INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA
CURSO SUPERIOR DE ENGENHARIA MECATRÔNICA

PEDRO HENRIQUE BUNN SCHMITT

APLICAÇÃO DE ALGORÍTMOS DO *AUTOMATED MACHINE LEARNING* EM UM CENÁRIO RELEVANTE

FLORIANÓPOLIS, 2022.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA
CURASO SUPERIOR DE ENGENHARIA MECATRÔNICA

PEDRO HENRIQUE BUNN SCHMITT

APLICAÇÃO DE ALGORÍTMOS DE *AUTOMATED MACHINE LEARNING* EM UM CENÁRIO RELEVANTE

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para a obtenção do título de Engenheiro Mecatrônico.

Orientador:
Prof. Me. Gregory Chagas da Costa Gomes

FLORIANÓPOLIS, 2022.

Ficha de identificação da obra elaborada pelo autor.

SCHMITT, PEDRO HENRIQUE BUNN
APLICAÇÃO DE ALGORÍTMOS DO AUTOMATED MACHINE LEARNING EM UM CENÁRIO RELEVANTE / PEDRO HENRIQUE BUNN SCHMITT; orientação de Gregory Chagas da Costa Gomes. - Florianópolis, SC, 2022.
72 p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal de Santa Catarina, Câmpus Florianópolis. Bacharelado em Engenharia Mecatrônica. Departamento Acadêmico de Metal Mecânica.
Inclui Referências.

1. Machine Learning. 2. Automated Machine Learning. 3. Predição de Temperatura. I. Gomes, Gregory Chagas da Costa . II. Instituto Federal de Santa Catarina. III. APLICAÇÃO DE ALGORÍTMOS DO AUTOMATED MACHINE LEARNING EM UM CENÁRIO RELEVANTE.

APLICAÇÃO DE ALGORÍTMOS DE *AUTOMATED MACHINE LEARNING* EM UM CENÁRIO RELEVANTE

PEDRO HENRIQUE BUNN SCHMITT

Este trabalho foi julgado adequado para obtenção do título em Bacharel em Engenharia Mecatrônica pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina e aprovado na sua forma final pela comissão avaliadora do Curso de Engenharia Mecatrônica.

Florianópolis, 18 de julho de 2022.

Banca Examinadora:

IFSC
Documento assinado digitalmente
GREGORY CHAGAS DA COSTA GOMES\;gregory.c
Data: 23/08/2022 15:18:16-0300
CPF: 073.008.969-01
Verifique as assinaturas em https://v.ifsc.edu.br

_____

Prof. Msc. Gregory Chagas da Costa Gomes
Orientador
Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

gov.br
Documento assinado digitalmente
Mauricio Edgar Stivanello
Data: 19/08/2022 09:34:26-0300
Verifique em https://verificador.iti.br

_____

Prof. Mauricio Edgar Stivanello, Dr.
Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

IFSC
Documento assinado digitalmente
MARIO DE NORONHA NETO
Data: 24/08/2022 08:22:00-0300
CPF: 003.859.519-22
Verifique as assinaturas em https://v.ifsc.edu.br

_____

Prof. Mario de Noronha Neto, Dr.
Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

AGRADECIMENTOS

Agradeço a minha grande família por estar sempre comigo em minhas decisões e ter me oferecido todo o suporte necessário para que eu pudesse fazer este trabalho.

Agradeço ao IFSC, aos servidores e professores que me guiaram e trocaram conhecimento comigo. Agradeço em especial ao meu orientador, Prof. Gregory que mesmo nas noites mal dormidas me orientou para que eu conseguisse criar e escrever este trabalho.

Agradeço aos meus amigos desta e nesta jornada que me fizeram refletir e trocar ideias tanto nas carteiras das salas do IFSC quanto nas mesas da Hercílio Luz.

Agradeço em geral a todos que tornaram possível a apresentação e criação deste trabalho.

Tudo claro
Ainda não era o dia
Era apenas o raio.
(Paulo Leminski)

RESUMO

O conceito de inteligência artificial consiste no desenvolvimento de sistema computadorizados que realizam tarefas análogas a inteligência humana. Oficializado em 1959, o termo toma relevância no mundo da tecnologia e dele surgem inúmeras ramificações, entre elas a do *Machine Learning* e posteriormente o *Automated Machine Learning.* Este trabalho teve como objetivo experienciar a utilização do AutoML (*Automated Machine Learning)* em uma aplicação de predição da temperatura de rotores de um conjunto de dados de motores elétricos. Apresentou-se os conceitos de inteligência artificial e suas ramificações, os de modelos de aprendizado do *Machine Learning*, quais sejam, Modelo de Aprendizado Supervisionado, Não Supervisionado e Reforçado. Foram vistos artigos que traziam comparações entre métodos tradicionais como D*eep Learning e Machine Learning,* em comparação ao presente método de estudo nas áreas de saúde e meio ambiente. Após a apresentação do tema e uma pequena contextualização sobre o campo de atuação da área, foi demonstrado qual o caminho percorrido até o encontro do modelo para predição da temperatura do rotor, evidenciando de modo gradual os programas, bibliotecas e códigos utilizados. Por fim, foi feita apresentação e comparação dos resultados do algoritmo encontrado em relação aos resultados disponíveis na plataforma *Kaggle.* O objetivo foi apresentar como foi realizado a criação de modelos de A*utomated Machine Learning* e a aplicação do mesmo para um conjunto de dados foi plenamente atingido através da produção e publicação dos *Notebooks* e este foi alcançado. Nesta pesquisa foi encontrado um algoritmo de *Machine Learning* para o conjunto de dados de motores elétricos capaz de fazer predições da temperatura dos rotores apresentando assim resultados promissores. No entanto, entende-se necessárias outras pesquisas buscando adquirir conhecimento e aprofundamento no assunto.

Palavras-Chave: *Machine Learning*. *Automated Machine Learning*. Predição de Temperatura.

ABSTRACT

The concept of artificial intelligence consists in the development of computerized systems that perform tasks analogous to human intelligence. Officialized in 1959, the term takes on relevance in the world of technology and from it arise numerous ramifications, including Machine Learning and later Automated Machine Learning. This work aimed to experience the use of AutoML (Au-tomated Machine Learning) in an application to predict the rotor temperature of a dataset of electric motors. The concepts of artificial intelligence and its ramifications, the Machine Le-arning learning models were presented, namely, Supervised, Unsupervised and Reinforced Learning Model. Articles were seen that brought comparisons between traditional methods such as Deep Learning and Machine Learning, compared to the present method of study in the areas of health and the environment. After the presentation of the theme and a little contextualization about the field of action of the area, it was demonstrated the path taken to find the model for predicting the temperature of the rotor, gradually showing the programs, libraries and codes used. Finally, the results of the algorithm found were presented and compared with the results available on the Kaggle platform. The objective was to present how the creation of Automated Machine Learning models was carried out and its application to a dataset was fully achieved through the production and publication of Notebooks and this was achieved. However, it is understood that further discussions and more research time would be needed that go beyond this work to acquire greater knowledge on the subject and reliability in the results found.

Keywords: *Machine Learning. Automated Machine Learning. Temperature Prediction.*

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Com o conhecimento adquirido nas últimas décadas, nosso planeta produziu uma grande quantidade de dados e informações em que o computador, seus softwares e hardwares foram imprescindíveis para o aprofundamento deste conhecimento. Desta forma se apresentou necessária a criação de ferramentas computacionais mais sofisticadas e autônomas que conseguissem gerir e utilizar dados. As máquinas começaram a aprender com suas experiências passadas, para gerar soluções de problemas, análise de comportamentos de consumidores, predição de valores dos mais diversos campos a partir de grupos de dados entre tantos outros casos de aplicação. Com esta proposta surge o campo do *Machine Learning,* também conhecido como aprendizado de máquina (FACELI, 2011; PETROVA, 2022). O Automated Machine Learning - AutoML, atua neste campo facilitando a parametrização dos modelos de Deep Learning a fim de não ser mais necessário a expertise de um usuário para "configuração do mesmo" (HE; ZHAO; CHU, 2021).

Para que seja possível a utilização do AutoML, é necessário entender o significam os termos *Machine Learning e Deep Learning.* O primeiro termo refere-se a um ramo das ciências da computação em que um computador é treinado, por meio de algoritmos e um conjunto de dados, para tomar decisões. Nestes tipos de modelo, normalmente existem algumas tarefas padrões como classificação, regressão, recomendação, *ranking* e *clustering*. O segundo termo surge, como um subcampo do *Machine Learning,* quando a extração dos dados não é de tão fácil processamento. Salvaris, Dean e Tok (2018) colocam que o *Deep Learning* se mostra como uma grande força nestes tipos de problemas como na extração de dados de imagens, áudios e textos. É dito que o crescimento desta área advém do crescente poder computacional pelo *cloud computing,* visto que se pode contratar mais recursos ou suspendê-los a medida da necessidade de poder computacional sem necessidade de investir em um *hardware* com mais poder de processamento, e do aumento da digitalização mundial com uma quantidade massiva de dados coletados (SALVARIS; DEAN; TOK, 2018).

O tema abordado neste trabalho possui uma gama de aplicações em diversas áreas, sendo apresentado como um grande ajudante para predições e estudos de caso de aplicações, como pode ser visto no trabalho feito sobre análise de água no rio Korattur, em Chennai na Índia. Neste estudo os autores compararam os

resultados entre dois algoritmos, um criado em *Automated Machine Learning,* que teve por objetivo automatizar a criação de algoritmos de *Deep Learning* fazendo com que não fosse necessária a assistência humana na escolha de parâmetros para o modelo e o outro modelo em *Machine Learning.* A finalidade deste trabalho citado apresentou o quanto de praticidade e segurança ocorre com a utilização de um modelo de AutoML para uma aplicação real e que a acurácia dos resultados foi maior quando utilizado as ferramentas de autoML em relação aos métodos tradicionais de *Machine Learning* (VENKATA *et al.*, 2021).

Outro trabalho que se destaca atua na predição de casos de depressão e ansiedade onde são feitas comparações entre três modelos, o primeiro sendo análise de regressão linear, o segundo como *Machine Learning* e o terceiro como *Automated Machine Learning.* Os autores verificaram que o último modelo foi o que trouxe resultados mais promissores quando acessado um *dataset* mais complexo com valores individuais (EEDEN *et al.*, 2021). Ainda no campo da saúde, existe um trabalho em que foi criado um algoritmo em AutoML para predizer a situação de risco de pacientes sofrerem ataques cardíacos (BANGASH, 2021).

Desta forma, o *Automated Machine Learning* pode ser empregado em várias frentes, demonstrando relevância crescente dentro das empresas com o interesse comercial em ascensão onde cada uma delas quer criar seu próprio algoritmo AutoML e pela ferramenta trazer a possibilidade da democratização do *Machine Learning* para não *experts* da área (HUTTER; KOTTHOFF; VANSCHOREN, 2019).

O Curso de Engenharia Mecatrônica do Instituto Federal de Santa Catarina tem em seu currículo diferentes disciplinas com conteúdos envolvendo engenharia da informática, engenharia da eletricidade, engenharia de controle e engenharia mecânica assim estabelecendo “os quatro pilares dos sistemas de automação atuais: sistemas mecânicos, eletrônica, controle e informática”. (SANTA CATARINA, 2012).

Durante o curso, disciplinas trouxeram conteúdos de sistemas de informática com foco na programação, os quais chamaram especial atenção do pesquisador, levando ao aceite da proposta realizada pelo orientador para a execução de um trabalho utilizando AutoML, sendo escolhido um grupo de dados de motores elétricos, onde seria predito qual a temperatura dos rotores. Desta forma este trabalho junta disciplinas vistas no curso, o desafio de explorar o campo da ciência de dados e o exercício da engenharia mecatrônica.

1.1 Objetivos

1.1.1 Objetivo Geral

- Aplicar e documentar o processo de utilização de algoritmos de *Automated Machine Learning*-AutoML para predizer a temperatura do rotor de motores elétricos utilizando uma base de dados pública.

1.1.2 Objetivos Específicos

- Utilizar uma base de dados pública que tenha relevância no contexto do curso de engenharia mecatrônica;
- Aplicar ferramentas de AutoML para geração de modelos de predição;
- Utilizar os ambientes e ferramentas de programação largamente utilizados pela comunidade de desenvolvimento na área de ciência de dados;
- Avaliar os resultados através da aplicação dos modelos gerados na base selecionada;
- Documentar o processo de desenvolvimento de forma pública e didática.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a fundamentação teórica da experiência realizada, a metodologia utilizada e seus passos, os resultados obtidos e a análise.

2.1 Inteligência Artificial e suas ramificações

A Inteligência Artificial se mostra um assunto amplamente discutido no campo acadêmico e de dados, fato visto ao ser realizada uma pesquisa pelo termo "*Artificial Intelligence"* em um banco de dados de publicações acadêmicas, no caso, o "Science Direct", encontramos documentos e publicações demostrando que o campo de estudo de inteligência artificial não é tão novo quanto se pode pensar.

O termo Inteligência Artificial- IA foi oficializado em 1959 na Faculdade de Dartmouth- Darmouth College- EUA, onde alguns dos maiores experts da área se reuniram para fazer uma discussão sobre "Inteligência Simulada". A pesquisa foi fomentada após Asimov escrever sobre as três leis da robótica e após Turing fazer uma publicação em 1950 onde ele trazia a proposta de que máquinas poderiam "pensar". Após este fato, houveram grandes investimentos financeiros pelo governo destinados para a área de criação de uma inteligência não humana (COREA, 2019).

Para Corea (2019), ao longo dos anos, pesquisas foram sendo desenvolvidas e com avanço das tecnologias de processamento, a inteligência artificial foi tomando campo e adquirindo credibilidade. Um evento que foi um gatilho para o aumento da tendência sobre o tema aconteceu em 4 de dezembro de 2012, onde um grupo de pesquisadores apresentou na conferência NIPS (Processamento de Sistemas de Informações Neurais) um estudo sobre redes de convoluções neurais que os fez ganhar o primeiro lugar no campeonato sobre classificação de imagens *ImageNet Classification.*

Outros três marcos contribuíram para o desenvolvimento da Inteligência Artificial sendo o primeiro deles a aquisição, em 2014, da empresa DeepMind, a qual possui foco em pesquisa e desenvolvimento na área de inteligência artificial, pela gigante da Tecnologia Google (DEEPMIND, 2022). Um segundo ponto consistiu em uma carta aberta do Instituto *Future of Life,* colocando a importância da IA para o futuro da humanidade, assinada por mais de oito mil pessoas dos mais diversos

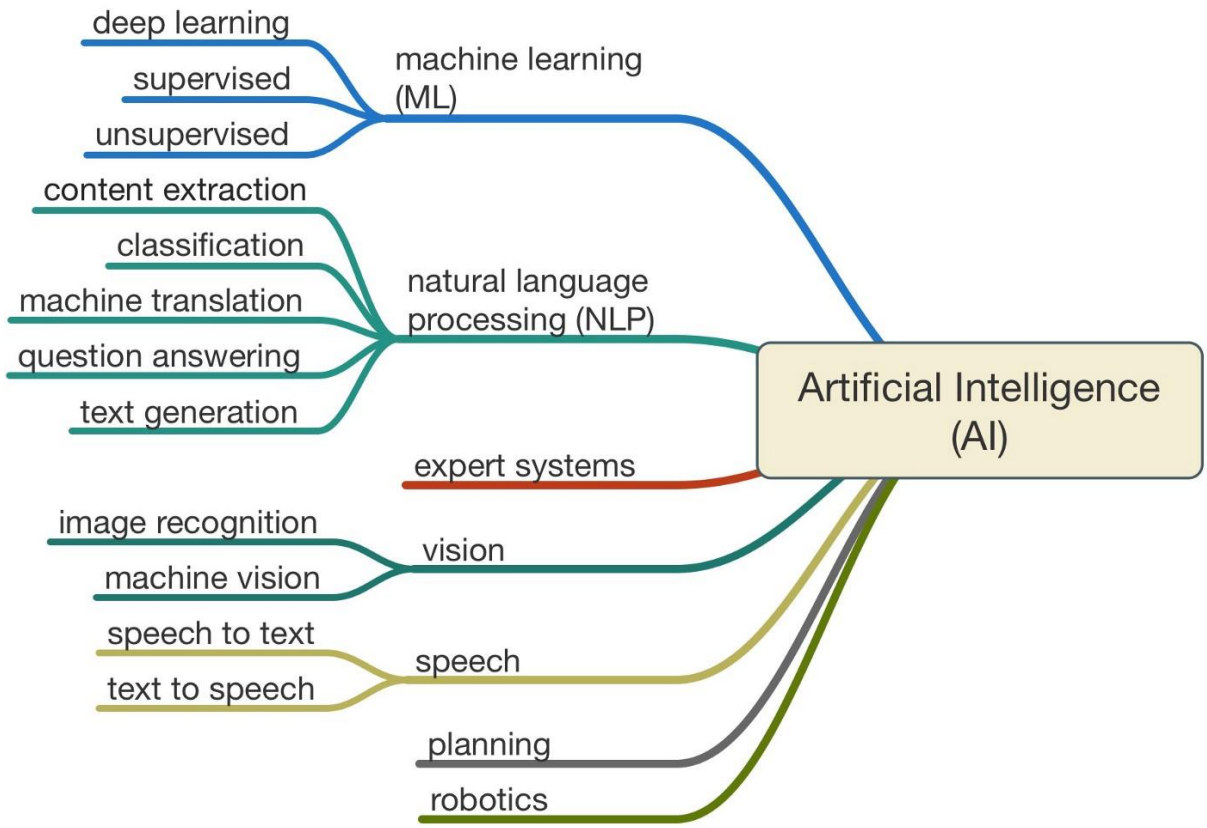
setores (FUTURE OF LIFE INSTITUTE, 2014) e por fim, o lançamento de uma matéria pela revista *Nature* sobre a vitória da IA, desenvolvida pela empresa *DeepMind,* no jogo *Go* sobre o jogador Lee Sedol, um dos melhores do mundo (COREA, 2019).

Após ser feita uma breve contextualização de como a IA foi tomando relevância no mundo da tecnologia, são apresentadas algumas definições sobre como ela funciona.

Para *The Organization for Economic Cooperation and Development*- OEDC, a IA é um sistema que possui a capacidade de imitar a inteligência humana sendo predefinido para fazer predições, recomendações e decisões sobre dados e estes, podendo influenciar em ambientes reais ou virtuais (STAHL, 2021). Uma segunda definição feita por John McCarthy traz que a IA seria a ciência e a engenharia de fazer máquinas inteligentes (SINGH; MISHRA; SAGAR, 2013). A terceira definição colocada pelo dicionário Oxford University Press (2022) coloca que se trata do desenvolvimento de sistemas computadorizados que conseguem realizar tarefas que requerem a inteligência humana como entendimento da linguagem falada, percepção visual, tomada de decisões.

Com algumas definições feitas sobre o campo da Inteligência artificial, categorias e subcategorias que abrangem o escopo do tema são apresentadas.

Figura 1 - Áreas de atuação da Inteligência Artificial



Fonte: Kumar (2018).

É possível observar pela Figura 1 que existem sete ramificações de categorias que partem do bloco da Inteligência Artificial, porém as de maior destaque, informadas pelo autor são a *NLP, Vision, Robotics e Machine Learning (*KUMAR*,* 2018*).*

A *NLP* tem a função de trazer ao computador a habilidade de entender a linguagem falada e escrita pelos humanos, a partir da combinação da linguagem computacional com estatística, *Machine Learning* e *Deep Learning* (IBM, 2020a). O *Vision* traz a visão computacional, a qual é apresentada como uma das ramificações do campo da IA que permite, a partir de imagens, vídeos e outras entradas visuais, tomar ações e fazer recomendações baseadas nessas informações. Existe uma definição interessante colocada pela IBM onde apresenta que a IA faz com que o computador pense e a visão computacional faz com que ele possa ver, entender e observar (IBM, 2022). A categoria *Robotics* é apresentada como o campo da robótica, em que o foco é dado na criação de robôs que executam tarefas difíceis, perigosas ou repetitivas para os humanos como trabalho em hospitais, limpeza, agricultura entre outros.

Após ser feita uma breve revisão das ramificações que surgem a partir da inteligência artificial, será dado enfoque na área do *Machine Learning* por ser o nosso objeto de estudo deste trabalho.

2.1.1 Machine Learning

Machine Learning-MI pode ser definido como o estudo da forma na qual os agentes computacionais podem realizar ações. Entende-se por agentes computacionais aqueles que por meio do ambiente através de seus sensores aperfeiçoam suas percepções e seus conhecimentos para tomarem decisões baseados em experiências e dados (RUSSEL; NORVIG, 2003).

A área do *Machine Learning* é uma das que mais cresce no campo das ciências da computação pois, tanto a quantidade de dados quanto a forma como eles são processados e transformados em conhecimento, aumenta de forma exponencial ao longo dos anos. Outro ponto importante é ressaltar a quantidade de tipos de dados que os algoritmos da área começaram a conseguir tratar ao longo do avanço da tecnologia, como: páginas *web*, arquivos de áudio, vídeos, gráficos entre tantos outros (ALPAYDIN, 2014). Com estas ponderações feitas, qual a definição de *Machine Learning*? De acordo com *Samuel (1959) apud Géron* (2017, p. 20) "*Machine Learning* é a arte de dar ao computador a habilidade de aprender a programar sem ser necessário explicitamente programar em linhas de código" ou por outra explicação mais técnica dada por Alpaydin *(*2014*),* ML utiliza a estatística para a criação de modelos matemáticos, os quais por meio de um algoritmo, aprendem com dados de treino ou experiências passadas e podem fazer predições, sendo estes modelos preditivos, ou que possuem o papel de "ganhar" conhecimento e descrever os dados passados para ele, sendo estes modelos descritivos, ou ainda pode atuar como uma junção destes dois tipos (ALPAYDIN, 2014). Então, pelo que pode ser visto, o método descrito é utilizado com a finalidade de fazer a máquina "aprender" por conta própria através de um algoritmo. Porém, existe algum benefício nisso?

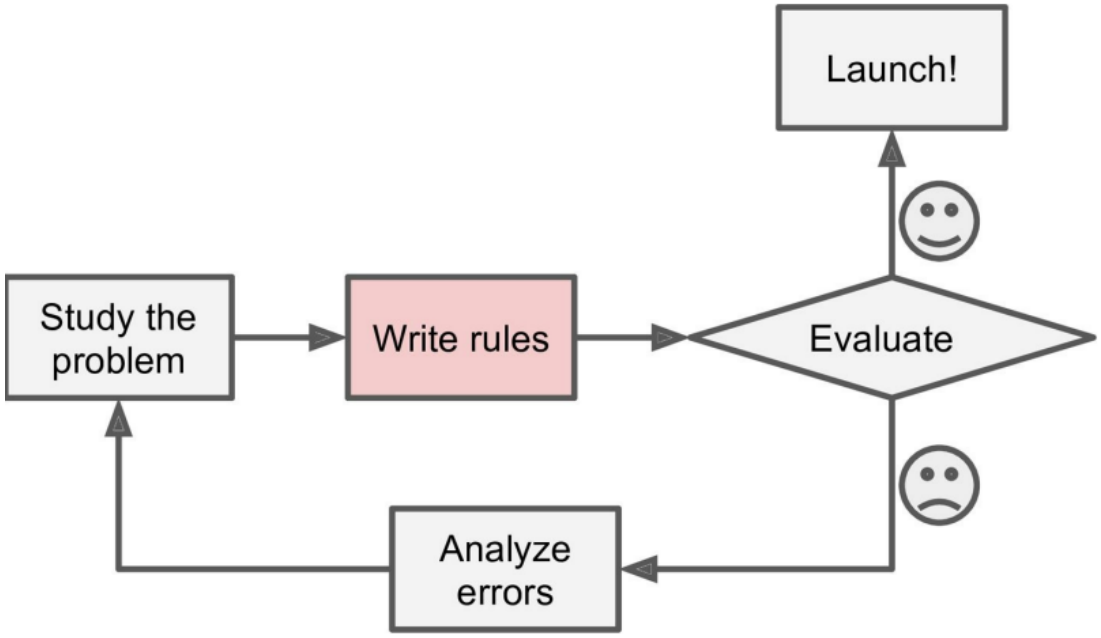
Para entender a utilização do ML, podemos usar como exemplo do controle de spam de um e-mail. Uma mensagem de correio eletrônico necessita passar por inúmeros filtros de palavras e endereços eletrônicos até que seja direcionada para a caixa de spam ou para a lixeira. Com isto em mente, a ideia de criar regras manuais

de programação para direcionar cada nova mensagem se torna inviável e pouco preciso, visto as inúmeras especificações de palavras chaves que precisam ser criadas diariamente. Com o uso do ML isto não é necessário. A partir da criação de um modelo preciso e bem organizado novas palavras seriam adicionadas automaticamente e o controle de spam possuiria mais acurácia e ainda uma baixa necessidade de manutenção.

O exemplo anterior apresentado foi em relação aos e-mails, porém poderia ser pensado em um algoritmo de predição de preços de casas, aplicativos de transporte como a *Uber,* sites de vendas de produtos no momento da busca de produtos como *Amazon* (INTEROP, 2019). Todos estes serviços apresentam, por *baixo dos panos*, um sistema de *Machine Learning* fazendo o gerenciamento de informações, predizendo valores e gerando recomendações a partir desta análise sobre esta imensa quantidade de dados. Estes dados que antes possuíam valor, porém, que não eram utilizados da melhor forma hoje são minerados e analisados gerando informações úteis e de alta valia para os negócios graças ao *Machine Learning* (GÉRON, 2017).

Voltando ao exemplo do controle de *spam* no correio eletrônico , Géron (2017) faz a comparação entre duas formas de resolver o problema, a forma tradicional com a programação sendo descrita pelas regras em que o programador faz a anotação de quais palavras tem mais tendência em aparecer nestas mensagens como: *“free”, “credit card”, “amazing”* ou alguma formatação de corpo de e-mail ou outro padrão na mensagem e faz a escrita do programa para que, caso um destes padrões apareça, seja feita o controle destas mensagens como desejado. Apresentamos na Figura 2 um esquema que representa o tratamento de dados no método tradicional.

Figura 2 - Método tradicional de tratamento de dados

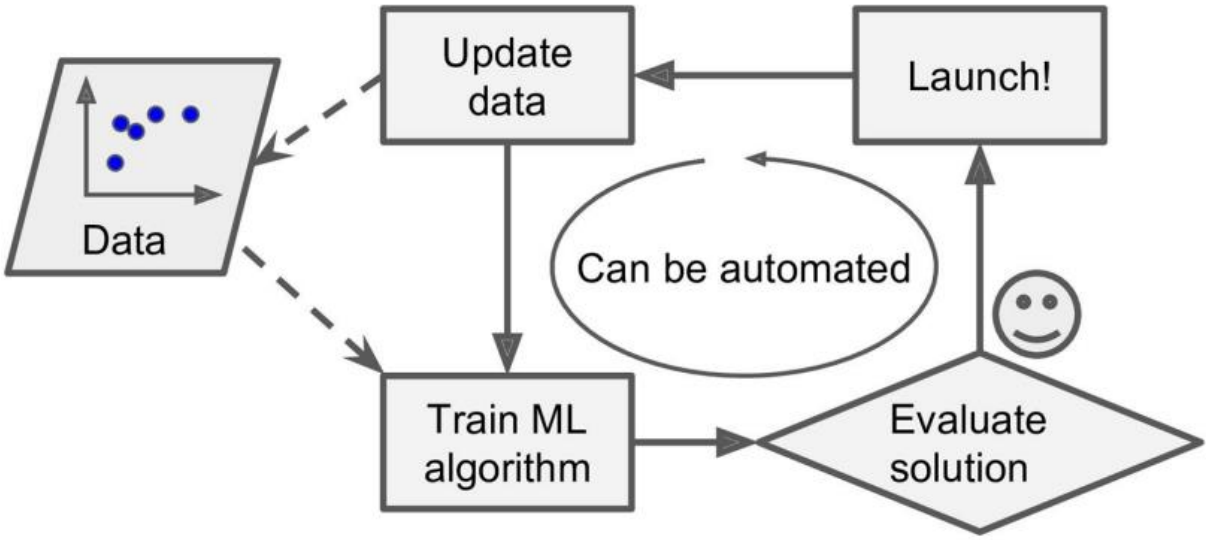


Fonte: Géron (2017, p. 21).

Como pode ser visualizado na Figura 2 após o estudo do problema são definidas uma série de regras para o modelo, a avaliação destas regras e realizado o tratamento para aquela informação trazida pelo modelo. Se o caminho traçado pelo *software* foi o desejado, o programa pode ser rodado, se algo foi diferente do esperado, é necessário fazer novamente o estudo do problema e reescrever as regras propostas até que seja obtido um programa com aplicação satisfatória. Porém existe o problema da manutenção deste programa pois, à medida que novas palavras e novos tipos de mensagens são adicionados, é necessário fazer um novo estudo de caso e tratamento dessas novas informações tornando a manutenção deste muito trabalhosa (GÉRON, 2017).

Em contraponto, o autor apresenta o modelo baseado em técnicas de ML, as quais tem o poder de aprender automaticamente quais palavras e frases são utilizadas para o controle de spam por meio de um treinamento do algoritmo construído. Segue abaixo a figura com o modelo explicitado.

Figura 3 - Modelo de adaptação automática com Machine Learning



Fonte: Géron (2017, p. 23).

Ao analisar a Figura 3, percebemos que no modelo com ML há automatização na forma como o computador aprende novos dados e os avalia. Esta forma de programar o cenário se apresenta mais rápida no momento de sua criação, não há necessidade de explicitar todas as regras de atuação do conjunto para o filtro de *spam,* pois o mesmo tem a autonomia de entender novos dados e os adicionar em seu filtro e, devido a este fato, também não é necessário implementar novas regras para o algoritmo como seria feito no modelo tradicional (GÉRON, 2017).

Com uma breve introdução sobre o que significa o termo *Machine Learning* e como ele pode trazer benefícios em relação aos métodos tradicionais, será visto como se dá a análise dos dados, desde sua extração até a criação de um modelo, a fim de apresentar as complexidades e facilidades que o algoritmo traz. Neste primeiro momento, serão apresentados para classificação três tipos de modelos de aprendizados: Modelo de Aprendizado Supervisionado, Modelo de Aprendizado Não Supervisionado e Modelo de Aprendizado Reforçado (RASCHKA, 2015).

2.1.1.1 Modelo de Aprendizado Supervisionado

No modelo supervisionado de aprendizado o objetivo principal é aprender com os dados já rotulados, mais conhecidos como *labeled*, e fazer predições a partir destes com dados ainda não processados ou não utilizados. Raschka (2015) utiliza como referência para exemplificar este modelo, o exemplo do *spam* dado na

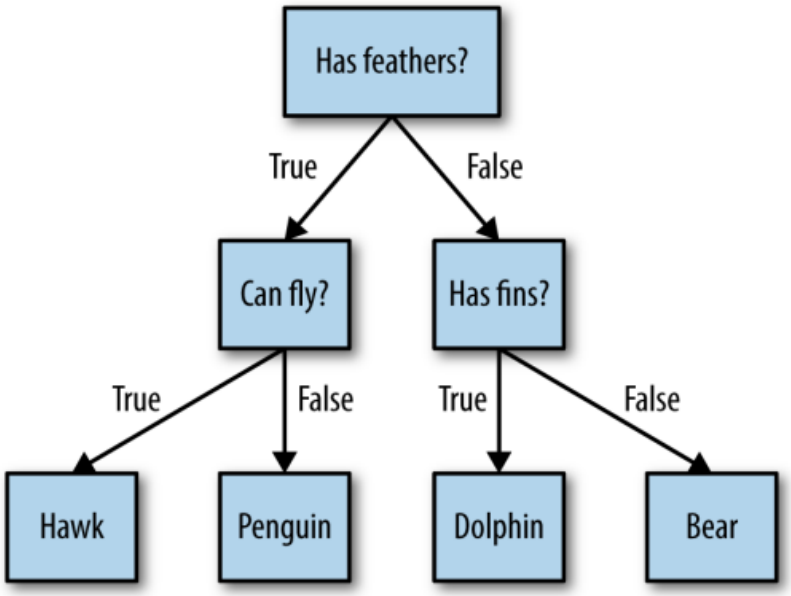
explicação da Figura 3 onde o algoritmo aprende com os dados rotulados e a partir dele faz novas predições para e-mails ainda não vistos os classificando como *spams* ou não. Ainda dentro do modelo supervisionado existem duas subcategorias, são elas: classificação e regressão.

O aprendizado supervisionado por classificação é o exemplificado anteriormente na filtragem dos e-mails que são *spams* ou não, onde temos uma resposta como sim ou não. Neste caso, existem alguns algoritmos que são mais utilizados para fazer esta classificação como: Árvore de Decisão, *Naive Bayes Classification* ou *Support Vector Machines* (SVM) (KUMAR, 2020).

A seguir são apresentados cada um dos algoritmos citados:

O algoritmo da Árvore de Decisão é utilizado por sua capacidade de transparência em relação à representação de suas regras. Ele é organizado em forma de árvore, como o nome sugere e possui nós, que são os retângulos apresentados na Figura 4, os quais trazem perguntas do tipo “se e se não”, amplamente conhecidas na programação como “*if* e *else”*, que atuam na divisão dos dados em grupo menores, essas divisões são chamadas de ramos. Também existe a raiz, que é apresentada como começo da árvore ou ponto de partida e o nó folha que é conhecido como o ponto final da mesma (MÜLLER; GUIDO, 2017).

Figura 4 - Modelo da árvore de decisão da classificação de animais



Fonte: Müller; Guido (2017, p. 85).

No exemplo a Árvore de Decisão realiza a classificação de alguns animais.

Ela começa com a primeira pergunta sendo se o animal possui penas, se ele possuir é feita a pergunta se ele consegue voar. Se ele não possui penas, é feita a pergunta se ele possui barbatanas. Se alguma das alternativas referentes às perguntas anteriores foram sim ou não, o animal é especificado como sendo uma águia, pinguim, golfinho ou urso. O exemplo é bem simples, porém, passa a ideia da hierarquia existente no modelo e que, a cada vez que é feita uma escolha, o conjunto de dados é dividido diminuindo o número de dados sucessivamente até ser encontrado o conjunto ou resultado alvo (MÜLLER; GUIDO, 2017).

Porém, no momento da construção de uma Árvore de Decisão, existem parâmetros principais que são necessários serem configurados a fim de se obter bons resultados. O primeiro deles atua no momento do cálculo da impureza da árvore onde é escolhido o método o qual este cálculo será feito podendo ser ele por entropia ou *gini*. A máxima profundidade da árvore é o segundo parâmetro e indica quão “funda” a árvore pode ir, sendo que quanto mais profunda a árvore, mais ramificações ela possuirá. O terceiro é a quantidade mínima de amostras para fazer uma divisão de um nó. A variação deste parâmetro pode ir de 10% até 100% e, à medida que a árvore aumenta se torna mais “embaraçado” ou denso, por precisar considerar mais amostras a cada nó e a quarta e última se apresenta como o máximo número de características para considerar em uma divisão de um nó (FRAJ, 2017).

O *Naive Bayes Classification* se apresenta como um método que pode ser utilizado para grandes grupos de dados por ser rápido e possuir poucos parâmetros para ajustes. Eles são apresentados como um bom ponto de partida no momento da criação de um modelo de *Machine Learning.* O funcionamento deste método é baseado no Teorema de Bayes, o qual possui como ideia inverter as probabilidades condicionais. Seria como descobrir a probabilidade que o evento A pode acontecer a partir do evento B, P(A|B), porém o que conhecido é exatamente o contrário, a probabilidade de B acontecer a partir de A, P (B|A) (OLIVEIRA, 2020).

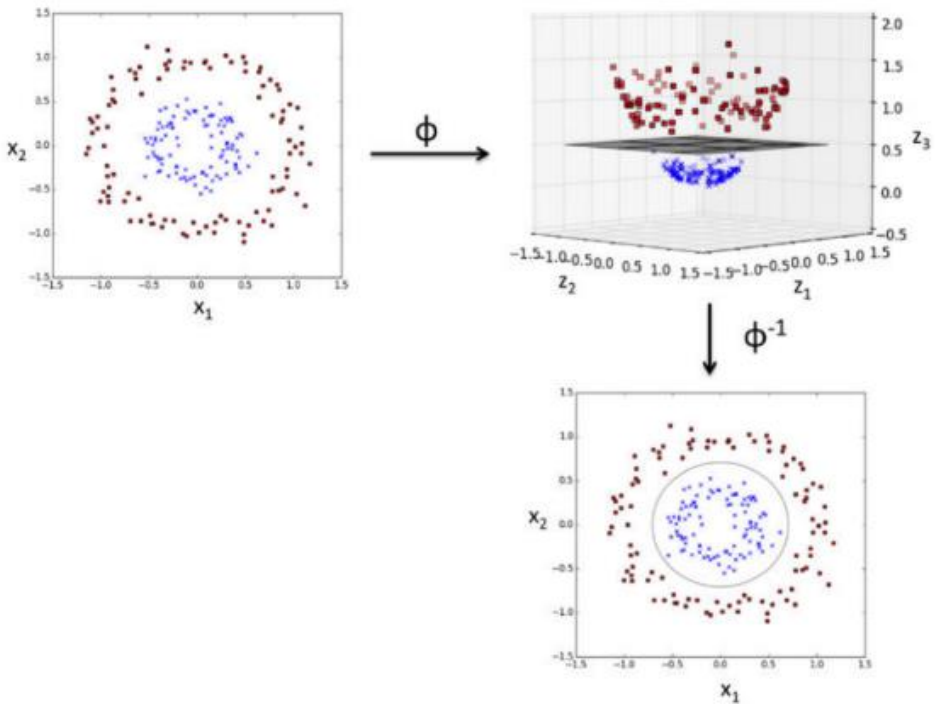
O grupo possui como principais algoritmos de classificação: *GaunssianNB, BernoulliNB e MultinomialNB* (MÜLLER; GUIDO, 2017). Demonstrando o funcionamento dos três algoritmos utilizados, vemos que o classificador de *BernoulliNB* se apresenta como um conjunto de dados binário e tem como princípio de funcionamento a contagem de quantas vezes as características de cada classe não são apresentadas como zero. Já os classificadores *GaunssianNB* e

MultinomialNB trabalham de forma um pouco diferente sendo que o classificador *GaussianNB atua* levando em conta que as características das classes seguem uma distribuição normal, enquanto o *MultinomialNB* leva em conta a média dos valores para cada característica de cada classe atuando com sistemas discretos (RAY, 2017). Para fazer as predições, os dados são comparados com os valores médios das estatísticas de cada classe a partir dos métodos e, a melhor classificação para cada um a partir das porcentagens é a predição utilizada (MÜLLER; GUIDO, 2017).

Os parâmetros principais utilizados para a configuração dos classificadores são o *alpha* que atua na complexidade do modelo, sendo que quanto maior o valor para *alpha*, menor a complexidade do modelo criado e vice-versa. Este parâmetro não é crítico para a performance do algoritmo, porém ele possui a capacidade de melhorar os resultados do algoritmo, o transformando em um modelo mais preciso. O *Fit-Prior* que pode ser colocado com verdadeiro ou falso e atua para o aprendizado ou não da classe *prior* e o *Class-Prior* que atua na probabilidade do ajuste da classe *prior* de acordo com o banco de dados utilizado (SCIKIT LEARN, 2022; SCIKIT LEARN, 2022a; SCIKIT LEARN, 2022b).

O Support Vector Machines (SVM) é colocado pelo autor como uma ferramenta versátil, poderosa e amplamente utilizada. Ela possui a capacidade de fazer classificações lineares, não lineares, regressão e *outlier detection* (GÉRON, 2017). A ideia por trás deste método é fazer a separação de dados linearmente inseparáveis e criar uma combinação não linear deles em mais uma dimensão, onde assim podem se tornar linearmente separáveis. Ou seja, transformar um conjunto de dados inseparável que possuía anteriormente duas dimensões para um conjunto separável de três dimensões. A seguir, é possível observar pela Figura 5 o que o modelo é capaz de fazer (RASCHKA, 2015).

Figura 5 - Separação de classes lineares



Fonte: Raschka (2015, p. 76).

Como pode ser observado na Figura 5, anteriormente era tido um gráfico com pontos azuis e vermelhos, porém não seria possível traçar uma reta para fazer a separação dos mesmos. Pelo método SVM foi possível aumentar uma dimensão e fazer a separação das duas classes sem problemas (RASCHKA, 2015).

Discutindo um pouco sobre a influência dos parâmetros sobre o modelo, é tido dois principais parâmetros: o *gamma* e o C. O *gamma* define o quanto de influência os dados de treinamento influenciam nas margens para separação das classes dos *datasets* enquanto o C tem influência em ser o parâmetro de regularização no SVM, ou seja, quanto menor o valor dele, maior a margem de dados que serão aceitas, assim diminuindo a precisão do modelo e quanto menor seu valor, a função de decisão terá uma margem menor sendo mais criteriosa com os dados (SCIKIT-LEARN, 2022c).

2.1.1.2 Modelo de Aprendizado Não Supervisionado

Quando é utilizado o aprendizado "unsupervised" ou não supervisionado, os dados que estão sendo tratados não possuem um rótulo, ou seja, não possuem

uma resposta para os dados de entrada. Uma exemplificação do caso pode ser dada na criação de um método para separação de animais como apresentado na Figura 4, porém quando fosse feita a divisão das características de cada animal, não é sabido o nome do animal e sim um grupo com animais de características similares ou idênticas (MÜLLER; GUIDO, 2017).

Sendo assim, esta técnica tem como finalidade explorar a estrutura dos dados, assim conseguindo informações úteis e padrões sobre eles sem saber qual será o valor resultante (RASHKA, 2015).

Outra explicação é dada por Das e Cakmak (2018) e coloca que o objetivo deste método é identificar padrões nos dados de entrada a partir das relações entre as características apresentadas por eles assim criando grupos de semelhantes com estes dados.

Segundo a IBM (2020), os modelos de aprendizados não supervisionados possuem três métodos principais, são eles: *Clustering, Association* e *Dimensionality Reduction.* O *Clustering* é um conjunto de técnicas que possui como objetivo criar grupos com dados semelhantes a partir dos dados de entrada entregues ao modelo. Para a criação destes grupos, existem alguns parâmetros que precisam ser determinados juntamente com a categoria do algoritmo que será utilizada, categorias estas que descrevemos a seguir.

A primeira categoria utilizada é a dos algoritmos hierárquicos, a qual a partir de uma matriz de proximidade e cálculos feitos pela distância entre dois grupos de dados, cria uma sequência de partições. Este tipo de algoritmo não lida bem com ruídos e *outliers*, termo utilizado para dados fora dos padrões, porém como ponto positivo, quando utilizado, não torna necessário as especificações de qual será o número de grupos ou *clusters* formados (FACELI *et al*., 2011).

Como segunda categoria, é apresentado os algoritmos particionados baseados em erros quadráticos. O objetivo principal destes é obter o menor erro quadrático para um número fixo de grupos. O modelo mais utilizado por este método é o K-means, disponível na biblioteca Scikit-Learn (FACELI *et al*., 2011). Para a parametrização dele, é necessário informar o número “K”, responsável pelo número de agrupamentos que serão feitos a partir dos dados. É necessário também definir a forma como os centróides serão calculados (SCIKIT LEARN, 2022d).

Na terceira categoria, é utilizado a densidade dos dados para fazer a

separação deles em grupos. Um algoritmo comumente utilizado para fazer este cálculo é o *denclue*, ele é baseado na estimativa de densidade de *kernel.* Este algoritmo apresenta dois momentos, o de pré agrupamento, que é responsável pelo mapeamento de todos os pontos relevantes dos dados, e o de agrupamento, o qual atua na identificação de atratores de densidade e pontos atraídos por eles criando assim, os grupos de pontos. Um problema apresentado pelo algoritmo *denclue* é sua sensibilidade a configuração dos seus parâmetros, *sigma* e *m*, os quais são difíceis de serem ajustados (FACELI *et al*., 2011).

Existem ainda mais três categorias que são divididas em grafos, que utilizam principalmente os algoritmos *HSC* e *CLICK,* redes neurais que utiliza principalmente o algoritmo *SOM* e a categoria *grid* que utiliza dos algoritmos *CLIQUE, MAFIA, Optigrid* e *STING* (FACELI *et al*., 2011).

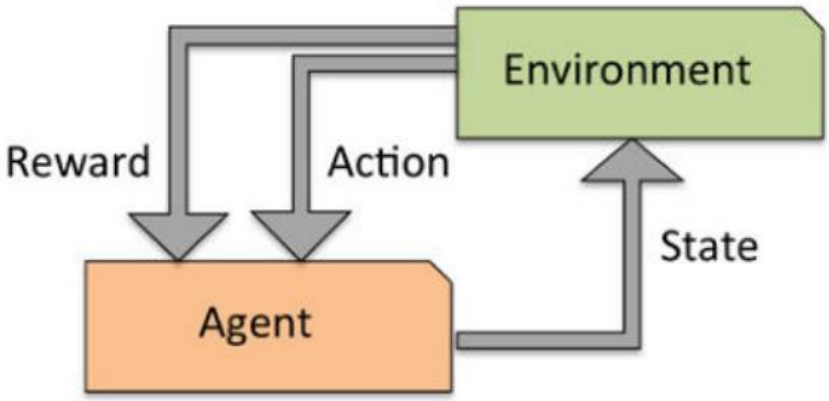
Em relação a *Association,* vemos um método baseado na descoberta de padrões entre os dados que são inseridos no modelo. As regras de associação são frequentemente utilizadas por empresas e organizações para analisar os hábitos dos consumidores a fim de entender seus costumes e, a partir deles criar estratégias de recomendações de produtos. Um exemplo deste método em ação pode ser visto em sites de compras online como *Amazon,* Mercado livre onde em suas lojas virtuais, a partir da escolha de um produto, são recomendados tantos outros itens que tem forte relação com o consumidor e seus padrões de consumo. Como exemplo de algoritmos utilizados podem ser citados *Eclat, FP-Growth* e *Apriori* (IBM, 2020).

O último modelo de Aprendizado Não Supervisionado consiste em *Dimensionality Reduction.* Quanto maior a quantidade de dados apresentadas ao modelo, maior pode ser a chance de se ter um sistema preciso, porém esta grande quantidade de dados pode trazer impactos negativos ao aprendizado do modelo, por exemplo o *overfitting*, e também pode tornar difícil a visualização e análise dos dados. Com isto em mente, surgem as técnicas e os algoritmos de redução de dimensões dos dados de treino, assim gerando grupos de dados com a quantidade de dados reduzidas, porém com seus comportamentos mantidos da forma mais fiel possível. Neste tipo de operação de pré processamento dos dados podem ser destacados os seguintes algoritmos: *PCA, SVD* e *Autoencoders.*

2.1.1.3 Modelo de Aprendizado Reforçado

O foco do modelo de aprendizado reforçado consiste em gerar um sistema, chamado de agente, que consiga melhorar a partir das interações com os dados. Ele faz isto com *rewards,* traduzido como prêmios, onde, quando o dado é colocado como correto, o sistema recebe um feedback positivo. Este feedback não é dado por um valor um por uma tabela verdade e sim pela função de recompensa. Um exemplo apresentado pelo autor é o do jogo de xadrez, onde a compensação é definida com a vitória ou perda do jogo. Segue abaixo a Figura 6 que utiliza deste caso como exemplo (RASHKA, 2015).

Figura 6 - Funcionamento do Reinforcement Learning



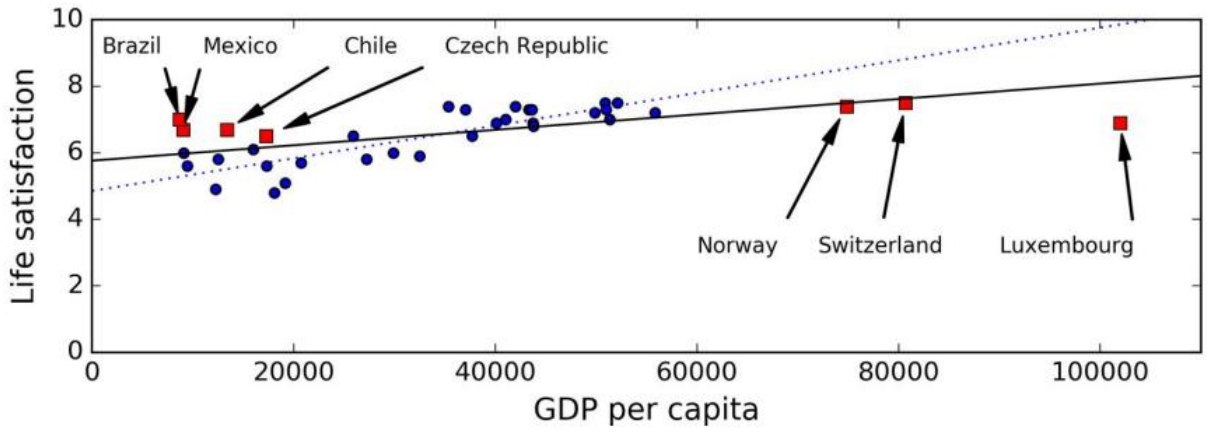
Fonte: Raschka (2015, p. 6).

Como é apresentado na Figura 6 existe o agente que faz uma série de decisões ou jogadas, e dependendo do estado que o ambiente, ou neste caso o tabuleiro ficar, pode ser recebido um uma recompensa ou uma penalidade, neste caso de vitória ou derrota no jogo (RASHKA, 2015).

Neste momento são colocados dois termos de importância na leitura e entendimento de dados utilizados pelos modelos, quais sejam, *Features* e *Labels*. As *Features* são as características de nossa variável que são utilizadas para fazer predições, uma vez que é considerando esses dados que o algoritmo faz inferências e cálculos para funções. Já as *Labels* seriam o somatório das características que dão um rótulo a alguma coisa, exemplificando o caso traz-se o *carro*, onde o conjunto formado por rodas, chassi, direção, vidros, sistema de potência definem o objeto carro, sendo esta a *label* (AWS, 2022).

Para Géron (2017) uma parte importante no momento de fazer treinamento do algoritmo é a qualidade dos dados que se está usando, estes podem apresentar vários problemas então a importância da limpeza e preparação dos dados de treinamento. O autor coloca que para fazer o treinamento de um algoritmo, é necessário um *dataset*, ou conjunto de dados, grande e nesta quantidade de informações podem ocorrer problemas como dados de treinamento que não representam bem o conjunto de casos. Ele coloca como exemplo uma sequência de dados de treinamento de países, onde o eixo X se trata da renda per capita e o eixo y trata da “felicidade” das pessoas nestes países, e quando se aplicou dados novos, alguns deles não estavam listados nos dados anteriores de aprendizado gerando alguns problemas como pode-se observar abaixo na Figura 7.

Figura 7 - Dataset do exemplo de países



Fonte: Géron (2017, p. 46).

Como é visto, os países adicionados que não estavam registrados estão na parte direita do gráfico. Pode-se observar um problema pois países ricos com mais renda per capita estão menos felizes que os países com menor renda. Isto mostra um possível erro no modelo devido ao conjunto de treinamento utilizado. Então o autor reforça a importância de utilizar um conjunto de dados que represente todos os casos que serão colocados como dados de entrada para geração de resultados e que, se o dataset utilizado for muito pequeno, isto pode resultar em uma *sampling noise* (Uma amostragem sem resultados confiáveis) ou quando se tem muitos dados de treinamento, pode-se gerar também uma resposta não representativa se os dados de amostragem forem defasados gerando um algoritmo não confiável (GÉRON, 2017).

Outros dois pontos de atenção colocados pelo autor consistem na baixa

qualidade dos dados de treinamento, podendo trazer ao modelo uma baixa performance e o segundo são as características irrelevantes que podem aparecer no conjunto de treinamento. Para fazer a tratamento destes campos que interferem no treino do algoritmo, existe o que se chama de *feature engineering* que envolve a escolha das características no conjunto de dados que são relevantes para preparação do algoritmo, a combinação das características que tem mesmo comportamento em apenas uma variável assim gerando um menor tempo de processamento para o programa e a criação de novas *features* devido a novos dados coletados (GÉRON, 2017).

Assim torna-se necessária também a avaliação dos resultados do modelo. Para ser analisada a performance do modelo e se o mesmo cumpre os requisitos especificados e pode ser utilizado, Rashka (2015) apresenta um método bem tradicional que tem o nome de *Cross-validation* podendo também ser chamado de validação cruzada.

O método nada mais é que a separação do *dataset* em frentes distintas, a de treino e a de teste, sendo que a de treino é utilizado para treinamento e aprendizado do modelo e a de teste é o utilizado para estimar a performance do algoritmo. Também é visto outro lado importante para testagem do modelo que é chamado de *model selection* o qual se refere a conseguir encontrar os melhores parâmetros de forma a otimizar o modelo. Nesta validação, são colocadas duas técnicas, o *holdout cross-validation* e *k-fold cross-validation*. Essas trazem o potencial de ser analisado o erro do modelo e o quão bem ele se comporta com dados novos (RASHKA, 2015).

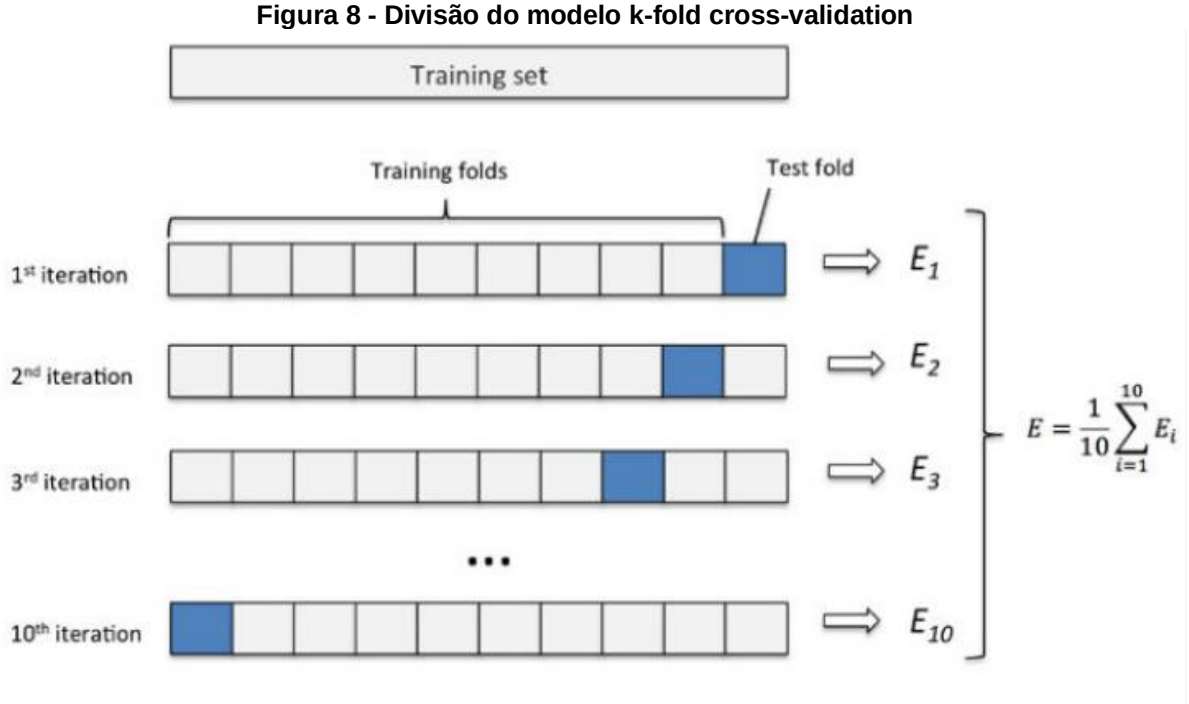
A melhor forma de utilizar a técnica de *holdout cross-validation* é fazer a separação dos dados em três partes: a de treino, a de validação e a de teste. A de treino atua na parte de treinamento do modelo, a de validação serve para escolha dos melhores parâmetros para o modelo e finalmente a de teste tem como objetivo analisar como o algoritmo se comporta com o recebimento de dados novos não vistos anteriormente nas etapas de treino e validação, assim buscando não obter um modelo tendencioso em seus resultados da performance. Uma questão a ser notada é que este método é muito sensível a como são particionados os três tipos de dados (RASHKA, 2015).

O *k-fold cross-validation* é colocado como um método mais robusto onde são separados os dados de treinamento várias vezes, e esta quantidade de

separações é colocado como variável “*k”* e são obtidos então “*k”* modelos e “*k”* estimativas de performance. Este desempenho calculado é medido a partir da média das performances dos “pacotes” da separação dos dados assim se mostrando um método menos sensível à escolha das partições dos dados, mostrando o porquê de se apresentar como um método mais robusto (RASHKA, 2015).

A Figura 8 explicita visualmente o método e como ele funciona.

Figura 8 - Divisão do modelo k-fold cross-validation



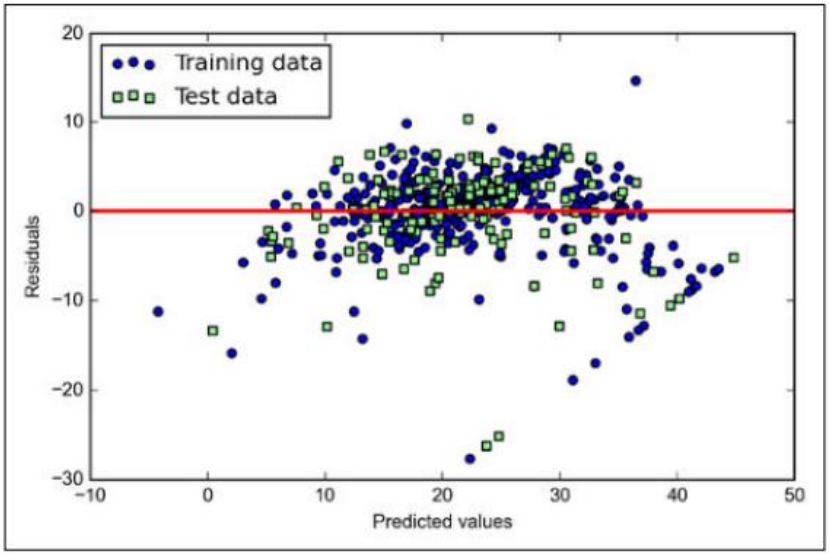
Fonte: Raschka (2015, p. 176).

Por padrão são feitas dez divisões dos dados, porém podem ser escolhidas menos divisões dependendo da quantidade de dados do *dataset.* A cada medida de performance de um grupo, o conjunto, tanto de dados de treino quanto de teste é escolhido de forma diferente, assim gerando menos divergência no momento da qualificação do modelo, isto é explicitado pela fórmula apresentada à direita da figura, onde é visto que a qualidade do modelo é feita pela soma de suas performances dividida pela quantidade de “pacotes” (RASHKA, 2015).

Uma forma mais específica de avaliar modelos de regressão, aquelas em que o objetivo é predizer resultados numéricos como exemplo valores de temperatura, calor, velocidade entre outros, acontece por meio do cálculo da diferença entre os valores encontrados pelo modelo em relação aos valores conhecidos que foram utilizados para o treinamento. Os valores encontrados pelo modelo são analisados por

meio dos gráficos de resíduo ou *residual plots* em que se observa os outliers, a não linearidade dos valores e seus erros (RASHKA, 2015).

Figura 9 - Gráfico dos resíduos entre os dados de teste e os dados previstos pelo modelo



Fonte: Raschka (2015, p. 295).

Analisando a Figura 9, o autor coloca que quanto mais os valores estiverem juntos a linha zero, melhor serão os valores da predição, porém, em um modelo prático é esperado que os erros estejam distribuídos próximos a linha central de forma aleatória. Neste mesmo gráfico é possível observar um padrão de pontos na parte inferior direita sendo isto um “vazamento” no modelo, ou seja, uma limitação do modelo para predizer alguns dados e se encontra também alguns pontos distantes do conjunto de pontos central, estes sendo *outliers* ou pontos fora da curva (RASHKA, 2015).

Além de se utilizar este gráfico de resíduos para medir a qualidade dos dados encontrados pelo modelo, utiliza-se o erro médio quadrático ou MSE (*Mean Squared Error),* o qual é a diferença quadrática média entre o valor estimado do modelo e o valor verdadeiro do grupo dados, e o coeficiente de determinação R^2 que varia de zero até um e quanto maior seu valor melhor o resultado do modelo. (RASHKA, 2015). Do MSE surge o RMSE que é a raiz do erro quadrático médio. Segue abaixo a fórmula 1 do RMSE:

Equação 1 – RMSE

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \quad (1)$$

Onde:

yi = Valores reais do grupo de dados

ŷi = Valores preditos

Como a equação apresenta, é feita a subtração dos valores preditos dos valores reais, e a partir disto o quadrado deste valor e posteriormente a média para todos os valores preditos. Quanto mais baixo o valor de RMSE, melhor a predição dos valores do modelo. (BRUCE; BRUCE, 2019)

Em relação a outro método de medição de acurácia chamado de coeficiente de determinação R^2, é apresentada a fórmula 2:

$$R^2 = 1 - \frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{\Sigma_{i=1}^{n}(y_i - \bar{y})^2} \quad (2)$$

Onde:

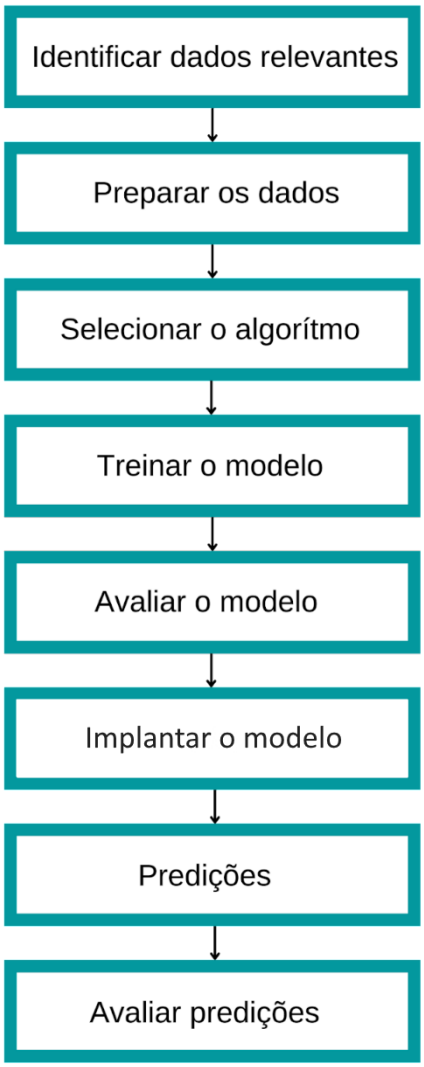
yi = Valores reais do grupo de dados

ŷi = Valores preditos

O R^2 mede a proporção de variação nos dados do modelo e pode variar de 0 até 1 sendo que quanto mais próximo de 1, melhor o resultado. Este parâmetro é principalmente utilizado em modelos de regressão e tem como função avaliar o quão bem o modelo acomoda os dados encontrados. (BRUCE; BRUCE, 2019)

Foram apresentados anteriormente de forma breve alguns dos pontos chaves para criação de um modelo de ML, desta forma pode-se exemplificar um esquema do ciclo de como isto deve ser feito no momento em que já se possui os dados. (BRUCE; BRUCE, 2019)

Figura 10 - Ciclo da criação de um modelo de Machine Learning



Fonte: Elaboração própria (2022).

Na Figura 10 são demonstradas oito etapas principais sendo que a primeira e a segunda são aquelas comentadas em que se deve fazer a preparação dos dados identificando quais deles são pertinentes para o estudo e a qualidade dos mesmos podendo gerar descartes de *features* e concatenação de dados que apresentam as mesmas variações, entendida como *data cleaning.*

Após isto é apresentada a seleção do modelo, esta parte vem da experiência do cientista de dados não existindo uma técnica exata para isto. Utiliza-se a parte de avaliação dos resultados do modelo exatamente nesta etapa, sendo então um ciclo de escolhas de algoritmos e avaliação dos resultados, esta etapa pode se apresentar bem longa e desgastante até ser encontrado o melhor método. Neste último momento é tida a última seção do fluxo que pode ser explicada de forma

conjunta com sua primeira parte se baseando no uso do modelo na aplicação, o *deploy,* e a avaliação deste, capturando as predições que o modelo estaria fazendo a fim de monitorar e analisar os dados e, se necessário retreinar e reaver o teste de qualidade do mesmo (HURWITZ; KIRSCH, 2018).

2.2 Automated Machine Learning

No capítulo anterior descreveu-se como se dá a formação do modelo de *Machine Learning*, os percalços no caminho do cientista de dados desde a parametrização até a criação de um sistema que demonstre bons resultados.

Então para deixar mais veloz, menos experimental, mais assertivo e menos dificultoso, surge o *Automated Machine Learning,* também chamado de AutoML*,* o qual traz como objetivo fazer estas tarefas repetitivas, como pré processamento das *features,* seleção de modelos, melhora dos *hyperparameters,* mais rapidamente para que o foco não seja mais nestes pontos, mas na análise dos resultados e interpretação dos mesmos (VENKATA *et al*., 2021).

Dentro de exemplos do uso de AutoML nas mais diversas áreas, vemos que no artigo de Eeden (2021) foi trazido o estudo da predição de comportamentos depressivos, bipolaridade, ansiedade na população. Neste trabalho foi apresentado que o aumento da capacidade de predizer estes comportamentos com mais precisão seria clinicamente relevante pois, em muitos casos o diagnóstico depende da experiencia dos profissionais da área juntamente com as informações dos livros. Porém, é colocado que, mesmo com conhecimento e a vivencia na profissão, é possível ignorar informações relevantes devido a uma grande ênfase em alguns sintomas não observando o quadro geral. Com estas premissas, surge a plano de se utilizar técnicas do *Machine Learning* para fazer predições mais eficientes assim ajudando com as decisões médicas sobre qual o quadro de saúde dos pacientes e posteriormente, qual seria a melhor forma de lidar com ele. Por meio de uma meta análise que incluía mais de 20 estudos sobre *therapeutic outcome of depression*, foi encontrada uma precisão de 82 % com 95 % de confiança.

Outra análise em um conjunto de dados da NESDA (The Netherlands Study of Depression and Anxiety), conseguiu uma acurácia na predição de recuperação dos quadros de ansiedade de 62% e em quadros de doenças mentais em geral, uma

precisão de 63 %. Com os dados apresentados, o objetivo do estudo é: mostrar a diferença entre métodos tradicionais de *Machine Learning,* utilizando o algoritmo de classificação de *Naive Bayesian* e *Logistic Regression*, com o método mais avançado do *Automated Machine Learning,* o *Auto-Sklearn* e, com a hipótese de que o algoritmo de *Automated Machine Learning* seria mais eficiente em detectar padrões mais complexos, assim apresentando melhores resultados que o método tradicional*.* O conjunto de dados seria um acompanhamento de predição de quadros psiquiátricos de 2,4,6 e 9 anos, e teria como variáveis a se predizer *clinician-rated, psychiatric diagnoses, self-reported depressioan na anxiety* e *demographic variables* (EEDEN *et al.*, 2021).

Como o resultado do estudo, foi encontrado que o AutoML não conseguiu performar em todos os conjuntos de predição como o melhor modelo em relação ao método tradicional de *Machine Learning,* porém se apresentou como um método mais consistente e com melhores resultados quando o dado predito era mais complexo (EEDEN *et al.*, 2021).

No segundo artigo de Venkata *et al.* (2021), é trazido o estudo que utilizou de técnicas de *Deep Learning* e *Auto Deep Learning* para análise e predição da qualidade de água do rio Korattur em Chennai.

Neste estudo, é apresentado que os recursos naturais como lagos e rios estão enfrentando fortes ameaças devido ao descarte de recursos não tratados de indústrias, esgoto e restos em geral então se apresenta como obrigatório o controle de qualidade da água que é consumida pelos seres humanos. A partir deste problema, existem os métodos tradicionais para execução deste controle, porém esses se apresentam como processos longos e que demandam muito tempo. Como um catalisador para isto, surge a ideia de se utilizar inteligência artificial então, as técnicas de D*eep Learning* e *Auto Deep Learning.* A partir deste ponto, é traçado o objetivo do estudo, fazer a comparação entre o *Auto Deep Leaning* comparando os resultados dele com os modelos de *Deep Learning* tradicionalmente utilizados (VENKATA *et al.,* 2021).

O estudo mostrou que os algoritmos do *Deep Learning* tradicionais conseguiram uma acurácia de 1.8% maior que os algoritmos de *Auto Deep Learning* quando se tratava de classes com apenas dois resultados, ou seja, classes binárias como exemplo, água poluída ou água limpa. Em relação a dados com várias classes,

ou seja, vários resultados possíveis como exemplo, classificação de espécies de plantas, o método tradicional de *Deep Learning* também prevaleceu com uma acurácia 1% maior que o método do *Auto Deep Learning*. Mesmo com estes dados em vista, os autores salientam que nos casos tradicionais, é necessário parametrizar e encontrar os melhores coeficiente para os modelos sendo esta uma tarefa que necessita de tempo e experiência enquanto, ao se utilizar de um modelo baseado no *Auto Deep Learning*, isto não é necessário (VENKATA *et al.*, 2021). Com alguns artigos fundamentando o tema, traz-se a ideia principal do AutoML novamente, tornar possível o desenvolvimento de um sistema de aprendizado efetivo em que o usuário não necessite de alto grau de conhecimento sobre os algoritmos e seus parâmetros. (NAGARAJAH; PORAVI, 2019).

São apresentados os níveis de automação do AutoML, eles podem ser separados pelas categorias de encontro de estimadores e preditores, encontro de algoritmos e encontro de algoritmos de *meta* aprendizado.

A primeira categoria se refere a tarefa de mapear a partir das entradas das variáveis os resultados das saídas, como exemplos podem ser dados a escolha do valor dos parâmetros de um algoritmo de regressão linear para uma tarefa específica ou um código de classificação de baixo nível, no qual se apresenta como necessário especificar cada uma das condições de entrada de um conjunto de valores de variáveis e, a partir deste conjunto de entrada gerar regras de saídas do conjunto, os resultados. Neste caso a ferramenta de AutoML atuaria nas escolhas dos parâmetros para o primeiro caso e no segundo caso, para o entendimento das condições de entrada e as ações tomadas a partir destas condições.

Para a segunda categoria apresentado, o papel da ferramenta do AutoML é dado pelo encontro dos algoritmos para o dado problema, escolha dos hiper parâmetros, análise e pré-processamento dos dados. Como terceira categoria é colocado o encontro e utilização de algoritmos de meta aprendizado, os quais possuem como objetivo melhorar os resultados de modelos existentes comparando os resultados encontrados por eles e sugerindo mudanças nos hiper-parâmetros ou recomendando o uso de outros algoritmos de aprendizado. (ESCALANTE, 2020)

As bibliotecas ou métodos destacados pelo autor para a segunda categoria que possui como objetivo o encontro de algoritmos de forma automatizada são *PSMS, Heterogeneous surrogate Evolution, Ensemble PSMS, TPOT e GPS: GAPSO-FMS.*

Para a terceira categoria utilizando algoritmos de meta aprendizado são apresentadas pelo as bibliotecas: *Auto-WEKA, Multiobjective surrogatebased FMS, AutoSkLearn e Neural Architecture Search* (ESCALANTE, 2020).

É colocado pelo autor alguns dos desafios que o campo do AutoML apresenta como a não transparência de como os modelos funcionam, a parte de pré processamento dos dados que é importante, porém ainda pouco explorada pela área, dificuldade para lidar com conjunto de dados muito extensos e reprodutibilidade dos resultados encontrados utilizando do método do AutoML. Porém mesmo com estes apontamentos Escalante (2020) traz que com o progresso na primeira década de utilização das ferramentas do AutoML foi muito interessante e que muito pode se esperar dos próximos anos desta área de estudo. (ESCALANTE, 2020).

3 METODOLOGIA

A ideia inicial do presente estudo consistia na criação de um algoritmo, por meio das bibliotecas de AutoML e de uma plataforma online disponibilizada por um terceiro para disposição interativa do algoritmo treinado, que faria a análise de alguns conjuntos de dados.

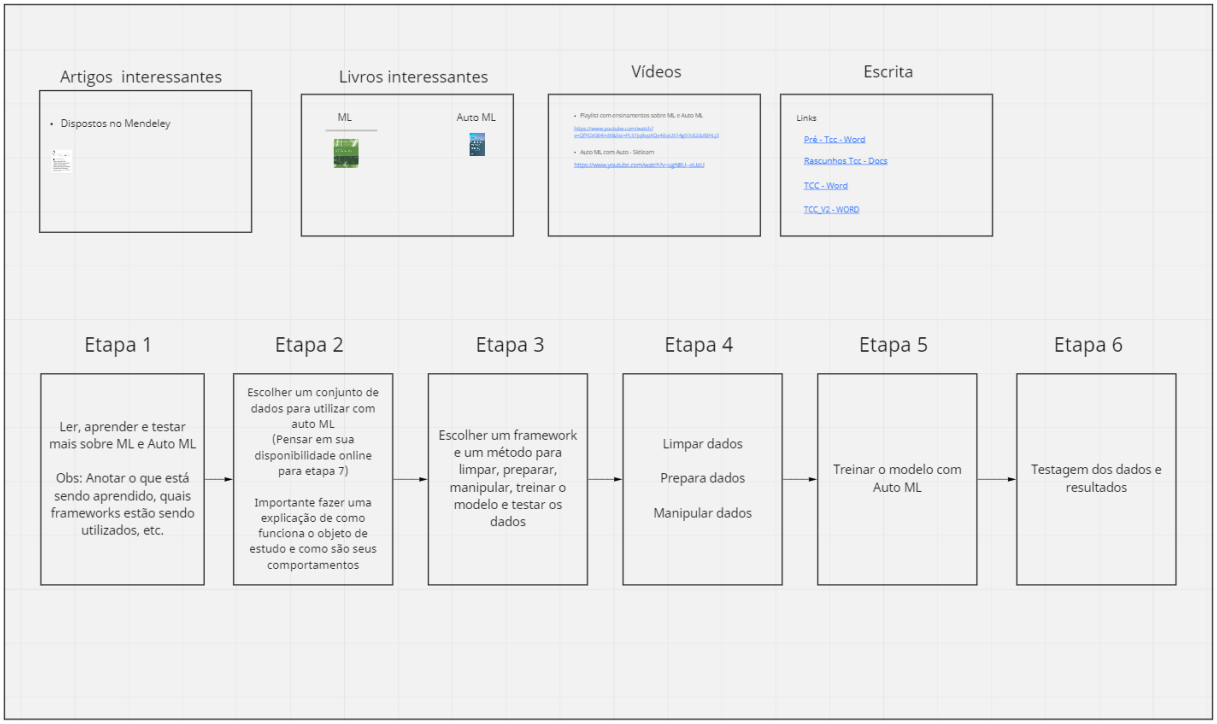
No decorrer do processo, foi discutido e visto que o objetivo precisava ser mais específico a fim de trazer resultados mais palpáveis à pesquisa, desta forma modificou-se o estudo para a escolha de um banco de dados único e a não utilização da plataforma online devido a complexidades que a tarefa traria consigo.

A partir desta escolha surgiu a concepção de um trabalho mais didático com objetivo de trazer resultados sobre um conjunto de dados de motores elétrico e, a partir dele a predição da temperatura dos rotores utilizando das bibliotecas *Auto-Sklearn, TPOT e H2O* do campo do AutoML, adentrando no mundo do *Machine Learning,* introduzindo os conceitos, os programas, estudos da utilização do AutoML em outras áreas do conhecimento, a apresentação das ferramentas e a comparação entre os resultados encontrados com os disponíveis na plataforma Kaggle.

Com a reelaboração do objetivo, se construiu um quadro organizacional demonstrando o fluxo de tempo e espaço para execução das tarefas. Na criação deste quadro organizacional, foi possível entender a complexidade do trabalho e analisar o que poderia ser entregue no tempo estipulado considerando as modificações da ideia inicial. Esta linha do tempo foi traçada utilizando o programa *miro, software web* que tem como objetivo principal simular um quadro branco.

Neste quadro foram idealizadas as datas de entrega de cada tópico, seus objetivos e constam alguns materiais como livros, artigos e vídeos que foram utilizados para o aprendizado e execução da atividade. A Figura 11 apresenta o quadro organizacional.

Figura 11 - Quadro organizacional elaborado para a pesquisa

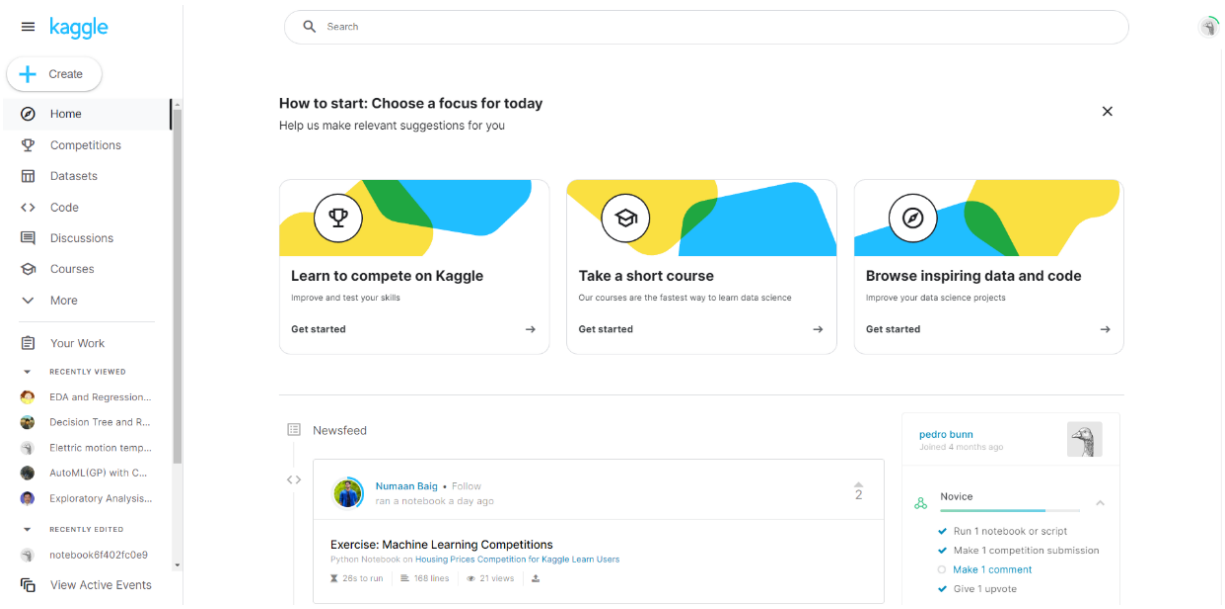


Fonte: Elaboração própria (2022).

A partir deste quadro, iniciou-se o trabalho da busca de conhecimentos por meio das mais diversas ferramentas, sites e artigos, a fim de entender um pouco mais sobre o mundo do *Data Science.* Os primeiros passos se deram para entender e se familiarizar com a linguagem *Python.*

Segundo Sebesta (2010) a profundidade da capacidade das pessoas de pensar é influenciada pelo poder da expressividade da linguagem que elas utilizam. A partir desta colocação, é possível entender porque foi esta a linguagem de programação escolhida para a pesquisa. Ela apresenta uma escrita de fácil entendimento, com diversas bibliotecas para visualização e análise de dados além de uma proporcionar uma rápida interação com os dados através dos terminais, uma tarefa necessária quando o assunto é *Machine Learning* (MÜLLER; GUIDO, 2017). Também foi necessário aprender sobre a biblioteca *pandas,* amplamente utilizada no tratamento e organização de dados e no campo do *Machine Learning,* entendendo melhor sobre seus algoritmos, classificações e métodos. Optou-se por fazer os cursos disponíveis na plataforma do *Kaggle,* site que foi de extrema importância na aquisição de conhecimento sobre o campo de estudo. A Figura 12 a seguir ilustra a página de apresentação do site.

Figura 12 - Pagina inicial Kaggle



Fonte: Kaggle (2022).

Na barra lateral da Figura 12, pode-se visualizar as abas de competições onde é possível criar modelos de *Machine Learning* a partir de grupos de dados disponíveis na plataforma. Observa-se a apresentação de diversos grupos de dados contendo grande variedade de categorias como negócios, músicas, saúde, trânsito o que permite competir com outros modelos gerados com estes mesmos dados por outros usuários instigando uma competição saudável entre eles. Também é visto a aba de cursos onde são oferecidas aulas gratuitas sobre diversos assuntos relacionados ao *Machine Learning* como visualização de dados, bibliotecas utilizadas, introdução ao *Deep Learning, Time Series, Data Cleaning* entre outros.

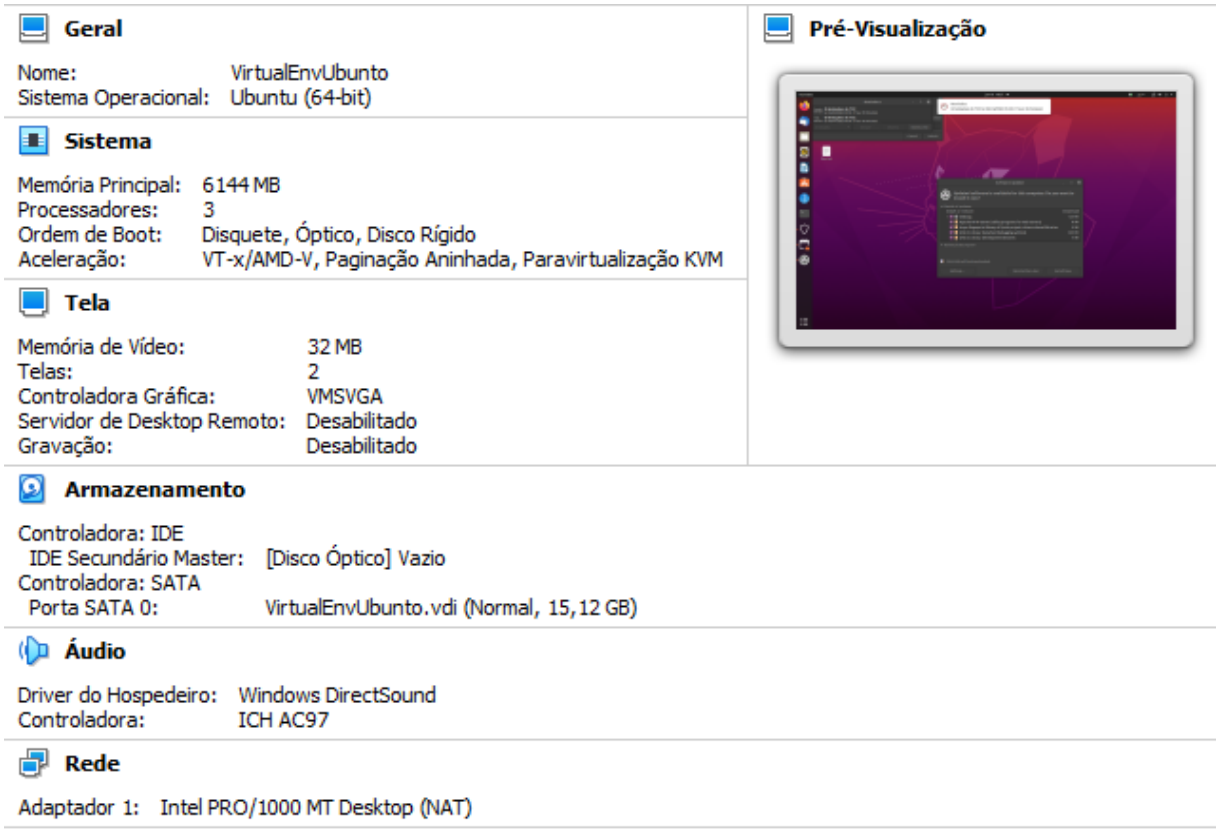
A ideia da plataforma é disponibilizar aos desenvolvedores e cientistas de dados um meio de compartilhas conhecimentos, praticar, entender, estudar, e resolver casos do mundo real *por meio do Machine Learning* (USMANI, 2017).

Feitos os cursos da plataforma, também foi necessário buscar como se dava o funcionamento das bibliotecas de *Automated Machine Learning. A*s bibliotecas que foram escolhidas para este estudo foram a *Auto-Sklearn,* TPOT e H2O, escolhidas por possuírem uma boa documentação, serem bem comentadas e apresentarem avaliações positivas pelos usuários do *GitHub* (BALAJI; ALLEN, 2018). Para experimentação destas bibliotecas utilizou-se o ambiente *Jupyter Notebook amplamente* empregado no campo do *Machine Learning,* sendo uma ótima ferramenta para análise

exploratória de dados (MÜLLER; GUIDO, 2017).

Para utilização da biblioteca *Auto-Sklearn,* é necessário um sistema operacional baseado em Linux, demanda suprida de duas formas, a primeira sendo através de uma máquina virtual executada pelo software *Oracle VM VirtualBox*, com as seguintes especificações apresentadas na Figura 13.

Figura 13 - Especificações da máquina virtual

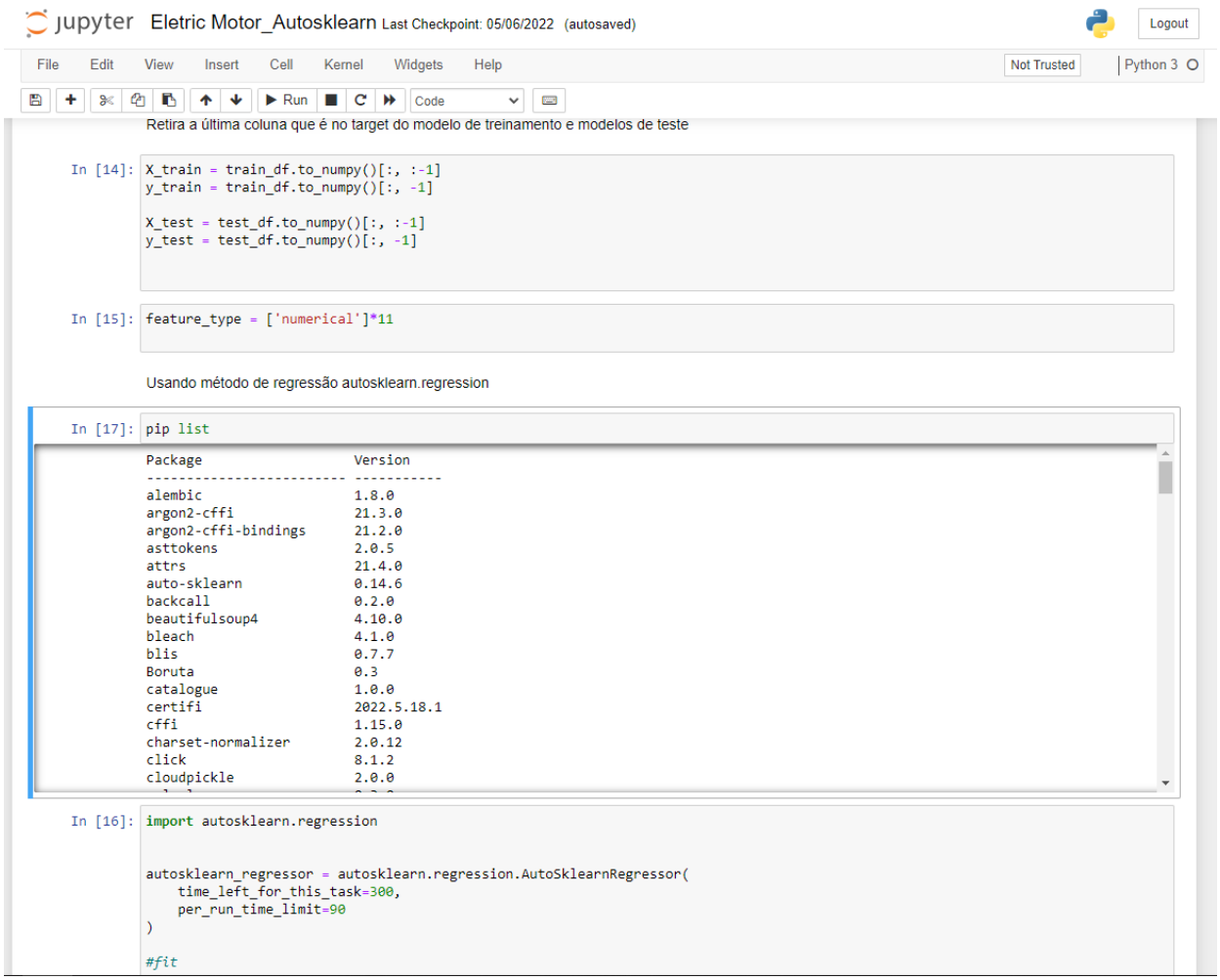


Fonte: Elaboração própria (2022).

A partir da máquina virtual, foi feita a instalação do *Jupyter Notebook e* também criados ambientes virtuais. A instalação do ambiente de desenvolvimento *Jupyter Notebook,* o qual tem seu ambiente principal apresentado pela Figura 14, se torna importante pois nele são geradas as linhas de código, a importação de bibliotecas, o tratamento dos dados do *dataset*, em geral tudo o que é necessário para execução do programa. A criação de ambientes virtuais se torna relevante pois como foram utilizadas três bibliotecas diferentes de *Automated Machine Learning*, cada uma delas necessita de dependências diversas para seu funcionamento se tornando interessante a presença de ambientes diferentes que contenha cada uma dessas

especificações (ISLINGTON, 2021). Todo este processo, realizado no terminal Linux, está descrito com maior detalhamento no Apêndice A.

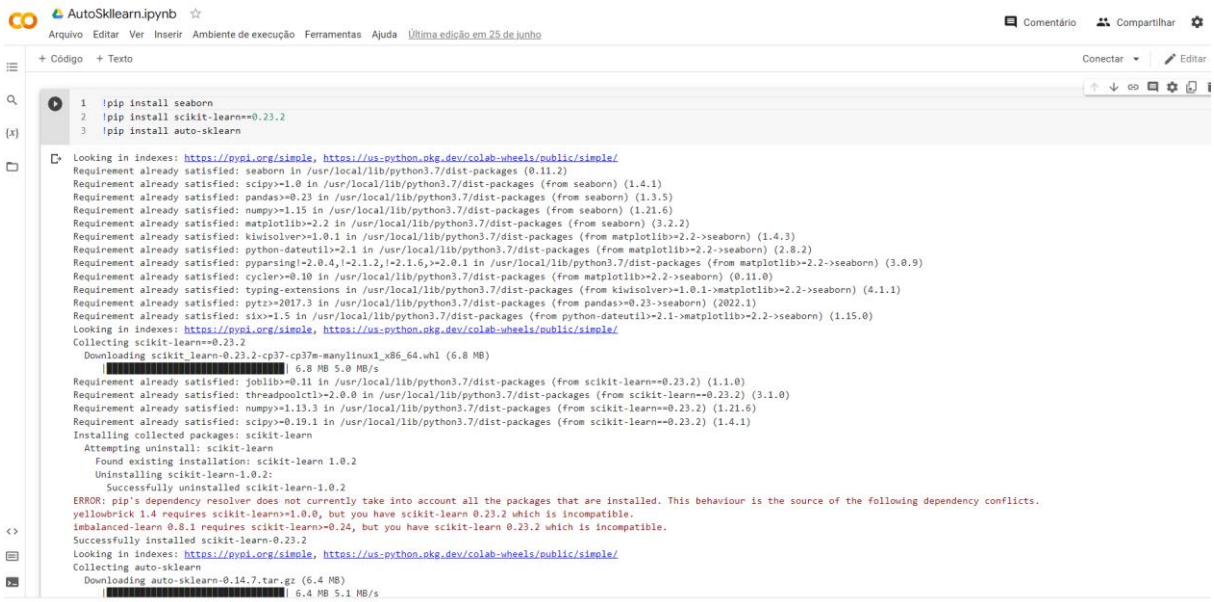
Figura 14 – Ambiente do *Jupyter Notebook*



Fonte: Elaboração própria (2022).

A segunda forma para utilização da biblioteca foi a partir do *Google Colab,* um produto desenvolvido pelo *Google Research* que permite a escrita e execução de códigos em Python principalmente utilizado para o campo do *Machine Learning, o* ambiente é apresentado pela Figura 15 *(*GOOGLE, *2022)*.

Figura 15 – Ambiente do *Google Colab*



Fonte: Google (2022a).

O recurso é utilizado diretamente pelo navegador e seu uso decorreu pela necessidade de maior poder de processamento para o método do AutoML utilizando a biblioteca *Auto-Sklearn.* O código utilizado é similar ao utilizado no Apêndice A, sua única particularidade é o caminho em que é “lido” o *dataset,* no caso do uso da máquina virtual com o *Jupyter Notebook,* o arquivo carregado estava diretamente na pasta onde o programa se encontrava, assim sendo bastava ser descrito o nome do arquivo para a leitura, no caso do *Colab* foi necessário especificar o caminho onde ele se encontrava (Apêndice B).

Com esta etapa realizada, o próximo passo foi encontrar um *dataset* relevante no contexto da Engenharia Mecatrônica sendo escolhido um grupo de dados sobre motores elétricos, na sequência preparar os dados para o treinamento do modelo a partir da biblioteca Pandas. Escolheu-se então o grupo de dados hospedado na plataforma *Kaggle* sobre a temperatura de um motor elétrico disponibilizado pela universidade de Paderborn (BÖCKER, 2021), sendo que o Quadro 1 demonstra as variáveis dos grupos de dados.

Quadro 1 - Variáveis do grupo de dados

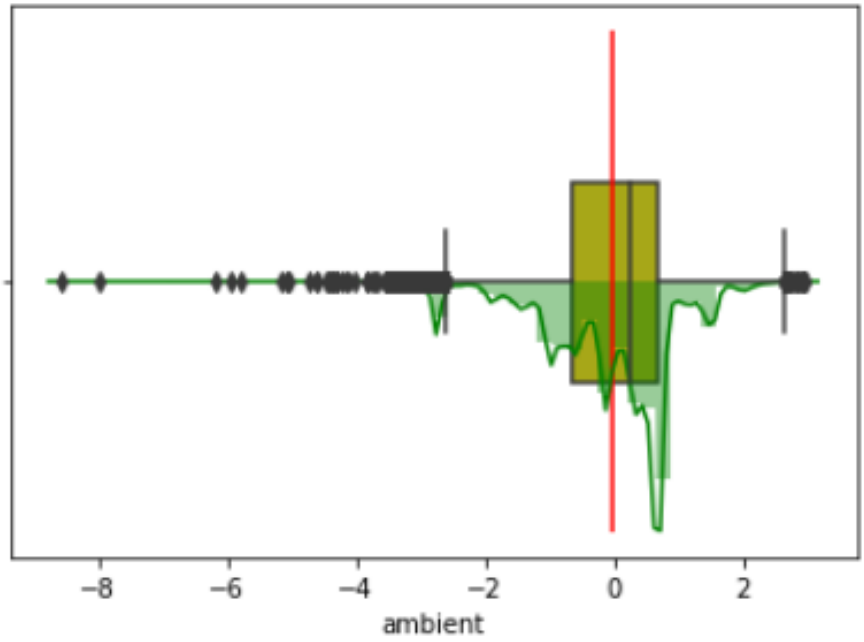
Variáveis	Funções	Unidade
u_q	Tensão do componente	V
Coolant	Temperatura do refrigerador	ºC
Stator_winding	Temperatura do bobina do estator	ºC
u_d	Tensão do componente	V
Stator_tooth	Temperatura dos dentes do estator	ºC
Motor Speed	Velocidade do motor	RPM
i_d	Corrente do componente	I
i_q	Corrente do componente	I
pm	Temperatura do rotor	ºC
Stator Yoke	Temperatura da coroa do estator	ºC

Fonte: Elaboração própria (2022).

É interessante ressaltar que as variáveis i_d, i_q, u_d e u_q são resultantes da estratégia de controle do motor que tem referência com a velocidade e o torque. Böcker (2021) não traz mais informações sobre estas variáveis, porém para o presente estudo foi necessário apenas conhecer seus valores, sem trazer interferências ao estudo.

Analisando o conjunto de colunas dos dados, observou-se que nenhum dos dados possuía valores nulos e que todos eles eram do tipo numérico "*float64".* Com isto em mente, não seria necessário fazer nenhum tratamento em relação a ajuste de valores. Foi observado a partir dos *BoxPlots* (ASARKAR, 2020), forma rápida para interpretação dos dados apresentado na Figura 16 que ilustra a distribuição dos pontos da temperatura do ambiente de teste do banco de dados utilizados neste estudo, que em cada uma das colunas os valores de média e mediana, linhas vermelha e marrom, eram bem próximos o que demonstra uma baixa assimetria na curva. Também pode ser visto que, em média, os grupos apresentaram poucos "*outliers",* valores destoantes do conjunto de dados (BRUCE; BRUCE, 2019). Por estes fatores, nenhum ajuste na tabela de dados foi executado.

Figura 16 - BoxPlot da temperatura do ambiente



Fonte: Asarkar (2020).

Para treinamento dos modelos, foi escolhida a variável *"pm",* temperatura do rotor, como a variável "*target"* e retirada a variável "*profile_id",* a qual se refere a duração das sessões de testes com o motor pois no estudo não foi levado em consideração os tempos de cada um dos testes. A variável "*pm"* foi escolhida como valor alvo a se predizer pois economicamente para uma indústria automotiva, conhecer este valor traz a possibilidade de fabricar motores elétricos com menos material e utilizar da capacidade máxima do equipamento por saber quais são os comportamentos do mesmo (BÖCKER, 2021).

Além disso, fez-se o embaralhamento dos dados da tabela para que todos os valores das classes fossem bem distribuídos quando separados em treino e teste (MÜLLER; GUIDO, 2017). Neste estudo, os dados foram separados em 75 % em treino e o restante de 25 % em teste. Para execução destes passos utilizou-se da biblioteca Pandas (Apêndice C).

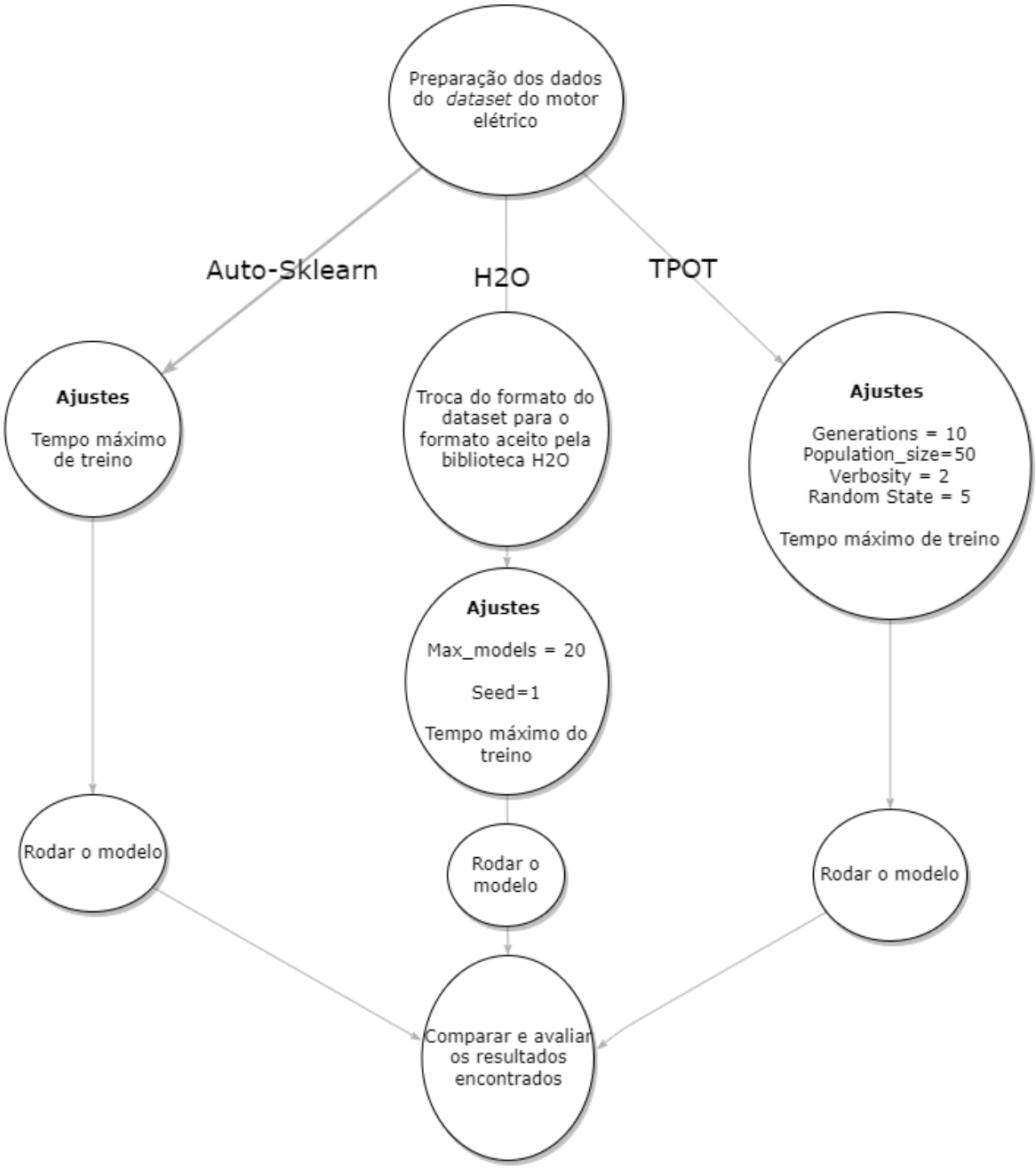
Quando utilizados os modelos *Auto-Sklearn* (Apêndice D) e TPOT (Apêndice E) bastou serem configurados os parâmetros do método de regressão. Para o primeiro modelo foi configurado o tempo de treino limite para cada algoritmo e o tempo limite de treino igual ao padrão. Para o segundo modelo foram estabelecidos a quantidade de gerações e populações, responsáveis pela quantidade de interações do *dataset*, respectivamente afetando no tempo de treinamento, a variável v*erbosity*

responsável pela quantidade de informações que o modelo se comunica, campo *random state* que possui o objetivo da repetibilidade dos resultados do treinamento do modelo e o tempo máximo de treinamento. Todos estes parâmetros foram encontrados de forma empírica partindo dos valores utilizados nos exemplos disponibilizados pelos criadores das bibliotecas (OLSON, 2022; AUTO SKLEARN, 2022).

A biblioteca H2O possui a particularidade de não compreender o conjunto de dados no formato de *DataFrames*, um formato de tabela que possui linhas e colunas rotuladas o qual foi utilizado pelos outros dois métodos. Assim sendo, foi necessária a troca do formato do *dataset* para o formato H2O (Apêndice F). Para seu treinamento, foram configuradas o máximo de modelos que poderiam ser treinados, o tempo máximo de treino por modelo, o tempo máximo de treinamento e o *seed,* que é responsável pela reprodutibilidade dos resultados.

Na Figura 17 se apresenta o caminho traçado até chegar aos resultados do treinamento pelas bibliotecas.

Figura 17 - Fluxograma das etapas desenvolvidas para predição dos algoritmos



Fonte: Elaboração própria (2022).

4 APRESENTAÇÃO DOS RESULTADOS

Após serem apontadas as etapas para o tratamento dos dados do *dataset* sobre motores elétricos e a criação de modelos de AutoML, foram gerados os resultados os quais são apresentados como algoritmos e hiperparâmetros. As métricas utilizadas para comparação dos resultados encontrados pelas bibliotecas foram aquelas expostas no capitulo 2.1.1.3 as quais são tidas como: erro quadrático médio (*Mean Squared error)*, coeficiente de determinação R^2 e o *k-fold cross-validation.* Inicia-se esta etapa com os resultados gerados a partir da biblioteca *Auto-Sklearn.*

Na utilização da biblioteca *Auto-Sklearn* o único parâmetro configurado foi o do tempo limite de treinamento de aproximadamente 3 horas (Apêndice G), este parâmetro foi necessário pois o ambiente de execução *Colab* na versão plano gratuito contém várias restrições, entre elas o tempo limite para a utilização da plataforma de modo contínuo de no máximo 12 horas (DAVID, 2021).

Pode-se levantar o questionamento do porque não utilizar o tempo máximo de execução que a plataforma oferece. Em testes realizados com tempos maiores a plataforma interrompeu o serviço de treinamento ou a biblioteca não apresentou o comportamento esperado sendo assim foi encontrado pelo autor o tempo de treino de aproximadamente 3 horas como sendo o ideal. Dada a execução do programa foi encontrado o resultado apresentado na Figura 18 disposto pela função *leaderboard* que tem a objetivo de mostrar os melhores algoritmos encontrados pela biblioteca *Auto-Sklearn*.

Figura 18 -Tabela de melhores algoritmos encontrados pela biblioteca Auto-Sklearn

```
1  print(autosklearn_regressor.leaderboard())
```

model_id	rank	ensemble_weight	type	cost	duration
43	1	0.40	k_nearest_neighbors	0.002416	37.370763
47	2	0.38	k_nearest_neighbors	0.002638	60.228681
28	3	0.18	k_nearest_neighbors	0.002641	65.127751
10	4	0.04	gradient_boosting	0.017248	109.899132

Fonte: Elaboração própria (2022).

É interessante notar que os três primeiros algoritmos encontrados na tabela da Figura 18 são os mesmos, *K_nearest_neighbors,* e que a diferença entre eles está no ajuste dos hiperparâmetros. Investigando mais a fundo os parâmetros do primeiro algoritmo com a função *Show_models* temos o apresentado na Figura 19.

Figura 19 - Parâmetros do primeiro algoritmo

```
1  print(autosklearn_regressor.show_models())
```

```
}>, 'sklearn_regressor': KNeighborsRegressor(n_neighbors=2, p=1, weights='distance')},
```

Fonte: Elaboração própria (2022).

Logo, a partir deste algoritmo e dos parâmetros encontrados pelo modelo foi executada a predição da temperatura do rotor, "*pm",* a partir do grupo de dados de motores elétricos. Com isto foram obtidos os resultados demonstrados na Figura 20.

Figura 20 - Predição e resultado dos valores de treino e teste

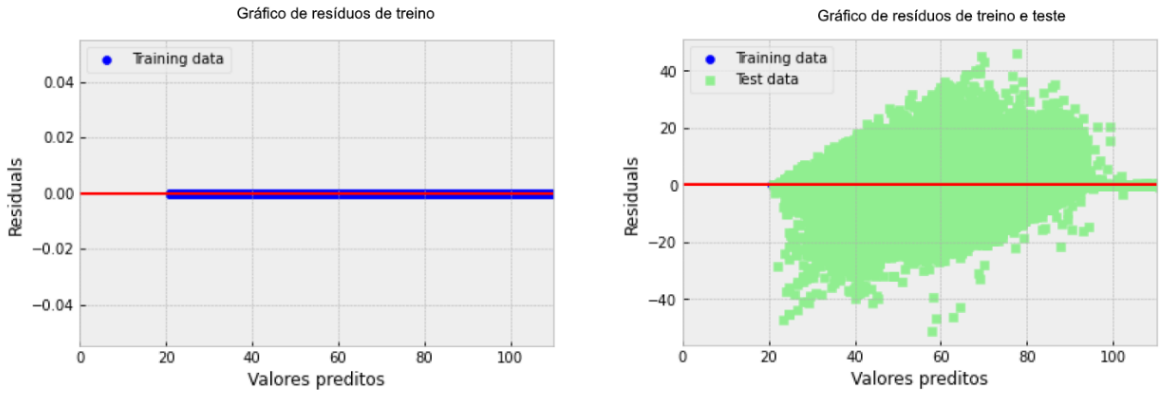
```
In [11]:

print("Scores R2 de treino", sklearn.metrics.r2_score(y_train,Pred_train_y))
print("Scores R2 de teste", sklearn.metrics.r2_score(y_test,Pred_test_y))
```

```
Scores R2 de treino 1.0
Scores R2 de teste 0.9716096096469911
```

Fonte: Elaboração própria (2022).

Como pode ser visto nas linhas do *notebook* o valor médio de R^2 também conhecido como coeficiente de determinação, um coeficiente que é considerado base para avaliar modelos de regressão (CHICCO; WARRENS; JURMAN, 2021), apresentou como resultado de treino o valor 1 e para o valor de teste 0,9716. Com os coeficientes encontrados, também foi executado os gráficos residuais de treino e de treino e teste, os quais podem ser vistos na Figura 21.

Figura 21 - Gráficos residuais Auto-Sklearn



Fonte: Elaboração própria (2022).

Com os gráficos residuais traçados, utilizou-se do método *Cross-validation* utilizando a função *KFold,* a qual foi parametrizada para dividir os dados em 10 grupos e com o embaralhamento dos valores ativo. Os resultados do método são apresentados pela Figura 22.

Figura 22 - Cross-Validation

```
knn_model= neighbors.KNeighborsRegressor(n_neighbors=2, p=1, weights='distance')
kfold  = KFold(n_splits=10, shuffle=True) # shuffle=True, Shuffle (embaralhar) the data.
result = cross_val_score(knn_model, X, y, cv = kfold)
rmse_cv=np.sqrt(-1*cross_val_score(knn_model,X,y ,cv=kfold,scoring='neg_mean_squared_error').mean())


print("K-Fold (R^2) Scores: {0}".format(result))
print("Média do R^2 para a validação cruzada K-Fold: {0}".format(result.mean()))
print("Erro quadrático médio", rmse_cv)
```

```
K-Fold (R^2) Scores: [0.95831087 0.95733623 0.95781197 0.95792054 0.95671842 0.95767563
 0.95731826 0.95779434 0.9583892  0.95819634]
Média do R^2 para a validação cruzada K-Fold: 0.9577471798436653
Erro quadrático médio 3.893667713835788
```

Fonte: Elaboração própria (2022).

O resultado médio do coeficiente de determinação foi de 0,9576 e o valor da raiz quadrática média dos erros foi igual a 3.8937. Com os resultados da biblioteca *Auto-Sklearn* retratados, serão apresentados os dados encontrados pela biblioteca TPOT. Os parâmetros que foram estabelecidos para o uso da ferramenta foram *Generation* com o valor 10, *Population_size* com o valor 50, *Verbosity* com valor 2, *Random_State* com valor 5 e o tempo máximo de treino de 3 horas. O modelo de

configurações dos modelos e o algoritmo encontrado pelo TPOT podem ser vistos na Figura 23.

Figura 23 - Tabela de melhor algoritmo encontrado pela biblioteca TPOT

```
#Neste momento estamos colocando os parâmetros ao TPOT
tpot = TPOTRegressor(generations=10,population_size=50,verbosity=2,random_state=5, max_time
#Agora iremos fazer o Fit de nossos dados ao modelo.
tpot.fit(X_train,y_train)
print(tpot.score(X_test,y_test))
```

```
HBox(children=(HTML(value='Optimization Progress'), FloatProgress(value=0.0,
max=50.0), HTML(value='')))


183.46 minutes have elapsed. TPOT will close down.
TPOT closed during evaluation in one generation.
WARNING: TPOT may not provide a good pipeline if TPOT is stopped/interrupted
in a early generation.


TPOT closed prematurely. Will use the current best pipeline.

Best pipeline: KNeighborsRegressor(input_matrix, n_neighbors=73, p=1, weight
s=distance)
-15.477387381984013
```

Fonte: Elaboração própria (2022)..

Como apresentado pela Figura 23, o tempo de treino para encontro do melhor valor, utilizando do *dataset* de motores elétricos, foi de aproximadamente 3 horas e o algoritmo encontrado foi o *KNeighborsRegressor* com seus respectivos parâmetros. Com estes dados em mãos, foi feito a predição dos valores do rotor para os conjuntos de treino e teste e foram obtidos os resultados apresentados na Figura 24.

Figura 24 - Predição e resultado dos valores de treino e teste

```
print("Scores R2 de treino", sklearn.metrics.r2_score(y_train,Pred_train_y))
print("Scores R2 de teste", sklearn.metrics.r2_score(y_test,Pred_test_y))
```

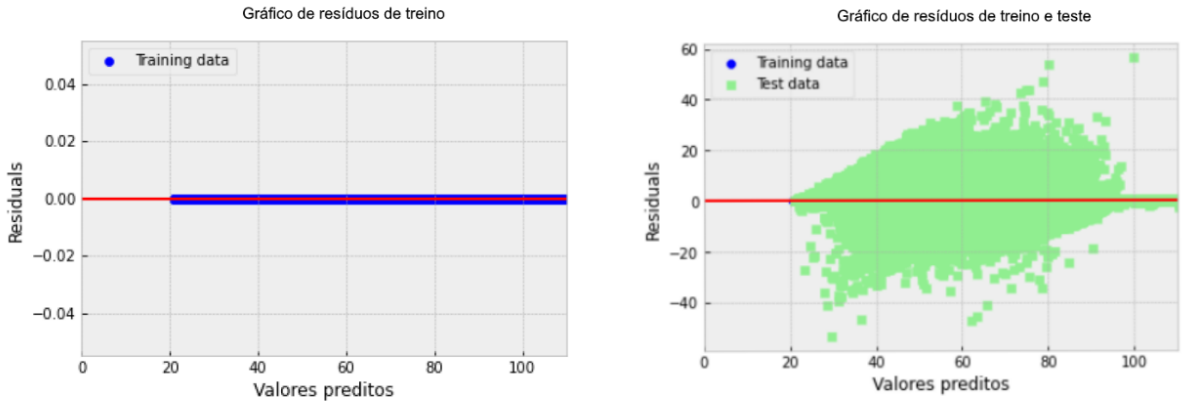
```
Scores R2 de treino 1.0
Scores R2 de teste 0.9574109366570978
```

Fonte: Elaboração própria (2022).

Então são apontados os valores do coeficiente de determinação para os

dados de treino e teste como sendo iguais a 1 e 0.9574. Após isto foi executado o gráfico de resíduos apresentado pela Figura 25.

Figura 25 - Gráficos residuais TPOT



Fonte: Elaboração própria (2022).

Pode-se notar que os gráficos de resíduos foram similares se comparados com os da Figura 21 em que se utilizou a biblioteca do *Auto-Sklearn.* Então foi feita a validação cruzada utilizando a função *Kfold* onde foram feitas a separação dos dados em 10 grupos assim como na validação com a biblioteca *Auto-Skleran.* O resultado pode ser visto pela Figura 26.

Figura 26 - Resultados da validação cruzada utilizando a biblioteca TPOT

```
knn_model= neighbors.KNeighborsRegressor(n_neighbors=73, p=1, weights='distance')
kfold  = KFold(n_splits=10, shuffle=True) # shuffle=True, Shuffle (embaralhar) the data.
result = cross_val_score(knn_model, X, y, cv = kfold)
rmse_cv=np.sqrt(-1*cross_val_score(knn_model,X,y ,cv=kfold,scoring='neg_mean_squared_error').mean())


print("K-Fold (R^2) Scores: {0}".format(result))
print("Média do R^2 para a validação cruzada K-Fold: {0}".format(result.mean()))
print("Erro quadrático médio", rmse_cv)
```

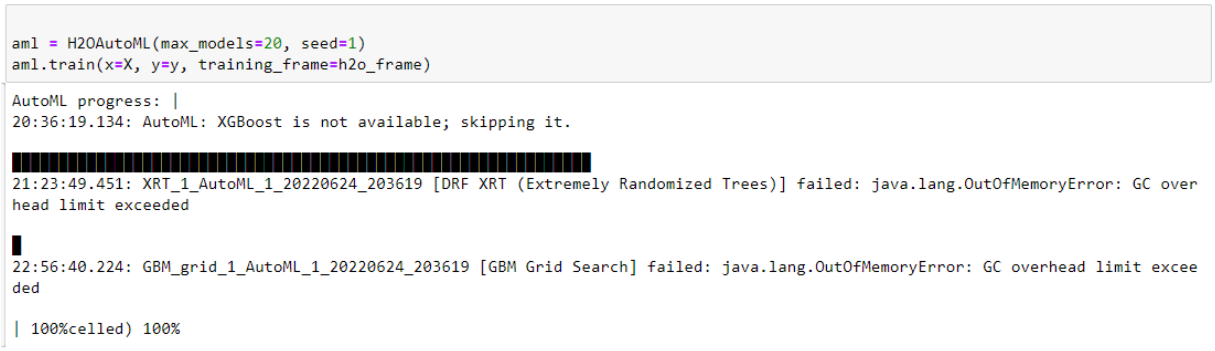
```
K-Fold (R^2) Scores: [0.94031931 0.9404739  0.94006137 0.94015522 0.93989519 0.93907088
 0.94074138 0.93973173 0.94198852 0.9412528 ]
Média do R^2 para a validação cruzada K-Fold: 0.9403690302339811
Erro quadrático médio 4.6368615650271945
```

Fonte: Elaboração própria (2022).

O resultado médio gerado pelo coeficiente de determinação foi de 0,9404

e o resultado da raiz quadrática média dos erros foi igual a 4,6369. Então foi executado o treinamento com a biblioteca H2O. Os parâmetros que foram utilizados são a limitação do máximo de modelos que poderiam ser treinados igual a 20 e o *seed,* que tem função de especificar números que são gerados de aleatória, igual a 1. O resultado do treino pode ser visto na Figura 27.

Figura 27 - Resultado encontrado pela biblioteca H2O



Fonte: Elaboração própria (2022).

Foram feitos testes com trocas de variáveis e especificados tempos de treino para o uso da biblioteca H2O, porém nenhuma dessas configurações foi capaz de gerar a predição de algoritmos para os testes alcançando o final de treino sem resultados.

4.1 Análise e discussão dos resultados

Um dos primeiros questionamentos do autor foi se realmente as parametrizações feitas pelos algoritmos de AutoML melhoravam os resultados no treinamento com o *dataset* de motores elétricos. Para o teste, o autor criou um modelo com o algoritmo *KNeighbors* encontrado pela biblioteca *Auto-Sklearn* sem parâmetros e outro modelo com os parâmetros fornecidos pelo modelo e fez a comparação dos coeficientes de determinação e raiz quadrada do erro quadrático médio (RMSE). Na Figuras 28 e 29 são apresentadas respectivamente as linhas de códigos e os resultados dos algoritmos após validação cruzada.

Figura 28 - Resultado da acurácia do algoritmo com parâmetro

```
knn_model= neighbors.KNeighborsRegressor(n_neighbors=2, p=1, weights='distance')
kfold  = KFold(n_splits=10, shuffle=True) # shuffle=True, Shuffle (embaralhar) the data.
result = cross_val_score(knn_model, X, y, cv = kfold)
rmse_cv=np.sqrt(-1*cross_val_score(knn_model,X,y ,cv=kfold,scoring='neg_mean_squared_error').mean())

print("Média do R^2 para a validação cruzada K-Fold: {0}".format(result.mean()))
print("Raiz quadrática do erro quadrático médio", rmse_cv)
```

```
Média do R^2 para a validação cruzada K-Fold: 0.9579376214412669
Raiz quadrática do erro quadrático médio 3.905702677535099
```

Fonte: Elaboração própria (2022).

Figura 29 - Resultados da acurácia do algoritmo sem parâmetros

```
knn_model= neighbors.KNeighborsRegressor()
kfold  = KFold(n_splits=10, shuffle=True) # shuffle=True, Shuffle (embaralhar) the data.
result = cross_val_score(knn_model, X, y, cv = kfold)
rmse_cv=np.sqrt(-1*cross_val_score(knn_model,X,y ,cv=kfold,scoring='neg_mean_squared_error').mean())

print("Média do R^2 para a validação cruzada K-Fold: {0}".format(result.mean()))
print("Raiz quadrática do erro quadrático médio", rmse_cv)
```

```
Média do R^2 para a validação cruzada K-Fold: 0.9471572268425781
Raiz quadrática do erro quadrático médio 4.374370851302784
```

Fonte: Elaboração própria (2022).

Na figura 28 é visto que os resultados encontrados para o erro quadrático médio e do coeficiente de determinação médio foram melhores se comparados com os resultados da acurácia do algoritmo sem parâmetros da figura 30, assim é visto que os parâmetros encontrados pela ferramenta de AutoML melhoraram o desempenho da regressão.

Com os resultados foi feita a análise de cada uma das bibliotecas e de seus achados. Tanto a biblioteca *Auto-Sklearn* quanto a biblioteca TPOT alcançaram, após seus respectivos treinos de mesmos intervalos, o mesmo algoritmo, o *KNeighborsRegressor,* porém apresentaram o parâmetro *n_neighbors* com valor diferente, sendo 73 para o TPOT e 2 para o *Auto-Sklearn.* Esta variação de parâmetro resultou em diferentes valores para a qualificação dos modelos e o que apresentou melhores resultados foi o encontrado a partir da biblioteca *Auto-Sklearn.* O valor do coeficiente de determinação encontrado pelo algoritmo sinalizado por esta biblioteca

foi de 0.9716 enquanto o algoritmo encontrado pela biblioteca TPOT teve seu resultado igual a 0,9574. Após isto, para maior confiança nos valores encontrados pelos modelos, foi executada a validação cruzada e o resultado do coeficiente de determinação diminuiu, porém seguiu o mesmo padrão, o modelo do *Auto-Sklearn* apresentando maior acurácia em seu algoritmo em relação ao algoritmo encontrado pelo modelo TPOT, estes valores comentados sobre a precisão dos modelos e seus valores de parâmetros podem ser consultados no Quadro 2. Com isto, o autor avaliou que os melhores resultados para este estudo foram os da biblioteca *Auto-Sklearn* porém levanta o ponto de restrição para sua utilização, a necessidade de um sistema operacional baseado em Linux.

Em relação ao não funcionamento da biblioteca H2O para geração de algoritmos, os quais fariam o treinamento e predição do *dataset*, seria necessário mais tempo para testes e mais conhecimento sobre a biblioteca, sobre Python e sobre *datascience* para que fosse possível entender o que ocorreu e o porquê da performance da mesma não ter sido a esperada. Levanta-se aqui a necessidade e possibilidade de um estudo para entendimento do acontecido.

Uma outra análise interessante foi verificar se os resultado do coeficiente de determinação encontrado pelo modelo *Auto-Sklearn,* após a validação cruzada, se mostravam compatíveis com os resultados dos algoritmos encontrados pelos usuários da plataforma *Kaggle.* Para isto, foi feita uma busca na plataforma por usuários que estavam utilizando o *dataset* do motor elétrico e predizendo o valor da temperatura do rotor, *‘pm’.* Com isto foram encontrados estudos em que apresentaram erros quadráticos médios (SAURAV, 2020; ASARKAR, 2020) menores dos que os encontrados pelo autor. Os resultados são apresentados pela tabela 2.

Quadro 2 - Resultados encontrados pelos algoritmos

Fonte de encontro do algoritmo	Algorítmo utilizado	Parâmetros	Raiz do erro quadrático médio (RMSE) após o *cross validation*	Coeficiente de determinação (R^2) após o *cross validation*
Autor - Auto-Sklearn	KNeighborsRegressor	n_neighbors=2 p=1 weights='distance'	3.8937	0,9574
Autor - TPOT	KNeighborsRegressor	n_neighbors=73 p=1 weights='distance'	4.6369	0,9716
Asarkar (2020)	*Ridge bag*	Base_estimator = ridge n_estimators = 2 Random_state = 0 n_jobs = -1	0,374	-
Saurav (2020)	Regressão linear	-	0,7722	-

Fonte: Elaboração própria (2022).

Como é apresentado pelo Quadro 2, os resultados das predições deste estudo não apresentaram os melhores valores se comparados com os outros estudos, porém a velocidade com que os modelos foram obtidos e a não necessidade de vasto conhecimento na área para encontro do algoritmo e seus parâmetros se apresentou como a vantagem do AutoML. É importante salientar que nos estudos de Saurav (2020) não foi utilizado da validação cruzada ou *Cross-validation,* podendo ser o fator para a diferença de precisão dos valores na medida do erro quadrático médio entre os resultados encontrados por ele e os deste estudo, sendo necessárias outras pesquisas para comprovar esta teoria.

É possível fazer um paralelo entre os resultados identificados neste estudo e aqueles encontrados nos artigos Eeden *et al.* (2021) na revisão do Capítulo 2. Os mesmos padrões de comportamento foram alcançados, ou seja, os resultados das predições encontradas pelas ferramentas do AutoML não apresentaram maior acurácia quando comparadas aos modelos encontrados de forma tradicional na plataforma *Kaggle* porém é importante ressaltar que estes valores de acurácia dos modelos encontrados pelo AutoML foram bons e não necessitaram de um vasto conhecimento na área de *datascience* para geração do algoritmo e seus parâmetros.

5 CONSIDERAÇÕES FINAIS

Este trabalho trouxe um breve histórico do surgimento da inteligência artificial e a partir de suas ramificações apresentou a área do *Machine Learning* e posteriormente o *Automated Machine Learning*, suas ferramentas, bibliotecas e divisões, com estas ferramentas fez-se a predição da temperatura do rotor de motores elétricos.

O AutoML foi explicado e foram apresentadas quais as etapas necessárias para o tratamento dos dados do *dataset*, bem como os programas utilizados e os testes realizados. Para entender a acurácia dos resultados gerados pelos modelos, esses foram comparados com outros trabalhos presentes na plataforma *Kaggle.* A partir disto foi feito o paralelo entre o comportamento dos resultados encontrados na pesquisa com os relatados nos artigos utilizados na revisão de literatura sendo observado o mesmo comportamento.

Para o pesquisador foi entendida como desafiadora a proposta inicial considerando a falta de conhecimento sobre a linguagem *Python* e sobre a ciência de dados, porém com a site *Kaggle,* as documentações das bibliotecas e o acervo digital trazido por meio da internet, foi possível atingir os objetivos e apresentar os caminhos tomados para encontro do modelo de predição utilizando as ferramentas de AutoML.

O planejamento elaborado para execução do trabalho foi realizado na integra e a partir dele, os objetivos traçados foram alcançados com sucesso, porém seriam necessárias mais pesquisas sobre as bibliotecas de AutoML, ferramentas de *datascience* e estatística aplicada na área da ciência de dados para aprofundamento da análise dos resultados. Uma outra observação importante se dá na não apresentação de resultados por parte da biblioteca H2O, na qual se sugere outros estudos buscando o entendimento do caso.

Em relação aos objetivos traçados para a pesquisa, conseguiu-se utilizar uma base de dados pública com relevância na área da Engenharia Mecatrônica. Foram aplicadas ferramentas do AutoML por meio de ambientes de programação largamente utilizados por cientistas de dados para geração de modelos de predição, no caso do presente estudo, predição da temperatura do rotor a partir do *dataset* de motores elétricos. Também foi realizada a avaliação dos resultados da acurácia dos mo-

delos gerados a partir do *dataset* de motores elétricos sendo todo este processo documentado desta forma atingindo-se os objetivos da pesquisa. Com o resultado final foi organizado um repositório no *GitHub* contendo os códigos e *dataset* utilizados,para mais fácil reutilização e aperfeiçoamento do mesmo.

LINK que será utilizado no Github: https://github.com/Pedrobum/AutoML_/tree/main/AutoML-main

O autor acredita que o resultado desta monografia foi satisfatório e que a proposta de um trabalho didático que apresenta os termos, um pouco do mundo do *datascience* e a utilização das ferramentas do AutoML na prática foi alcançado.

REFERÊNCIAS

ASARKAR, P. **PMSM Rotor Temperature Detection**. 2020. Disponível em: https://www.kaggle.com/code/pratikasarkar/pmsm-rotor-temperature-detection/notebook. Acesso em: 02 jul. 2022.

ALPAYDIN, E. **Introduction to Machine Learning**. 3rd. ed. London: The MIT Press, 2014.

AWS. **What is data labeling for machine learning?** 2022. Disponível em: https://aws.amazon.com/pt/sagemaker/data-labeling/what-is-data-labeling/. Acesso em: 27 jun. 2022.

BANGASH, A. H. *et al.* Amalgamation of Auto Machine Learning and Ensemble Approaches to Achieve State-of-the-Art Post-Heart Failure Survival Predictions. **American Heart Journal**, v. 242, p. 153, dez. 2021. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0002870321002660?via%3Dihub. Acesso em: 28 fev. 2022.

BALAJI, A; ALLEN, A. **Choosing the best AutoML Framework**. 2018. Disponível em: https://medium.com/georgian-impact-blog/automatic-machine-learning-aml-landscape-survey-f75c3ae3bbf2. Acesso em: 10 jul. 2022.

BÖCKER. **Electric Motor Temperature**. 2021. Disponível em: https://www.kaggle.com/datasets/wkirgsn/electric-motor-temperature. Acesso em: 28 jun. 2022.

BRUCE, P.; BRUCE, A. **Estatística Prática para Cientistas de Dados**. Rio de Janeiro: Alta Books, 2019.

CHICCO, D.; WARRENS, M. J.; JURMAN, G. The coefficient of determination R-squaredis more informative than SMAPE, MAE,MAPE, MSE and RMSE in regressionanalysis evaluation. **PeerJ Comput. Sci,** [s.l.], v. 7, p. e623, 2021. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8279135/. Acesso em: 28 jun. 2022.

COREA, F. **Studies in Big Data 50 An Introduction to Data Everything You Need to Know About AI, Big Data and Data Science**. v. 50. Switzerland: Springer, 2019.

DAS, S.; CAKMAK, U. M. **Hands-On Automated Machine Learning A beginner's guide to building automated machine learning systems using AutoML and Python.** Reino Unido: Packt, 2018.

DAVID. **Alternative to Colab Pro:** Comparing Google's Jupyter Notebooks to Gradient Notebooks. 2021. Disponível em: https://blog.paperspace.com/alternative-to-google-colab-pro/. Acesso em: 28 jun. 2022.

DEEP MIND. **AI could be one of humanity's most useful inventions**. 20-?. Disponível em: https://www.deepmind.com/about. Acesso em: 8 mai. 2022.

EEDEN, W. A. V. *et al.* Predicting the 9-year course of mood and anxiety disorders with automated machine learning: A comparison between auto-sklearn, naïve Bayes classifier, and traditional logistic regression. **Psychiatry Research,** v. 299, p. 113823, 2021. Disponível em: https://www.sciencedirect.com/science/article/pii/S0165178121001207?via%3Dihub Acesso em: 28 fev. 2022

FACELI, K. *et al.* **Inteligência artificial:** uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, 2011.

FRAJ, M. B. **InDepth:** Parameter tuning for Decision Tree. 2017. Disponível em: https://medium.com/@mohtedibf/indepth-parameter-tuning-for-decision-tree-6753118a03c3. Acesso em: 02 jul. 2022.

FUTURE OF LIFE INSTITUTE. **Research priorities for robust and beneficial artificial intelligence**. 2014. Disponível em: https://futureoflife.org/2015/10/27/ai-open-letter/. Acesso em: 9 mai. 2022.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. Massachusetts (USA): O'Reilly, 2017.

GOOGLE. **Colaboratory Frequently Asked Questions**. 2022. Disponível em: https://research.google.com/colaboratory/faq.html. Acesso em: 28 jun. 2022.

HE, X.; ZHAO, K.; CHU, X. AutoML: A survey of the state-of-the-art. **Knowledge-Based Systems,** [s.l.], v. 212, jan. 2021. Disponível em: https://www.sciencedirect.com/science/article/pii/S0950705120307516. Acesso em: 23 fev. 2022.

HURWITZ, J.; KIRSCH, D. **Machine Learning for dummies.** Nova Jersey (USA): IBM Limited Edition, 2018.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. **Automated Machine Learning Methods, Systems, Challenges.** 2017. Disponível em: https://link.springer.com/chapter/10.1007/978-3-030-05318-5_1. Acesso em: 22 fev. 2022.

ESCALANTE, H. **Automated Machine Learning** – a brief review at the end of the early years. 2022. Disponível em: https://arxiv.org/pdf/2008.08516.pdf Acesso em: 22 jul. 2022.

IBM CLOUD EDUCATION. **Natural Language Processing (NLP)**. 2020a. Disponível em: https://www.ibm.com/cloud/learn/natural-language-processing. Acesso em: 21 mai. 2022.

IBM CLOUD EDUCATION. **Unsupervised Learning**. 2020. Disponível em: ibm.com/cloud/learn/unsupervised-learning. Acesso em: 25 mai. 2022.

IBM. **What is computer vision?** 2022. Disponível em:

https://www.ibm.com/topics/computer-vision. Acesso em: 21 mai. 2022.

INTEROP. **Exemplos de Machine Learning aplicados ao dia a dia**. 2019. Disponível em: https://www.interop.com.br/blog/exemplos-de-machine-learning/. Acesso em: 27 mai. 2022.

ISLINGTON, M. **Creation of virtual environments**. 2021. Disponível em: https://github.com/python/cpython/blob/3.10/Doc/library/venv.rst. Acesso em: 02 jul. 2022.

KUMAR, A. **What is Supervised Learning and its different types?** 2020. Disponível em: https://www.edureka.co/blog/supervised-learning/. Acesso em: 02 jul. 2022.

KUMAR, C. **Artificial Intelligence:** Definition, Types, Examples, Technologies. 2018. Disponível em: https://chethankumargn.medium.com/artificial-intelligence-definition-types-examples-technologies-962ea75c7b9b. Acesso em: 11 mai. 2022.

MACHINE LEARNING PROFESSORSHIP FREIBURG. **Auto-sklearn**. 2022. Disponível em: https://automl.github.io/auto-sklearn/master/#example. Acesso em: 02 jul. 2022.

MÜLLER, A. C.; GUIDO, S. **Introduction to Machine Learning with Python.** A guide for data scientists. Massachusetts (USA): O'Reilly, 2017.

OLIVEIRA, L. **Teorema de Bayes & Probabilidade**. 2020. Disponível em: https://medium.com/data-hackers/teorema-de-bayes-probabilidade-d5ead2df1379. Acesso em: 02 jul. 2022.

OLSON, R. S. **Overview**. 2022. Disponível em: http://epistasislab.github.io/tpot/examples/#titanic-survival-analysis. Acesso em: 02 jul. 2022.

OXFORD UNIVERSITY PRESS. **Artificial intelligence**. 2022. Disponível em: https://www.oxfordlearnersdictionaries.com/definition/english/artificial-intelligence. Acesso em: 26 jun. 2022.

PETROVA, S. **This is How Companies are Applying Machine Learning to Remain Competitive**. 2022. Disponível em: https://adevait.com/machine-learning/companies-using-machine-learning. Acesso em: 10 jul. 2022.

RASCHKA, S. **Python machine learning:** unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics. Reino Unido: Packt, 2015.

RAY, Sunil. **6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R**. 2017. Disponível em: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. Acesso em: 02 jul. 2022.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence A Modern Approach Third**

Edition. 3rd. ed. Nova Jersey (USA): Prentice Hall, 2010 .

SALVARIS, M.; DEAN, D.; TOK, W. H. **Deep learning with azure: b**uilding and deploying artificial intelligence solutions on the microsoft AI platform. Califórnia (USA): Apress Media LLC, 2018.

SANTA CATARINA. IFSC. **Projeto Pedagógico do Curso Engenharia Mecatrônica**. 2012. Disponível em: http://cs.ifsc.edu.br/portal/files/florianopolis_PPC_engenharia_mecatronica.pdf. Acesso em: 10 jul. 2022.

SAURAV. **Exploratory Analysis of Electric Motor temperature**. 2020. Disponível em: https://www.kaggle.com/code/sauravmom/exploratory-analysis-of-electric-motor-temperature. Acesso em: 2 jul. 2022.

SCIKIT LEARN. **Sklearn.naive_bayes.BernoulliNB**. 2022. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html. Acesso em: 30 mai. 2022.

SCIKIT LEARN. **Sklearn.naive_bayes.MultinomialNB**. 2022a. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html. Acesso em: 30 mai. 2022.

SCIKIT LEARN. **Sklearn.naive_bayes.GaussianNB**. 2022b. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html. Acesso em: 30 mai. 2022.

SCIKIT LEARN. **RBF SVM parameters**. 2022c. Disponível em: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html. Acesso em: 30 mai. 2022.

SCIKIT LEARN. **Clustering**. 2022d. Disponível em: https://scikit-learn.org/stable/modules/clustering.html. Acesso em: 30 mai. 2022.

SEBESTA, R. W. **conceitos de linguagem de programação**. 9. ed. Porto Alegre: Artmed, 2010.

SINGH, G.; MISHRA, A.; SAGAR, D. An overview of artificial intelligence. **SBIT Journal of Sciences and Technology,** [s.l.], v. 2, n. 1, 2013.

STAHL, B. C. **Artificial Intelligence for a Better Future An Ecosystem Perspective on the Ethics of AI and Emerging Digital Technologies**. Reino Unido: Springer, 2021.

NAGARAJAH, T.; PORAVI, GUHANATHAN **A Review on Automated Machine Learning (AutoML) Systems**. 2019. Disponível em: http://dlib.iit.ac.lk/xmlui/bitstream/handle/123456789/416/Naga.pdf?sequence=1&isAll

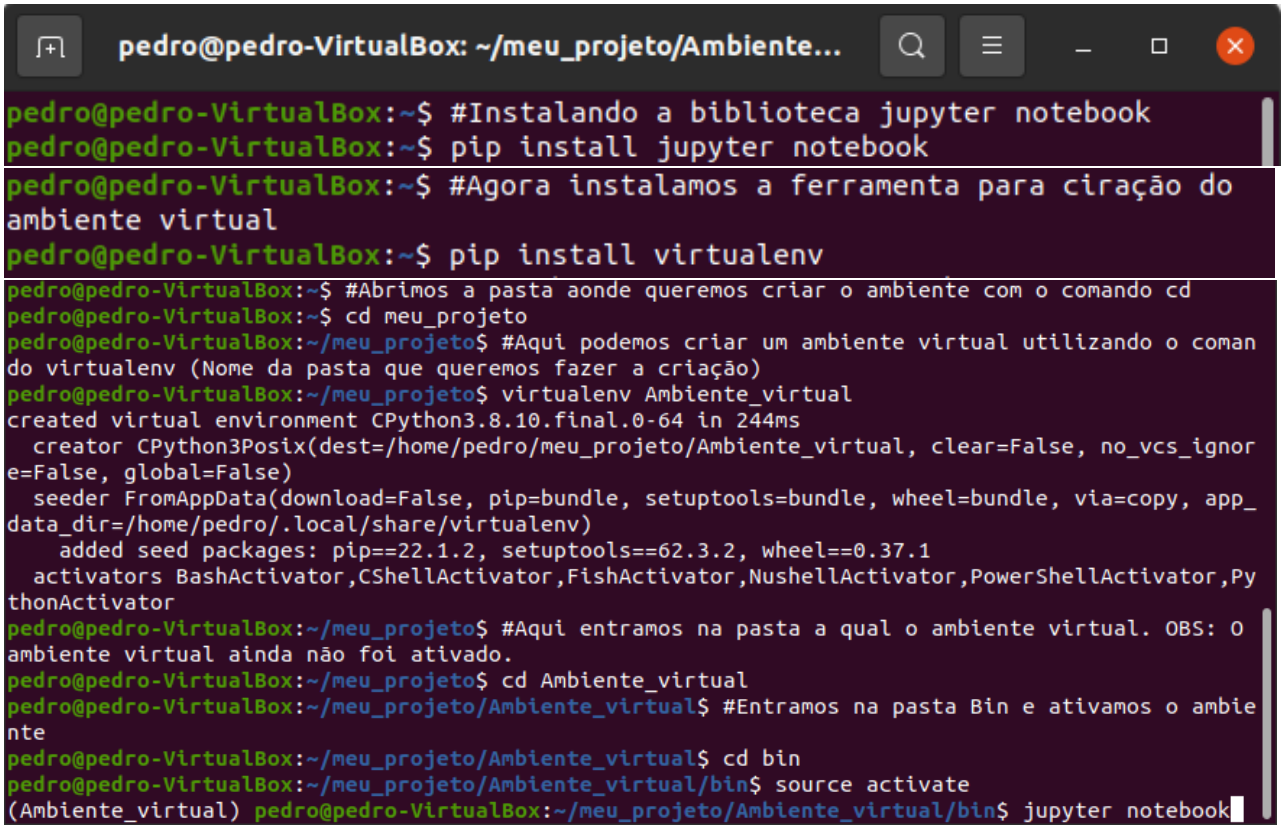
owed=y. Acesso em: 27 jul. 2022.

USMANI, Z. U. H. **What is Kaggle, Why I Participate, What is the Impact?** 2017. Disponível em: https://www.kaggle.com/getting-started/44916. Acesso em: 02 jul. 2022.

VENKATA, V. P. D. *et al.* Automating water quality analysis using ML and auto ML techniques. **Environmental research,** [s.l.], v. 202, p. 111720, 2021. Disponível em: https://pubmed.ncbi.nlm.nih.gov/34297938/ Acesso em: 28 fev. 2022

APÊNDICES

APÊNDICE A – Criação da máquina virtual e instalação do Jupyter Notebook



APÊNDICE B – Leitura do dataset pela plataforma Colab

```
[ ]  1  from google.colab import drive
     2  drive.mount("/content/drive")
     3
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ]  1  import io
     2  df=pd.read_csv('/content/drive/MyDrive/meu_projeto/meu_projeto_env/bin/measures_v2.csv', usecols=[0,1,2,3,4,5,6,7,8,9,10,11])
     3
```

APÊNDICE C – Leitura, tratamento e divisão de treino e teste do dataset

```
import pandas as pd #Biblioteca para manipulação e análise de dados
```

```
df=pd.read_csv('measures_v2.csv', usecols=[0,1,2,3,4,5,6,7,8,9,10,11])
```

In [3]:
```
df.head()
```

Out[3]:

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197

In [10]:
```
target = df.pop('pm') #Temperatura do rotor
```

In [11]:
```
df = pd.concat([df, target], axis=1)
```

In [12]:
```
df = df.sample(frac=1,random_state=0) #embaralha os dados do dataframe #Ajuda a previnir o overfitting
df.reset_index(drop=True, inplace=True) #Faz com que o Index volte a ser o que era antes
df
```

Out[12]:

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	stator_yoke	ambient	torque	pm
0	41.938923	18.744030	66.684830	-123.478027	46.080647	4749.964355	-187.964111	77.635490	33.169899	20.551407	7.486735e+01	40.030872
1	-0.431508	59.902590	85.079312	-0.878644	76.299257	0.057160	-2.000745	1.097449	67.216376	24.467134	-2.614878e-06	72.503215
2	-1.541598	33.149664	48.669293	-0.333442	45.330586	0.001482	-2.000673	1.097664	39.476950	23.169348	5.436636e-14	43.026863
3	42.387482	44.949261	104.791174	-123.337533	90.274398	5112.368164	-181.587703	68.767176	72.123242	27.374628	6.631759e+01	74.825417
4	15.335679	18.755226	113.366333	-130.067474	84.144737	3999.963135	-205.157623	97.281723	57.477020	23.940556	9.412620e+01	89.903763
...	...	...	...	...	...	...	...	...	...	...	...	...
1330811	12.093378	18.362038	19.795088	0.766441	19.273512	249.997833	-1.999840	1.097852	19.066999	22.962929	1.049427e-01	25.814903
1330812	23.644573	18.671892	39.454746	-22.069843	32.786079	499.995819	-43.512096	132.618576	27.362280	23.022329	1.033085e+02	35.359821
1330813	31.839993	54.416758	102.315358	56.278286	88.524355	1096.309584	-62.842545	-174.414836	75.441479	27.976668	-1.377123e+02	82.917509
1330814	94.615028	18.818180	36.839790	-1.240177	36.204067	1999.979004	-1.999915	1.098047	30.371397	23.878441	-4.387210e-01	68.415085
1330815	-1.087644	18.247478	20.067276	1.469395	19.850620	-0.004758	-1.999998	1.099235	19.825804	25.826746	5.552618e+00	46.172703

In [13]:
```
split_index=int(len(df) * 0.75)

train_df = df[:split_index] #Primeiros 75%
test_df = df[split_index:] #outros 25% restantes

train_df.info()
test_df.info()
```

Retira a última coluna que é no target do modelo de treinamento e modelos de teste

In [14]:
```
X_train = train_df.to_numpy()[:, :-1]
y_train = train_df.to_numpy()[:, -1]

X_test = test_df.to_numpy()[:, :-1]
y_test = test_df.to_numpy()[:, -1]
```

APÊNDICE D – Utilização da biblioteca autosklearn

```
!pip install scikit-learn==0.23.2
!pip install auto-sklearn
```

```
import autosklearn.regression
import sklearn.metrics
from pprint import pprint

autosklearn_regressor = autosklearn.regression.AutoSklearnRegressor(
```

```
[ ] autosklearn_regressor.fit(X_train,y_train)
```

APÊNDICE E – Utilização da biblioteca TPOT

```
[ ]  !pip install tpot
```

```
from tpot import TPOTRegressor
```

```
#Neste momento estamos colocando os parâmetros ao TPOT
tpot = TPOTRegressor(generations=10,population_size=50,verbosity=2,random_state=5)
#Agora iremos fazer o Fit de nossos dados ao modelo.
tpot.fit(X_train,y_train)
```

APÊNDICE F – Utilização da biblioteca H2O

```
import h2o
from h2o.automl import H2OAutoML
h2o.init()
```

```
h2o_frame=h2o.H2OFrame(train_df)
```

```
X=h2o_frame.columns
y='pm'
X.remove(y)
```

```
aml = H2OAutoML(max_models=20, seed=1)
aml.train(x=X, y=y, training_frame=h2o_frame)
```

APÊNDICE G – Treino com a biblioteca autosklear

```
6   autosklearn_regressor = autosklearn.regression.AutoSklearnRegressor(
7       per_run_time_limit=200,
8
9   )
```

```
1   autosklearn_regressor.fit(X_train,y_train)
```