo Leite de Sousa,*** matriculado sob o nº 161050115, tendo como banca examinadora a sua orientadora, *a Profª. Drª. Mara Lúcia Martins Lopes, a Pós-doutorando Carlos Roberto dos Santos Júnior e o Doutorando Reginaldo José da Silva,* apresentou o Trabalho de Graduação do Curso de Engenharia Elétrica (FEIS-UNESP) intitulado **"Ciência de dados como ferramenta para manutenção preditiva para equipamentos industriais sensorizados"** obtendo a nota 10 (DEZ) e conceito APROVADO .

Murilo Leite de Sousa

Murilo Leite de Sousa

- discente -

Mara R. M. Lopes

Profª. Drª. Mara Lúcia Martins Lopes

- orientadora -

Carlos R S Jr.

Pós-doutorando Carlos Roberto dos Santos Júnior

- Membro da Banca -

Reginaldo José da Silva

Doutorando Reginaldo José da Silva

- Membro da Banca -

Faculdade de Engenharia de Ilha Solteira
Avenida Brasil Centro, 56. Caixa Postal 31, CEP: 15385-000, Ilha Solteira – São Paulo – Brasil.
Pabx (18) 37431000 - Fax (18) 3742-2735 - www.feis.unesp.br

AGRADECIMENTOS

É muito importante para mim estar escrevendo esses agradecimentos. Ilha Solteira me mostrou um outro lado da vida, me trouxe novas experiências, novas pessoas para guardar no coração e muita história para contar. Primeiramente, agradeço a Deus por toda força, ensinamento, lutas e vitórias que vivi durante os anos de universidade e mais o tempo para inserção do mercado de trabalho, de fato fui muito abençoado por ter feito a escolha de cursar Engenharia Elétrica na FEIS. Agora, partindo para o que é terreno, meu maior agradecimento deve-se a Sra. Maria Lourdes Vicário Zanin, *"*Vó Lurdinha*"* como desde criança a chamo. Foi pelos esforços e ajuda dela e do Sr. Edson Geraldo Zanin, o *"*Jovem*"*, que hoje estou finalizando esse curso e escrevendo este trabalho de graduação. Tão importantes quanto, meus pais, Ricardo Antônio Leite de Sousa e Daiana Lígia Batista de Sousa, que me deram todo suporte emocional, confiança, ensinamentos e me mostram todos os dias que Deus deve ser o primeiro lugar da nossa vida. À minha irmã, Ana Clara Leite de Sousa Gomes, minha melhor amiga, e meu irmão-cunhado Thiago Henrique Gomes, meu irmão de vida que posteriormente se tornou cunhado (é uma história engraçada).

Em ordem cronológica de acontecimentos (não de importância), não posso deixar de agradecer à minha família de Ilha Solteira: a Família Keoma (Leonardo, Paula, Dom e Romeu), que conheci através da Igreja Fonte da Vida. Paula e Léo, obrigado por me acolherem, me *"*adotarem*"*, me ensinarem e me manterem firme sempre quando eu achava que não ia conseguir, continuo sendo forte e corajoso, sal e luz em todo lugar que estou. Não posso deixar de citar todos os integrantes da equipe principal do ministério de adolescentes Dokmos da igreja Fonte da Vida, vocês são muito importantes para mim, guardo todos no coração! (Roger, Gada, Zu, Gaby, Onaga, Fê, Mari, AnaLu, MiniAr, Mari, Túlio, Gui e Pâmela), estendo os agradecimentos às famílias de cada um também.

À melhor turma que a Engenharia Elétrica da FEIS já viu, 16/1, vocês são incríveis! Agora, aos meus irmãos da igreja em que congrego atualmente, Paz Church Lençóis Paulista, os *"*colegas*"*, sou grato à Deus pela vida de vocês, muito obrigado por tudo!

Por fim, à toda CIDA, minha equipe de Ciência de Dados da RT Dados PRG do Itaú Unibanco, onde há dois anos me aturam e me ensinam todos os dias na questão pessoal e profissional.

RESUMO

Sabe-se que atualmente, o principal objetivo das indústrias está em maximizar seus resultados e, consequentemente, obter maior lucro frente aos seus concorrentes. Desta forma, não somente no viés de vendas, as indústrias buscam otimizar ao máximo todos seus processos e isso inclui a manutenção de equipamentos industriais.

Com a indústria 4.0, a possibilidade de acompanhar as performances desses ativos industriais foi possível devido à atuação de sensores e de dados provenientes desse sensoriamento. Com isso, a manutenção preditiva traz uma solução que busca otimizar as manutenções de equipamentos com o objetivo de redução de paradas desnecessárias na produção utilizando para isso conceitos da ciência de dados para a predição de falhas no processo produtivo.

Desta forma, esse trabalho de graduação tem o intuito de mostrar como estruturar um projeto de ciência de dados para ser utilizado na agenda de manutenção preditiva das indústrias, realizando a predição de falhas com até 15 aferições dos sensores de antecedência para 100 equipamentos industriais sensorizados, utilizando as informações provenientes dos sensores como variáveis preditoras. Assim, é possível utilizar a saída do modelo de aprendizado de máquina para estruturar com antecedência as manutenções em máquinas, prevenindo paradas desnecessárias na produção.

Para isso, utilizou-se de conceitos de ciência de dados, aprendizado de máquina e manutenção preditiva. A metodologia utilizada para o desenvolvimento do projeto foi a CRISP-DM, uma metodologia usada em larga escala no mundo dos dados. Essa metodologia passa por todos os pilares necessários para um projeto de ciência de dados de sucesso, como o entendimento de negócio, onde foi necessário definir qual o problema a ser resolvido e qual variável resposta que se espera; o entendimento dos dados, em que foi realizado uma visualização geral dos dados, volumetria, qual o significado de cada variável preditora e adEquação da base de dados para a utilização em algoritmos de aprendizado de máquina; preparação de dados, momento em que foram definidos pontos de importância para serem explorados na base de dados para serem realizadas possíveis transformações e onde a variável resposta foi criada; modelagem que é onde o modelo foi treinado, otimizando hiperparâmetros e fazendo validação cruzada buscando reduzir o sobreajuste; avaliação do modelo, mostrando métricas de avaliação como precisão e sensibilidade para as quais foram obtidos valores de 78% e 91%, respectivamente, para um limiar de decisão de 0,35. Também é apresentada a curva ROC encontrada a partir da matriz de confusão obtida com a base de teste e, de maneira geral, é realizada uma discussão sobre os motivos e impactos dos resultados obtidos por meio dessas métricas encontradas. Por fim, é discutida uma proposta de orientação à equipe de manutenção sobre como utilizar esse modelo para estruturar as agendas de manutenção. Diante disso, este trabalho de graduação mostra como realizar a

integração das tecnologias atuais com o conceito de manutenção, em específico, a utilização da ciência de dados e algoritmos de aprendizado de máquina para prever falhas em equipamentos industriais, como utilizar os passos da metodologia do CRISP-DM visando organização e completude de um projeto, como interpretar resultados provenientes de algoritmos de aprendizado de máquina de classificação e de que forma uma equipe de manutenção pode utilizar tais resultados com o objetivo de reduzir falhas inesperadas nos equipamentos que constituem as linhas de produção de indústrias.

Palavras-chave – Manutenção Preditiva. Ciência de Dados. Aprendizado de Máquina. CRISP-DM.

ABSTRACT

It is known that currently, the main goal of industries is to maximize their results and, consequently, achieve greater profits compared to their competitors. In this regard, industries seek to fully optimize all their processes, including the maintenance of industrial equipment.

With Industry 4.0, the possibility of monitoring the performance of these industrial assets became feasible due to the use of sensors and data derived from this sensing. Therefore, predictive maintenance provides a solution that aims to optimize equipment maintenance with the goal of reducing unnecessary production downtime, using data science concepts for predicting faults in the production process.

Thus, this undergraduate thesis aimed to demonstrate how to structure a data science project for use in industries in the context of predictive maintenance, predicting faults with up to 15 cycles of anticipation for 100 sensorized industrial equipment, using sensor values as predictor variables. Thus, it is possible to use the output of the machine learning model to plan maintenance in advance, preventing unnecessary production stoppages.

For this purpose, concepts of data science, machine learning, and predictive maintenance were utilized. The methodology used for the project development was CRISP-DM, a methodology widely used in the world of data. This methodology goes through all the necessary pillars for a successful data science project, such as business understanding, where it was necessary to define the problem to be solved and the expected response variable; data understanding, where an overview of the data, volume, the meaning of each predictor variable, and data preparation for use in machine learning algorithms were performed; data preparation, where points of importance were defined to be explored in the database for possible transformations, and where the response variable was created; modeling, which is where the model was trained, optimizing hyperparameters and performing cross-validation to reduce overfitting; finally, model evaluation, showing evaluation metrics such as accuracy and sensitivity, for which values of 78% and 91% were obtained, respectively, with a decision threshold of 0,35. The ROC curve derived from the confusion matrix obtained from the test dataset is also presented. Overall, a discussion is conducted on the reasons and impacts of the results obtained through these metrics. Finally, a proposal for guidance to the maintenance team on how to use this model to structure maintenance schedules is discussed.

In light of this, this graduation thesis demonstrates how to integrate current technologies with the maintenance concept, specifically leveraging data science and machine learning algorithms to predict failures in industrial equipment. It outlines the steps of the CRISP-DM methodology for

project organization and completeness, interprets results from classification machine learning algorithms, and explores how a maintenance team can utilize such results to reduce unexpected failures in the equipment comprising industrial production lines.

Keywords - Predictive Maintenance. Data Science. Machine Learning. CRISP-DM.

LISTA DE ILUSTRAÇÕES

Figura 1 - Diagrama de Venn com as disciplinas que envolvem ciência de dados. 14
Figura 2 – Regressão Linear aplicada a um problema de vendas.....18
Figura 3 - Árvore de Decisão Simples 20
Figura 4 - Utilização da função Logit 22
Figura 5 - Treinamento de um algoritmo de Bagging 25
Figura 6 - CRISP-DM e suas fases.....28
Figura 7 - Gráfico de barras verticais.....37
Figura 8 - Gráfico de Setor.....37
Figura 9 - Gráficos de Dispersão.....38
Figura 10 - Histograma 39
Figura 11 - Box Plot 40
Figura 12 - Matriz de Correlação 41
Figura 13 - Validação Cruzada 43
Figura 14 - Matriz de Confusão 45
Figura 15 - Curva ROC 47
Figura 16 - Barreiras internas que dificultam a adoção das tecnologias digitais 48
Figura 17 - Fluxograma geral da Manutenção 50
Figura 18 - Arquivo de texto com as informações do problema 52
Figura 19 - Informações da base de dados 53
Figura 20 - Solução em *Python* para renomear colunas.....55
Figura 21 - Colunas renomeadas na base recebida.....55
Figura 22 - Corte na base de dados para entendimento da disposição das informações 56
Figura 23 - Quantidade de ciclos até a falha por Máquina.....57
Figura 24 - Histograma da quantidade de ciclos até a falha por equipamento.....58
Figura 25 - Média, Mediana e Moda.....58
Figura 26 - Mapa de calor considerando a correlação entre cada sensor 60
Figura 27 - Gráficos de dispersão dos sensores mais correlacionados.....61
Figura 28 - Gráficos de dispersão dos sensores mais correlacionados com distinção no momento da falha 62
Figura 29 - Gráficos de dispersão dos sensores mais correlacionados com distinção na iminência da falha 62
Figura 30 - Visualização dos tipos das variáveis e valores nulos.....63
Figura 31 - Código gerador da variável resposta (target).....65
Figura 32 - Estatísticas relacionadas às variáveis 65
Figura 33 - Código gerador de base de treino e teste 67
Figura 34 - Código em *Python* para definição dos melhores hiperparâmetros 69
Figura 35 - Treinamento do modelo utilizando Florestas Aleatórias e os melhores hiperparâmetros.....70
Figura 36 - Código para extração das propensões à iminência de falha 71

Figura 37 - Matriz de Confusão para as predições feitas com limiar de 0.5 72
Figura 38 - Valores de Precisão e Sensibilidade respectivamente 72
Figura 39 - Variação de precisão e recall de acordo com o limiar da matriz de confusão 74
Figura 40 - Curva ROC 74

LISTA DE TABELAS

Tabela 1 - Nome e descrição das colunas da base de dados recebida 54

SUMÁRIO

1 Introdução 13
2 Referencial teórico 14
2.1 Ciência de dados 14
2.2 Aprendizado de Máquina 15
2.2.1 Aprendizado Supervisionado 16
2.2.1.1 Algoritmos de Regressão 17
2.2.1.1.1 Regressão Linear 17
2.2.1.1.2 K Vizinhos Mais Próximos (KNN) – Aplicação em Regressão 18
2.2.1.1.3 Árvores de Decisão – Aplicação em Regressão 19
2.2.1.2 Algoritmos de Classificação 21
2.2.1.2.1 Regressão Logística 21
2.2.1.2.2 K Vizinhos Mais Próximos (KNN) – Aplicação em Classificação 23
2.2.1.2.3 Árvores de Decisão – Aplicação em Classificação 23
2.2.1.3 Algoritmos *Ensemble* – Agrupamento de Modelos 24
2.2.1.3.1 *Bagging* e Florestas Aleatórias 25
2.2.1.3.2 Boosting 26
2.2.2 Aprendizado não supervisionado 27
2.2.2.1 K Means Clustering 27
2.3 CRISP-DM – Projetos em ciência de dados 28
2.4 Estatística descritiva 30
2.4.1.1 Medidas de Localização 30
2.4.1.2 Média 30
2.4.1.3 Mediana 31
2.4.1.4 Quartis e Percentis 31
2.4.1.5 Outliers 31
2.4.1.6 Moda 32
2.4.2 Medidas de Variabilidade 32
2.4.2.1 Desvios 32
2.4.2.2 Variância 32
2.4.2.3 Desvio Padrão 33
2.5 Preparação dos dados 33
2.5.1 Tratamento de Valores Faltantes 33
2.5.2 Tratamento de variáveis categóricas 34
2.5.3 Escalonamento de variáveis 34
2.5.4 Seleção de Variáveis 35
2.6 Visualização de Dados 36
2.6.1 Gráficos de Barras 36

2.6.2 Gráficos de Setor 37
2.6.3 Gráficos de Dispersão 38
2.6.4 Histogramas 38
2.6.5 Box plot 39
2.6.6 Correlação 40
2.7 Métricas de Avaliação de Modelos 41
2.7.1 Validação cruzada 42
2.7.2 Avaliação em modelos de regressão 43
2.7.3 Avaliação em modelos de classificação 44
2.7.3.1 Matriz de Confusão 44
2.7.3.2 Sensibilidade 46
2.7.3.3 Precisão 46
2.7.3.4 Especificidade 46
2.7.3.5 Curva ROC e Área sob a Curva ROC 47
2.8 Manutenção Preditiva 47
2.8.1 A evolução da manutenção 49
3 Desenvolvimento 51
3.1 Entendimento de Negócio 51
3.2 Entendimento dos dados 52
3.2.1 Obtenção dos dados para o projeto 52
3.2.2 Entendimento e primeira visualização da base de dados 52
3.2.3 Análise Exploratória de Dados 56
3.3 Preparação dos dados 63
3.3.1 Pré-preparação dos dados 63
3.3.2 Criação da variável resposta (*target*) 64
3.3.3 Definição do algoritmo a ser utilizado e a necessidade de escalonamento 65
3.4 Modelagem 66
3.4.1 Definição de Solução 66
3.4.2 Divisão treino e teste para modelagem 67
3.4.3 Treinamento do modelo utilizando Florestas Aleatórias 68
3.5 Avaliação 71
3.5.1 Avaliação e entendimento dos resultados na base de teste 71
3.5.2 Aconselhamento à equipe de manutenção 75
4 Conclusão 77
REFERÊNCIAS 78
Apêndice A 80

1 INTRODUÇÃO

Nos últimos anos, o setor industrial testemunhou uma revolução transformadora impulsionada pelo avanço da tecnologia, especialmente no que diz respeito ao acompanhamento de informações obtidas por meio de sensores, ciência de dados e big data. O avanço da Indústria 4.0 trouxe consigo a possibilidade de obtenção de um grande volume de dados gerados por inúmeros meios, entre eles, equipamentos industriais sensorizados, criando oportunidades para otimizar processos de manutenção e evitar paradas não programadas. A manutenção preditiva, uma técnica que utiliza dados e algoritmos para prever falhas em equipamentos antes que elas ocorram, surgiu como uma resposta inovadora para os desafios de manutenção enfrentados pelas indústrias modernas. [1]

O desenvolvimento das ideias que envolvem manutenção preditiva é diretamente ligado ao avanço da ciência de dados e da tecnologia de sensores. A capacidade de coletar dados em tempo real de equipamentos sensíveis, juntamente com a análise avançada desses dados, revolucionou a forma como as indústrias abordam a manutenção. A abundância de dados agora disponível não apenas permite a identificação precoce de possíveis falhas, mas também oferece a possibilidade de detecção de padrões de desgaste e de como o equipamento está operando ao longo de sua vida útil. [2]

A utilização de algoritmos de aprendizado de máquina e técnicas de análise estatística permite às organizações antecipar falhas e implementar estratégias de manutenção de forma proativa e não mais reativa (manutenção realizada somente quando o equipamento apresentava a falha). Nesse contexto, a ciência de dados desempenha um papel fundamental, transformando dados brutos em informações acionáveis, proporcionando uma visão clara do estado de saúde dos equipamentos e agregando valor ao negócio para evitar perdas na produção e alavancando lucros, uma vez que essa é uma necessidade de mercado visto a competitividades entre as empresas atuais. [2]

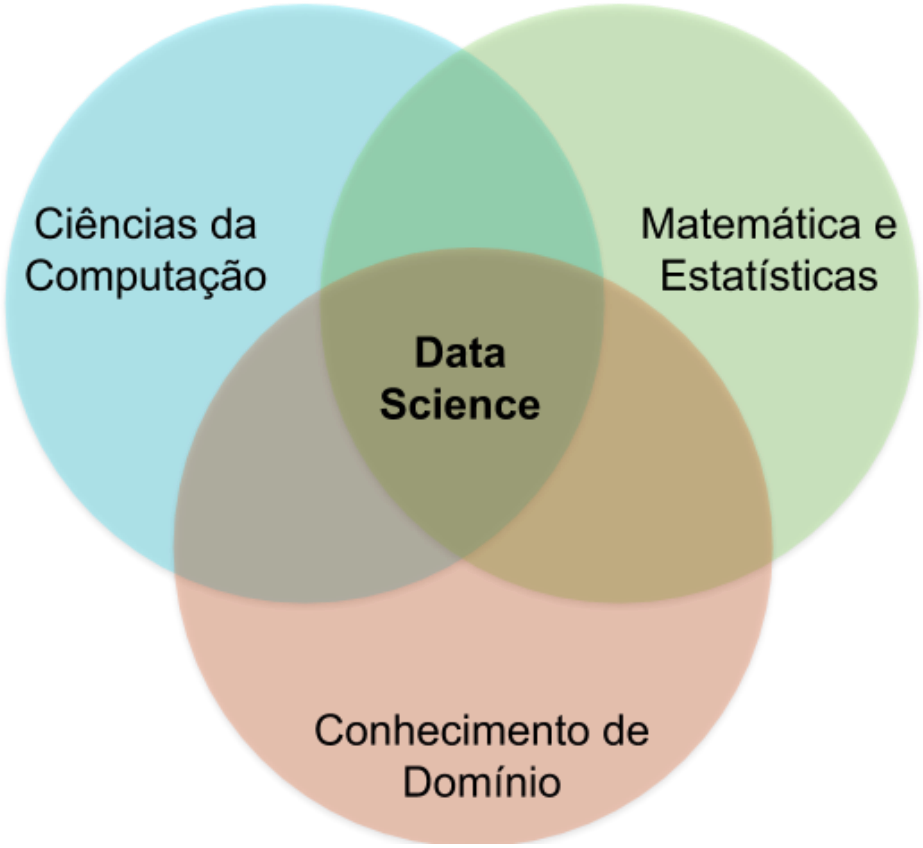
Este trabalho de graduação explora a intersecção entre a ciência de dados e a manutenção preditiva em um contexto industrial, realizando a previsão de falhas em 100 ativos industriais sensorizados com uma antecipação de 15 leituras dos sensores, as quais ocorrem duas vezes ao dia, destacando a importância crítica da análise de dados na prevenção de falhas e na maximização da eficiência operacional utilizando estratégias de manutenção inteligentes.

2 REFERENCIAL TEÓRICO

2.1 Ciência de dados

A Ciência de dados, de forma sumarizada, é a intersecção de três áreas: ciência da computação, matemática e estatística e conhecimento de domínio (conhecimento da área em que se deseja atuar) guiada pela necessidade de resolver problemas do mundo real. Envolve a aplicação de métodos científicos, algoritmos e sistemas para extrair conclusões e informações importantes a partir de dados obtidos por diversos meios. O cientista de dados emprega técnicas estatísticas avançadas, aprendizado de máquina e engenharia de dados para criar modelos preditivos, identificar padrões e tomar decisões que tragam valor ao negócio [3].

Figura 1 - Diagrama de Venn com as áreas que envolvem ciência de dados.



Fonte: [3].

A Ciência de Dados, como é conhecida hoje, surgiu como uma resposta à evolução tecnológica nas últimas décadas. Com o advento da internet e o aumento exponencial na geração de dados, as organizações perceberam a necessidade de métodos sistemáticos para transformar esses dados em informações úteis. Isso levou ao desenvolvimento de técnicas avançadas de análise de dados e ao surgimento da Ciência de Dados como uma área distinta [3].

Dado avanço tecnológico, a infinidade de dados disponíveis para utilização e a necessidade de usá-los de forma simultânea buscando encontrar padrões, foi de extrema importância o desenvolvimento de ferramentas que hoje são amplamente utilizadas para análise e modelagem de dados. O *Python*, uma linguagem de programação versátil e poderosa, é frequentemente preferido devido à sua rica coleção de bibliotecas para análise de dados, como *Pandas*, *NumPy* e *SciPy*. Além disso, o *Scikit-Learn* e o *TensorFlow* são *frameworks* populares para aprendizado de máquina e desenvolvimento de modelos de inteligência artificial. Essas ferramentas proporcionam aos cientistas de dados a capacidade de construir, treinar e avaliar modelos preditivos de forma eficaz e eficiente [3].

Atualmente, a Ciência de Dados é aplicada em uma variedade de setores e campos, desde finanças até medicina e manufatura. Nas indústrias, a análise preditiva é particularmente relevante. A manutenção preditiva, por exemplo, utiliza técnicas de Ciência de Dados para prever falhas em equipamentos antes que ocorram, permitindo a intervenção proativa e a redução de custos associados a paradas não programadas [1].

2.2 Aprendizado de Máquina

De forma didática um programa de computador aprende pela experiência E em relação a algum tipo de tarefa T e alguma medida de desempenho P se o seu desempenho em T, conforme medido por P, melhora com a experiência E. O aprendizado de máquina é um assunto que está extremamente em alta no mundo corporativo e acadêmico e está diretamente ligado ao ramo da inteligência artificial. De forma objetiva é um conjunto de técnicas que são, em sua maior parte, supervisionados e não supervisionados, que permitem aos sistemas computacionais melhorarem sua performance em uma tarefa específica à medida que são expostos a mais dados. Ao contrário dos sistemas tradicionais de programação, em que os computadores seguem instruções específicas para realizar uma tarefa, os algoritmos de aprendizado de máquina aprendem padrões nos dados possibilitando realizar previsões ou tomar decisões com base nesses padrões [4].

O livro "Mãos à Obra: Aprendizado de máquina com *Scikit-Learn* e *TensorFlow*" de Aurélien Géron [4] é uma fonte valiosa para entender os fundamentos do aprendizado de máquina. No contexto desta referência, o aprendizado de máquina não é apenas um conjunto de técnicas, mas uma revolução que está transformando nossa compreensão de como os sistemas podem aprender e melhorar conforme a experiência. Essa metodologia é especialmente útil em situações em que é difícil ou inviável programar regras específicas para resolver um problema complexo [4].

O aprendizado de máquina, portanto, oferece uma solução poderosa para resolver uma ampla gama de problemas do mundo real, desde reconhecimento de padrões até tomada de decisões, impulsionando avanços significativos e agregando valor para suas inúmeras aplicações.

2.2.1 Aprendizado Supervisionado

No aprendizado supervisionado os algoritmos são treinados usando dados rotulados, ou seja, dados em que as saídas desejadas já são conhecidas. Esse tipo de aprendizado é essencial para prever ou classificar dados futuros com base em padrões aprendidos a partir de dados passados [4].

O filtro de *Spam*, por exemplo, mostra uma aplicação de aprendizado supervisionado onde é preciso distinguir entre as classes da máquina (*e-mails spam* e *e-mails* não *spam*). Desta forma, o algoritmo é treinado com um conjunto de e-mails rotulados como *spam* ou não *spam*, analisa as características dos *e-mails*, como palavras-chave, remetente e estrutura, aprende os padrões associados a cada categoria e, uma vez treinado, o sistema pode classificar automaticamente novos e-mails como spam ou não spam com base nos padrões aprendidos. O modelo desenvolvido neste trabalho de graduação, por exemplo, o qual utiliza informações preditoras de sensores de ativos industriais já rotulados com as classes de falha e não falha, utiliza o aprendizado supervisionado [4].

Com um olhar mais matemático, podemos definir esse tipo de aprendizado como: para cada observação $x_i, i = 1, \dots, n$ existe um rótulo (ou classe) y_i associado [5].

Existem vários tipos de algoritmos de aprendizado supervisionado. Regressão Linear, por exemplo, é usado para prever valores contínuos, como preços ou temperaturas. Por outro lado, Algoritmos de Classificação, como Regressão Logística, são usados para classificar dados em categorias específicas, como sim/não ou *spam*/não *spam* e, por fim, algoritmos que possuem a capacidade de serem utilizados para ambos os aprendizados: K vizinhos mais próximos (KNN) que identificam padrões com base na proximidade dos dados no espaço e árvores de decisão que utilizam de cortes em variáveis “mais importantes” criando uma série de decisões para chegar a um resultado [4, 5].

Desta forma, aprendizado supervisionado é uma ferramenta que forma a base para muitas aplicações do mundo real, incluindo diagnósticos médicos, reconhecimento de voz e automação inteligente, transformando dados em soluções e impulsionando inovações em diversos setores.

2.2.1.1 Algoritmos de Regressão

Os algoritmos de regressão são uma categoria fundamental no aprendizado supervisionado, empregados para prever valores numéricos (quantitativos), por exemplo, custos de produção, valores de residências de acordo com sua localização e preço de produtos. Eles estabelecem uma relação matemática entre variáveis de entrada e saída, permitindo fazer previsões precisas para novos dados [4, 5].

2.2.1.1.1 Regressão Linear

A regressão linear é um algoritmo simples e muito utilizado em casos em que é possível encontrar uma relação linear entre as variáveis preditoras e a variável resposta. Em outras palavras, ele modela a relação linear entre uma variável de entrada (ou mais) e uma variável de saída. Por exemplo, é possível prever o preço de uma casa com base em sua localidade, utilizando a regressão linear para encontrar uma Equação que relaciona essas variáveis.

De forma simples, como mostra a Equação (1) se existe uma relação linear entre uma variável preditora X e a variável resposta Y, podemos dizer que:

$$Y \approx \beta_0 + \beta_1 X \tag{1}$$

A Equação (2) mostra o caso em que temos n variáveis preditoras:

$$Y \approx \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n \tag{2}$$

Na Equação (2), Y é a variável resposta, β_0 o coeficiente linear (intercepto) e β_n o coeficiente angular para cada variável preditora. Uma vez que os betas são determinados por meio do treinamento com os dados conhecidos, é possível determinar valores desconhecidos para Y apenas substituindo o X_n. [5]

Entretanto, é necessário encontrar os valores desses parâmetros beta e uma das formas mais comuns para sua determinação é o método dos mínimos quadrados. Para isso, faz-se necessário entender que $\hat{y}_i$ representa o valor predito e y_i o valor real da variável resposta para um determinado valor de X. Desta forma, define-se o conceito de resíduo como: $res = y_i - \hat{y}_i$. Assim, determina-se a soma do quadrado dos resíduos, como na Equação (3):

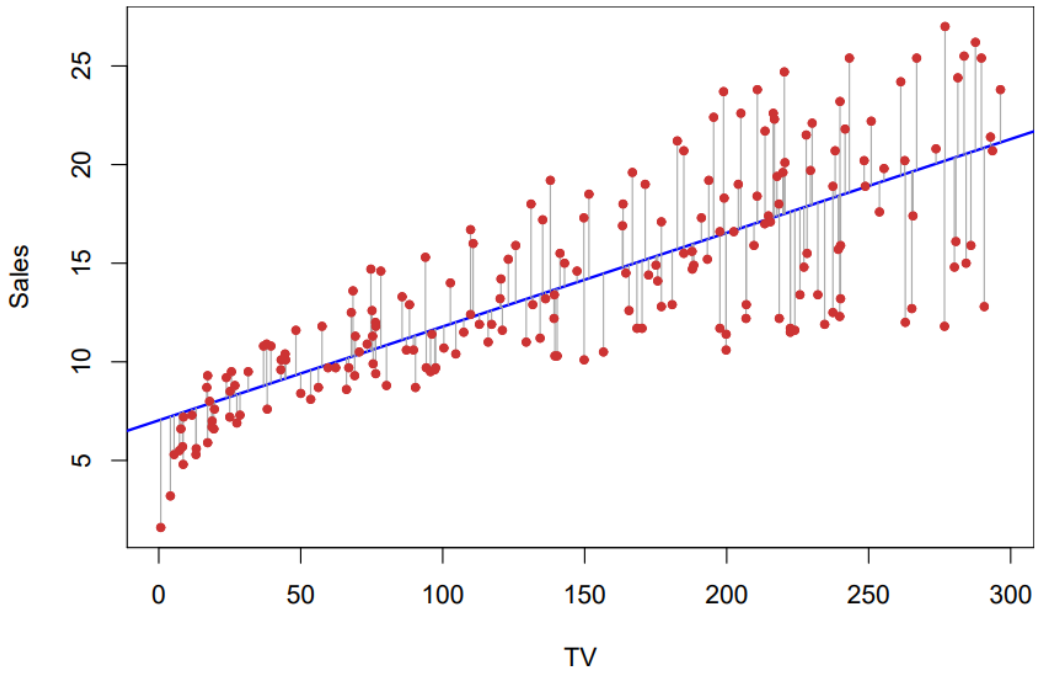
$$SQR = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3}$$

E, com isso, apenas reestruturando a Equação (3), encontra-se a Equação (4):

$$SQR = (y_i - \beta_0 - \beta_1 X_{1pred})^2 + \cdots + (y_i - \beta_0 - \beta_n X_{npred})^2 \tag{4}$$

O método dos mínimos quadrados então, encontra os valores de beta que reduzem ao máximo a soma do quadrado dos resíduos SQR. A Figura 2 mostra um exemplo de regressão linear aplicada à uma situação em que se deseja encontrar como a quantidade de propagandas na TV influencia nas vendas (*sales*) de um determinado produto [5].

Figura 2 – Regressão Linear aplicada a um problema de vendas.



Fonte: [5].

Uma vez definido os parâmetros betas usando o método dos mínimos quadrados é possível concluir, por exemplo, qual é a relação entre cada variável preditora com a variável resposta, uma vez que o valor de beta define o quanto a variável resposta irá variar a cada unidade acrescida na variável preditora. Assim é possível determinar variáveis mais ou menos importantes para o problema em questão [5].

2.2.1.1.2 K Vizinhos Mais Próximos (KNN) – Aplicação em Regressão

O algoritmo K Vizinhos Mais Próximos (*KNN – K Nearest Neighbors*) é utilizado tanto para problemas de regressão quanto para classificação, o que difere os dois tipos é somente a forma com que a previsão é realizada, ou seja, todos os passos realizados pelo algoritmo são os mesmos para ambos os casos. É um algoritmo baseado em instância, o que significa que ele não cria um modelo durante a fase de treinamento. Em vez disso, durante a fase de treinamento, o algoritmo armazena os exemplos de treinamento. Para fazer previsões ou classificações, o KNN encontra os

"k" (parâmetro definido pelo usuário) observações de treinamento mais próximos (vizinhos) do novo ponto de dados que precisa ser previsto ou classificado. Os passos de seu funcionamento são: [4]

1. Inicialmente o KNN utiliza uma métrica de distância, como a distância euclidiana por exemplo, para calcular a distância no espaço entre a observação a ser realizada, a previsão e todas as demais observações no treinamento.
2. Em seguida, o algoritmo identifica os "k" observações mais próximas ao novo ponto com base na métrica de distância escolhida.
3. Como neste tópico falamos sobre regressão, o KNN calcula a média dos "k" vizinhos mais próximos e atribui esse valor como a previsão para a nova observação.

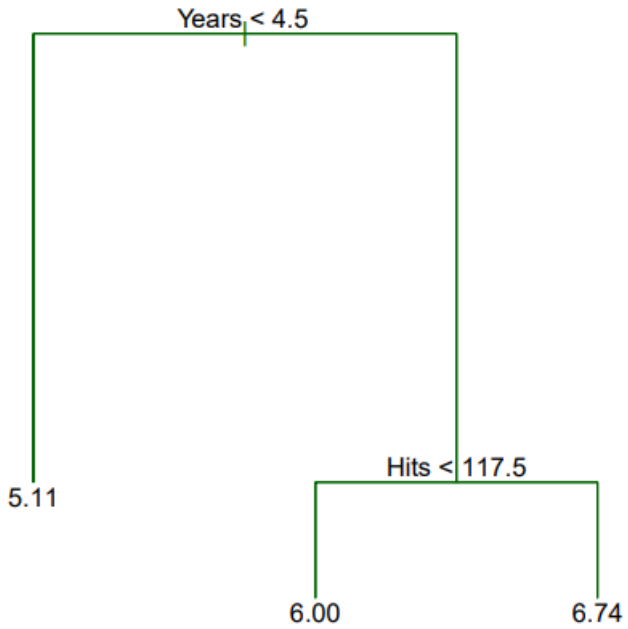
Um grande problema ao se utilizar o KNN é qual valor utilizar para o parâmetro "k". Um valor muito pequeno pode tornar a previsão sensível a outliers, enquanto um valor muito grande pode suavizar demais as previsões. Portanto, é importante experimentar diferentes valores de "k" para encontrar o melhor desempenho no conjunto de dados específico [4].

É importante mencionar também que o KNN pode ser computacionalmente custoso, especialmente em conjuntos de dados grandes, pois precisa calcular a distância entre o novo ponto e todos os pontos de dados de treinamento durante a previsão. Além disso, o desempenho do KNN pode ser sensível à escala das variáveis, portanto, normalizar ou padronizar os dados pode ser necessário para obter melhores resultados [4].

2.2.1.1.3 Árvores de Decisão – Aplicação em Regressão

Este algoritmo pode ser usado tanto para regressão quanto para classificação e só difere quanto à função de custo e ao passo final para realizar as previsões. Basicamente, as árvores de decisão realizam as previsões baseadas em um conjunto de regras, onde a cada nível da árvore acontece uma partição dos dados por meio de uma decisão (partição recursiva). O valor da decisão é escolhido para otimizar algum tipo de função de custo, como por exemplo o erro quadrático médio, que calcula a média do quadrado dos resíduos de cada folha. A Figura 3 mostra uma árvore de decisão simples que busca realizar a previsão do log do salário de um jogador de *baseball* baseado em anos de experiência e total de batidas [5].

Figura 3 - Árvore de Decisão Simples.



Fonte:[5].

Passos para a criação de uma árvore de decisão [4]:

1. A árvore de decisão escolhe uma característica do conjunto de dados que melhor separa os dados em subgrupos mais puros. Para dados de regressão, a característica escolhida é aquela que minimiza a variância das variáveis resposta dentro dos subgrupos. A ideia é dividir os dados de tal forma que as variáveis resposta dentro de cada subgrupo sejam as mais homogêneas possíveis.
2. Para cada característica escolhida, a árvore de decisão determina um valor de separação que divide os dados em dois subgrupos. [Por exemplo, se a característica escolhida é os anos de experiência do jogador de *baseball* é 4,5], os dados seriam divididos em dois grupos: um com anos de experiência menor que 4,5 e outro com anos de experiência maior ou igual que 4.5. Desta forma, podemos afirmar que se trata de um algoritmo binário, uma vez que sempre divide os subgrupos em outros dois subgrupos.
3. A árvore de decisão utiliza diferentes critérios para medir a impureza dos subgrupos, sendo que o critério mais comum para problemas de regressão é o Erro Quadrático Médio (MSE - *Mean Squared Error*). O MSE calcula a média dos quadrados das diferenças entre os valores alvo reais e as previsões para cada subgrupo. A divisão que minimiza o MSE é escolhida como a melhor divisão em cada nó da árvore. Por essa característica chamamos as árvores de decisão de algoritmo “guloso”.

4. O processo é repetido para cada subgrupo gerado. A árvore de decisão continua dividindo os subgrupos em subgrupos menores até atender a um critério de parada, como a profundidade máxima da árvore ou o número mínimo de amostras em um nó, este processo é chamado de partição recursiva.
5. Para realizar a previsão, no caso de problemas de regressão, o algoritmo realiza a média das observações presentes no subgrupo (folha) final e atribui esse valor ao dado desconhecido.

Por fim, pode-se dizer que as árvores de decisão apresentam alguns prós ao serem utilizadas devido ao seu padrão de funcionamento. O primeiro deles é a escolha das melhores características para fazer a divisão, o que "naturalmente" já se configura como uma seleção de variáveis (será abordado com mais profundidade na seção 2.5.4); é um algoritmo pouco sensível aos *outliers* devido aos cortes realizados nos subgrupos serem ortogonais; possuem fácil interpretação pois com as bibliotecas disponíveis para o *Python*, por exemplo, é possível plotar uma imagem de árvore de regressão como a da Figura 5 [4].

Entretanto, um grande contra das árvores de decisão é que são necessários critérios de parada, pois há uma grande probabilidade de sobre ajustar aos dados (modelo não generalizar o suficiente, ter ótimo desempenho para conjunto de treino e péssimo desempenho para conjunto de teste) [5].

2.2.1.2 Algoritmos de Classificação

Os algoritmos de classificação buscam realizar previsões de classes, podendo ser binárias ou multiclasses. São utilizados para os mais diversos fins, entre eles: filtros de spam, previsões de clientes devedores e em processamentos de imagens médicas [4].

2.2.1.2.1 Regressão Logística

A regressão logística, por mais que tenha "regressão" no título para previsão de variável categórica binária (duas classes) com base em uma ou mais variáveis independentes [6], pode ser utilizada em situações como: prever se um e-mail é *spam* ou não e se um paciente tem uma doença específica ou não. Trata-se de um algoritmo paramétrico, ou seja, existe uma função que relaciona as variáveis independentes com a variável resposta, por meio de parâmetros β [6]. Em resumo, é a aplicação de um modelo linear para problemas de classificação e tem como saída uma probabilidade de pertencimento à classe desejada. O problema da regressão logística consiste em linearizar um problema de classificação por meio de uma função chamada "*logit*", estimar uma

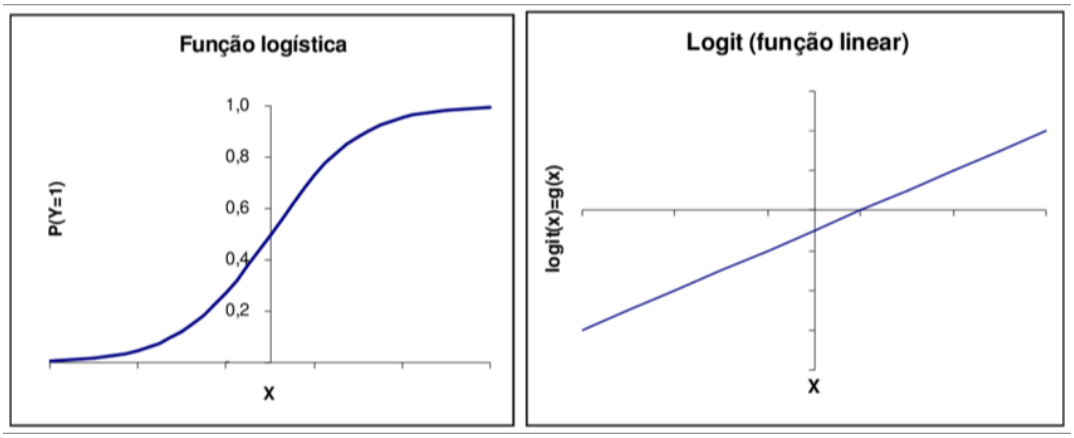
função linear que melhor se encaixa no comportamento das variáveis preditoras e utiliza uma função sigmoide para retornar ao domínio do modelo de classificação [6]. Os passos para realizar uma regressão logística são [6]:

1. Realizar a transformação do eixo y da distribuição dos dados utilizando a função *logit*, mostrada na Equação (5).

$$logit(P) = \ln\left(\frac{P}{1-P}\right) \tag{5}$$

Em que P é a probabilidade de que a variável resposta seja 1 ($P(Y = 1)$).
Ao utilizar essa função, observações com $P(Y = 1)$ são levadas ao $+\infty$ enquanto às observações com $P(Y = 0)$ são levadas ao $-\infty$. A Figura 4 mostra um exemplo de transformação do eixo y e, vale ressaltar, que tal relação só é linear com relação aos parâmetros β, entretanto entre as possibilidades não é linear.

Figura 4 - Utilização da função *Logit*.



Fonte: [7].

2. A melhor reta é a que maximiza a função de verossimilhança por métodos numéricos como o gradiente descendente.
3. Com a função linear e seus parâmetros definidos utiliza a Equação 6, a função sigmoide, para retornar o eixo y para o intervalo de [0, 1].

$$f(x) = \frac{1}{1+e^{-x}} \tag{6}$$

É vantajoso utilizar a regressão logística: É possível melhor interpretabilidade da importância das variáveis com o valor dos βs da função, temos probabilidades calibradas na saída do modelo, que são simples de implementar e rápidas para realizar novas inferências. Como

desvantagens observa-se que problemas sem possibilidade de serem linearizados são difíceis de se resolverem e as variáveis preditoras não podem se correlacionar entre si (devem ser independentes, uma não deve explicar a outra) [6].

2.2.1.2.2 K Vizinhos Mais Próximos (KNN) – Aplicação em Classificação

Como já citado no tópico 1.2.1.1.2 – K Vizinhos Mais Próximos (KNN) – Aplicação em Regressão, o KNN também é um algoritmo que pode ser aplicado para prever variáveis qualitativas, ou seja, que representam uma classe ou estado. O funcionamento deste algoritmo é idêntico ao descrito no tópico de regressão, exceto pelo modo que são realizadas as previsões.

Uma vez definido o valor K de vizinhos mais próximos e calculadas todas as distâncias entre a observação a qual se quer prever a saída e as observações instanciadas no treino, é realizada o voto majoritário da classe mais presente entre todos os K vizinhos, ou seja, em um exemplo hipotético no qual existem 10 vizinhos sendo, 7 iguais à 1 e 3 iguais a 0, o valor a ser previsto na nova observação será 1. Caso ocorra empate entre as classes, o algoritmo soma o total de distância da nova observação para todas as observações rotuladas como 1 e todas rotuladas como 0 e classifica a nova instância com a classe em que a soma da distância for menor [4, 5].

2.2.1.2.3 Árvores de Decisão – Aplicação em Classificação

Assim como já comentado no tópico 1.2.1.1.3 – Árvores de Decisão – Aplicação em Regressão, esse algoritmo também pode ser utilizado em problemas de classificação. O que difere as duas aplicações é o momento em que o algoritmo busca medir a impureza nas folhas utilizando uma função de custo. No caso da classificação, as medidas utilizadas são a impureza de Gini e o método da Entropia. O método de Gini calcula a impureza de uma folha, sendo igual a 0 para nós puros (que todas as observações possuem a mesma variável resposta). O método da entropia, também chamado de Ganho de Informação, mede o ganho de informação em cada folha, em outras palavras, se todas as instâncias em uma folha forem da classe 1, não há informação alguma a ser medida, então o Ganho de Informação é 0 [4].

Por fim, para definição da classe da observação a ser prevista, o algoritmo realiza o voto majoritário (entende-se por moda na estatística) entre todas as observações presentes na última folha da árvore. Sendo assim, a classe que tiver maior número de elementos na folha será a classe com a qual a nova observação será classificada [4].

2.2.1.3 Algoritmos *Ensemble* – Agrupamento de Modelos

Um algoritmo *Ensemble* combina inúmeros blocos de algoritmos de forma a utilizá-los em conjunto para obtenção de um único resultado, que é mais assertivo e fora de qualquer risco associado aos algoritmos quando usados isoladamente [5].

Os *ensembles* utilizam de um conceito chamado de *sabedoria das multidões.* Para entender esse conceito, imagine uma pesquisa de campo sobre determinado assunto e, para isso, será comparada a resposta de um único especialista com a combinação de 1000 respostas de pessoas aleatórias. Na grande maioria dos casos a resposta agregada das 1000 pessoas aleatórias será mais assertiva do que a do especialista [4].

Isso por conta do fato de que se existem n observações independentes em uma amostra, cada uma com variância σ^2, então encontra-se $\bar{z}$, que é a variância da média, conforme a Equação (7):

$$Var_{avg} = \frac{\sigma^2}{n} \tag{7}$$

Obtendo-se a média das observações, diminui-se a variância, resultando em uma previsão melhor. [5]

Assim, uma forma de diminuir a variância é definir diferentes conjuntos de treinamento da população, para cada um desses conjuntos, ter um modelo independente e, em seguida, encontrar a previsão final pela média dos resultados. Em outras palavras, calcula-se previsões separadas para B conjuntos de treinamento distintos e obtém-se um único modelo estatístico de aprendizado com menor variação ao tirar a média dessas previsões [5].

É possível, então, definir modelos ($\widehat{f_l}$(x)), para B diferentes tipos de conjuntos de treino e encontrar a média dos resultados de forma a ter um único resultado com uma menor variância frente aos modelos individuais, como mostra a Equação (8) [5]:

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x) \tag{8}$$

Existem duas formas mais comuns de se utilizar o ensemble: *Bagging* e *Boosting*. Os algoritmos *Bagging* agregam os modelos de forma paralela, enquanto os de *Boosting* fazem a agregação em série.

2.2.1.3.1 *Bagging* e Florestas Aleatórias

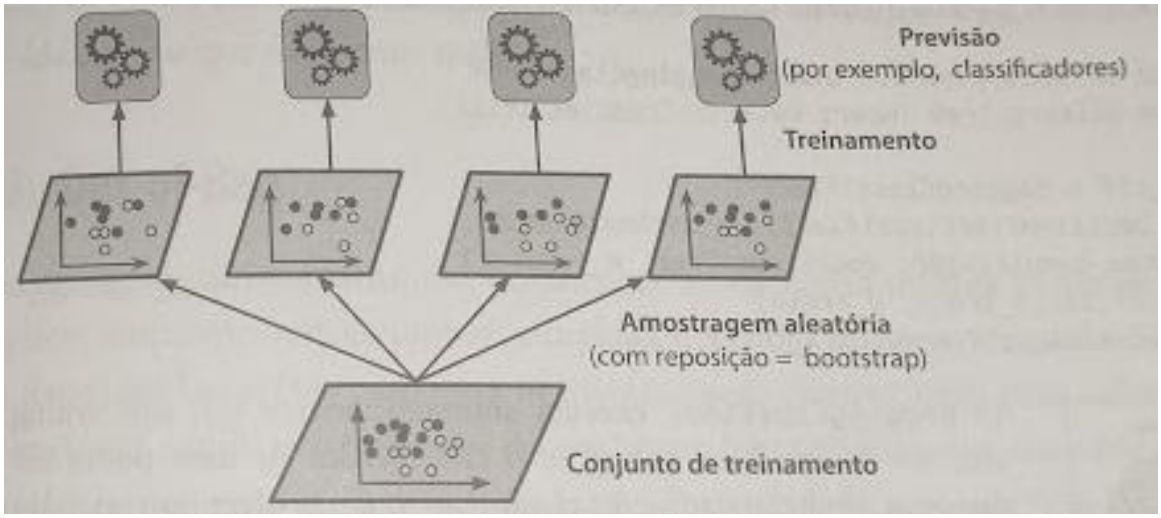
Árvores de decisão possuem um problema crônico: alta variância, em outras palavras, se forem utilizadas duas amostras aleatórias das observações de treino muito provavelmente as duas terão árvores de decisão com resultados diferentes. Desta forma, *Bagging* ou *Bootstrap Aggregation* foram criadas para reduzir a variância do resultado de modelos estatísticos [5].

O algoritmo funciona da seguinte forma [4]:

1. Inicia-se definindo qual tipo de algoritmo será usado como base, as mais utilizadas são as árvores de decisão por justamente sofrerem mais com os problemas de alta variância, possuírem um treinamento simples e muitos hiper parâmetros para ajustes;
2. Para cada árvore a ser treinada, realiza-se uma amostra utilizando o método de *Bootstrap*, ou seja, uma amostra aleatória com reposição;
3. O treinamento de todas as árvores de decisão é realizado, lembrando que o resultado de cada árvore é independente uma da outra pela arquitetura do *bagging* ser em paralelo;
4. A previsão final do algoritmo é:
    a. Para Regressão: Média das previsões finais de cada árvore;
    b. Para Classificação: Moda das previsões finais de cada árvore.

A Figura 5 mostra todo o processo de treinamento de um algoritmo de *bagging*.

Figura 5 - Treinamento de um algoritmo de *Bagging*.



Fonte: [4].

As Florestas Aleatórias (*Random Forests*), são uma extensão dos algoritmos de *Bagging*, entretanto, com uma aleatoriedade adicional além da amostragem de *bootstrap*. Ao se realizar a construção de cada árvore, ao invés de considerar todo o conjunto de variáveis preditoras, a

Floresta Aleatória faz uma amostra aleatória das variáveis preditoras, resultando em mais aleatoriedade entre as árvores reduzindo ainda mais a variância [4].

2.2.1.3.2 Boosting

Seguindo o objetivo de melhorar a precisão dos modelos estatísticos, ao contrário do *bagging*, onde tem-se um grande número do mesmo algoritmo treinados com uma amostra aleatória com reposição dos dados, o *boosting* treina uma série de algoritmos simples (modelos fracos) organizados em série, onde cada modelo tenta corrigir os erros cometidos pelo modelo anterior. O modelo final é uma combinação ponderada desses modelos fracos, onde os modelos que têm melhor desempenho em dados mal classificados recebem maior peso [4].

Árvores de decisão também são muito utilizadas como modelos fracos nos algoritmos de Boosting. Para isso utilizam de árvores simples, com pouca profundidade e poucas quebras. O algoritmo funciona da seguinte forma [4]:

1. Um modelo fraco é treinado com o conjunto de dados original;
2. Cada observação é ponderada, dando mais peso às observações que são classificadas incorretamente, dessa forma, o próximo modelo fraco terá mais atenção com essa observação;
3. Após um novo modelo fraco ser treinado com as observações anteriores ponderadas, os pesos são novamente atualizados, mas agora com base nas previsões do treino atual;
4. Por fim, a previsão final é a combinação ponderada da saída de cada modelo fraco envolvido no algoritmo.

Pode-se dizer que, da mesma forma que o *Bagging*, o *Boosting* realiza uma melhora incremental no modelo utilizando a ponderação para classificar melhor as observações mais difíceis, melhor precisão em previsões e redução da variância pelo motivo de focar nos erros dos treinamentos anteriores. Existem inúmeros algoritmos variantes do *Boosting*, entre eles: *AdaBoost* e *GradientBoost* que são os mais famosos. A ideia é basicamente a mesma, entretanto no *GradientBoost* a árvore subsequente visa otimizar os resíduos da árvore anterior e não as observações ponderadas em si [4].

2.2.2 Aprendizado não supervisionado

O aprendizado não supervisionado é uma abordagem de aprendizado de máquina em que o algoritmo é treinado em dados não rotulados, ou seja, dados que não possuem rótulos ou categorias predefinidas. Em vez de prever uma saída como no aprendizado supervisionado, o objetivo do aprendizado não supervisionado é explorar a estrutura ou padrões intrínsecos nos dados. Esta é uma técnica crucial em análise de dados, mineração de dados e diversas outras aplicações, em que muitas vezes os padrões não são conhecidos no início [5].

Com isso, é possível encontrar padrões que podem não ser evidentes à primeira vista; segmentar clientes em um estudo de mercado com base em seu comportamento; reconhecimento de padrões em imagens e áudio e criação de sistema de recomendação que pode definir um grupo de clientes com um gosto em comum, por exemplo [5].

É necessário pontuar que para o treinamento de um modelo não supervisionado é necessário ter um bom pré-processamento de dados, o que será discutido posteriormente.

Existe atualmente uma vasta gama de algoritmos não supervisionados, os mais comuns são os de clusterização, que buscam encontrar padrões entre as observações e agrupá-las de acordo com suas semelhanças.

2.2.2.1 K Means Clustering

O algoritmo *K Means* tem como objetivo agrupar os dados em K grupos desde que não haja sobreposições, ou seja, uma observação não pode pertencer a dois grupos ao mesmo tempo, e toda observação deve estar presente em pelo menos um *cluster* [5].

O K *Means*, funciona da seguinte forma [5]:

1. Inicialmente, o algoritmo seleciona aleatoriamente K observações como centros iniciais de cada *cluster*, chamado de centroide.
2. Cada observação é atribuída ao cluster cujo centroide é mais próximo. A proximidade é geralmente medida pela distância euclidiana, mas outras métricas de distância também podem ser usadas.
3. Uma vez que todas as observações são atribuídas aos clusters, os centroides dos clusters são recalculados como a média dos pontos atribuídos a cada *cluster*.
4. Os passos de atribuição de *cluster* e atualização de centroides são repetidos iterativamente até que não haja mudanças significativas na atribuição dos pontos aos clusters ou um número máximo de iterações seja atingido. Essa mudança é medida

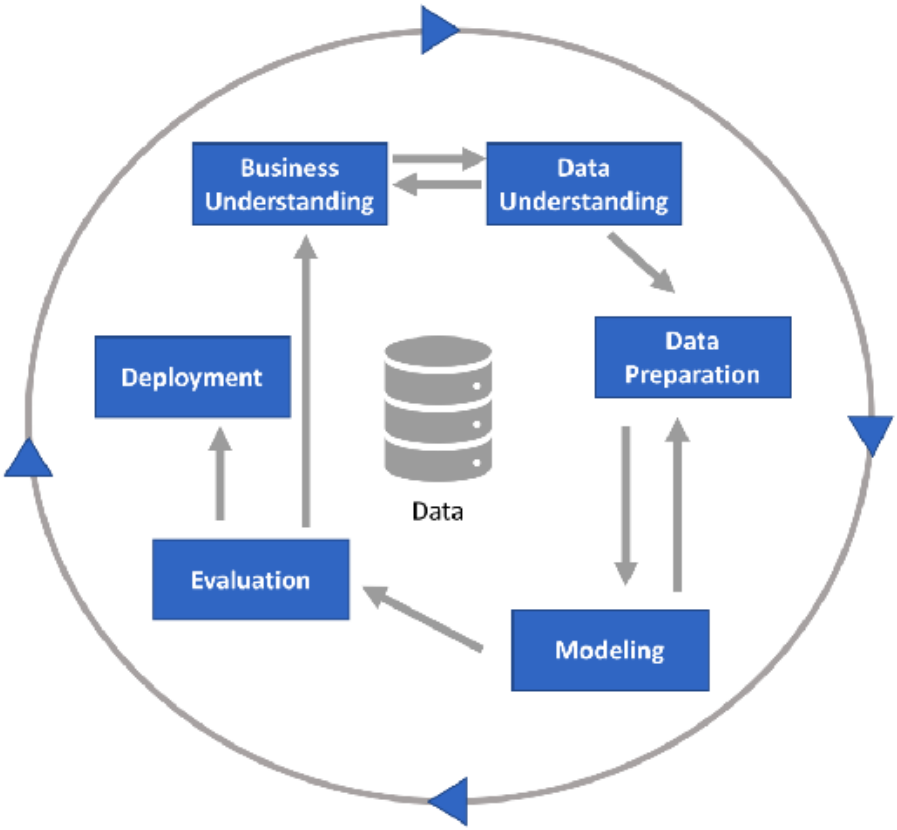
utilizando o cálculo da variância, em que se busca a menor variância possível dentro de cada cluster.

Para utilização do K *Means* é necessário atentar-se que dependendo de como é inicializado, ou seja, a posição dos centroides iniciais, pode interferir no resultado. Além disso, o número de clusters também pode influenciar, por isso deve-se escolher um valor de K adequado, utilizando o método do joelho, por exemplo [5].

2.3 CRISP-DM – Projetos em ciência de dados

O CRISP-DM (*Cross-Industry Standard Process for Data Mining*) [16] é uma metodologia padrão utilizada como base para projetos de ciência de dados. Ele fornece uma estrutura sistemática e abrangente com os passos essenciais para guiar os profissionais de dados durante o processo de descoberta, análise, tratamento e modelagem de dados. O CRISP-DM é composto por seis fases inter-relacionadas, como mostra a Figura 6 [16].

Figura 6 - CRISP-DM e suas fases.



Fonte: [16].

De forma sucinta as 6 fases podem ser descritas como [16]:

1. Entendimento de Negócio (*Business Understanding*): Uma fase extremamente importante, é onde o profissional de dados busca entender qual o problema que irá trabalhar. O que deve ser solucionado, quais as tentativas já realizadas pelo time de negócio, quais bases de dados e/ou variáveis que o time de negócio julga ser importantes para o desenvolvimento do projeto, alinhar expectativas, ganhos e soluções para o projeto final;
2. Entendimento dos Dados (*Data Understanding*): Fase em que o profissional de dados começa a buscar todas as possíveis variáveis que podem ser explicativas relacionadas ao problema em questão. Nessa fase, todo conhecimento estatístico do profissional de dados deve ser posto em prática, pois as conclusões sobre as relações entre as variáveis começam a ficar claras por meio de padrões, anomalias e tendências;
3. Preparação dos Dados (*Data Preparation*): É nesse momento em que os dados são transformados e tratados para ficarem suficientemente prontos para servirem de entrada nos modelos de aprendizado de máquina. Remove-se outliers (se necessário), preenchem-se os valores ausentes, encontra-se valores inconsistentes, modifica os tipos de variáveis, criam-se outras variáveis, etc;
4. Modelagem (*Modeling*): Fase em que a base de treino serve de entrada para o modelo de aprendizado de máquina. Define-se qual modelo será usado conforme o tipo de problema (classificação, regressão, clusterização) e treina-se o modelo buscando os melhores resultados possíveis seguindo as métricas de avaliação correspondentes e adequadas;
5. Avaliação (*Evaluation*): Fase em que são avaliados os resultados do modelo seguindo os critérios definidos com o time de negócio na fase de entendimento de negócio. É uma fase crítica, pois deve-se decidir se os resultados atendem aos objetivos do negócio. Se não, pode ser necessário voltar a uma fase anterior para revisar e refinar o processo;
6. Implantação (*Deployment*): Implementar o modelo em ambiente produtivo em que pode ser utilizado pela equipe de negócio sempre que necessário e utilizando os dados mais atualizados.

Destaca-se então a importância em se utilizar o CRISP-DM em projetos de dados, pois oferece uma estrutura clara e organizada para esses tipos de projetos, garantindo que todas as etapas essenciais sejam abordadas. Sua flexibilidade permite se adaptar a diferentes tipos de problemas e conjuntos de dados e a possibilidade de interação entre as fases permite que aconteçam ajustes à medida que novas necessidades são definidas ou problemas são identificados. Além disso

a metodologia imprime um forte foco nos objetivos do negócio, garantindo que a análise de dados seja relevante e valiosa para a organização.

2.4 Estatística descritiva

A estatística desempenha um papel fundamental em projetos de ciência de dados, sendo a base sobre a qual são construídos modelos e inferências significativas, sendo assim é fundamental em várias etapas do processo de análise de dados e em diversas situações em projetos de ciência de dados. Permite uma compreensão profunda dos dados, validação de modelos e interpretação confiável de resultados. Utilizar técnicas estatísticas de maneira eficaz não só aumenta a precisão dos modelos, mas também proporciona uma compreensão mais profunda dos fenômenos estudados, sendo fundamental para o sucesso do projeto em si [6].

2.4.1.1 Medidas de Localização

As medidas de localização ou tendência central, como a média, mediana e moda, são fundamentais em projetos de ciência de dados por representarem o ponto central em torno do qual os dados se agrupam. Elas oferecem uma compreensão concisa e quantitativa da distribuição dos dados, permitindo identificar padrões, comparar conjuntos de dados e realizar análises precisas. [6] Além disso, essas medidas ajudam a interpretar resultados, tomar decisões baseadas em dados e detectar outliers, garantindo a validade e a confiabilidade das análises [6].

2.4.1.2 Média

Essa é a medida de tendência central mais básica. De forma simples, é a soma de todas as observações dividida pelo número total de observações, como mostra a Equação (9) [6]:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{9}$$

Note que na Equação (8) o número total de observações é representado por n minúsculo. Isso se dá pois estamos nos referindo à uma amostra [6].

A média em si, é uma boa estimativa de localização e, dependendo dos dados, representa muito bem todo o conjunto. Entretanto, deve-se citar que é uma medida muito sensível à valores

extremos (outliers), portanto, deve-se sempre tomar cuidado ao utilizar a média como representação geral dos dados.

2.4.1.3 Mediana

Diferente da média, a mediana representa o valor exato que divide um conjunto de dados ordenado em duas partes iguais: metade dos valores são menores e metade são maiores, caso o número de observações da amostra seja par, a mediana torna-se a média entre os dois valores centrais da distribuição dos dados. Isso a torna robusta à valores extremos, sendo especialmente útil em conjuntos de dados com distribuições assimétricas ou outliers. Ela não apenas fornece uma visão mais realista sobre os dados, mas também é uma ferramenta poderosa em análises comparativas, ajudando a identificar disparidades e tendências em diferentes grupos de dados [6].

2.4.1.4 Quartis e Percentis

Percentis são uma maneira útil de entender a distribuição dos dados, especialmente quando se lida com grandes conjuntos de dados. São valores que dividem um conjunto de dados ordenado em 100 partes iguais, representando a posição relativa de um determinado valor em relação aos outros valores no conjunto de dados. Por exemplo, o 25º percentil (ou primeiro quartil) divide os dados em que 25% dos valores são menores que ele, enquanto o 75º percentil (ou terceiro quartil) divide os dados em que 75% dos valores são menores que ele [6].

2.4.1.5 Outliers

Outliers são as observações com valores extremos que podem, ou não, enviesar os resultados. É muito complicado dizer que um outlier é algo ruim dentro de ciência de dados, pois dependendo do projeto são justamente os outliers que se busca, são os chamados modelos de detecção de anomalias. Não existe uma regra clara sobre como os outliers são definidos em um estudo, mas existe uma convenção que define um valor extremo usando os conceitos de quartil [6].

O outlier é definido como todo valor que for maior que o limite superior (LS) e menor que o limite inferior (LI), sendo:

$$LS = Q_3 + 1.5 * (Q_3 - Q_1)$$

$$LI = Q_1 - 1{,}5 * (Q_3 - Q_1)$$

Sendo que Q_3 *e* Q_1 são os quartis e $Q_3 - Q_1$ também é chamado de Intervalo Interquartil ou IQR.

2.4.1.6 Moda

A moda representa o valor ou valores mais frequentemente observados em um conjunto de dados. Com essa medida de localização é possível descobrir tendências dominantes e comportamentos comuns, auxiliando em tomadas de decisão para qualquer problema [9].

Além disso, a moda também é fundamental na detecção de padrões anômalos. Quando um valor tem uma frequência significativamente maior do que outros, pode indicar uma mudança no fenômeno que está sendo estudado [9].

2.4.2 Medidas de Variabilidade

As medidas de variabilidade, também conhecidas como medidas de dispersão, são indicadores estatísticos que revelam o quanto os dados em um conjunto estão espalhados ou dispersos. Elas oferecem *insights* valiosos sobre a consistência dos dados e a sua homogeneidade [6].

As principais medidas de variabilidade que medem a dispersão dos dados em torno da média são: variância e desvio padrão.

2.4.2.1 Desvios

Desvios são a diferença entre os dados que são observados e a estimativa de localização (média, mediana, moda, etc.), como mostra a Equação (10) [6]:

$$desvio = x_i - \overline{x} \tag{10}$$

2.4.2.2 Variância

Entende-se por variância a soma da média do quadrado dos desvios, como mostra a Equação (11) [6, 9]:

$$var(x) = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n} \tag{11}$$

2.4.2.3 Desvio Padrão

O desvio padrão, assim como a variância, é uma medida de dispersão que mostra o quanto os dados estão dispersos em torno do valor central, no nosso caso a média, como mostra a Equação 12 [6, 9]:

$$std = \sqrt{var}(\mathrm{x}) \tag{12}$$

2.5 Preparação dos dados

O pré-processamento de dados é um passo de extrema importância para qualquer projeto na área de dados, pois é através dele que análises e conclusões importantes para o fenômeno estudado são dadas de forma correta. De forma sucinta, são um conjunto de técnicas utilizadas para transformar dados brutos em um formato adequado para análises e modelagem [4]. Serão detalhadas nos próximos tópicos as principais técnicas de preparação de dados.

2.5.1 Tratamento de Valores Faltantes

Devido à falhas na obtenção dos dados ou até mesmo inexistência deles, algumas informações podem vir sem valor algum e a maioria dos algoritmos de aprendizado de máquina requer que estes valores faltantes sejam preenchidos de alguma forma. Para isso, existem 3 principais estratégias [4]:

1. Excluir a observação que possui o dado faltante. Essa estratégia só deve ser realizada caso a exclusão da observação toda não interfira nas previsões do algoritmo. Em casos em que existe uma concentração considerável de valores faltantes, essa estratégia não é recomendada [4];
2. Excluir toda a variável. Da mesma forma que a estratégia anterior, deve-se ter certeza que a variável a ser excluída possui pouco poder preditivo para o algoritmo a ser usado e se estiver com uma grande concentração de valores nulos [4];
3. Preencher os valores faltantes utilizando estatísticas básicas. Ao encontrar um valor faltante, é interessante observar o todo, caso os valores presentes na variável tenham uma baixa variância, é interessante substituir os valores faltantes pela média de todos os valores da característica. Caso existam outliers, pode-se utilizar a mediana, por exemplo, por ser mais robusta a esse tipo de dado. Em último caso, pode-se

simplesmente substituir o valor faltante por zero, pois existem algoritmos, como os de árvore, que conseguem lidar bem com esse tipo de solução [4].

Na estratégia 3 deve-se tomar cuidado com o vazamento de dados. É necessário dividir a base de dados em treino e teste, para que seja aplicada a estratégia de preenchimento de dados faltantes no treino e a mesma estratégia ser aplicada nos dados de teste. Isso é necessário pois caso a estratégia seja realizada para a base de dados total, os dados de teste terão os dados preenchidos com uma estratégia que utilizou os dados de treino, enviesando o resultado [4].

2.5.2 Tratamento de variáveis categóricas

Variáveis categóricas são aquele tipo de variável obtidas no formato de *string,* representando uma classe ou categoria. Isso significa que, a princípio, o dado não está em formato numérico, sendo este formato o que os algoritmos de aprendizado de máquina preferem trabalhar. Portanto, é necessário converter cada classe/categoria em um número inteiro que a represente. Para este fim, tem-se algumas estratégias [4]:

1. *One Hot Enconding*: O OHE cria uma nova variável binária (coluna) para cada classe existente em uma variável. Excelente estratégia para uma coluna de tamanho de roupas, por exemplo, onde temos: P, M, G e GG. Serão criadas 4 novas colunas binárias, uma para cada tamanho. Nota-se que essa estratégia não é viável em casos em que existem 10 categorias diferentes, por exemplo, pois aumentaria o espaço dimensional do modelo em si [4].
2. *Label Encoding*: Cada classe/categoria é atribuída à um número. No exemplo dos tamanhos de roupas: P = 0, M = 1, G = 2 e GG = 3. O problema nesse tipo de tratamento é que se existe um tipo ordenação nos dados, isso não é considerado.
3. *Ordinal Encoding*: Justamente para solucionar o problema do *Label Encoding*. Neste caso o valor numérico é atribuído para cada classe, mas respeitando uma ordem preestabelecida [4].

2.5.3 Escalonamento de variáveis

Para muitos algoritmos de aprendizado de máquina, principalmente os baseados em distâncias, é necessário que todas as variáveis estejam na mesma escala, para não haver

sobreposição de importância em variáveis de escalas muito grandes. As duas estratégias mais utilizadas são [4]:

1. Escalonador Mínimo – Máximo (Normalização): Os valores das variáveis são redimensionados para variar no intervalo de 0 a 1. Isso é factível devido à Equação (13) [4]:

$$y = \frac{x - x_{minimo}}{x_{máximo} - x_{mínimo}} \tag{13}$$

2. Padronização: Ajusta os valores dos dados para valores formando uma distribuição normal padrão, ou seja, média 0 e variância unitária. Assim como mostra a Equação (14), entretanto, caso os dados não sejam normalmente distribuídos antes da padronização, eles também não serão após a transformação [4].

$$y = \frac{x - \mu}{\sigma} \tag{14}$$

É necessário dizer que a padronização, por conta da variância unitária, não vincula valores em um intervalo em específico, entretanto é menos sensível à outliers. Assim como em todas as transformações, é necessário realizar o escalonamento no conjunto de treino e depois no conjunto de teste para que não haver vazamento de dados [4].

2.5.4 Seleção de Variáveis

Selecionar variáveis é extremamente importante para reduzir a dimensionalidade do modelo e, assim, melhorar a precisão dos modelos, reduzir o *overfitting*, diminuir o tempo de treinamento e facilitar a interpretação dos resultados. Quando se fala de seleção de variáveis, três abordagens principais se destacam: *Filter, Embedded e Wrapper* [10, 11].

1. O método *filter* utiliza métricas estatísticas para avaliar a relação entre cada variável e a variável de saída. As variáveis são classificadas com base nesses critérios e um subconjunto é selecionado. Este método é rápido e eficiente para conjuntos de dados grandes, mas não leva em consideração interações entre variáveis [10];
2. Métodos *Embedded* incorporam a seleção de variáveis diretamente no processo de treinamento do modelo. Algoritmos como árvores de decisão, e regressões lineares incluem mecanismos internos para atribuir pesos às variáveis durante o treinamento. Variáveis com baixos pesos ou importâncias são automaticamente consideradas

menos importantes, eliminando a necessidade de uma etapa separada de seleção de variáveis [10];

3. O método *Wrapper* envolve a avaliação de diferentes subconjuntos de variáveis. Algoritmos de aprendizado de máquina são treinados usando diferentes combinações de variáveis, e a performance de cada subconjunto é avaliada usando uma métrica como a precisão ou o erro médio. Existem várias técnicas de *Wrapper*, incluindo a *Forward Selection* (adição iterativa de variáveis), *Backward Elimination* (remoção iterativa de variáveis) e *Recursive Feature Elimination* (eliminação recursiva de variáveis). Embora seja computacionalmente intensivo, o método *Wrapper* leva em conta interações entre variáveis, resultando em seleções mais precisas [10].

2.6 Visualização de Dados

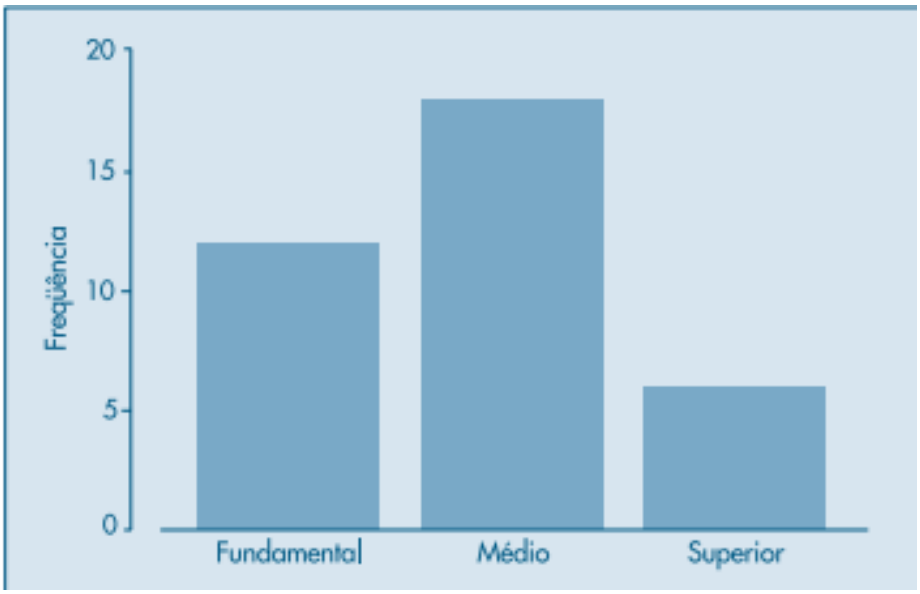
A visualização de dados é utilizada para transformar os dados brutos em representações gráficas claras e compreensíveis. Ela desempenha um papel fundamental em diversas áreas, pois serve para auxílio na avaliação de riscos e tomada de decisão. É importante pois serve para simplificar informações abstratas, tornando-as fáceis de interpretar por qualquer que seja o público. As representações gráficas não apenas tornam os dados mais atraentes, mas também permitem uma análise intuitiva [4].

Pode-se dizer que gráficos bem estruturados e informativos tornam-se mais memoráveis e impactantes, facilitando a comunicação eficaz em apresentações com o time demandante do projeto. Em suma, a visualização de dados conta histórias, guia a tomada de decisões, e auxilia as organizações a agirem com confiança em qualquer projeto.

2.6.1 Gráficos de Barras

Esse tipo de representação envolve a criação de retângulos ou barras, em que uma dimensão da barra está relacionada à magnitude a ser indicada, enquanto a outra é arbitrariamente escolhida, mas permanece constante para todas as barras. Essas barras são organizadas de maneira paralela, seja na horizontal ou na vertical. A Figura 7 mostra um gráfico de barras verticais [9].

Figura 7 - Gráfico de barras verticais.



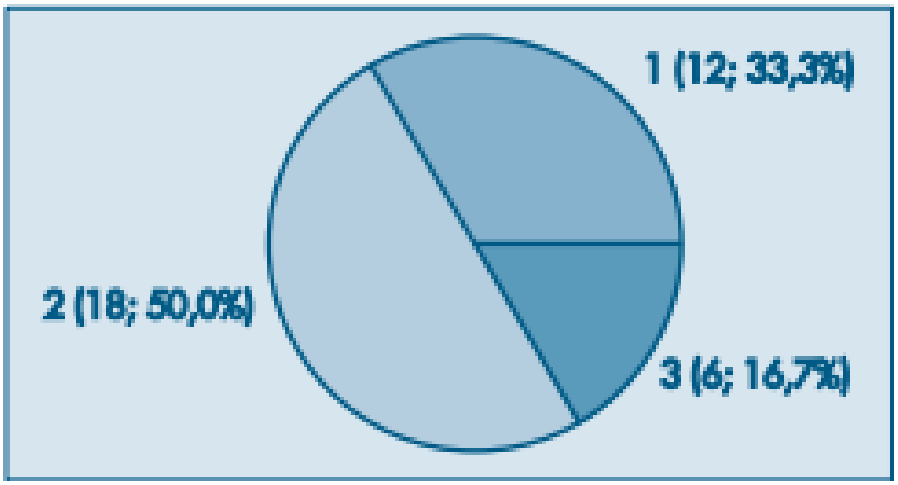
Fonte: [9].

O gráfico da Figura 7 mostra a frequência com que as categorias “Fundamental”, “Médio” e “Superior” aparecem.

2.6.2 Gráficos de Setor

O gráfico de setor, também conhecido como gráfico de pizza, é uma representação visual usada para mostrar a composição proporcional de um todo. Ele consiste em um círculo dividido em setores, onde cada setor representa a porcentagem ou a fração de uma categoria em relação ao total. É importante evitar o uso excessivo de setores para evitar distorções na interpretação dos dados e poluição da imagem. A Figura 8 mostra um gráfico de setor.[9]

Figura 8 - Gráfico de Setor.

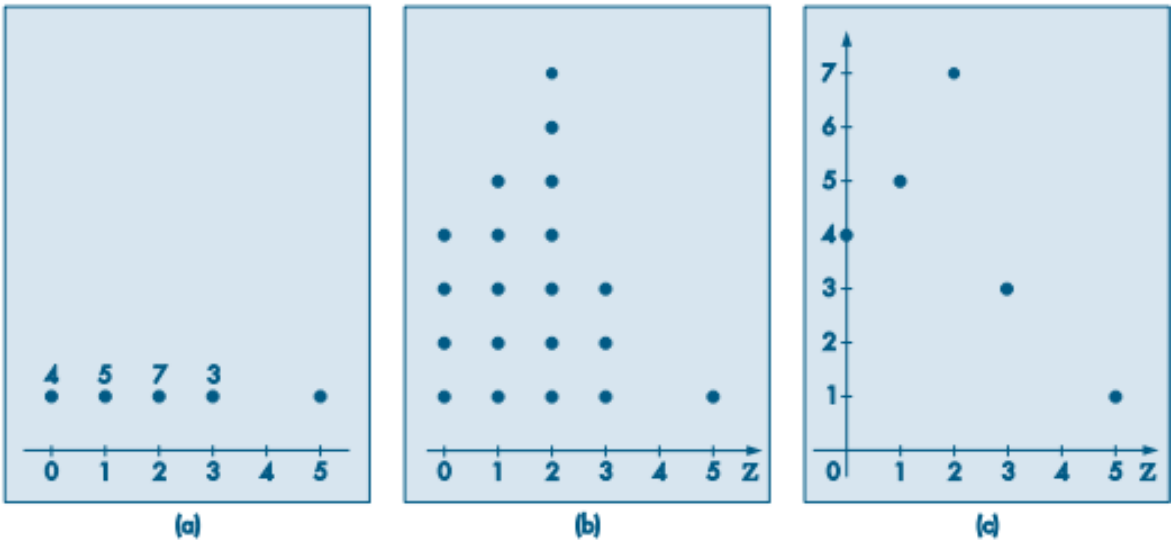


Fonte: [9].

2.6.3 Gráficos de Dispersão

O gráfico de dispersão é uma ferramenta visual utilizada para representar a relação entre duas variáveis em um conjunto de dados. Ele é composto por pontos posicionados em um plano cartesiano, em que cada ponto representa um par de valores das variáveis em questão. Ao observar a dispersão dos pontos no gráfico, é possível identificar correlações (positivas ou negativas). Além disso, permite identificar outliers e padrões não lineares. A Figura 9 mostra três possibilidades para os gráficos de dispersão utilizando no eixo x o número de filhos de algumas famílias pesquisadas e no eixo y a quantidade de famílias que possuem aquele determinado número de filhos [9].

Figura 9 - Gráficos de Dispersão.



Fonte: [9].

Observa-se na Figura 9 (a) e (c), que ambos os gráficos descrevem uma relação (linear) entre as duas variáveis em questão, de forma negativa na Figura 9 (a) e positiva na Figura 9 (c). Na Figura 9 (c) não é possível ver nenhuma associação entre as variáveis x e y.

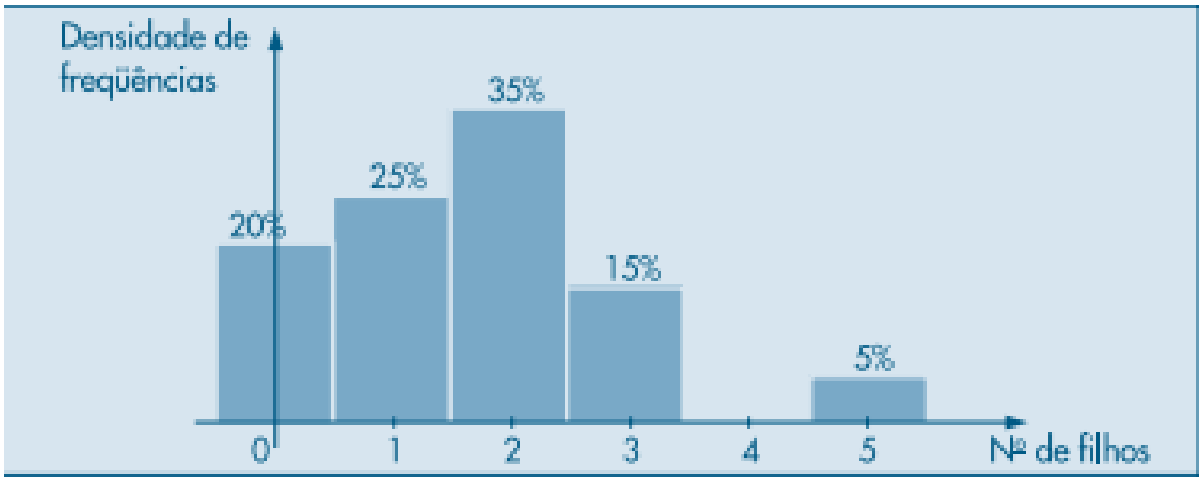
2.6.4 Histogramas

O histograma mostra a distribuição de uma variável quantitativa. Pode ser até confundido com um gráfico de barras ao olhar a primeira vista, mas o histograma não utiliza barras separadas, e sim retângulos adjacentes para mostrar a frequência ou a densidade das observações em diferentes intervalos ao longo do eixo x [6].

Cada retângulo representa a quantidade de dados em uma determinada faixa, enquanto a altura do retângulo indica a frequência ou densidade dos dados nesse intervalo. Histogramas são particularmente úteis para identificar distribuições simétricas ou assimétricas, bem como para detectar outliers e compreender a variabilidade dos dados [9].

A Figura 10 mostra um histograma para as mesmas variáveis de número de filhos e porcentagem das famílias com o número de filhos em questão.

Figura 10 – Histograma.



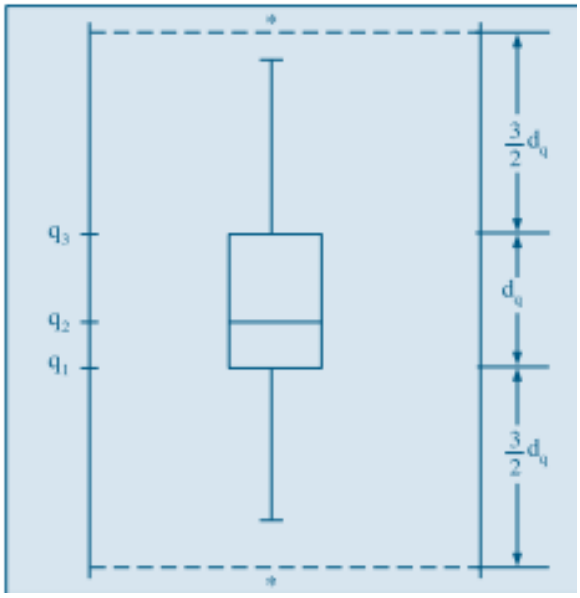
Fonte: [9].

2.6.5 Box plot

Outra excelente representação gráfica é o *box plot*. É uma maneira eficiente de demonstrar a distribuição dos dados, pois basicamente mostra como está a concentração de dados em cada quartil. [6]

A Figura 11 mostra um box plot genérico.

Figura 11 - Box Plot.



Fonte: [9]

Para a construção de um box plot é necessário [9]:

1. Cálculo dos quartis (Q_3 *e* Q_1), e mediana (Q_2);
2. O quartil 1 é a aresta mais baixa do retângulo, a mediana é a linha entre as duas extremidades do retângulo e o quartil 3 é a aresta mais alta;
3. As linhas prolongas após as extremidades da caixa são chamadas de *whiskers* [6] e a parte final delas são os limites inferior e superior.;
4. Os limites inferior e superior são calculados conforme já foi mostrado no tópico de *Outliers* no capítulo de medidas de localização.

2.6.6 Correlação

Quando se trata de projetos de dados, é sempre importante observar as relações entre as variáveis que se está utilizando. Tanto entre as variáveis preditoras entre si, quanto das variáveis preditoras com a variável resposta. Estarem positivamente correlacionadas significa que a variável Y cresce conforme X cresce, e negativamente se Y decresce conforme X cresce [6].

Uma matriz de correlação é uma tabela em que tanto nas linhas quanto nas colunas as variáveis do projeto estão dispostas e, para cada intersecção (célula da tabela), há o valor da correlação entre as duas variáveis [6].

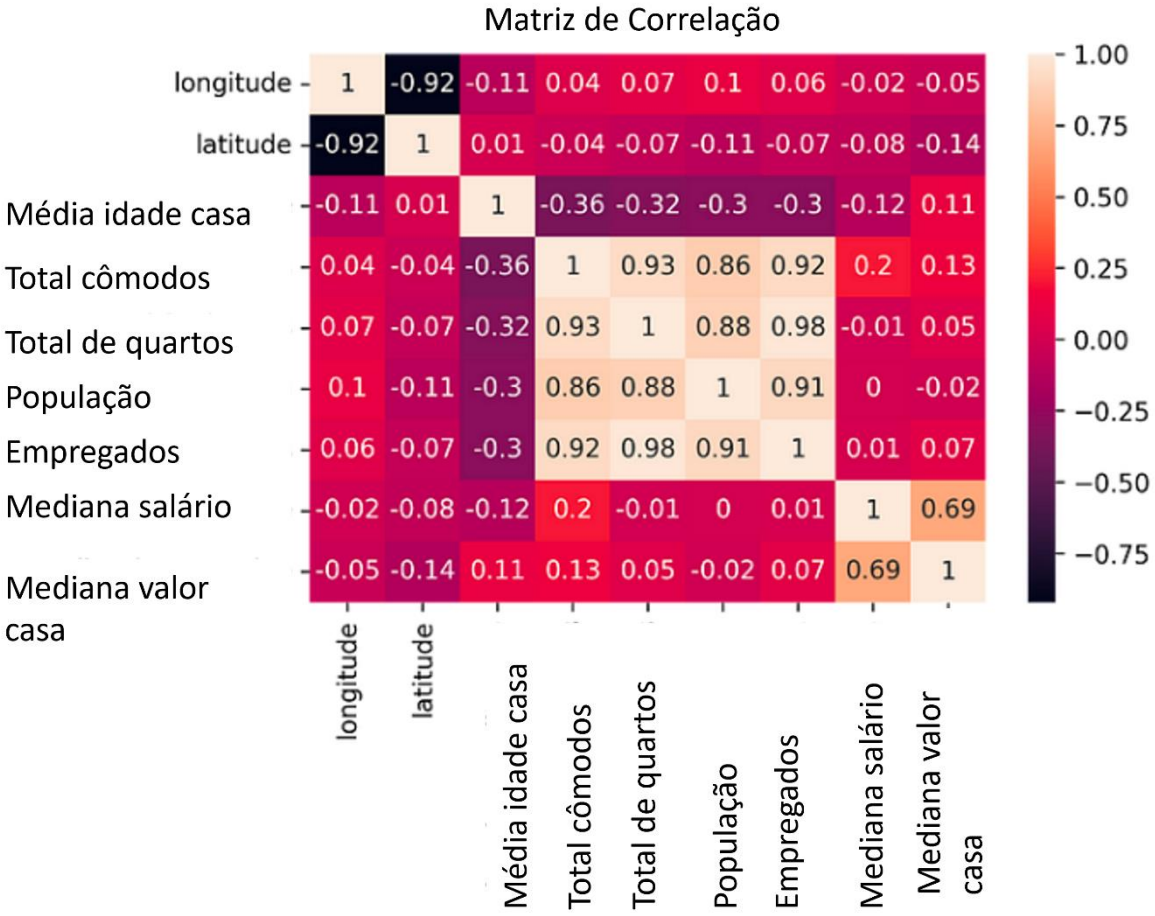
Esse valor de correlação é dado pelo coeficiente de correlação de *Pearson* e entrega uma estimativa de correlação sempre na mesma escala, no intervalo [-1, 1]. O coeficiente de correlação de *Pearson* pode ser calculado simplesmente como a multiplicação dos desvios da média da

variável 1 pelos da variável 2 e, divididos pelo produto do desvio-padrão [6], como mostra a Equação (15):

$$r = \frac{\sum_{i=1}^{n}\left[\left(x^{(i)} - \mu_x\right)\left(y^{(i)} - \mu_y\right)\right]}{\sqrt{\sum_{i=1}^{n}\left(x^{(i)} - \mu_x\right)^2}\sqrt{\sum_{i=1}^{n}\left(y^{(i)} - \mu_y\right)^2}} = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{15}$$

Na Figura 12, mostra uma matriz de correlação plotada em forma de mapa de calor, para facilitar a interpretação das correlações de acordo com as cores [11].

Figura 12 - Matriz de Correlação.



Fonte: Adaptado de [11].

2.7 Métricas de Avaliação de Modelos

As métricas de avaliação de modelos quantificam a precisão, robustez e eficácia dos modelos, permitindo aos cientistas de dados e pesquisadores avaliar quão bem seus modelos estão performando em dados de teste. Essas métricas ajudam na escolha do melhor modelo para um

problema específico, oferecendo *insights* sobre sua capacidade de generalização para novos dados, além de auxiliar na identificação de possíveis problemas como *overfitting* ou *underfitting*.[4] Ao utilizar métricas adequadas, como precisão, recall, F1-score ou área sob a curva ROC, é possível decidir sobre ajustes no modelo ou escolha de abordagens alternativas, melhorando assim a qualidade e confiabilidade das previsões e classificações [4, 5].

2.7.1 Validação cruzada

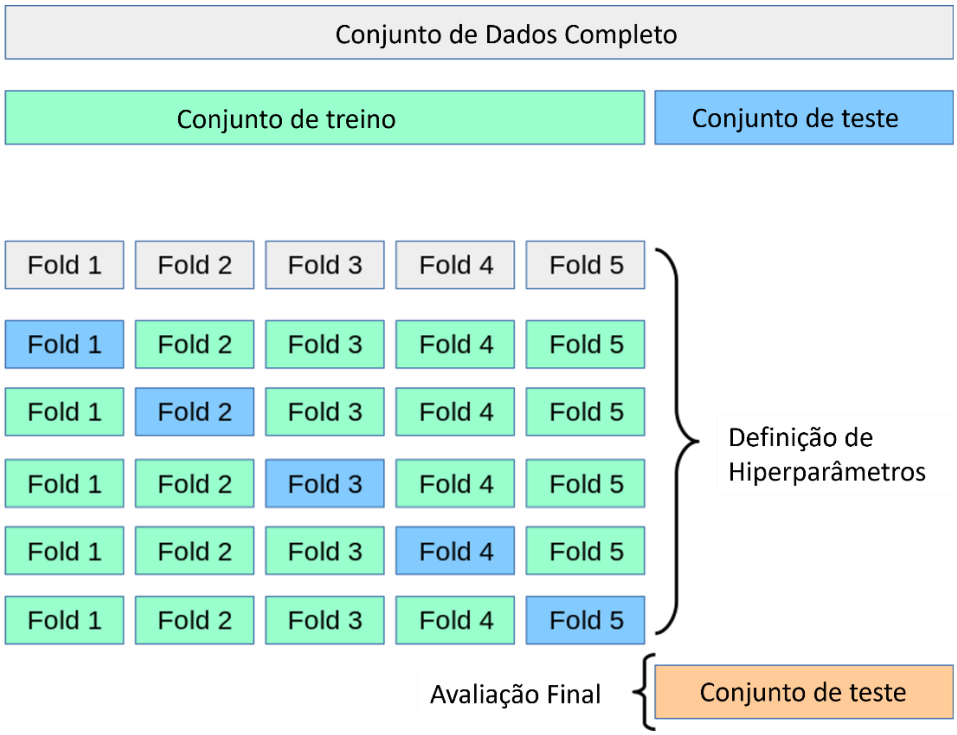
A melhor forma de avaliar modelos é dividir o conjunto de dados em duas partes: uma para treino e outra para teste. É interessante ter um conjunto de teste pois é possível avaliar se o modelo terá um bom desempenho em dados que ele nunca observou [5, 4].

Algo que necessita de atenção especial é quando se tem duas opções de modelagem para seguir. O caminho é óbvio: treinar as duas possibilidades e comparar a performance (usando as métricas adequadas que serão discutidas posteriormente) no conjunto de teste. Uma vez definido qual algoritmo será utilizado, é necessário selecionar os melhores hiperparâmetros e então, mais uma vez, serão treinados inúmeros modelos com todas as combinações de hiperparâmetros possíveis até que se encontre o melhor dentre todos comparando o conjunto de teste. Após colocar o modelo em produção percebe-se um desempenho aquém do esperado [4].

Acontece que o modelo foi validado inúmeras vezes para o mesmo conjunto de validação, fazendo com que todas as melhorias realizadas ao longo dos treinamentos fossem feitas sob a ótica desse conjunto de dados, o que reduz a possibilidade de generalização do modelo [4].

Para solucionar isso, a validação cruzada divide o conjunto de treinamento em subcobjuntos e todas as possibilidades de treino são feitas com cada combinação possível de subconjuntos de dados e validado com os subconjuntos restantes. Um algoritmo de validação cruzada muito utilizado é o *K-Fold Cross Validation*, mostrado na Figura 13 [5, 12].

Figura 13 - Validação Cruzada.



Fonte: Adaptado de [12].

O *K-Fold Cross Validation* é uma técnica que divide o conjunto de dados em K partes iguais, chamadas "*folds*". O modelo é treinado K vezes, cada vez usando K-1 *folds* como dados de treino e 1 *fold* como dados de validação. Esse processo é repetido K vezes, garantindo que cada *fold* seja usado como conjunto de validação exatamente uma vez. Ao final, são calculadas métricas de desempenho médias a partir das K iterações, oferecendo uma estimativa mais robusta do desempenho do modelo [5].

2.7.2 Avaliação em modelos de regressão

Uma vez que se tem um modelo de regressão treinado, é necessário avaliá-lo utilizando um conjunto de teste. Sabe-se que a variável resposta dos modelos de regressão é quantitativa, ou seja, um valor numérico, desta forma como medir se o modelo está realizando as previsões de forma mais generalizada possível? Isso não significa acertar tudo, mas ter erros baixos e aceitáveis para dados desconhecidos.

A qualidade dos modelos de regressão é dada por duas métricas: Erros padrão residual (RSE) e o coeficiente de determinação (R^2). A Equação (16) mostra como o RSE é calculado [5]:

$$RSE = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{16}$$

Essa métrica mede a precisão geral dos modelos de regressão e com ela é possível realizar a comparação com outros tipos de modelos [6]. Em outras palavras, é a média do valor com que as previsões irão divergir da linha de regressão definida. [5]

Para definir se o resultado de RSE é bom ou ruim, será necessário avaliar com qual problema está se lidando. Um RSE que pareça ser baixo e bom para um tipo de problema, pode ser um valor inaceitável para outros.

O R^2 por sua vez, também chamado de coeficiente de determinação, é uma métrica que varia de 0 a 1 e mede a proporção de variação dos dados no modelo, ou seja, o quão bem os dados se ajustam ao modelo definido. [5, 6] O R^2 é definido como mostra a Equação (17).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{17}$$

O denominador do segundo termo da Equação (15) calcula a variância da variável resposta y, enquanto o numerador determina a variabilidade que fica sem explicação após a definição do modelo [5, 6].

Como resultado, quanto mais próximo de 1 tem-se um modelo "perfeito", que explica toda a variabilidade dos dados.

2.7.3 Avaliação em modelos de classificação

Assim como em regressão, os modelos de classificação, os quais utilizam como variável resposta, dados categóricos (classes, dados qualitativos, etc.) também necessitam de formas para avaliar sua performance e para comparar os algoritmos entre si, buscando maior precisão nas previsões realizadas [6].

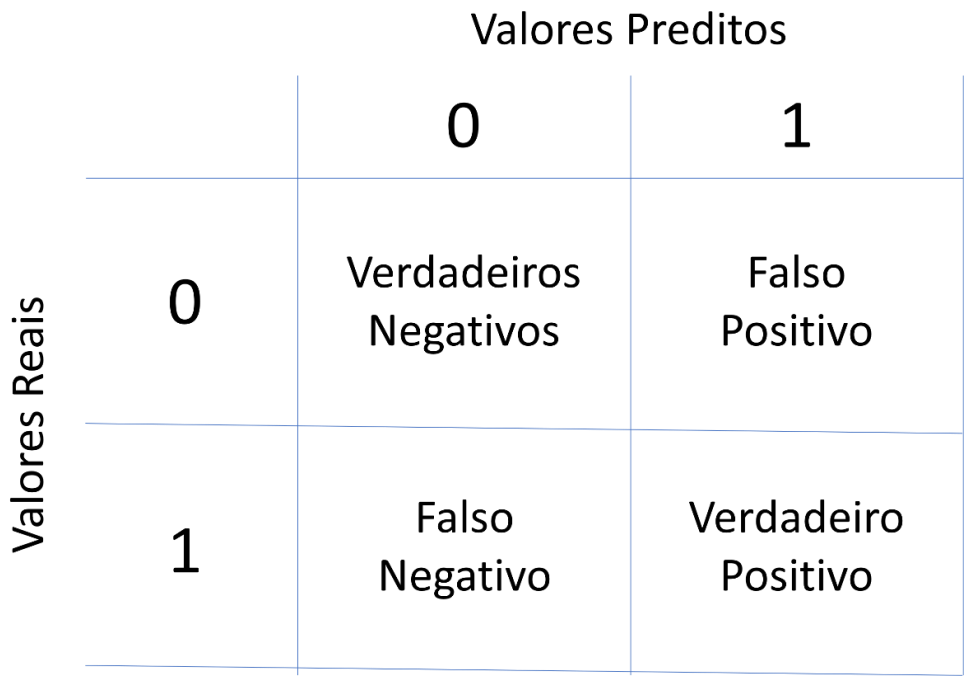
Para definir todas as métricas de avaliação relacionadas aos modelos de classificação primeiro é necessário entender o que é a Matriz de Confusão.

2.7.3.1 Matriz de Confusão

A matriz de confusão é a base para todas as métricas de avaliação que serão discutidas posteriormente nesta seção. É uma tabela que mostra quais previsões são corretas e incorretas, mas categorizadas pelo agrupamento do valor real e valor previsto [6].

Considerando um modelo de classificação binário, em que os resultados são somente 0 para não pertencimento à uma classe e 1 para pertencimento. A matriz de confusão se dá como mostra a Figura 14.

Figura 14 - Matriz de Confusão.



Fonte: Elaborado pelo autor.

Na Figura 14, as colunas são valores preditos e as linhas os valores reais. Temos então, que na diagonal principal dessa matriz, temos os valores que foram previstos corretamente pelo modelo [5].

Então, através da matriz de confusão são obtidos quatro resultados [5]:

1. Verdadeiros Negativos: É a quantidade de observações não pertencentes à classe de interesse e que o modelo acertou a previsão;
2. Falsos Positivos: São as observações que o modelo previu pertencimento à classe de interesse, entretanto as observações não pertenciam à essa classe;
3. Falsos Negativos: Observações em que o modelo realiza a previsão de não pertencimento à classe, mas na realidade as observações eram consideradas pertencentes;
4. Verdadeiros Positivos: Observações pertencentes à classe de interesse e o modelo acertou a previsão.

Entretanto, deve-se atentar ao tipo de problema de classificação trabalhado. Existem problemas como detecção de fraudes, diagnósticos médicos raros e defeitos em produtos onde a classe 1 é minoritária (extremamente baixo), e o modelo acaba enviesando pendendo para o lado

da variável majoritária. Dessa forma, caso o modelo classifique a maioria dos dados de teste com a classe negativa, terá uma alta taxa de acerto, mas não por ser um bom modelo, mas somente por ter maior parte dos dados de treino classificados como 0. Isso pode ser facilmente observado por uma métrica chamada Sensibilidade [5].

2.7.3.2 Sensibilidade

A sensibilidade é a proporção de acertos para todas as observações que de fato pertencem à classe positiva. Em um problema de detecção de fraudes, pode-se dizer que a sensibilidade é: de todas as observações do conjunto de teste que de fato eram fraudes, quantas o modelo classificou corretamente. A Equação (18) mostra como é calculada a sensibilidade [5].

$$Sensibilidade = \frac{VP}{FN+VP} \tag{18}$$

2.7.3.3 Precisão

A precisão identifica a proporção de acertos para todas as previsões realizadas para a classe positiva. A Equação (19) mostra como é calculada a precisão [5].

$$Precisão = \frac{VP}{FP+VP} \tag{19}$$

2.7.3.4 Especificidade

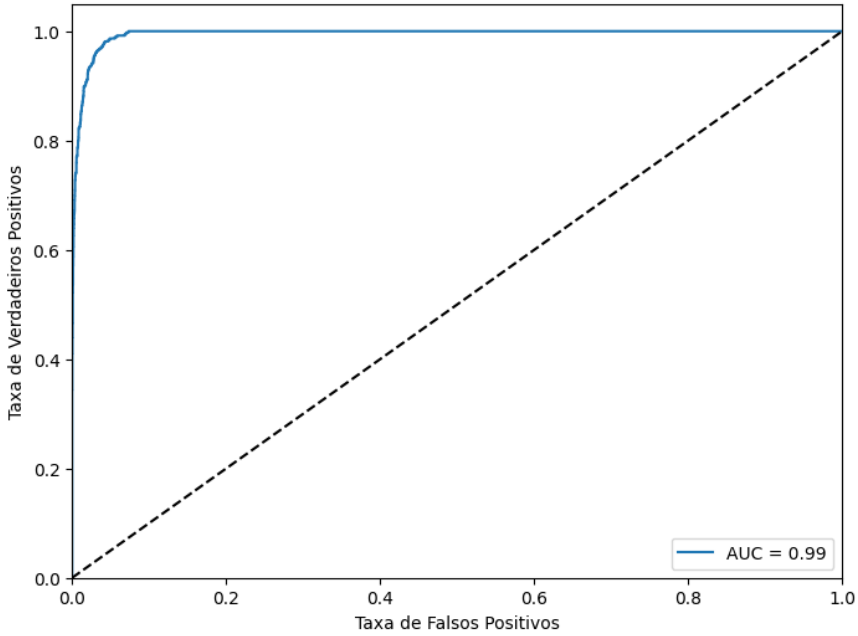
É o contrário da sensibilidade, ou seja, a proporção de acertos do modelo frente todas as observações que realmente pertencem à classe negativa. A Equação (20) define a especificidade [5].

$$Especificidade = \frac{VN}{VN+FP} \tag{20}$$

2.7.3.5 Curva ROC e Área sob a Curva ROC

Tendo em vista que Sensibilidade e Especificidade são métricas opostas, fica claro que existe uma troca entre elas. A Curva ROC é uma ferramenta que utiliza as duas métricas em questão para mostrar de forma gráfica a relação entre elas. A Figura 15 mostra uma curva ROC [5].

Figura 15 - Curva ROC.



Fonte: Elaborado pelo autor

Na curva ROC o eixo x representa a taxa de falsos positivos, ou seja, 1 – especificidade, enquanto no eixo y apresenta a sensibilidade (taxa de verdadeiros positivos). A especificidade também pode ser usada ao invés de "1 – sensibilidade" o que mudaria seria a posição da curva. A linha pontilhada mostra somente o que seria um modelo classificador de chance aleatória [5].

De forma simplificada, a curva ROC é traçada considerando diferentes possibilidades de limiares. Para cada limiar os pares ordenados das grandezas dos eixos x e y são plotadas para formar a curva [5].

A área sob a curva ROC é a métrica almejada. Quanto maior a área, significa que o modelo tem maior separação entre as classes.

2.8 Manutenção Preditiva

A manutenção preditiva, uma das novidades na área da indústria moderna, é conhecida por utilizar dados e técnicas analíticas para prever quando uma máquina ou equipamento pode falhar.

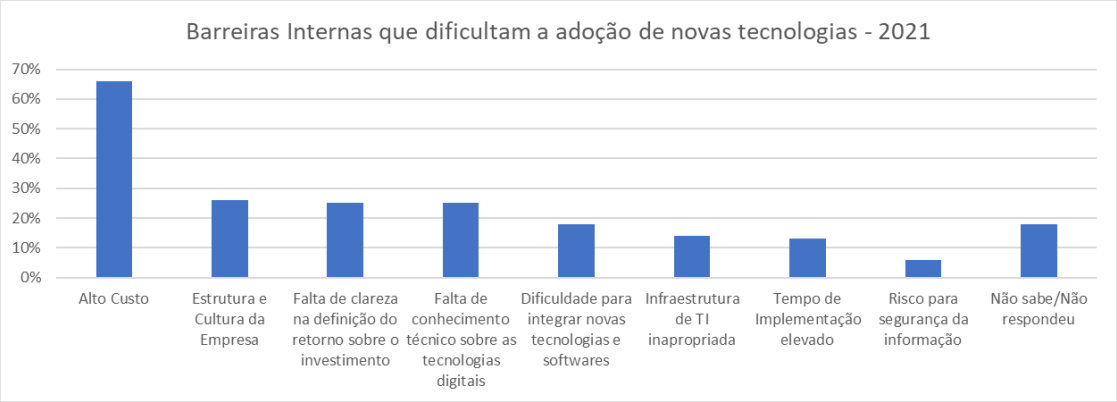
Antes da era digital e do mundo movido a dados, a manutenção era predominantemente reativa ou preventiva. No entanto, com o avanço da tecnologia, sensores, *big data* e algoritmos de aprendizado de máquina, a manutenção preditiva se tornou uma prática muito utilizada em diversos ramos da indústria visando otimização dos processos e redução de custos evitando paradas desnecessárias na produção [14].

Uma das necessidades mais intrínsecas à manutenção preditiva é a coleta contínua de dados dos equipamentos em operação, como vibração, temperatura e consumo de energia. Esses dados são analisados usando algoritmos de aprendizado de máquina para identificar padrões. Quando anomalias são detectadas, os técnicos podem intervir antes que ocorra uma falha, dessa forma se tem prolongamento da vida útil dos equipamentos, transformação e otimização das operações industriais, redução de custos e melhoria da segurança [14].

Entretanto, para que as indústrias comecem a usufruir dos benefícios dessa técnica de manutenção é necessário um investimento grande, uma vez que são necessários dados de qualidade para a interpretação e consumo dos modelos de aprendizado de máquina, a necessidade de alternativas para armazenamento de dados em larga escala (armazenamento em nuvem é uma alternativa muito utilizada atualmente, mas que gera um custo), ter equipe especializada para gerenciar e manusear toda a plataforma de sistemas e operações envolvidas e ter todo capital disponível para esses investimentos iniciais [13].

Uma pesquisa realizada em 2021 por [13] com diversas empresas mostra como elas enxergam as barreiras internas que dificultam a adoção das novas tecnologias e, fica claro, que o alto custo é, de longe, o principal impeditivo para que as empresas comecem a evoluir, como mostra a Figura 16.

Figura 16 - Barreiras internas que dificultam a adoção das tecnologias digitais.



Fonte: Adaptado de [13].

2.8.1 A evolução da manutenção

Antigamente a manutenção era vista principalmente em seu âmbito preventivo, conservativo e corretivo, focalizando aspectos técnicos e operacionais. No entanto, uma transformação significativa vem ocorrendo nos últimos tempos, ligados ao aspecto humano, de confiabilidade e de custos [15].

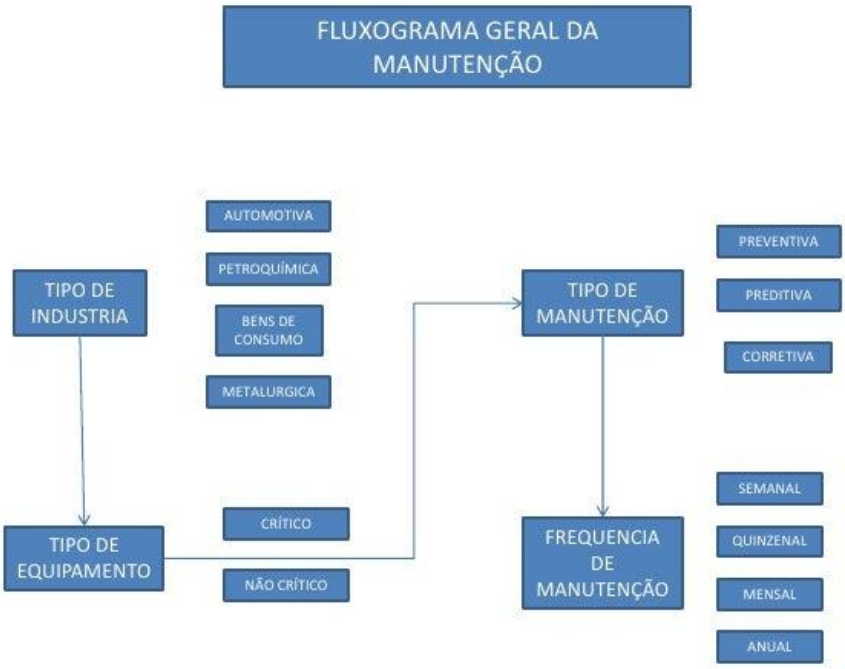
Essa transformação reflete a crescente importância e responsabilidade do setor de manutenção dentro das organizações modernas. O reconhecimento dos aspectos humanos mostra que existe a necessidade de habilidades interpessoais e trabalho em equipe na execução das atividades do dia a dia. Afinal, a colaboração e a comunicação entre os membros da equipe desempenham um papel importante na garantia da eficiência operacional e na minimização de falhas [15].

Além disso, a consideração dos aspectos de custos é vital em um mundo em que a otimização financeira é essencial. Compreender os custos associados à manutenção ajuda as organizações a tomar decisões informadas sobre investimentos em tecnologias e práticas de manutenção. Isso não apenas otimiza os recursos, mas também contribui para a sustentabilidade a longo prazo das operações [15].

A confiabilidade é outro pilar fundamental na definição moderna de manutenção. Em um cenário onde a confiabilidade operacional é crucial para a reputação e o sucesso das organizações, garantir que os equipamentos estejam sempre em pleno funcionamento torna-se uma prioridade. A manutenção, portanto, não é apenas sobre corrigir falhas quando elas ocorrem, mas também sobre implementar estratégias para preveni-las, melhorando assim a confiabilidade dos sistemas [15].

A Figura 17 mostra o atual fluxograma geral da manutenção.

Figura 17 - Fluxograma geral da Manutenção.



Fonte: [16].

Percebe-se que existe uma organização por trás das ações de manutenção. Deve-se definir criticidade, tipo de manutenção e com qual frequência será realizada. Desta forma, é possível realizar os agendamentos das ações de forma assertiva, aumentando a sinergia da equipe, maximizando lucros, minimizando falhas e otimizando custos devido ao fato de paradas desnecessárias serem reduzidas.

3 DESENVOLVIMENTO

Como ferramenta para o desenvolvimento deste trabalho de conclusão de curso será utilizado a plataforma *Visual Studio Code*, um software editor de código-fonte desenvolvido pela *Microsoft* que inclui inúmeras ferramentas internas úteis para desenvolvimento de projetos utilizando qualquer linguagem de programação. Será utilizada a extensão do *Jupyter Notebook* para o *Visual Studio Code*, para que seja possível desenvolver os códigos utilizando a linguagem de programação *Python* e se beneficiando dos blocos de códigos característicos do *Jupyter*.

Python será utilizada neste projeto pois é uma linguagem de programação de alto nível, versátil e fácil de aprender, devido à sua sintaxe clara, torna-se uma linguagem acessível mesmo para aqueles que estão apenas começando no mundo da programação. Além disso, é muito utilizada por profissionais de dados devido às suas bibliotecas especializadas, como *NumPy, Pandas e Scikit-Learn*, que fazem de *Python* uma ferramenta poderosa para análise de dados, modelagem estatística e aprendizado de máquina.

Para todo desenvolvimento do projeto será utilizado o CRISP-DM (Seção 1.3), medidas estatísticas, gráficos e modelagem de aprendizado de máquina que estão descritas na Introdução deste trabalho de graduação.

3.1 Entendimento de Negócio

Para o primeiro passo do projeto a ser desenvolvido é importante entender qual a importância e criticidade do que será realizado. Uma vez que a manutenção preditiva de equipamentos industriais e a antecipação de falhas tornou-se uma prioridade para as indústrias modernas, é necessário adicionar o conceito de Ciência de Dados, combinada com o CRISP-DM para oferecer uma estrutura sólida para enfrentar esse desafio complexo, permitindo uma análise preditiva robusta e orientada por dados.

Para este trabalho, o desafio é realizar a predição de falhas com no mínimo 15 ciclos de aferição de antecedência. Este número mínimo de ciclos foi estipulado pela equipe demandante do projeto, acredita-se que seja pelo fato de que sejam ciclos suficientes para realizar as intervenções dos equipamentos sem comprometimento das demais demandas da equipe. Ao total tem-se 100 equipamentos industriais que possuem 23 sensores e 3 pontos de configuração (variáveis presentes na base de dados de comandos realizados em cada máquina) que realizam as aferições duas vezes ao dia, ou seja, ao realizar a previsão de falhas em no mínimo 15 ciclos é possível estruturar uma agenda de manutenção com 7 dias de antecedência. Desta forma, é possível revolucionar a gestão de ativos industriais e trazer vantagem competitiva para as organizações no cenário industrial atual.

3.2 Entendimento dos dados

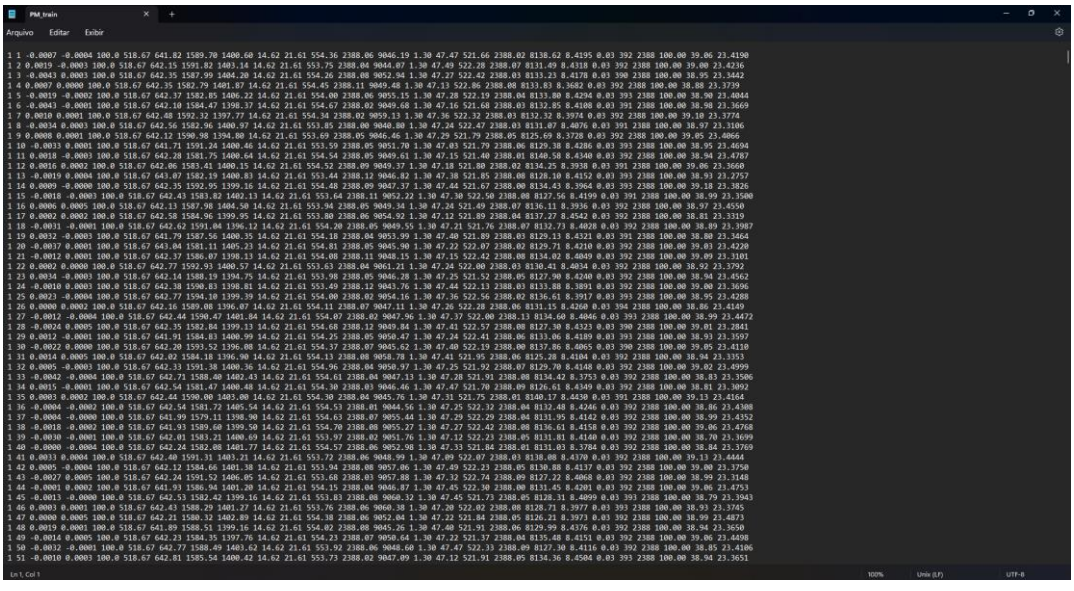
3.2.1 Obtenção dos dados para o projeto

Tratando-se de um projeto de ciência de dados, é necessário que a equipe demandante do projeto forneça também os dados necessários para todas as análises e modelagens, bem como a garantia da qualidade e integralidade dos dados fornecidos. Para este projeto, a base de dados obtida foi fornecida por uma indústria de celulose brasileira, a qual o nome não será divulgado por questões de privacidade, como parte de um processo seletivo interno para o cargo de Cientista de Dados. Tanto as bases para treino quanto a de teste referem-se à 100 equipamentos industriais monitorados por 23 sensores e com 3 pontos de configuração.

3.2.2 Entendimento e primeira visualização da base de dados

Existem inúmeros formatos de arquivos que podem servir como base de dados. Dependendo do grau tecnológico da empresa fornecedora dos dados, pode existir até mesmo um banco de dados que consegue armazenar uma infinidade de informações. Entretanto, para esse caso, os dados foram fornecidos por meio de um arquivo de texto (*.txt*), o delimitador entre cada coluna é definido por um espaço entre as informações. A Figura 18 mostra a primeira visualização desse arquivo de texto contendo as informações do problema.

Figura 18 - Arquivo de texto com as informações do problema.



Fonte: Elaborado pelo autor.

Após o carregamento das informações no Jupyter Notebook, podemos utilizar a função *info()* do *Python* para obtenção de informações básicas a respeito de cada coluna. Segundo a Figura 19 é possível notar os seguintes problemas com a base de dados:

1. Os nomes de cada coluna estão representados por números e não com as informações que eles realmente se referem, ou seja: a máquina, o ciclo de aferição dos sensores, os pontos de conFiguração e os sensores de monitoramento.
2. As duas últimas colunas (números 26 e 27) que correspondem aos sensores de número 22 e 23 estão completamente vazias.

Figura 19 - Informações da base de dados.

```
data.info()
[4] ✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20631 entries, 0 to 20630
Data columns (total 28 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       20631 non-null  int64
 1   1       20631 non-null  int64
 2   2       20631 non-null  float64
 3   3       20631 non-null  float64
 4   4       20631 non-null  float64
 5   5       20631 non-null  float64
 6   6       20631 non-null  float64
 7   7       20631 non-null  float64
 8   8       20631 non-null  float64
 9   9       20631 non-null  float64
 10  10      20631 non-null  float64
 11  11      20631 non-null  float64
 12  12      20631 non-null  float64
 13  13      20631 non-null  float64
 14  14      20631 non-null  float64
 15  15      20631 non-null  float64
 16  16      20631 non-null  float64
 17  17      20631 non-null  float64
 18  18      20631 non-null  float64
 19  19      20631 non-null  float64
 20  20      20631 non-null  float64
 21  21      20631 non-null  int64
 22  22      20631 non-null  int64
 23  23      20631 non-null  float64
 24  24      20631 non-null  float64
 25  25      20631 non-null  float64
 26  26      0 non-null      float64
 27  27      0 non-null      float64
dtypes: float64(24), int64(4)
memory usage: 4.4 MB
```

Fonte: Elaborado pelo autor.

Nota-se, também, que se tem um total de 20631 linhas, ou seja, foram realizados 20631 ciclos de medição dos sensores, no total, para os 100 equipamentos disponíveis nessa base. Os formatos de cada coluna (Dtype), a princípio, não se pode afirmar se estão corretos ou não, entretanto deve-se ter uma atenção especial para os tipos de dados que serão utilizados no projeto, pois sabe-se que os algoritmos de aprendizado de máquina necessitam ter os tipos de dados adequados para

conseguir generalizar o problema em questão. Neste caso, o tipo *int64* refere-se à valores numéricos, inteiros, com até 64 *bits* (varia de -2^{63} à $2^{63}-1$). E o tipo *float64* refere-se a dados numéricos, decimais, com até 64 *bits*, variando no mesmo intervalo que o tipo anterior.

Para solucionar o primeiro problema encontrado, foi necessário criar uma solução em *Python* que substituísse os números dos cabeçalhos das colunas para o seu respectivo nome. Sabe-se, devido às informações obtidas com a equipe demandante, que a ordem das colunas é a disponível na Tabela 1:

Tabela 1 - Nome e descrição das colunas da base de dados recebida.

Nome da Coluna	**Descrição**
Id	Número identificador do equipamento.
Runtime	Quantidade de ciclos de aferição realizados
Setting 1	Ponto de configuração 1
Setting 2	Ponto de configuração 2
Setting 3	Ponto de configuração 3
tag_1	Valor de aferição obtido pelo sensor 1
tag_2	Valor de aferição obtido pelo sensor 2
tag_3	Valor de aferição obtido pelo sensor 3
tag_4	Valor de aferição obtido pelo sensor 4
tag_5	Valor de aferição obtido pelo sensor 5
tag_6	Valor de aferição obtido pelo sensor 6
tag_7	Valor de aferição obtido pelo sensor 7
tag_8	Valor de aferição obtido pelo sensor 8
tag_9	Valor de aferição obtido pelo sensor 9
tag_10	Valor de aferição obtido pelo sensor 10
tag_11	Valor de aferição obtido pelo sensor 11
tag_12	Valor de aferição obtido pelo sensor 12
tag_13	Valor de aferição obtido pelo sensor 13
tag_14	Valor de aferição obtido pelo sensor 14
tag_15	Valor de aferição obtido pelo sensor 15
tag_16	Valor de aferição obtido pelo sensor 16
tag_17	Valor de aferição obtido pelo sensor 17
tag_18	Valor de aferição obtido pelo sensor 18
tag_19	Valor de aferição obtido pelo sensor 19
tag_20	Valor de aferição obtido pelo sensor 20
tag_21	Valor de aferição obtido pelo sensor 21
tag_22	Sensor danificado - desconsiderar
tag_23	Sensor danificado - desconsiderar

Fonte: Elaborado pelo autor.

Agora, com posse dessas informações, desenvolveu-se uma solução utilizando *Python* (Figura 20) para renomear cada coluna, resultando no que mostra a Figura 21.

Figura 20 - Solução em *Python* para renomear colunas.

```python
#Cria lista com o nome real de cada coluna
lista_colunas = [
'id',
'runtime',
'setting1',
'setting2',
'setting3']
for tag_number in range(1, 22):
    tag = f'tag_{tag_number}'
    lista_colunas.append(tag)

#Cria dicionário que possui nas chaves o nome antigo das colunas e nos valores o novo nome
dict_map = {old_name: new_name for old_name, new_name in zip(data.columns, lista_colunas)}

#Utiliza a função rename da biblioteca pandas para substituir os nomes antigos pelos novos utilizando o dicionário acima.
data.rename(columns=dict_map, inplace=True)
```

Python

Fonte: Elaborado pelo autor.

Figura 21 - Colunas renomeadas na base recebida.

#	Column	Non-Null Count	Dtype
0	id	20631 non-null	int64
1	runtime	20631 non-null	int64
2	setting1	20631 non-null	float64
3	setting2	20631 non-null	float64
4	setting3	20631 non-null	float64
5	tag_1	20631 non-null	float64
6	tag_2	20631 non-null	float64
7	tag_3	20631 non-null	float64
8	tag_4	20631 non-null	float64
9	tag_5	20631 non-null	float64
10	tag_6	20631 non-null	float64
11	tag_7	20631 non-null	float64
12	tag_8	20631 non-null	float64
13	tag_9	20631 non-null	float64
14	tag_10	20631 non-null	float64
15	tag_11	20631 non-null	float64
16	tag_12	20631 non-null	float64
17	tag_13	20631 non-null	float64
18	tag_14	20631 non-null	float64
19	tag_15	20631 non-null	float64
20	tag_16	20631 non-null	float64
21	tag_17	20631 non-null	int64
22	tag_18	20631 non-null	int64
23	tag_19	20631 non-null	float64
24	tag_20	20631 non-null	float64
25	tag_21	20631 non-null	float64
26	26	0 non-null	float64
27	27	0 non-null	float64

Fonte: Elaborado pelo autor.

Fez-se necessário entender, de fato, como os dados foram obtidos e dispostos ao longo da base de dados importada para a plataforma de desenvolvimento. Ao enviar os dados, a equipe de

manutenção afirmou que cada linha de informações corresponde a uma aferição realizada pelos sensores para aquela máquina (coluna *"id"*) em específico. A coluna *"runtime"* é o número da aferição realizada, sendo que, a última aferição de uma máquina é o momento da falha. Sendo assim, a próxima linha da tabela já é relacionada à máquina seguinte (*"id"* + 1) em sua primeira aferição. A Figura 22 mostra um corte na base de dados que mostra a última aferição da máquina de id igual à 1 (momento da falha) e primeiras aferições para a máquina de id igual à 2.

Figura 22 - Corte na base de dados para entendimento da disposição das informações.

```
index_limit = data[(data.id == 1) & (data.runtime == data[data.id == 1].runtime.max())].index
data.iloc[index_limit[0]-5:index_limit[0]+5]
```

[67] ✓ 0.0s Python

	id	runtime	setting1	setting2	setting3	tag_1	tag_2	tag_3	tag_4	tag_5	...	tag_14	tag_15	tag_16	tag_17	tag_18	tag_19	tag_20	tag_21	26	27
186	1	187	-0.0047	-0.0000	100.0	518.67	643.32	1592.10	1427.27	14.62	...	8115.67	8.5218	0.03	396	2388	100.0	38.42	23.0822	NaN	NaN
187	1	188	-0.0067	0.0003	100.0	518.67	643.75	1602.38	1422.78	14.62	...	8117.69	8.5207	0.03	396	2388	100.0	38.51	22.9588	NaN	NaN
188	1	189	-0.0006	0.0002	100.0	518.67	644.18	1596.17	1428.01	14.62	...	8117.51	8.5183	0.03	395	2388	100.0	38.48	23.1127	NaN	NaN
189	1	190	-0.0027	0.0001	100.0	518.67	643.64	1599.22	1425.95	14.62	...	8112.58	8.5223	0.03	398	2388	100.0	38.49	23.0675	NaN	NaN
190	1	191	-0.0000	-0.0004	100.0	518.67	643.34	1602.36	1425.77	14.62	...	8114.61	8.5174	0.03	394	2388	100.0	38.45	23.1295	NaN	NaN
191	1	192	0.0009	-0.0000	100.0	518.67	643.54	1601.41	1427.20	14.62	...	8110.93	8.5113	0.03	396	2388	100.0	38.48	22.9649	NaN	NaN
192	2	1	-0.0018	0.0006	100.0	518.67	641.89	1583.84	1391.28	14.62	...	8137.72	8.3905	0.03	391	2388	100.0	38.94	23.4585	NaN	NaN
193	2	2	0.0043	-0.0003	100.0	518.67	641.82	1587.05	1393.13	14.62	...	8131.09	8.4167	0.03	392	2388	100.0	39.06	23.4085	NaN	NaN
194	2	3	0.0018	0.0003	100.0	518.67	641.55	1588.32	1398.96	14.62	...	8140.58	8.3802	0.03	391	2388	100.0	39.11	23.4250	NaN	NaN
195	2	4	0.0035	-0.0004	100.0	518.67	641.68	1584.15	1396.08	14.62	...	8140.44	8.4018	0.03	391	2388	100.0	39.13	23.5027	NaN	NaN

Fonte: Elaborado pelo autor.

Fica claro então, no exemplo da Figura 22, que a máquina 1 possui 192 ciclos de aferição até o momento de sua falha e, logo após, ficam dispostas as aferições dos sensores responsáveis pela máquina 2. Isso é realizado para os 100 equipamentos industriais disponíveis nessa base de dados.

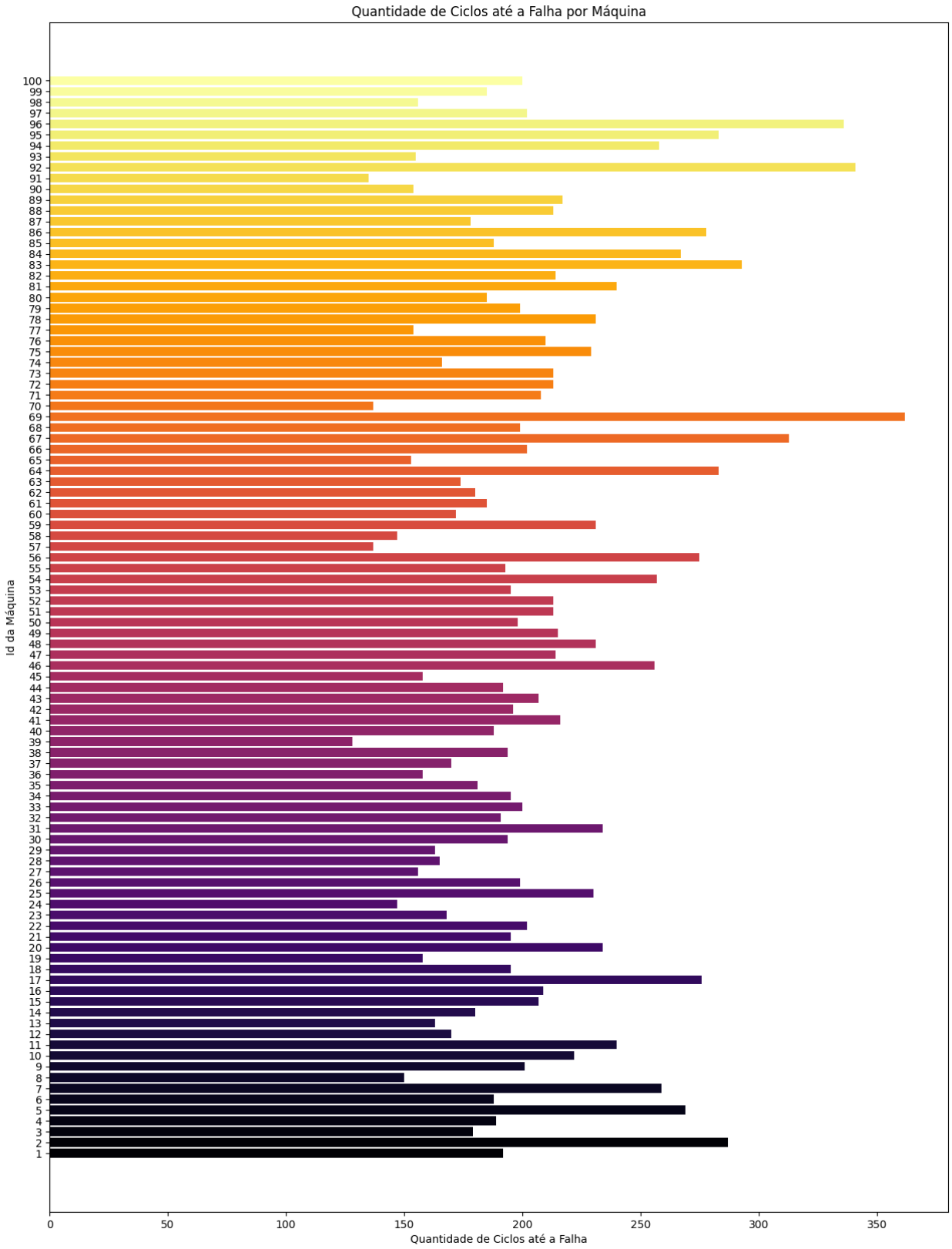
3.2.3 Análise Exploratória de Dados

Partindo do entendimento de como estão disponibilizadas as aferições é possível obter uma valiosa informação para o problema que está sendo trabalhado: Quantos ciclos de aferição cada máquina realizou até a falha? Pensando nisso, foi construído um gráfico que mostra essas informações, disponível na Figura 23.

Para esse tipo de visualização o gráfico de barras horizontal foi escolhido devido à quantidade de máquinas e considerando uma melhor visualização gráfica.

Percebe-se, pela Figura 23, que as máquinas 96, 92, 69 e 67 são as 4 máquinas que tiveram mais de 300 ciclos de aferição até a falha, ou seja, demoraram mais para necessitarem de manutenção, enquanto as máquinas 39, 57, 70 e 91 foram as que necessitaram de manutenção com menos de 150 ciclos de aferição.

Figura 23 - Quantidade de ciclos até a falha por Máquina.

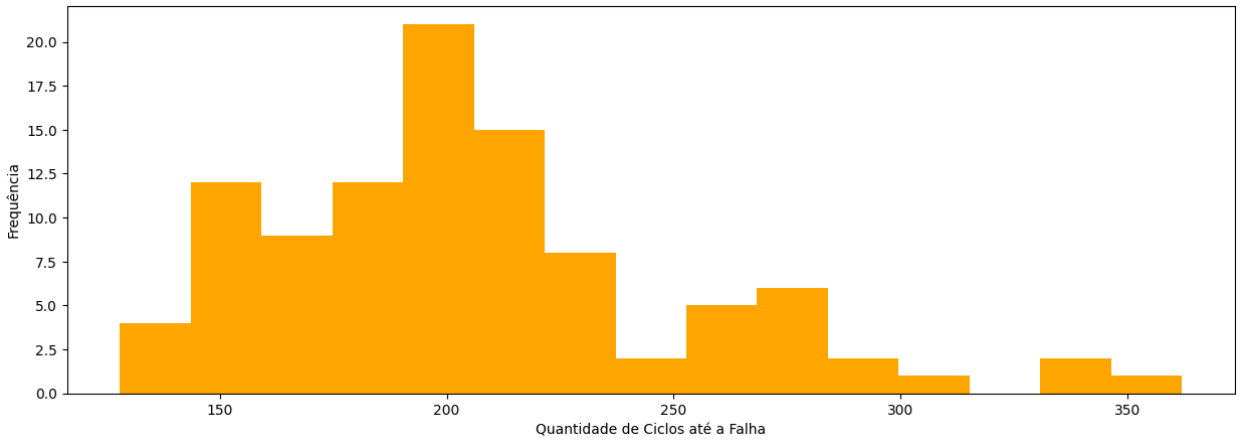


Fonte: Elaborado pelo autor.

Um histograma também foi plotado com o intuito de observar qual é o intervalo de ciclos de aferição em que mais acontecem falhas. A Figura 24 mostra que existe uma grande concentração

de máquinas que falham por volta dos 200 ciclos. Nota-se, também que se trata de uma distribuição assimétrica positiva, com essa conclusão podemos afirmar que, com relação ao máximo de ciclos de cada equipamento até a falha, a média é maior que a mediana que, por sua vez, é maior que a moda.

Figura 24 - Histograma da quantidade de ciclos até a falha por equipamento.



Fonte: Elaborado pelo autor.

Para comprovar essa afirmação, foram calculadas: média, moda e mediana da quantidade máxima de ciclos até a falha, como mostra a Figura 25.

Figura 25 - Média, Mediana e Moda

```
    print('Média: ',df_falha['ultima_afericao'].mean())
    print('Mediana: ',df_falha['ultima_afericao'].median())
    print('Moda: ',df_falha['ultima_afericao'].mode()[0])
[165]  ✓ 0.0s
...  Média:  206.31
     Mediana:  199.0
     Moda:  213
```

Fonte: Elaborado pelo autor.

Como é possível identificar na Figura 25, a afirmação que foi realizada apenas observando a assimetria do histograma está errada. A moda é maior que a mediana e a média, e isso não está errado. Ao observar atentamente o histograma da Figura 24, vê-se que a assimetria positiva é causada devido aos valores mais altos à direita da distribuição que não são tão discrepantes do resto da distribuição enviesando muito pouco a média. Outro ponto é que o local de maior

frequência da distribuição é um local muito próximo aos altos valores que formam a cauda à direita, dessa forma, a moda é maior que média e mediana como comprovado.

Outro ponto a ser estudado é qual a relação entre cada sensor, ou seja, se um sensor está aferindo uma grandeza pode ser que outro sensor também esteja aferindo uma consequência dessa grandeza, por exemplo: Se um sensor que mede rotação está com valores altos, é muito provável que um sensor que mede vibração também esteja aferindo valores altos, uma vez que são grandezas correlacionadas. Desta forma, foi criada uma matriz de correlações.

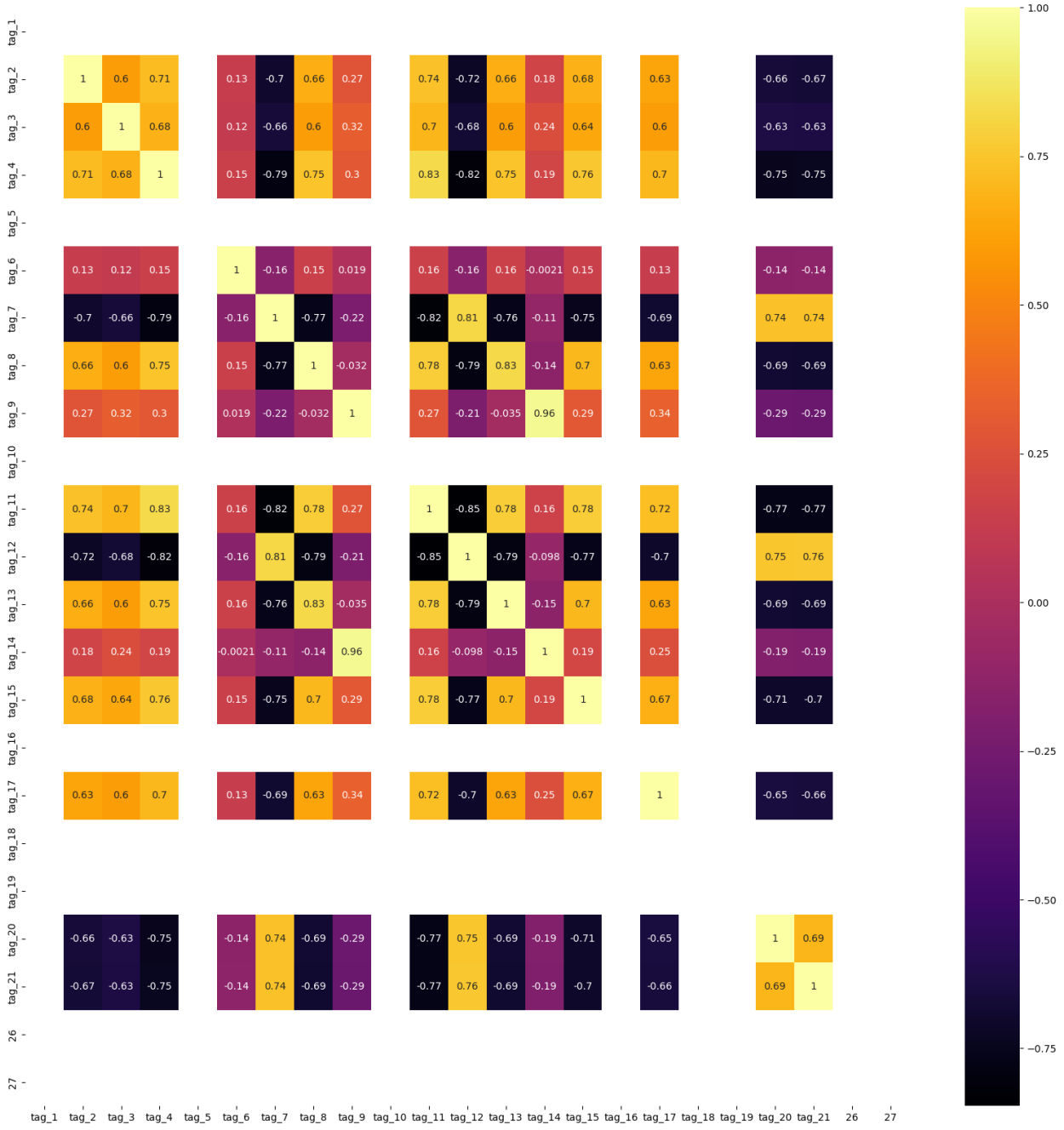
A Figura 26 mostra todos os sensores e suas correlações, em que foi utilizado para cálculo de correlação o método do coeficiente de Pearson já discutido anteriormente. Por meio da Figura 26 é possível tirar algumas conclusões:

1. Os sensores de números: 1, 5, 10, 16, 18 e 19 não possuem correlação com nenhum outro sensor. Isso acontece pois, como veremos a seguir, esses sensores realizam aferições constantes (mesmo valor em todos os ciclos), sendo assim, a variância dos dados desses sensores é nula, criando uma indeterminação ao calcular o coeficiente de Pearson.
2. Existem sensores fortemente correlacionados positivamente, como por exemplo:
    a. 4 e 11;
    b. 7 e 12;
    c. 8 e 13.
3. E os fortemente correlacionados negativamente, como por exemplo:
    a. 4 e 7;
    b. 4 e 20;
    c. 12 e 11.

A correlação entre os sensores, seja ela positiva ou negativa, pode mostrar à equipe de manutenção:

1. Quais grandezas estão correlacionadas;
2. Se uma destas correlações ficarem muito discrepantes é possível que exista um distúrbio no funcionamento da máquina;
3. Quais sensores observar caso seu par correlacionado esteja com defeito.

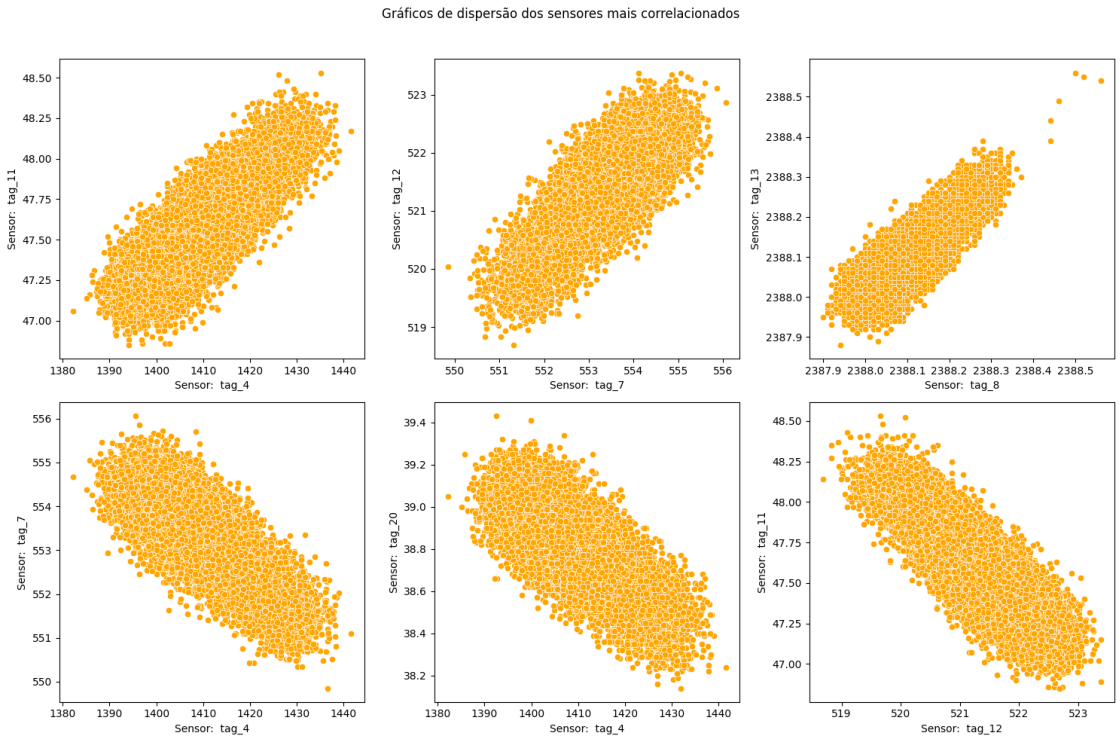
Figura 26 - Mapa de calor considerando a correlação entre cada sensor.



Fonte: Elaborado pelo autor.

Elencados os 3 pares de máquinas com forte correlação positiva e negativa, é possível verificar, por meio de um gráfico de dispersão, como é esta relação de forma visual. A Figura 27 mostra os 6 gráficos de dispersão relacionando os pares de sensores já verificados anteriormente.

Figura 27 - Gráficos de dispersão dos sensores mais correlacionados.



Fonte: Elaborado pelo autor.

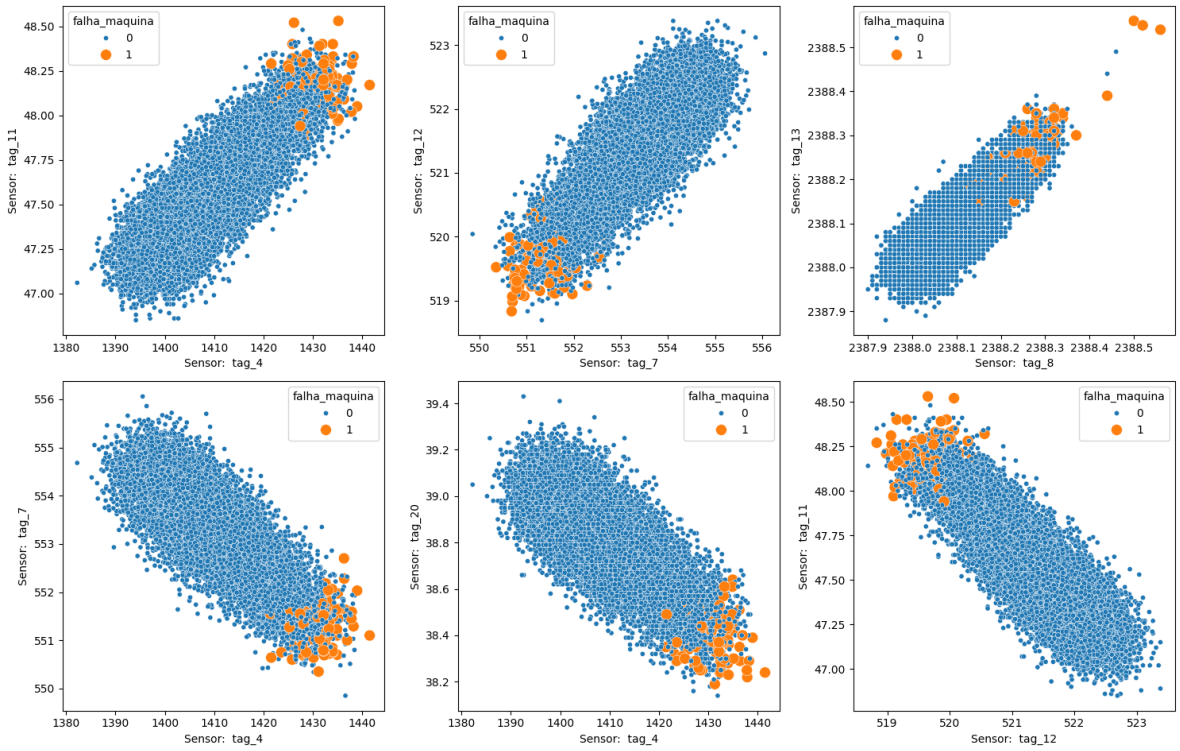
De fato, ao observar a Figura 27, vê-se que existe a correlação positiva e negativa muito forte entre os sensores descritos, entretanto isso é uma informação muito vaga visto que neste projeto o objetivo é prever falhas. Então, buscou-se definir o quanto os sensores mais correlacionados influenciam nas falhas das máquinas.

Com o intuito de responder essa questão, foram plotados os gráficos da Figura 28, que mostram os mesmos sensores, entretanto destacando o valor destes sensores nos momentos de falha para cada equipamento da base de dados.

Percebe-se então que as falhas nas máquinas também estão fortemente correlacionadas com os valores dos sensores que estão mais correlacionados entre si. No primeiro gráfico de dispersão, que mostra a correlação entre os sensores “tag_11” e “tag_4”, as falhas acontecem em seus picos de atuação, ou seja, quando seus valores são os mais altos registrados na base de dados.

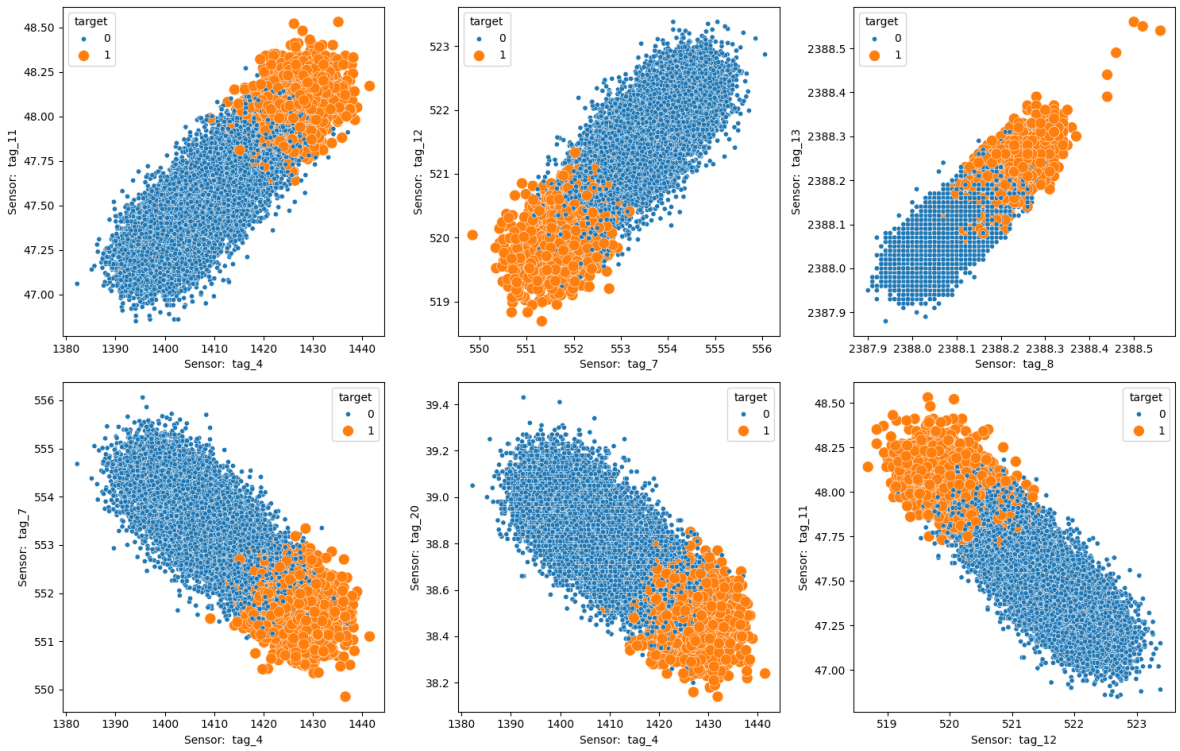
Por fim, uma vez que o objetivo do projeto é prever falhas, mas mais precisamente, com uma antecipação de 15 ciclos de aferição (*target*), foi verificado que os sensores mais correlacionados também estão correlacionados com a iminência da falha por meio dos gráficos construídos na Figura 29.

Figura 28 - Gráficos de dispersão dos sensores mais correlacionados com distinção no momento da falha.



Fonte: Elaborado pelo autor.

Figura 29 - Gráficos de dispersão dos sensores mais correlacionados com distinção na iminência da falha.



Fonte: Elaborado pelo autor.

3.3 Preparação dos dados

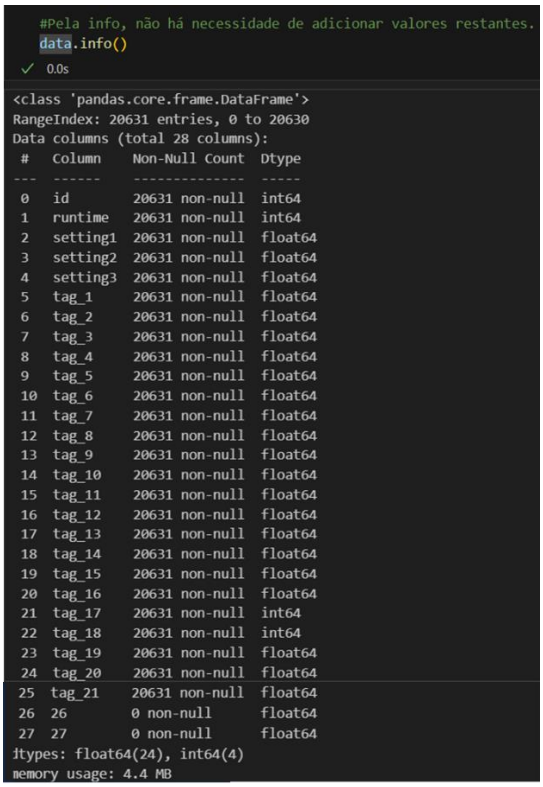
3.3.1 Pré-preparação dos dados

Nesta etapa do projeto, como já comentado anteriormente, é o momento em que os dados são transformados, tratados e até mesmo modificados para se tornarem novas variáveis. Portanto, de início é necessário ter em mente as seguintes questões:

1. Quais os tipos de dados provenientes da tabela que será utilizada para que seja possível identificar possíveis variáveis preditoras em que seja necessário a transformação do tipo de variável;
2. Existem colunas com valores nulos? Quantas? É necessário retirá-los?
3. Existem colunas com valores em escalas completamente diferentes? É necessário tratá-las?
4. Pré-seleção de variáveis: existem variáveis que podem ser retiradas da análise?

Ao utilizar a biblioteca pandas no *Python* e utilizar o método *info()*, pode-se responder algumas destas questões, como pode ser na Figura 30, muito semelhante à Figura 19, entretanto com os nomes das variáveis.

Figura 30 - Visualização dos tipos das variáveis e valores nulos.



Fonte: Elaborado pelo autor.

Pode-se notar que, como já citado, tem-se duas variáveis (26 e 27) com todos os valores nulos, essas variáveis serão logo retiradas da análise por não possuírem poder preditivo para o projeto. Das restantes, todas possuem os valores não-nulos, desta forma não será necessária aplicação de nenhum tipo de estratégia para preenchimento de valores faltantes.

As outras variáveis provenientes de sensores (*tags*), configurações(*settings*) número de ciclos de aferição do sensoriamento (*runtime*) e identificação da máquina (*id*) estão todas com seus respectivos tipos de dados corretos, uma vez que apenas será trabalhado com variáveis numéricas e decimais. Não será necessário a mudança do tipo de variável para esse projeto. Isso pode ter acontecido devido a forma com que os dados foram extraídos, armazenados e enviados para o time de dados. Entretanto, se o projeto for algo em que se utilize um número alto de variáveis preditoras e ter procedência duvidosa, é necessário se atentar mais aos tipos de dados para não resultar em complicações no treinamento.

3.3.2 Criação da variável resposta (*target*)

Como parte da preparação de dados, é necessário criar a variável que será responsável por identificar a falha na máquina para que os algoritmos de aprendizado de máquina recebam essa variável como entrada juntamente com os dados preditores e assim aprenda os padrões relacionados ao fenômeno da falha.

O objetivo do projeto é realizar a previsão de quando a máquina irá falhar com 15 ciclos de aferição de antecedência, ou seja, este projeto configura-se como um projeto de classificação, onde busca-se realizar a previsão para uma variável categórica (falha ou não falha).

Para criar a variável *target* será utilizado a seguinte estratégia: utilizar como referência a variável de contagem de ciclos de aferição (*runtime*), utilizar os 15 últimos ciclos antes da falha e marcá-los como 1 (iminência de falha) o restante será marcado como 0 (funcionamento normal).

Como mostrado nas linhas de código em *Python* na Figura 31, inicialmente cria-se uma coluna na tabela chamada *"target"* somente com zeros; uma lista chamada "equipamentos_unicos" também é criada em que recebe somente os *ids* da coluna de *id* que são únicos, ou seja, de 1 até 100; agora uma estrutura de repetição (*loop for*) é criada onde para cada *id* único da lista "equipamentos_unicos" é encontrado o índice máximo (número da linha), ou seja, é o último registro de aferição dos sensores ou o momento da falha; com esse valor guardado na variável "indice_falha", cria-se uma condicional onde se o índice da falha for maior ou igual à 15 atribui o valor 1 à coluna "*target*" para todos os índices no intervalo: "índice máximo do equipamento-14" até índice máximo do equipamento, caso contrário atribui o valor 1 à todas as

observações daquela máquina, uma vez que a falha aconteceu antes de completar o décimo quinto ciclo de aferição.

Figura 31 - Código gerador da variável resposta (*target*).

```
data['target'] = 0
#Criando target: Prever falha em até 15 ciclos antes
equipamentos_unicos = data['id'].unique()
for equipamento in equipamentos_unicos:
    # Encontre o índice da linha onde ocorre a falha
    indice_falha = data[data['id'] == equipamento].index.max()

    # Defina as últimas 15 linhas até a falha como target = 1
    if indice_falha >= 15:
        data.loc[indice_falha - 14:indice_falha, 'target'] = 1
    else:
        data.loc[:indice_falha, 'target'] = 1
```

✓ 0.0s

Fonte: Elaborado pelo autor.

3.3.3 Definição do algoritmo a ser utilizado e a necessidade de escalonamento

Por fim, é necessário entender se temos dados em escalas completamente diferentes, uma vez que caso sejam utilizadas regressões lineares ou algoritmos baseados em distâncias, é necessário que as variáveis preditoras estejam na mesma escala para que escalas maiores não tenham peso maior na predição da variável resposta.

A Figura 32 mostra uma função em *Python* chamada *describe* em que é possível observar grande parte das estatísticas relacionadas às variáveis da tabela.

Figura 32 - Estatísticas relacionadas às variáveis.

```
data[lista_colunas].drop(['id','runtime'], axis = 1).describe()
```

[110] ✓ 0.0s Python

	setting1	setting2	setting3	tag_1	tag_2	tag_3	tag_4	tag_5	tag_6	tag_7	...	tag_12	tag_13
count	20631.000000	20631.000000	20631.0	20631.00	20631.000000	20631.000000	20631.000000	2.063100e+04	20631.000000	20631.000000	...	20631.000000	20631.000000
mean	-0.000009	0.000002	100.0	518.67	642.680934	1590.523119	1408.933782	1.462000e+01	21.609803	553.367711	...	521.413470	2388.096152
std	0.002187	0.000293	0.0	0.00	0.500053	6.131150	9.000605	1.776400e-15	0.001389	0.885092	...	0.737553	0.071919
min	-0.008700	-0.000600	100.0	518.67	641.210000	1571.040000	1382.250000	1.462000e+01	21.600000	549.850000	...	518.690000	2387.880000
25%	-0.001500	-0.000200	100.0	518.67	642.325000	1586.260000	1402.360000	1.462000e+01	21.610000	552.810000	...	520.960000	2388.040000
50%	0.000000	0.000000	100.0	518.67	642.640000	1590.100000	1408.040000	1.462000e+01	21.610000	553.440000	...	521.480000	2388.090000
75%	0.001500	0.000300	100.0	518.67	643.000000	1594.380000	1414.555000	1.462000e+01	21.610000	554.010000	...	521.950000	2388.140000
max	0.008700	0.000600	100.0	518.67	644.530000	1616.910000	1441.490000	1.462000e+01	21.610000	556.060000	...	523.380000	2388.560000

Fonte: Elaborado pelo autor.

Pela linha *"mean"*, que mostra a média das observações para cada coluna, percebe-se que existe uma diferença de escala para algumas variáveis em que vemos na Figura 32, como por exemplo: *"*setting2*"* possui uma média de $2x10^{-6}$ enquanto que *"*tag_2*"* possui uma média de 642,68 em suas observações. A questão é: de fato é necessário realizar o escalonamento? A resposta é: depende.

Como já tratado anteriormente, o escalonamento é de suma importância caso o problema em questão seja de regressão ou que utilize qualquer algoritmo que seja baseado em distâncias: K-Vizinhos Mais Próximos, por exemplo. Entretanto, sabe-se que atualmente os algoritmos baseados em árvore (Florestas Aleatórias, por exemplo) estão dominando os projetos de ciência de dados, devido aos seus benefícios e justamente à sua robustez frente à dados fora de escala e outliers. Não serão testados outros algoritmos pelo fato de que o objetivo desse trabalho é simular um projeto dentro de um contexto de mercado, em que é necessário que datas sejam cumpridas, desta forma de primeira mão já é escolhido um algoritmo que entregará um resultado satisfatório.

Portanto, para este projeto, será utilizado o algoritmo de florestas aleatórias com viés de classificação para determinar as falhas em equipamentos industriais sensorizados com 15 ciclos de aferição de antecedência.

3.4 Modelagem

3.4.1 Definição de Solução

Na abordagem da manutenção preditiva, há diversas soluções possíveis para otimizar o desempenho das máquinas industriais. O ponto de partida é definir claramente os objetivos, focando no lucro da indústria. A manutenção preditiva surge como a opção mais econômica, contribuindo para prolongar a vida útil das máquinas, evitar paradas não programadas, otimizar a alocação de recursos e prevenir danos e acidentes.

Ao utilizar a ciência de dados como ferramenta para alavancar esse processo é possível utilizar de modelos de aprendizado de máquina para antecipar falhas com base em dados coletados dos equipamentos. Para este projeto onde deseja-se definir a falha com 15 ciclos de aferição de antecedência a abordagens de solução adotada foi a de utilizar algoritmos de classificação, mais precisamente os baseados em árvores devido à sua robustez, performance e facilidade de implementação.

É evidente que, com a previsão de falhas, a equipe de manutenção terá a oportunidade de criar estratégias de manutenção, agendar manutenções em equipamentos críticos e agir sem que a

produção pare por completo. Desta forma, é possível atingir os objetivos do projeto de maximização de lucros com manutenções.

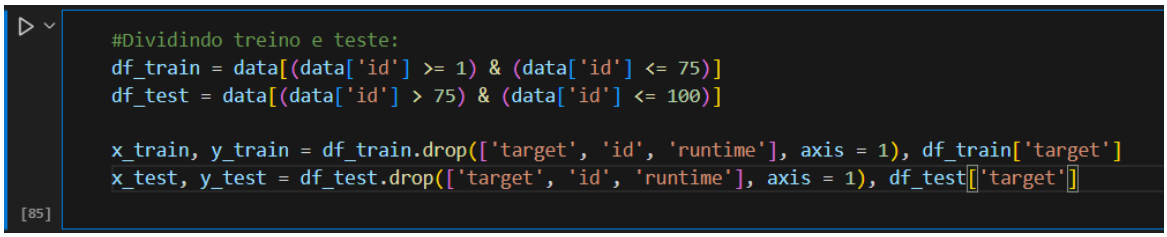
3.4.2 Divisão treino e teste para modelagem

Como todo projeto de ciência de dados que utiliza algoritmos de aprendizado de máquina, fez-se necessário a divisão da base de dados entre treino e teste. A base de treino é aquela que será utilizada como entrada no algoritmo para que ele aprenda os padrões entre os dados e a base de teste, são dados que o algoritmo desconhece e que será testado sua eficácia por meio das métricas de avaliação.

Entretanto, para esse projeto em específico, é necessário desenvolver uma estratégia própria para a divisão entre treino e teste. Usualmente, para projetos em que cada observação pode corresponder a um cliente único, dividem-se as observações com uma proporção fixa de observações para base de treino e o restante para base de teste (geralmente utiliza-se 80% para treino e 20% para teste). Mas, neste projeto, cada observação corresponde à uma aferição de sensores de uma máquina em específico, podendo uma única máquina, ter centenas de observações. Uma vez que é necessário que o algoritmo encontre os padrões para predição das falhas, é necessário que o corte para divisão em treino e teste seja feito considerando o *id* da máquina e não o número de observações totais. Nesse caso, para manter uma proporção próxima ao padrão, a base de treino contém as informações das máquinas de *id* 1 até 75 e a base de teste contém as informações das máquinas de *id* de 76 até 100. Desta forma, a base de treino resultou em 15159 observações, enquanto a base de teste 5472 observações. Tem-se então, uma proporção de 74% da base total para treino e 26% para teste.

O código em *Python* em que essa divisão foi realizada é mostrado na Figura 33.

Figura 33 - Código gerador de base de treino e teste.



Fonte: Elaborado pelo autor.

É necessário pontuar, que os algoritmos de aprendizado de máquina que são provenientes da biblioteca *Scikit-Learn* utilizam dois parâmetros principais de entrada:

1. X: São as variáveis preditoras, ou seja, todas as observações de sensores e configurações disponíveis na base de dados. No código é a variável *"*x*_train"*;
2. Y: É a variável resposta, que deve estar na mesma ordem que as variáveis preditoras. No código é a variável *"y_train"*.

Nas duas últimas linhas do código, em que as variáveis de *"x_train"*, *"y_train"*, *"x_test"* e *"y_test"* foram criadas, foram retiradas as colunas de *"id"* e *"runtime"* uma vez que a primeira é utilizada somente para identificação da máquina e a segunda é utilizada para criar a variável resposta, ou seja, aconteceria vazamento de informações caso ela estivesse sendo utilizada como variável preditora.

3.4.3 Treinamento do modelo utilizando Florestas Aleatórias

Como já citado no tópico 1.2.1.3.1 – *Bagging* e Florestas Aleatórias, esse tipo de algoritmo de aprendizado de máquina utiliza de inúmeras árvores para realizar a predição, fazendo uma amostra aleatória das observações para cada árvore e também (no caso das florestas aleatórias) uma amostra aleatória das próprias variáveis preditoras, ou seja, incluindo dois passos de aleatoriedade no algoritmo, resultando então em menos sobreajuste (*overfitting*) para o modelo. Isso tudo, a própria biblioteca do Scikit-Learn realiza automaticamente apenas recebendo como entrada as variáveis preditoras da base de treino (*x_train)* e a variável resposta *(x_test*).

Entretanto, somente esses parâmetros não são suficientes para resultar em um modelo generalizado e com previsões precisas. O algoritmo também possui entradas chamadas de Hiperparâmetros que, basicamente, são entradas que modificam a forma como o algoritmo é treinado. Para o treinamento deste modelo para previsão de falhas em máquinas com 15 ciclos de aferição de antecedência, serão utilizados os seguintes hiperparâmetros:

1. *n_estimators:* Representa o número de árvores na floresta, quanto mais árvores, mais previsões o algoritmo possui para realizar a decisão final por meio do voto majoritário. O incremento desse hiperparâmetro geralmente aprimora a precisão do modelo, contudo, há uma contrapartida no aumento do custo computacional;
2. *max_depth:* Refere-se à profundidade máxima de cada árvore na floresta, o que é essencial para controlar a complexidade do modelo, evitando sobreajuste. Valores inferiores são preferidos para prevenir árvores excessivamente profundas;
3. *min_samples_split:* Indica o número mínimo de amostras necessário para realizar uma divisão em um nó interno. Este hiperparâmetro desempenha um papel crucial na

prevenção de divisões que resultam em nós com poucas amostras, contribuindo assim para a mitigação do sobreajuste;

4. *min_samples_leaf:* Estabelece o número mínimo de amostras permitidas em uma folha (nó terminal), ou seja, controla o tamanho mínimo das folhas, desempenhando um papel significativo na suavização do modelo e prevenção do sobreajuste. Valores mais elevados tendem a gerar árvores mais simples.

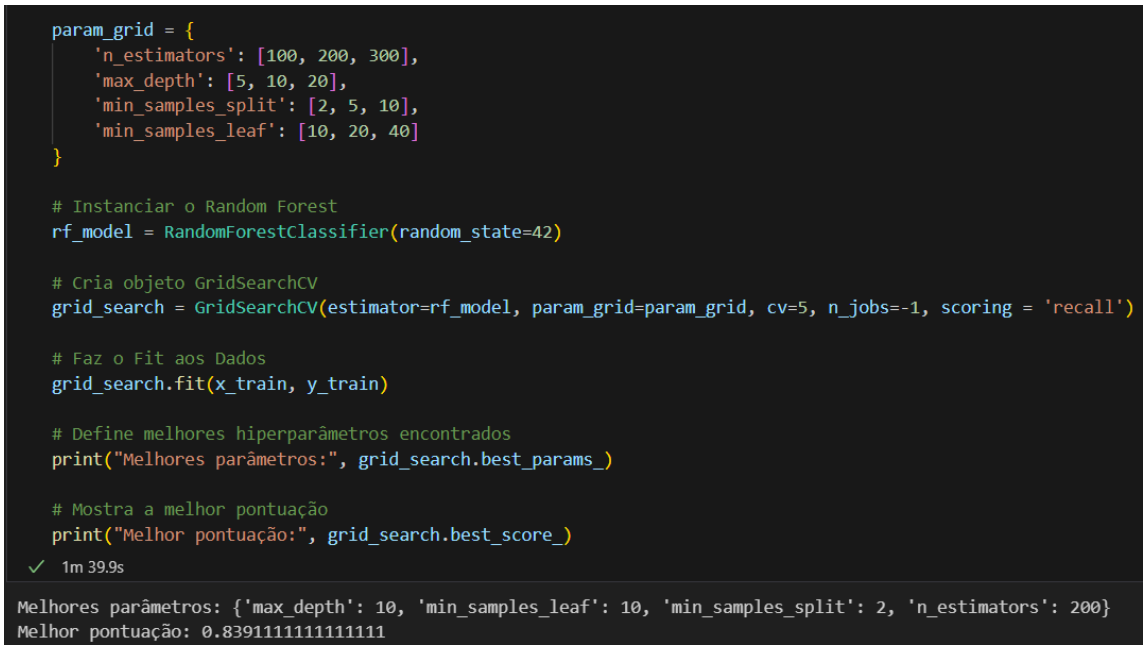
Estes hiperparâmetros desempenham um papel importante no desempenho do modelo de florestas aleatórias, possibilitando a adaptação do algoritmo aos padrões específicos do conjunto de dados em questão. Essa capacidade de ajuste fino permite aos cientistas de dados encontrar o equilíbrio adequado entre complexidade do modelo e generalização, fundamentais para o sucesso da aplicação dessa técnica em diferentes contextos.

Outra necessidade, como está disposto no item 1.5.4 – Seleção de Variáveis, é selecionar as variáveis mais importantes do modelo para o treinamento. Das 3 técnicas que já foram descritas, será utilizada a *Embedding,* a qual utiliza-se do próprio algoritmo do modelo para selecionar as variáveis mais importantes.

Por fim, para encontrar quais são os melhores hiperparâmetros para o modelo, será utilizada a técnica de validação cruzada (item 1.7.1 – Validação Cruzada), onde o número de *folds* do *K-Fold Cross Validation* será de 5.

A Figura 34 mostra as linhas de código utilizadas para a definição dos melhores hiperparâmetros.

Figura 34 - Código em *Python* para definição dos melhores hiperparâmetros.



```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [10, 20, 40]
}

# Instanciar o Random Forest
rf_model = RandomForestClassifier(random_state=42)

# Cria objeto GridSearchCV
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5, n_jobs=-1, scoring = 'recall')

# Faz o Fit aos Dados
grid_search.fit(x_train, y_train)

# Define melhores hiperparâmetros encontrados
print("Melhores parâmetros:", grid_search.best_params_)

# Mostra a melhor pontuação
print("Melhor pontuação:", grid_search.best_score_)
```

✓ 1m 39.9s

```
Melhores parâmetros: {'max_depth': 10, 'min_samples_leaf': 10, 'min_samples_split': 2, 'n_estimators': 200}
Melhor pontuação: 0.8391111111111111
```

Fonte: Elaborado pelo autor.

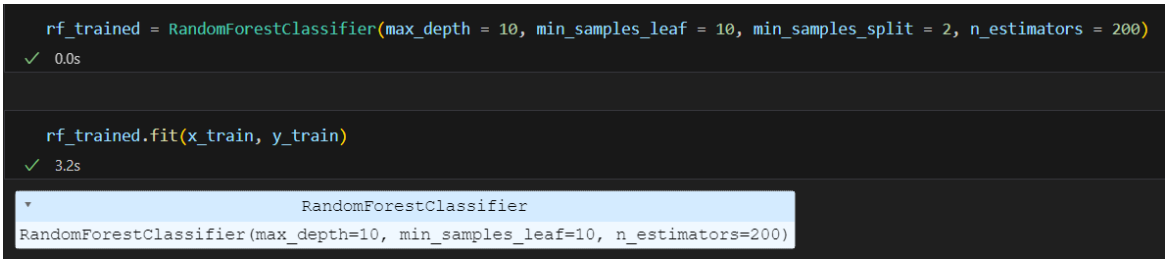
Conforme a Figura 34, foi criado um dicionário em *Python,* que contém os hiperparâmetros que serão utilizados para treinamento do modelo e, associado a eles, uma lista com três possibilidades de valores a serem testados para cada um. Os valores definidos nos dicionários foram definidos pelo próprio autor devido à experiências com esse tipo de modelagem. O *Scikit-Learn* possui um método chamado *GridSearchCV* (*Grid Search Cross Validation*) que basicamente cria modelos utilizando todas as combinações possíveis entre os valores de hiperparâmetros disponíveis no dicionário criado. Além disso, utiliza uma validação cruzada para definição de quais hiperparâmetros são melhores de acordo com uma métrica de avaliação. Como resultado foram obtidos os seguintes hiperparâmetros:

1. *n_estimators*: 200;
2. *max_depth*: 10;
3. *min_samples_leaf*: 10;
4. *min_samples_split*: 2.

E, para essa configuração, a pontuação final foi de 0.84 (84%), utilizando o *recall* (sensibilidade), como métrica a ser maximizada pelo *GridSearchCV*. A escolha da sensibilidade como métrica a ser maximizada será discutida adiante na seção de avaliação do modelo.

Por fim, com os melhores valores de hiperparâmetros em mãos, foi possível realizar o treinamento do modelo, como mostra a Figura 35.

Figura 35 - Treinamento do modelo utilizando Florestas Aleatórias e os melhores hiperparâmetros.



Fonte: Elaborado pelo autor.

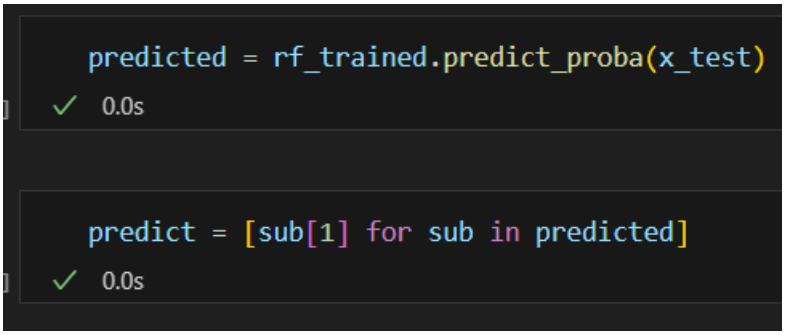
O objeto *Python "rf_trained",* é o objeto que possui todas as informações de treinamento, padrões e características da base de treino. É ele o responsável por realizar a predição dos valores na base de teste (*"x_test" e "y_test"*).

3.5 Avaliação

3.5.1 Avaliação e entendimento dos resultados na base de teste

O momento da avaliação é de extrema importância para a finalização do período de modelagem, pois é por meio dele que é possível avaliar quão bem o modelo está performando em dados que ele não conhece. Uma vez com o modelo treinado, deve-se utilizar a base de teste que foi dividida no item 2.4.2 (*x_test* e *y_test*). A Figura 36 mostra as linhas de código em *Python* utilizadas para realizar as previsões do modelo, note que a função utilizada é a *predict_proba,* essa função tem como saída um valor de propensão à target e não um valor fixo de 0 ou 1. A escolha dessa função aconteceu justamente para que seja possível modificar o limiar da matriz de confusão para melhorar a precisão ou sensibilidade do modelo, dependendo de qual for a mais importante para o projeto. A segunda linha de código da Figura 36 faz com que somente o item de índice 1 das predições (*predicted*) sejam armazenados no objeto *predict,* pois a função *predict_proba* tem como saída a propensão ao valor 0 e ao valor 1, ambos complementares.

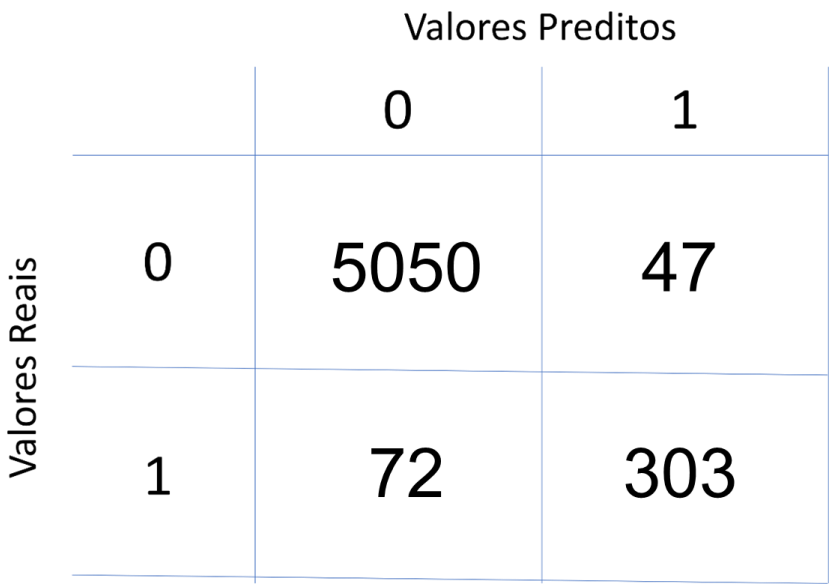
Figura 36 - Código para extração das propensões à iminência de falha.



Fonte: Elaborado pelo autor

De início, será utilizada uma matriz de confusão (mostrada no item 1.7.3.1), com um limiar de 0,5, ou seja, previsões que possuem valores maiores ou iguais à 0,5 serão classificadas como 1 (iminência de falha) e 0 caso contrário. A Figura 37 mostra a matriz de confusão com os resultados obtidos.

Figura 37 - Matriz de Confusão para as predições feitas com limiar de 0,5.

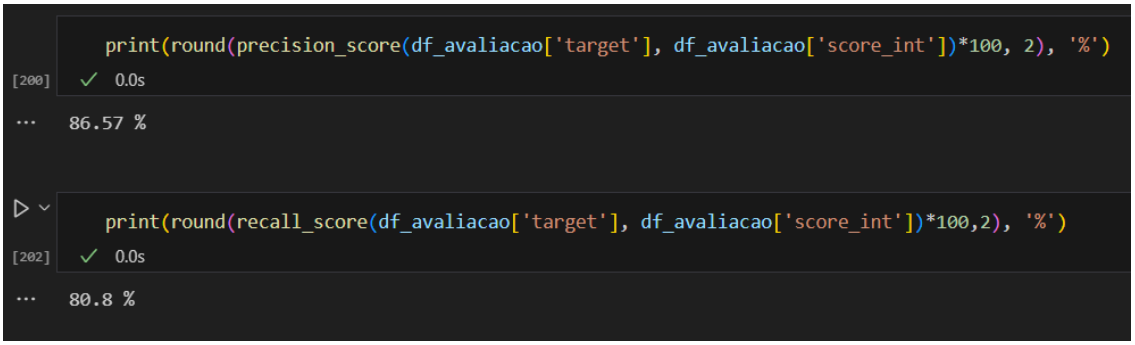


Fonte: Elaborado pelo autor.

Somente com a matriz de confusão, não existe muito que seja possível afirmar. Para se tomar conclusões específicas a respeito dos resultados, é necessário avaliar as métricas de precisão, sensibilidade e a curva ROC.

A Figura 38 mostra os resultados obtidos para precisão e sensibilidade.

Figura 38 - Valores de Precisão e Sensibilidade respectivamente.



Fonte: Elaborado pelo autor.

Agora sim, é possível tirar algumas conclusões baseadas nas informações que estão disponíveis. Nota-se que a precisão e sensibilidade estão na faixa dos 80%, entretanto, é necessário definir qual métrica será considerada a mais importante no contexto em que está inserida. Sabendo que a precisão é a taxa de acertos sobre tudo aquilo que foi previsto e a sensibilidade é a taxa de

acertos sobretudo que de fato estava na iminência de falha, como escolher uma das métricas para avaliar?

A resposta é simples, entretanto é necessário pensar em como as métricas são calculadas.

Sabe-se, pelo item 1.7.3.2 que a Equação da sensibilidade é a quantidade de verdadeiros positivos sobre a quantidade de positivos reais, ou seja, verdadeiros positivos somados aos falsos negativos. Em contrapartida, o item 1.7.3.3 mostra que a precisão é a quantidade de verdadeiros positivos sobre todos os positivos previstos, ou seja, verdadeiros positivos somados aos falsos positivos.

Fica evidente que a única diferença entre as equações está no denominador, onde a sensibilidade usa a quantidade de falsos negativos e a precisão usa a quantidade de falsos positivos. Portanto, para saber qual das duas métricas é importante ao problema é necessário entender se é mais crítico reduzir falsos negativos ou reduzir falsos positivos.

Somente com essa questão ainda pode ser difícil de se imaginar, mas é possível contextualizar com este projeto. Ao reduzir falsos negativos significa reduzir a quantidade de máquinas que irão falhar, mas o modelo erra na previsão e reduzir os falsos positivos significa reduzir a quantidade de máquinas que eu o modelo prevê que irão falhar, mas não falham. Nessa linha, é necessário pensar no custo. O modelo errar em uma previsão de máquinas que irão falhar e ter a possibilidade de parar toda a produção ou o modelo prever que uma máquina irá falhar e a equipe de manutenção realizar uma ação sobre aquele ativo industrial em uma ou duas horas de trabalho?

Parar a produção é muito mais custoso do que mobilizar a equipe de manutenção para realizar uma manutenção preditiva em uma máquina. Ou seja, nesse projeto, o objetivo é reduzir a quantidade de falsos negativos, isso resulta em maximizar a sensibilidade.

Para maximizar a sensibilidade é necessário aumentar o número de previsões que resultam em 1, isso significa flexibilizar o limiar da matriz de confusão. A Figura 39 mostra como variam precisão e sensibilidade para os limiares de 0.45, 0.4 e 0.35 respectivamente.

Figura 39 - Variação de precisão e recall de acordo com o limiar da matriz de confusão.

```
print('Precisão: ', round(precision_score(df_avaliacao['target'], df_avaliacao['score_int_2'])*100, 2), '%')
print('Sensibilidade: ', round(recall_score(df_avaliacao['target'], df_avaliacao['score_int_2'])*100,2), '%')
✓ 0.0s

Precisão:  85.03 %
Sensibilidade:  84.8 %
```

```
print('Precisão: ', round(precision_score(df_avaliacao['target'], df_avaliacao['score_int_3'])*100, 2), '%')
print('Sensibilidade: ',round(recall_score(df_avaliacao['target'], df_avaliacao['score_int_3'])*100,2), '%')
✓ 0.0s

Precisão:  82.34 %
Sensibilidade:  88.27 %
```

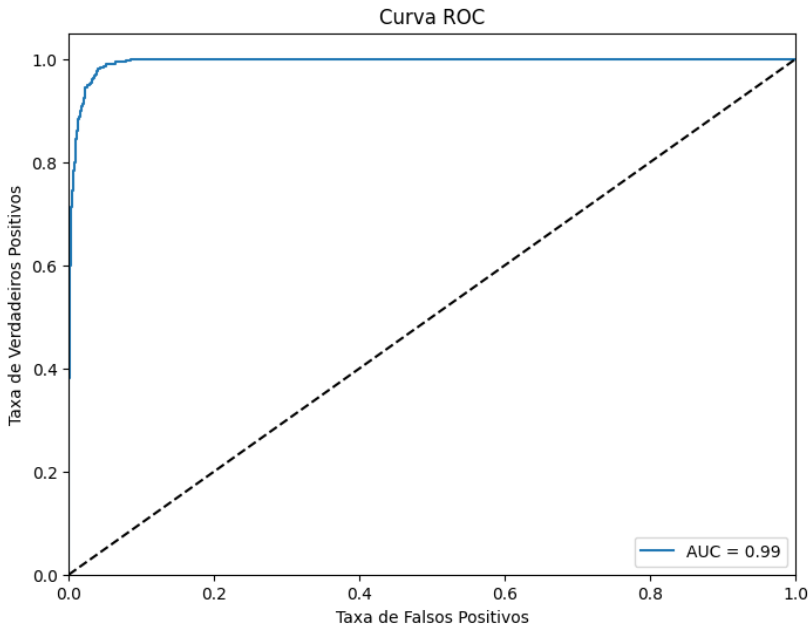
```
print('Precisão: ', round(precision_score(df_avaliacao['target'], df_avaliacao['score_int_4'])*100, 2), '%')
print('Sensibilidade: ',round(recall_score(df_avaliacao['target'], df_avaliacao['score_int_4'])*100,2), '%')
✓ 0.0s

Precisão:  78.03 %
Sensibilidade:  90.93 %
```

Fonte: Elaborado pelo autor.

Percebe-se então, a importância de definir qual métrica é mais importante para o projeto. Nesse caso, foi possível atingir uma sensibilidade de aproximadamente 91% utilizando um limiar de 0.35 nas previsões feitas pelo modelo.

A curva ROC foi plotada e está mostrada na Figura 40.

Figura 40 - Curva ROC.



Fonte: Elaborado pelo autor.

Nota-se que a curva ROC possui um excelente resultado, isso já era esperado devido à alta precisão que o modelo apresentou com os limiares testados, uma vez que a curva ROC relaciona a taxa de verdadeiros positivos com a taxa de falsos positivos. Isso significa que o modelo está performando bem e consegue distinguir com facilidade as máquinas que estarão na iminência de falha das que não estarão.

3.5.2 Aconselhamento à equipe de manutenção

Com os resultados em mãos, é necessário apresentá-los para a equipe de manutenção e, não somente isso, mas também aconselhá-los sobre qual a melhor maneira de utilizar tais resultados para chegar no objetivo do projeto priorizando o que for essencial.

A estratégia de priorização com base nas propensões à iminência de falha é fundamental para otimizar a agenda de manutenção. O modelo de aprendizado de máquina fornece propensões associadas a cada equipamento, permitindo uma classificação eficiente por risco. Ao priorizar os equipamentos com maiores probabilidades de falha, a equipe de manutenção pode direcionar seus esforços para onde são mais necessários, minimizando o risco de paradas não programadas.

O estabelecimento de ciclos de aferição de sensores alinhados com as especificações do modelo (15 ciclos) é crucial para garantir a precisão contínua das previsões. Além disso, a adaptação da frequência de inspeção com base na propensão de falha e na criticidade do equipamento permite uma abordagem mais proativa. Equipamentos com maior probabilidade de falha e maior impacto operacional podem exigir inspeções mais frequentes, enquanto aqueles com menor risco podem ter intervalos mais espaçados, otimizando os recursos da equipe.

A designação de técnicos especializados para equipamentos específicos é essencial. Isso permite que a equipe desenvolva um profundo conhecimento sobre as características técnicas e requisitos de manutenção de cada equipamento. A especialização aumenta a eficiência da equipe, acelerando diagnósticos e intervenções, resultando em tempos de inatividade reduzidos.

A constituição de uma equipe de resposta rápida é uma medida preventiva para falhas críticas. Equipamentos com alta criticidade, independentemente de sua probabilidade de falha, são monitorados de perto. A equipe de resposta rápida está pronta para intervir imediatamente, reduzindo o impacto operacional e mitigando possíveis danos associados a falhas inesperadas.

O fornecimento de treinamentos regulares é de extrema importância para manter a equipe atualizada. Isso inclui a familiarização contínua com as características dos equipamentos e a interpretação dos resultados do modelo. A formação contínua também permite que a equipe esteja preparada para as evoluções tecnológicas e ajustes no modelo preditivo.

A implementação de um ciclo de feedback contínuo relaciona a previsão do modelo e a realidade operacional. Informações de campo, como o desempenho real dos equipamentos, são incorporadas ao modelo regularmente. Isso melhora a capacidade do modelo de se adaptar a mudanças nas condições operacionais e de aprender com experiências passadas. Por isso, é de extrema importância a realização de testes no modelo em aferições dos sensores mais recentes em um intervalo de tempo definido. Isso evita que o modelo fique desatualizado e reduz a chance de erros nas previsões.

A geração de relatórios periódicos sobre o desempenho do modelo é necessária para avaliar sua eficácia. Destacar áreas de melhoria e sucessos permite à equipe ajustar continuamente suas estratégias. Relatórios transparentes também facilitam a comunicação com as partes interessadas, promovendo a confiança no uso do modelo para aprimorar a gestão de manutenção preditiva.

Em conjunto, essas estratégias formam uma abordagem abrangente para implementar e otimizar a manutenção preditiva em muitos equipamentos industriais sensorizados, destacando a importância da integração de tecnologia, especialização da equipe e adaptação contínua.

4 Conclusão

Este trabalho de conclusão de curso teve como objetivo aplicar conceitos e técnicas de aprendizado de máquina, ciência de dados e manutenção preditiva para desenvolver um modelo capaz de prever falhas em equipamentos industriais sensorizados com até 15 ciclos de aferição de antecedência. Utilizou-se dos conceitos de manutenção preditiva, buscando otimizar o momento da intervenção apenas quando necessário. O método CRISP-DM guiou as etapas do projeto, incluindo entendimento de negócio, entendimento dos dados, preparação, modelagem, avaliação e a etapa de implantação explorado somente no viés da equipe de manutenção por meio de aconselhamento sobre como utilizar as previsões do modelo e extrair o melhor resultado delas.

No pilar de entendimento de dados, foi necessário entender junto à equipe de manutenção todos os significados das variáveis disponíveis na base, qual a expectativa para o projeto e qual a resposta que é esperada pela equipe. Durante o entendimento de dados, foi encontrada uma dificuldade em definir os nomes das variáveis para cada sensor e equipamento de configuração, uma vez que essas variáveis não vieram nominadas na base de dados original.

Durante a etapa de preparação dos dados, foram conferidas todas as possibilidades de transformação, criação ou exclusão de variáveis e, devido à confiabilidade e qualidade das informações fornecidas na base de dados, essas transformações não foram necessárias. Posteriormente, a seleção de variáveis mais importantes foi realizada utilizando o método *Embedding* pelo próprio algoritmo de Florestas Aleatórias, tornando o modelo mais leve, reduzindo o tempo de treinamento e melhorando suas métricas de avaliação.

Conjuntamente ao treinamento, foram realizados os ajustes nos hiperparâmetros, buscando otimizar o desempenho de cada algoritmo. Para isso utilizou-se da função *GridSearchCV* da biblioteca *Scikit-Learn* que além de testar todas as possibilidades de hiperparâmetros também realiza a validação cruzada utilizando o método do *K-Fold Cross Validation*. A avaliação dos modelos foi realizada focando no valor de sensibilidade, considerando o contexto de manutenção preditiva, em que prever falhas é crítico e o custo de manutenção desnecessária é menor do que o custo de uma falha.

Os resultados obtidos na base de teste utilizando o algoritmo de florestas aleatórias, especialmente na métrica de sensibilidade, mostraram que o modelo performa muito bem, uma vez que foi possível maximizar essa métrica reduzindo o limiar condicional para a falha, obtendo valores de 90% de sensibilidade e 70% de precisão, desta forma é possível prever os equipamentos na iminência da falha, facilitando assim a atuação da equipe de manutenção focando especificamente em integração de tecnologia, especialização da equipe e adaptação contínua.

REFERÊNCIAS

[1] ACHOUCH, M.; DIMITROVA, M.; ZIANE, K.; SATTARPANAH K.; DHOUIB, R.; IBRAHIM, H.; ADDA, M. On Predictive Maintenance in Industry 4.0: Overview, Models, and Challenges. Applied Sciences, Vol. 12, No. 8081, 2022.

[2] CINAR, Z. M.; NUHU, A. A.; ZEESHAN, Q.; KORHAN, O.; ASMAEL, M.; SAFAEI, B. Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0, Sustainability, Vol 12, No. 8211, 2020.

[3] SOUSA, Evandro F. Data Science – Um panorama geral, 2018. Disponível em: https://medium.com/trainingcenter/data-science-um-panorama-geral-87edbbd35885. Acesso em: 07 out 2023.

[4] GÉRON, A. **Mãos à Obra: Aprendizado de máquina com Scikit-Learn e TensorFlow: Conceitos, ferramentas e técnicas para a construção de sistemas inteligentes**. Rio de Janeiro, RJ: O*'*'Reilly, 2019.

[5] JAMES, G.; WITTEN, D.; HASTIE, T.; TIBISHIRANI, R. **An Introduction to Statistical Learning : with Applications in R**. New York :Springer, 2013.

[6] BRUCE, P.; BRUCE, A**. Estatística Prática para Cientistas de Dados: 50 conceitos essenciais**. Rio de Janeiro, RJ: O*'*'Reilly, 2019.

[7] BITTENCOURT, H. Regressão Logística Politômica: revisão teórica e aplicações. ACTASCIENTIAE, Vol. 05, No. 01, p.p. 77-86, 2003.

[8] PROVOST, F.; FAWCETT, T. **Data Science para negócios**. Rio de Janeiro, RJ: Alta Books, 2016.

[9] MORETTIN, P.; BUSSAB, W. **Estatística Básica**, 6.ed. São Paulo, SP: Saraiva, 2010.

[10] Brownlee, J. An Introduction to Feature Selection. Machine Learning Mastery, 2021. Disponível em: https://machinelearningmastery.com/an-introduction-to-feature-selection/. Acesso em: 18 out 2023.

[11] Mastrandrea, G. Correlation Matrix, Demystified, 2022. Disponível em: https://towardsdatascience.com/correlation-matrix-demystified-3ae3405c86c1 . Acesso em: 18 out 2023.

[12] Khan, S. What is K-fold Cross Validation? Working of Cross validation on K-fold data, 2023. Disponível em: https://towardsmachinelearning.blogspot.com/2023/02/what-is-k-fold-cross-validation-working.html. Acesso em: 21 out 2023.

[13] GARCIA, N. Indústria 4.0: 69% das indústrias brasileiras fazem uso de tecnologia digital. Portal da Indústria, 2022. 26/04/2022. Disponível em: https://noticias.portaldaindustria.com.br/noticias/inovacao-e-tecnologia/industria-40-69-

das-industrias-brasileiras-fazem-uso-de-tecnologia-digital-no-brasil/. Acesso em: 20 dez 2023

[14] ALMEIDA, P. S. d. **Manutenção mecânica industrial: princípios técnicos e operações**. 1. ed. São Paulo: Érica, 152 p., 2015.

[15] SOUZA, V. **Organização e gerência da manutenção**. 2° ed. Cidade: All Print, 2007.

[16] SHEARER, C. **The CRISP-DM model: The new blueprint for data mining**. Journal of Data Warehousing, Vol. 5, No. 4, pp. 13-22, 2000.

APÊNDICE A

Link para o Github onde está localizado o notebook do projeto e a base de dados utilizada: https://github.com/muleite96/manutencao_preditiva.git . Acessado em 05 de dezembro de 2023.