



UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

GILGLEISON PAULINO LIMA

**MOMENT - SISTEMA DE APRENDIZADO DE MÁQUINA PARA MANUTENÇÃO
PREDITIVA**

QUIXADÁ
2021

GILGLEISON PAULINO LIMA

MOMENT - SISTEMA DE APRENDIZADO DE MÁQUINA PARA MANUTENÇÃO
PREDITIVA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Wagner Guimarães Al-Alam

QUIXADÁ

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L698m Lima, Gilgleison Paulino.
MOMENT : sistema de aprendizado de máquina para manutenção preditiva / Gilgleison Paulino Lima. –
2021.
48 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá,
Curso de Engenharia de Computação, Quixadá, 2021.
Orientação: Prof. Dr. Wagner Guimarães Al-Alam.

1. Aprendizado do computador . 2. Manutenção preditiva. I. Título.

CDD 621.39

GILGLEISON PAULINO LIMA

MOMENT - SISTEMA DE APRENDIZADO DE MÁQUINA PARA MANUTENÇÃO
PREDITIVA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: ____/____/____

BANCA EXAMINADORA

Prof. Dr. Wagner Guimarães Al-Alam (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Elvis Miguel Galeas Stancanelli
Universidade Federal do Ceará (UFC)

Prof. Dr. Criston Pereira de Souza
Universidade Federal do Ceará (UFC)

RESUMO

Este trabalho apresenta o desenvolvimento do MOMENT (Sistema de Aprendizado de Máquina para Manutenção Preditiva), um sistema capaz de identificar e indicar se o filtro de ar do carro precisa ou não de manutenção, a partir de uma abordagem de manutenção preditiva, e assim evitando os riscos e altos custos gerados pelas manutenções corretivas, e o desperdício gerado pelas manutenções preventivas. A coleta dos dados é feita por uma aplicação capaz de ler em tempo real os dados gerados pelos sensores presentes no carro. Esses dados são lidos a partir do sistema de diagnóstico eletrônico do automóvel, e então aplicados em um algoritmo de aprendizado de máquina para que seja possível detectar pequenas variações nos padrões de funcionamento do carro, diferenciando um funcionamento normal de um comportamento com avaria iminente, identificando assim que a peça deve ser substituída, ou precisa de manutenção. Esse monitoramento é voltado à injeção eletrônica.

Palavras-chave: Aprendizado de máquina. Manutenção preditiva.

ABSTRACT

This work presents the development of the MOMENT (Sistema de Aprendizado de Máquina para Manutenção Preditiva), a system capable of identifying and indicating whether the car's air filter needs maintenance or not, from a predictive maintenance approach, thus avoiding the risks and high costs generated by corrective maintenance, and the waste generated by preventive maintenance. The data collection is done by an application capable of reading in real time the data generated by the sensors present in the car. This data is read from the car's electronic diagnostic system, and then applied to a machine learning algorithm so that it is possible to detect small variations in the car's operating patterns, differentiating normal operation from behavior with imminent damage, identifying so the part must be replaced, or needs maintenance. This monitoring is aimed at electronic injection.

Keywords: Machine learning. Predictive maintenance.

LISTA DE ILUSTRAÇÕES

Figura 1 – Classificação binária	17
Figura 2 – Regressão	18
Figura 3 – Interações aprendizado por reforço	19
Figura 4 – Agrupamento	20
Figura 5 – Redução de Dimensionalidade	20
Figura 6 – <i>Overfitting</i> , <i>underfitting</i> e caso ideal de treinamento de modelos	21
Figura 7 – <i>Exemplo de classificação com KNN</i>	23
Figura 8 – Árvore de decisão para seleção de uma atividade diária	24
Figura 9 – Classificação binária com SVM	25
Figura 10 – <i>Holdout cross-validation</i>	26
Figura 11 – Validação cruzada com $k = 3$	27
Figura 12 – Forma geral de uma matriz de confusão	27
Figura 13 – Exemplo de uma matriz de confusão	28
Figura 14 – Motor 4 tempos	30
Figura 15 – Diagrama injeção eletrônica	31
Figura 16 – Conector J1962	34
Figura 17 – Estrutura do código do DTC	36
Figura 18 – Importância das <i>features</i> selecionadas para o modelo	39
Figura 19 – Matriz de confusão para os dados de validação do algoritmo SVC	43
Figura 20 – Matriz de confusão para os dados de validação do algoritmo RF	43
Figura 21 – Matriz de confusão para os dados de validação do algoritmo KNN	44
Figura 22 – Primeira avaliação do protótipo	45
Figura 23 – Segunda avaliação do protótipo	45
Figura 24 – Primeiro caso de erro	45
Figura 25 – Segundo caso de erro	46

SUMÁRIO

1	INTRODUÇÃO	8
1.1	Objetivos	10
1.2	Trabalhos Relacionados	10
2	FUNDAMENTAÇÃO TEÓRICA	14
2.1	Manutenções	14
2.2	Aprendizado de Máquina	16
2.2.1	<i>Categorias de Aprendizado de Máquina</i>	16
2.2.2	<i>Overfitting e Underfitting</i>	21
2.2.3	<i>Métodos de normalização de dados</i>	21
2.2.4	<i>Algoritmos de aprendizado supervisionado</i>	22
2.2.5	<i>Validação e avaliação de modelos</i>	26
2.3	Injeção Eletrônica	29
2.3.1	<i>Engine Control Unit (ECU)</i>	33
2.3.2	<i>On Board Diagnostics (OBD)</i>	33
2.3.3	<i>Diagnostic Trouble Codes (DTCs)</i>	36
2.3.4	<i>Parameter IDs (PIDs)</i>	37
3	MOMENT	38
4	CONCLUSÕES E TRABALHOS FUTUROS	47
4.1	Trabalhos futuros	47
	REFERÊNCIAS	48

1 INTRODUÇÃO

Atualmente os carros não são mais puramente mecânicos, pois evoluíram muito com o passar dos anos. Tendo muitas de suas funções sendo controladas computacionalmente, desde sistemas que proveem o funcionamento principal, como motor e injeção eletrônica, até sistemas voltados ao conforto como multimídia e ar-condicionado.

Esses sistemas são compostos por sensores, ECUs (*Engine Control Unit*) e atuadores. Sensores são dispositivos capazes de ler grandezas físicas reais e transformá-las em sinais elétricos. A partir do momento em que o carro está em funcionamento eles são verificados constantemente. Pode-se dizer que as ECUs são o cérebro do automóvel, sendo responsáveis por toda a parte computacional e controlando quase todos os sistemas do carro. Atuadores são dispositivos que através de informações de sinais elétricos, são capazes de coordenar e ajustar outros dispositivos através de informações enviadas pela ECU. Os sistemas são capazes de se adaptar em tempo real de acordo com o envio dos dados dos sensores para a ECU, se baseando em informações salvas em memória. Os cálculos necessários são feitos, e então as informações são enviadas da ECU para os atuadores.

A injeção eletrônica funciona de modo a coletar informações de sensores e os envia para a ECU, que faz cálculos baseados em mapas armazenados em memória, em seguida envia as informações necessárias para os atuadores para que os ajustes sejam feitos visando uma melhor eficiência, definindo a melhor mistura ar-combustível e respeitando leis estabelecidas para poluentes (BOSCH, 2005; SOCORRO, 2017). Durante esse processo, as ECUs também verificam se as leituras são condizentes com o esperado ou se caracterizam alguma falha. Caso uma falha seja detectada, a mesma é armazenada em memória em formato de códigos de erro chamados de DTCs (*Diagnostic Trouble Codes*) para que sejam lidos posteriormente em uma manutenção ou pelo proprietário do carro através de *scanners* que são encontrados facilmente no mercado, e em alguns casos são reportadas ao condutor através de uma luz indicativa de falha no painel.

Apesar de o carro ser capaz de fazer ajuste no funcionamento de seus sistemas em tempo real, a capacidade de detecção de erros é muito limitada, dado que uma falha só é detectada e reportada quando já veio a prejudicar o automóvel de alguma forma.

A correção e substituição de peças depois que uma falha ocorreu, seja porque foi detectada pelo sistema do carro ou porque o erro se tornou perceptível ao motorista, é chamada manutenção corretiva. Esse tipo de abordagem além de ser caro, pois um mau funcionamento de

uma peça pode acarretar danos a outras partes do carro, também é perigosa, visto que um mau funcionamento do carro pode acarretar acidentes.

Um tipo de manutenção que evita os ônus gerados pela manutenção corretiva, é a manutenção preventiva, que é a manutenção e troca de peças do carro em intervalos de tempo regulares. Embora seja uma melhor opção que a manutenção corretiva, essa manutenção pode gerar desperdícios de peças, visto que o desgaste das peças pode variar de acordo com o uso, podendo acontecer antes do prazo previsto ou levando a troca de uma peça em pleno estado de funcionamento.

Por fim temos a manutenção preditiva, que é definida como o ato de inferir sobre o estado futuro de algo a partir do monitoramento do histórico e do estado atual. Essa manutenção possibilita a troca da peça no momento correto, dessa forma evitando os malefícios das manutenções corretiva e preventiva. Os carros modernos possuem uma interface de diagnóstico eletrônico padronizada, chamada de OBD2 (*On Board Diagnostics 2*). Uma das especificações estabelecidas pela padronização é a interface física, por onde é possível conectar dispositivos externos e ter acesso os dados provenientes dos sensores em tempo real ou ler os códigos de erro armazenados em memória. Isso nos deixa muitas possibilidades, pois através dessas informações é possível sabermos muito sobre o estado atual do veículo, que nos possibilita trabalhar com os dados lidos em diferentes aplicações.

Atualmente também existe uma área de pesquisa que está cada vez mais presente no nosso cotidiano, o aprendizado de máquina, este é caracterizado pela capacidade de um programa de computador de aprender e tomar decisões com base em experiências adquiridas, ou seja, algoritmos de aprendizado de máquina conseguem aprender a partir de informações que lhe foram apresentadas.

Este trabalho apresenta o desenvolvimento do MOMENT (Sistema de Aprendizado de Máquina para Manutenção Preditiva), um sistema de aprendizado de máquina para manutenção preditiva de veículos a partir da leitura dos sensores. Esse monitoramento é voltado ao filtro de ar do motor. Partindo da leitura dos dados dos sensores usados para treinar, validar e testar, os algoritmos de aprendizado de máquina, e o protótipo desenvolvido. O protótipo consegue fornecer um método confiável para avaliação da peça, identificando se a peça está em bom estado ou precisa de manutenção.

1.1 Objetivos

O objetivo principal desse trabalho, é desenvolver um sistema confiável capaz de identificar se uma peça do carro deve ou não ser substituída, partindo de uma abordagem de manutenção preditiva, assim evitando os riscos e altos custos gerados pelas manutenções corretivas, e o desperdício gerado pelas manutenções preventivas. Os dados são lidos em tempo real por meio de aplicativos que se conectam aos *scanners* usados na interface OBD2, e enviados para um site, para ser acessado posteriormente. Ao fim, os dados lidos são aplicados em um algoritmo de aprendizado de máquina para ser possível detectar pequenas variações nos padrões de funcionamento do carro, diferenciando o funcionamento normal de uma peça, do funcionamento de uma peça que precisa de manutenção, neste trabalho o componente do carro abordado é o filtro de ar do motor.

Objetivos específicos:

1. Fazer a leitura das informações geradas pelos sensores.
2. Identificar os sensores que produzem dados relevantes para a predição da falha.
3. Identificar o algoritmo de aprendizado de máquina que obterá o melhor resultado com os dados do problema proposto.
4. Desenvolver um protótipo capaz de detectar se uma peça precisa ou não de manutenção.

1.2 Trabalhos Relacionados

Esta seção aborda as obras que se relacionam de alguma forma com o presente trabalho, seja pela área de estudo, tipo de aplicação ou abordagem. Entre os trabalhos descritos estão abordagens de manutenção preditiva, aprendizagem de máquina, e análise ou estudo dos dados lidos dos sensores do carro.

Kriebe *et al.* (2019) propuseram uma abordagem onde buscam encontrar novos padrões de DTCs (*Diagnostic Trouble Codes*) com uma combinação de técnicas de clusterização e lógica booleana, pois apesar dos DTCs salvarem dados das condições em que aconteceu, não revelam as reais razões que levaram ao erro. Novos padrões de DTCs são encontrados de acordo com regras, caso vários DTCs aconteçam na mesma semana e em um intervalo de 50 km eles serão do mesmo grupo, para isso é usada uma abordagem de aprendizado não supervisionado que permite que os DTCs classificados no mesmo grupo estejam agrupados durante o aprendizado. Após o agrupamento são observados os padrões encontrados. Essa técnica permite que padrões

recorrentes sejam facilmente detectados. Como semelhança com o presente trabalho, também classificaremos padrões de funcionamento, porém a intenção é diagnosticar a falha antes que ela seja detectada e um DTC seja gerado.

Em Shafi *et al.* (2018) é apresentada uma abordagem para previsão de falhas para quatro subsistemas de um veículo: sistema de combustível, sistema de ignição, sistema de exaustão e sistema de refrigeração. Códigos DTCs (*Diagnostic Trouble Codes*) são transmitidos ao *smartphone* via *bluetooth* e depois enviado para o servidor. Os dados eram analisados usando quatro classificadores de aprendizado de máquina: KNN, *Random Forest*, *Decision Tree* e SVM. Os padrões aprendidos com esses classificadores eram usados para detectar falhas em outros veículos. A comparação de precisão de todos os classificadores foram realizadas com base nas curvas ROC (*Receiver Operating Characteristics*). A abordagem dos autores tinha como objetivo final aumentar a tempo de funcionamento de veículos sendo feitos testes em 70 veículos Corolla, da marca Toyota. Este artigo aborda a previsão de falhas a partir da análise de DTCs, diferentemente da abordagem de leitura e aplicação dos dados dos sensores, como é proposto pelo presente trabalho.

O artigo Kowalik (2018) apresenta uma proposta onde é usado um carro com defeito no termostato responsável por liberar o líquido de refrigeração do motor. O monitoramento e coleta de dados é feita com o uso de um ELM327 que é um dispositivo que se comunica com a porta OBD2 do carro, e as informações são enviadas para um celular. Algumas variáveis foram observadas, como temperatura do líquido de arrefecimento do motor e porcentagem de carga do motor. Para efeito de comparação, as características de funcionamento são comparadas com as de um carro com um bom funcionamento do termostato. Como conclusões é relatado que é quase impossível prever esse tipo de falha baseado em qualquer dado lido da interface de diagnóstico. O presente trabalho também tem como característica diagnosticar a falha, porém o diagnóstico será voltado ao filtro de ar do motor indicando quando o mesmo precisará de manutenção, diferente do trabalho descrito que aborda um sistema já defeituoso.

O trabalho proposto por Goyal *et al.* (2020) trata a previsão de falhas em veículos focando em 4 subsistemas do carro, combustível, ignição, exaustão e sistema de resfriamento, usando uma arquitetura IOT de 3 camadas, onde a primeira camada é constituída por uma *raspberry pi*, um *scanner* OBD2 e alguns sensores propostos pelo autor, a segunda camada é o processamento de dados na nuvem e a terceira trata notificações para o usuário. Além dos sensores já presentes no carro o autor propõe mais 3 sensores, que são: um sensor de vibração

usado para detectar rachaduras no eixo do veículo, um sensor de gás e um sensor de flutuação usado para medir o nível do líquido de arrefecimento. A *raspberry pi* é responsável tanto por ler os sensores propostos pelo autor, quanto por receber os DTCs provindos da interface OBD2 do veículo que são lidos usando um *scanner* OBD2 LM327 e recebidos pela *raspberry* via *bluetooth*. Os dados lidos pela *raspberry* são enviados para o servidor para que sejam processados, e caso alguma anomalia seja detectada o usuário será notificado. Além dos aspectos do sistema já descritos, o autor propõe um aplicativo onde o usuário pode verificar a situação do veículo em uma linguagem acessível, além de propor uma listagem dos mecânicos mais próximos basando-se na localização do veículo no caso de detecção de anomalias.

No trabalho de Massaro *et al.* (2020) é focado no *design* e desenvolvimento de um sistema eletrônico compacto e inteligente para o monitoramento de uma frota de ônibus, fornecendo previsão sobre desgaste do veículo e sendo capaz de prever a manutenção para cada veículo classificando o comportamento do motorista. O algoritmo *k-means* capaz de fornecer *clusters* de motoristas indicando os comportamentos corretos e inadequados, divididos nas categorias 0, 1 e 2. A primeira é composta por motoristas que viajam baixas velocidades e aceleram lentamente, esse é tido como o comportamento mais eficiente na condução. A segunda é descrita como sendo uma categoria menos eficiente sendo de motoristas que viajam com baixa velocidade, mas forçando o motor. A terceira e ultima categoria é descrita como sendo o estilo de direção de mais contribui para o desgaste do veículo, mas não são atribuídas características a essa categoria. A rede neural artificial perceptron multicamadas (MLP-ANN) é capaz de prever a manutenção de cada veículo classificando o comportamento do motorista. Com base no estresse do motor, que é uma função do comportamento do motorista é possível replanejar a manutenção programada de cada veículo antecipando a manutenção do ônibus em caso de previsão de um alto desgaste do motor. Com base na análise dos resultados, são definidas metodologias de indicadores-chave de desempenho (KPIs), correlacionando o comportamento do motorista com o estresse do motor, definindo os critérios do plano de manutenção do ônibus. Todos os resultados são reunidos em uma plataforma em nuvem mostrando painéis de eficiência da frota.

A Tabela 1 mostra a comparação entre as diferentes abordagens dos trabalhos relacionados e do trabalho proposto. Para atender ao requisito de análise de DTC, indica que o trabalho foi baseado na leitura e análise de DTCs gerados pelo carro. O requisito de análise dos sensores, que dizer que a abordagem usada no trabalho usa os dados dos sensores. Aprendizado de máquina, indica que o trabalho usou algum algoritmo de aprendizado de máquina para análise

dos dados. Manutenção preditiva, indica que o trabalho usou alguma abordagem para prognóstico de falha nas peças do veículo. Motorista, indica que faz o monitoramento do comportamento de direção do motorista. E novos sensores, indica que o autor adiciona novos além dos que já estão presentes no carro para fazer análises.

Quadro 1 – Comparação das abordagens para cada trabalho relacionado e trabalho proposto

Trabalho	Análise de DTC	Análise dos sensores	Aprendizado de máquina	Manutenção preditiva	Motorista	Novos sensores
Kriebe <i>et al.</i> (2019)	X	-	X	-	-	-
Shafi <i>et al.</i> (2018)	X	-	X	X	-	-
Kowalik (2018)	-	X	X	-	-	-
Goyal <i>et al.</i> (2020)	-	X	X	X	-	X
Massaro <i>et al.</i> (2020)	X	-	X	X	X	-
Trabalho proposto	-	X	X	X	-	-

Fonte: Autoria própria

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são apresentados os principais conceitos abordados no trabalho. Na seção 2.1 serão apresentados os principais tipos de manutenções e as diferenças entre elas, bem como as vantagens da manutenção preditiva que é o principal conceito abordado neste trabalho. A seção 2.2 apresenta o aprendizado de máquina que será responsável por encontrar padrões no funcionamento que nos possibilitará prever as falhas. Por sua vez, a seção 2.3 fala sobre a injeção eletrônica que proverá os dados necessários para a análise e aplicação nos algoritmos de aprendizado de máquina. Esses dados serão lidos através do sistema de diagnóstico eletrônico presente no carro através da porta OBD2 que é a interface padrão para leitura dos dados.

2.1 Manutenções

As formas de manutenção são classificadas em manutenção corretiva, manutenção preventiva e manutenção preditiva. A manutenção corretiva pode ser dividida em duas categorias, que são, manutenção corretiva planejada e não planejada, onde a manutenção corretiva não planejada é aquela que é feita após uma falha acontecer. Geralmente o número de defeitos tende a aumentar com o passar do tempo caso a falha não seja corrigida o quanto antes. Esse tipo de manutenção é considerado o mais caro, uma vez que o equipamento precisa parar para que a manutenção seja feita. Esse tipo de manutenção tende a ficar mais caro à medida que o equipamento envelhece (SILVA NETO; LIMA, 2002). Esta é uma boa opção quando o custo para evitar a falha é maior que o custo de conserto da mesma (MARCORIN; LIMA, 2003). Diferentemente, a manutenção corretiva planejada se faz com uma intervenção planejada através de acompanhamento preditivo (XAVIER, 2005a).

Já a manutenção preventiva se baseia em intervenções periódicas em um equipamento, os intervalos de tempo são geralmente definidos pelo fabricante. Essa manutenção pode gerar desperdícios de peças, visto que o desgaste das peças de um equipamento pode variar, levando a troca de uma peça em pleno estado de funcionamento. Mas por ser uma checagem anterior a falha, e conseqüentemente a parada do equipamento, ela já é de certa forma uma melhor alternativa que a manutenção corretiva. Sendo uma boa alternativa no caso em que a troca precoce da peça tenha um menor custo que a parada do equipamento em decorrência da falha (MARCORIN; LIMA, 2003).

Por fim, a manutenção preditiva faz o uso do monitoramento de variáveis com o

intuito de inferir sobre um estado futuro da máquina, ou seja, uma forma de fazer prognóstico e descobrir futuras falhas. Esse tipo de manutenção se coloca como o de maior custo-benefício, uma vez que a parada devido a uma falha pode trazer grandes perdas (MARCORIN; LIMA, 2003). Esse tipo de manutenção também nos permite que o equipamento se mantenha em funcionamento pelo maior tempo possível, fazendo manutenção ou troca de peças apenas quando realmente for necessário, diferente da manutenção preventiva, onde a troca de peças ocorre em decorrência do tempo, muitas vezes não sendo necessária (XAVIER, 2005b).

O monitoramento da manutenção preditiva pode ser feito de duas formas distintas, sendo elas, monitoração subjetiva ou monitoração objetiva. A monitoração subjetiva pode ser caracterizada como feita pelos sentidos humanos, como por exemplo, ao sentir a vibração gerada por algum equipamento. Já a monitoração objetiva é o monitoramento feito através de equipamentos específicos, que fornecerá valores exatos. Ela pode ser classificada de duas formas, continua que é a medição constante, e pontual em intervalos definidos(XAVIER, 2005b).

Algumas características devem ser preservadas para qualquer técnica de manutenção preditiva: permitir que a coleta de dados seja feita com o equipamento em funcionamento ou com o mínimo de interferência possível, e permitir a coleta dos dados para que seja possível fazer análises; gerar a menor interferência possível, embora algumas técnicas não sigam essas regras (XAVIER, 2005b).

As conclusões sobre qual o melhor tipo de manutenção em relação de custo pode ser observada com o auxílio da Tabela 1, que mostra uma relação dos preços de cada tipo de manutenção.

Tabela 1 – Comparação dos custos das manutenções.

Tipo de manutenção	Custo US\$/HP/ANO
Manutenção corretiva não planejada	17 a 18
Manutenção preventiva	11 a 13
Manutenção preditiva + Manutenção corretiva planejada	7 a 9

Nota: HP (*Horse Power*) é a potência do equipamento

Fonte: Adaptado de (XAVIER, 2005b)

Com o auxílio da Tabela 1 podemos ver que o custo da manutenção preditiva é 50% menor quando comparado com o custo da manutenção preventiva. Já se compararmos com a

manutenção corretiva não planejada essa diferença é significativa, com a manutenção preditiva tendo metade do custo.

2.2 Aprendizado de Máquina

Aprendizado de máquina é uma área de IA (Inteligencia Artificial), vista como a capacidade de um programa de computador de aprender e tomar decisões com base em experiências adquiridas (MONARD; BARANAUSKAS, 2003). Na IA existem duas linhas de pensamento, o indutivo e o dedutivo, com aprendizado de máquina nos preocuparemos apenas com o indutivo (CORCOVIA; ALVES, 2019). Sendo a indução a capacidade de inferir sobre algo a partir de conhecimentos prévios sobre esse determinado assunto, essa informação deduzida pode ser ou não verdade. O aprendizado indutivo é feito a partir de exemplos fornecidos, porém se os exemplos escolhidos não forem adequados ou se o número de exemplos forem insuficientes isso pode não gerar bons resultados, mesmo assim, a indução é um dos principais métodos para obtenção de novos conhecimentos. (MONARD; BARANAUSKAS, 2003)

O aprendizado de máquina, além de não exigir que humanos gerem regras para analisar manualmente grandes quantidades de dados, também nos dá um método mais eficiente para aprender com essas informações, tornando cada vez mais confiáveis os modelos preditivos, tomando decisões baseados nesses dados. O aprendizado de máquina está se tornando mais importante não apenas na pesquisa científica, mas também está cada vez mais presentes nas nossas vidas, como por exemplo em filtros de *spam* em *e-mails*, reconhecimentos de voz e texto, entre outras aplicações (RASCHKA; MIRJALILI, 2017).

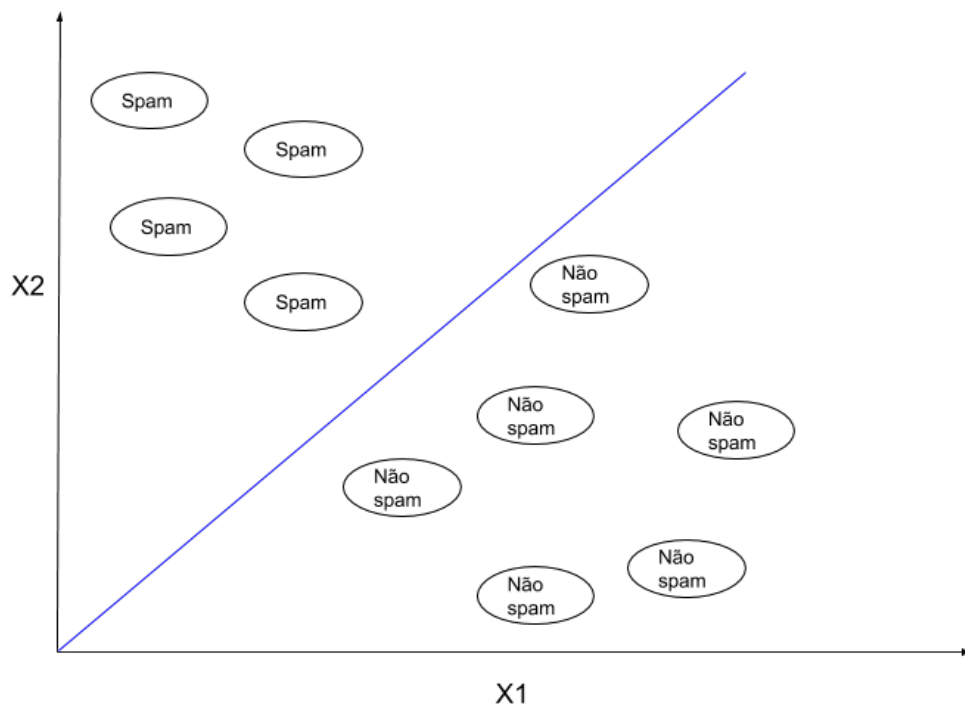
2.2.1 Categorias de Aprendizado de Máquina

Os tipos de aprendizagem de máquina são apresentados por Raschka e Mirjalili (2017) como sendo 3, aprendizado supervisionado (*Supervised learning*), aprendizado não supervisionado (*Unsupervised learning*) e aprendizado por reforço (*Reinforcement learning*). As mesmas categorias de aprendizado também são apontados por Russel *et al.* (2013) como sendo os principais, mas também apresenta o aprendizado semi-supervisionado (*Semi-supervised learning*) como sendo um misto de supervisionado e não supervisionado. Já Nassif *et al.* (2019) apresenta mais uma categoria que é aprendizado profundo (*Deep learning*). Nesta seção serão apresentadas as categorias descritas por Raschka e Mirjalili (2017).

O aprendizado supervisionado nos permite aprender com dados de treinamento e então inferir o resultado de um conjunto de dados desconhecidos ou futuros. Usando como exemplo um filtro de spam, onde os *e-mails* usados para treinar o algoritmo estão classificados corretamente como sendo ou não spam. Então, quando um novo *e-mail* for aplicado no algoritmo ele será classificado na categoria correspondente (RASCHKA; MIRJALILI, 2017).

Uma classificação é caracterizada por ter um número limitado de valores possíveis como resposta, o exemplo do spam é uma classificação binária, onde somente dois valores são possíveis como resultado (RASCHKA; MIRJALILI, 2017). Para ilustrarmos esse exemplo, considere que são usados 10 *e-mails* para treinamento, onde 4 deles são spam e 6 não. Então assumindo $X1$ para um e-mail que não é um spam e $X2$ para um *e-mail* com spam, teremos ilustração da Figura 1. A linha azul significa a divisão entre eles, e qualquer novo dado que for apresentado será classificado entre uma dessas duas categorias. É importante ressaltar que foram escolhidas apenas 10 amostras de dados apenas para uma melhor ilustração, pois em uma situação real, quanto mais dados para treino, mais chances do algoritmo ter um bom resultado. É importante ressaltar que a figura não ilustra os falsos positivos nem falsos negativos, que seriam as classificações de $X1$ classificados como $X2$ ou $X2$ classificados como $X1$.

Figura 1 – Classificação binária



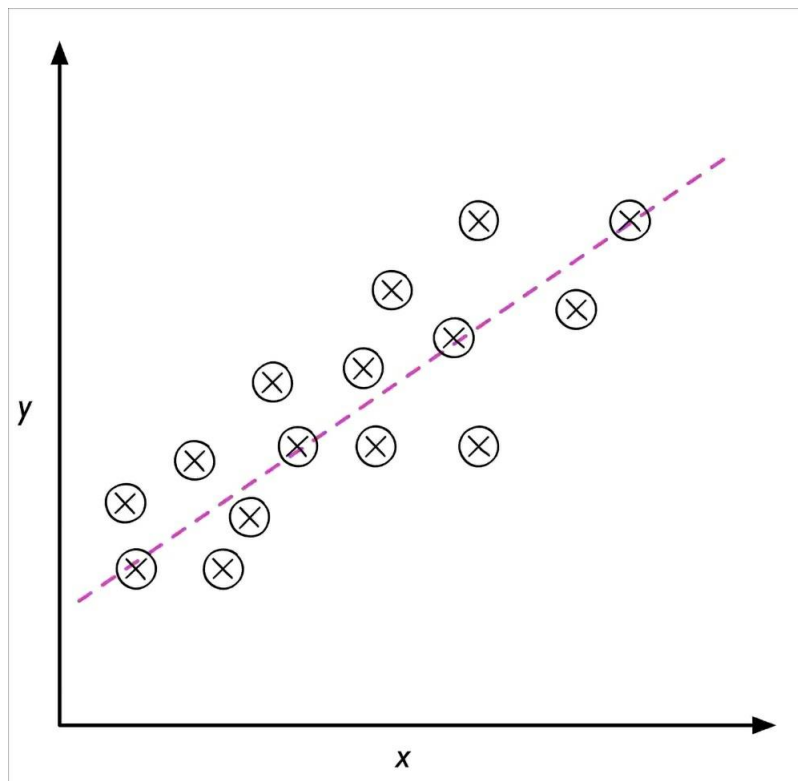
Fonte: Adaptado de (RASCHKA; MIRJALILI, 2017)

Uma classificação não precisa ser somente binária. Um algoritmo de aprendizado

supervisionado pode classificar um dado com qualquer resultado aprendido durante o treinamento. Um exemplo de classificação multi-classe é do alfabeto escrito a mão, onde um conjunto de dados de alfabetos manuscritos é coletado para treinamento. Então, quando um novo dado de um caractere manuscrito for apresentado ele será categorizado na letra correspondente com uma certa precisão. Porém, não conseguirá prever corretamente números de 0 a 9, pois nosso modelo não foi treinado para reconhecê-los (RASCHKA; MIRJALILI, 2017).

Existe outra subcategoria do aprendizado supervisionado que é a regressão, onde valores contínuos são possíveis como resultado (RASCHKA; MIRJALILI, 2017). Como um exemplo de regressão, digamos que queremos prever uma nota de matemática de um aluno e tenhamos uma relação entre o tempo de estudo e a nota, caso usemos esses dados para treinamento do nosso modelo de regressão será possível estimar a nota final do aluno, já que prever a nota exata é quase impossível (RASCHKA; MIRJALILI, 2017; RUSSEL *et al.*, 2013). A Figura 2 ilustra essa regressão, onde o x representa o tempo de estudo e y representa a nota que vai ser predita.

Figura 2 – Regressão

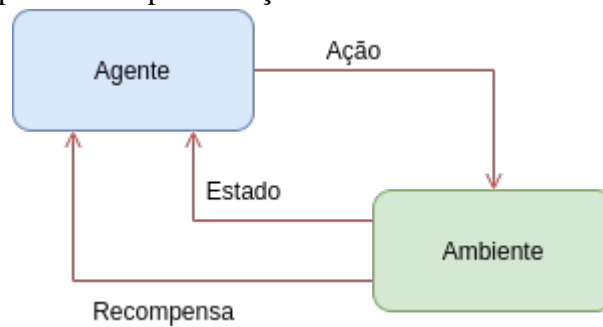


Fonte: (RASCHKA; MIRJALILI, 2017)

Já o aprendizado por reforço é caracterizado pelo sistema aprimorar seu desempenho conforme o agente interage com o ambiente, isso se deve a um sistema de recompensas positivas

ou negativas que são medidas de acordo o quão bem o sistema se saiu com as interações (RASCHKA; MIRJALILI, 2017). A Figura 3 ilustra as interações do sistema, onde o agente faz uma ação no ambiente, e este responderá com o estado atual e a recompensa gerada com a interação.

Figura 3 – Interações aprendizado por reforço



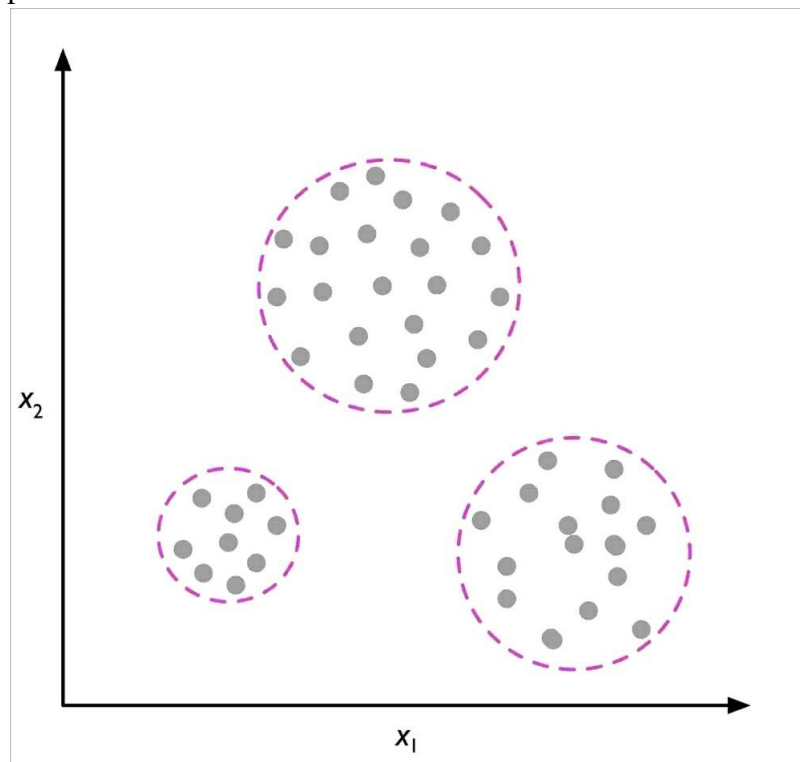
Fonte: Adaptado de (RASCHKA; MIRJALILI, 2017)

Um exemplo famoso de aprendizado por reforço é o de uma partida de xadrez, que o sistema será penalizado caso perca a partida ou recompensado caso consiga a vitória, ou mesmo cada movimento pode ser visto como um estado diferente do ambiente, como por exemplo a remoção de uma peça do adversário seria associado a um resultado positivo e perder uma peça é visto como um resultado negativo (RASCHKA; MIRJALILI, 2017).

Diferente do aprendizado supervisionado que as amostras tinham as respostas, e do aprendizado por reforço o sistema aprendia com as interações com o ambiente de acordo com o sistema de premiações, com aprendizado não supervisionado podemos extrair informações dos dados sem qualquer sistema de resposta (RASCHKA; MIRJALILI, 2017). Uma técnica de aprendizado não supervisionado é o agrupamento (*clustering*) que nos permite separar os dados em subgrupos (*cluster*) mesmo sem informações sobre associações entre eles. Dados que compartilham um certo grau de similaridade são agrupados no mesmo *cluster* ao mesmo tempo que diferem dos dados que representam os outros subgrupos. Um exemplo de aplicação é no *marketing* que permite descobrir diferentes grupos de clientes baseando-se em características de preferência, permitindo criar programas distintos para cada grupo (RASCHKA; MIRJALILI, 2017). A Figura 4 ilustra um exemplo de agrupamento com 3 subgrupos distintos.

Outro subcampo de aprendizado não supervisionado é a redução de dimensionalidade, é comum trabalharmos com dados com alta dimensionalidade, dado que cada amostra vem com um grande número de medições, mas isso pode ser um problema tanto em questão de armazenamento quanto em relação a desempenho dos algoritmos de aprendizado de máquina.

Figura 4 – Agrupamento

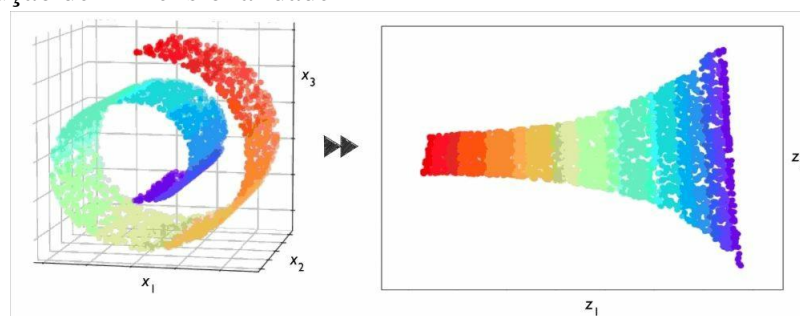


Fonte: (RASCHKA; MIRJALILI, 2017)

A redução de dimensionalidade é usado para compressão dos dados sem perda de informações relevantes, também é comumente usada como técnica de pre-processamento para remoção de ruídos que podem prejudicar o desempenho do algoritmo (RASCHKA; MIRJALILI, 2017).

A Figura 5, ilustra um exemplo onde a redução de dimensionalidade foi usada para compressão de dados de 3 para 2 dimensões. O exemplo também nos mostra uma aplicação útil para redução de dimensionalidade, onde dados com alta dimensionalidade podem ser comprimidos para ser visualizados em gráficos de dispersão 2D, por exemplo (RASCHKA; MIRJALILI, 2017).

Figura 5 – Redução de Dimensionalidade

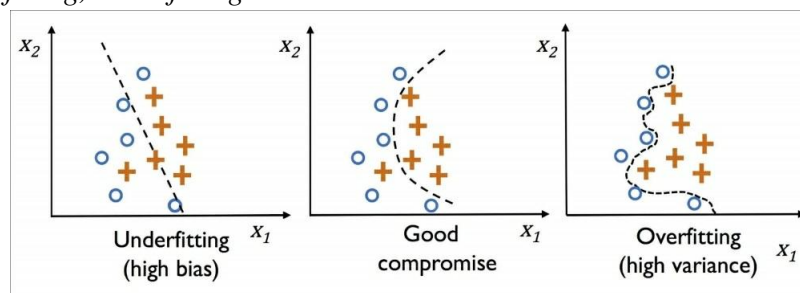


Fonte: (RASCHKA; MIRJALILI, 2017)

2.2.2 *Overfitting e Underfitting*

Overfitting é um problema comum em aprendizagem de máquina, que é caracterizado pelo fato do modelo se ajustar muito bem aos dados de treino, mas não conseguir generalizar para dados desconhecidos. Em contrapartida temos o *underfitting* que significa que o modelo não conseguiu capturar o padrão dos dados, não conseguindo se ajustar. Nos dois casos o modelo terá um baixo desempenho, a Figura 6 ilustra os problemas citados e o caso ideal (RASCHKA; MIRJALILI, 2017).

Figura 6 – *Overfitting, underfitting* e caso ideal de treinamento de modelos



Fonte: (RASCHKA; MIRJALILI, 2017)

2.2.3 *Métodos de normalização de dados*

A normalização de dados é um requisito comum para muitos modelos de aprendizagem de máquina, pois se uma *feature* tem características de variância em magnitudes diferentes o modelo pode ser prejudicado e não conseguir aprender bem com os dados (SCIKIT-LEARN, 2021).

Um tipo de normalização comum é o *Min-Max Scaling* onde os dados são redimensionados em intervalos conhecidos para valores máximos e mínimos, geralmente entre 1 e 0. O *Min-Max Scaling* considera os valores máximos e mínimos de uma coluna do *dataset* para os cálculos, como demonstrado na seguinte fórmula: $X_{Normalizado} = \frac{X - (Maior\ valor)}{(Maior\ valor) - (Menor\ valor)}$, onde X se refere a amostra que está sendo normalizada, maior e menor valor se referem aos maiores e menores valores para aquela coluna (*feature*) do *dataset* (RASCHKA; MIRJALILI, 2017).

Embora seja comum, o *Min-Max Scaling* é mais usado em casos que a *features* precisam estar em um intervalo específico. O método chamado *Standardization* é bem mais prático para alguns modelos, dadas suas características de desvio padrão unitário e centralização de dados em média 0. A fórmula de cálculo é dada por: $X_{Normalizado} = \frac{X - (Mdia\ dos\ valores)}{Desvio\ padro}$, onde

X se refere a amostra que está sendo normalizada, média e desvio padrão se referem aos cálculos de média e desvio padrão para a coluna (*feature*) do *dataset* a que se refere a amostra que está sendo normalizada (RASCHKA; MIRJALILI, 2017). Podemos comparar a diferença no resultado da aplicação entre os dois métodos na Tabela 2

Tabela 2 – Comparação dos métodos de normalização

Entrada	Standardization	Min-Max Scaling
0,0	-1,46385	0,0
1,0	-0,87831	0,2
2,0	-0,29277	0,4
3,0	0,29277	0,6
4,0	0,87831	0,8
5,0	1,46385	1,0

Fonte: (RASCHKA; MIRJALILI, 2017)

2.2.4 Algoritmos de aprendizado supervisionado

Algoritmos de aprendizado de máquina se distinguem em técnicas de aprendizagem, processamento e predição dos dados. Essa seção abordará alguns algoritmos usados ou cogitados a serem usados no presente trabalho, juntamente com suas vantagens e desvantagens, essências para escolha dos modelos usados no trabalho.

Os modelos de aprendizado podem ser agrupados dividindo modelos paramétricos e não paramétricos, que se distinguem de modo que modelos paramétricos aprendem com os dados e não precisam guardar instancias de dados originais de treino. Já os não paramétricos precisam guardar instancias de treino em memória e o número de instâncias aumenta conforme a quantidade de dados de treinamento (RASCHKA; MIRJALILI, 2017).

O algoritmo *KNN* (*k-nearest neighbors*) faz parte de uma subcategoria de algoritmos não paramétricos ditos como algoritmos de aprendizagem baseada em instância, esse tipo de algoritmo se caracteriza pela memorização dos dados de treino e o aprendizado chamado *lazy learner* é um caso especial dessa categoria, que tem como característica custo zero na fase de aprendizagem.

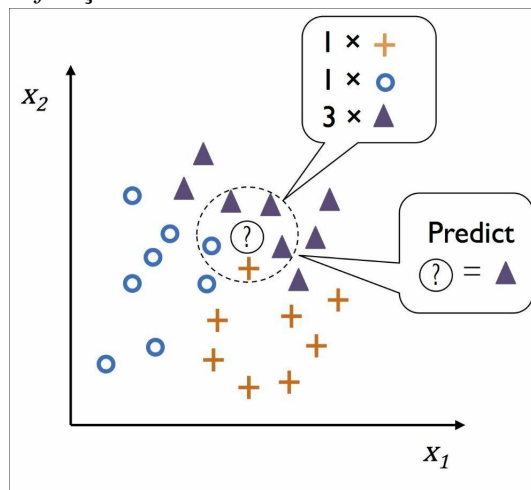
Segundo Raschka e Mirjalili (2017) o *KNN* é um algoritmo muito simples, e pode ser descrito apenas usando uma sequência de 3 passos:

1. Selecionar um número k , onde k é o número de vizinhos e uma métrica de distância.

2. Encontrar os k vizinhos mais próximos ao novo dado que queremos classificar.
3. Classificar o novo dado de acordo com a classe com maior número dentre os k vizinhos.

Baseado na métrica de distância que foi selecionada, o KNN encontrará os k vizinhos do *dataset* de treino que mais se assemelham com o novo dado. A figura 7 mostra um algoritmo com $k=5$ o novo dado está representado pelo símbolo '?' pela maioria o novo dado foi classificado como triângulo, pois dentre os 5 mais próximos vizinhos selecionados 3 eram triângulos enquanto as outras duas classe apenas um de cada foi selecionado (RASCHKA; MIRJALILI, 2017).

Figura 7 – Exemplo de classificação com KNN



Fonte: (RASCHKA; MIRJALILI, 2017)

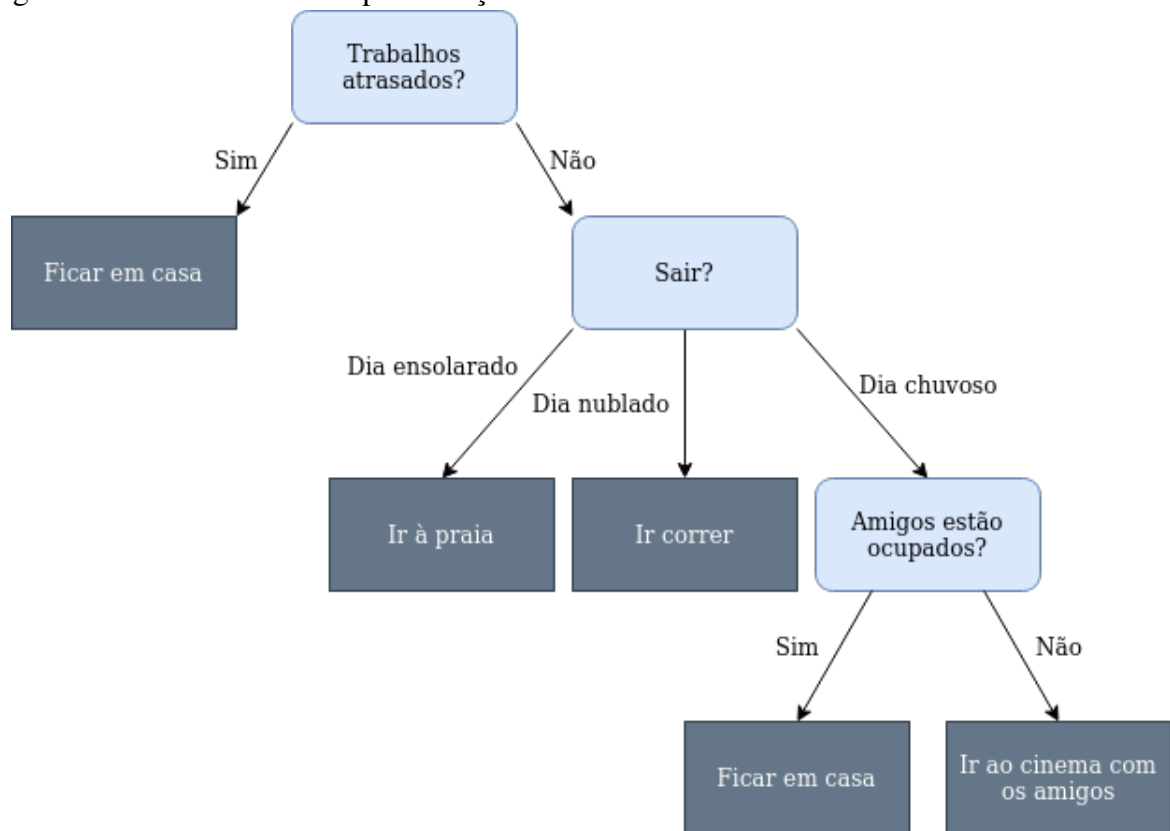
A principal vantagem dessa abordagem é que o modelo se ajusta automaticamente a medida que novos dados são classificados, uma desvantagem é que a complexidade computacional cresce a medida que o conjunto de dados aumenta. Em caso de empate a implementação do algoritmo presente na biblioteca *scikit-learn* usará como critério de desempate será o vizinho que está mais próximo, caso as distâncias sejam semelhantes será selecionado a classe que vem primeiro no *dataset*. A seleção de um k adequado é essencial, dado que pode ser ele o balanço entre *overfitting* e *Underfitting*, e garantir também que a métrica de distância escolhida é adequada ao *dataset* (RASCHKA; MIRJALILI, 2017).

As árvores de decisão (*Decision Trees*), também usam métodos de aprendizagem supervisionado não paramétricos, usadas tanto em classificação quanto em regressão. É possível prever o valor de uma variável aprendendo regras de decisão deduzidas a partir dos dados apresentados, ou seja, se baseando no *dataset* de treino a árvore de decisão aprende uma série de indagações para identificar a classe das amostras (RASCHKA; MIRJALILI, 2017; SCIKIT-LEARN, 2021).

Algumas das vantagens dessa abordagem são: fácil análise e interpretação, e não tem a necessidade de dados com muito pre-processamento. As desvantagens são: árvores complexas que não consigam generalizar os dados podem gerar *overfitting* e árvores de decisão podem ser instáveis, pois mesmo variações pequenas nos dados podem acarretar em árvores completamente diferentes (SCIKIT-LEARN, 2021).

Um exemplo de árvore de decisão é mostrado na Figura 8 onde é usada para decidir qual atividade será feita em um dia específico.

Figura 8 – Árvore de decisão para seleção de uma atividade diária



Fonte: Adaptado de (RASCHKA; MIRJALILI, 2017)

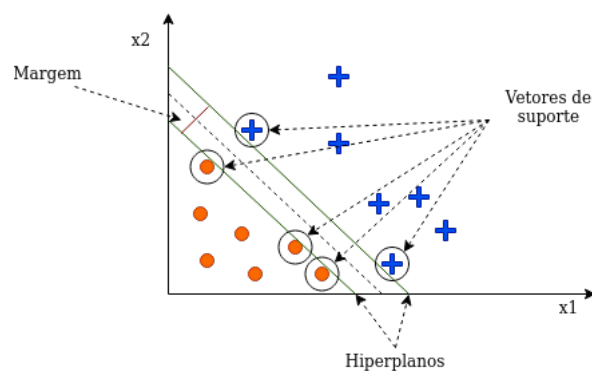
Outra abordagem de modelo de árvore é o algoritmo de floresta randômica (*Random Forest*), que pode ser interpretado como sendo a combinação de um conjunto de árvores de decisão para gerar um estrutura mais robusta. É possível destacar as seguintes vantagens: possui um ótimo desempenho em classificação de dados, é escalável, fácil de usar, possui uma ótima capacidade de generalização, é menos suscetível a *overfitting* se comparado á arvores de decisão usadas individualmente e não exige um grande esforço com ajuste de hiperparâmetros. Uma desvantagem é que se comparado a uma árvore de decisão é que os resultados não são tão facilmente interpretáveis. Para classificação em uma floresta randômica com N árvores, o

resultado das N árvores é combinado usando critério de maioria de votos para obtenção do resultado (RASCHKA; MIRJALILI, 2017).

Já o SVM (*Support Vector Machine*) consegue combinar vantagens dos dois tipos de algoritmo, tanto paramétrico quanto não paramétrico dependendo do *kernel*. Esse é um algoritmo popular e atraente por vários motivos, alguns deles são: as estratégias usadas fazem com que consiga generalizar bem os dados, é resistente a *overfitting* e consegue trabalhar bem com dados com muitas dimensões. Como desvantagens podem ser listados os seguintes problemas: é custoso computacionalmente, aprendizado lento e difícil interpretabilidade dos resultados (RUSSEL *et al.*, 2013; BHAVSAR; GANATRA, 2012).

Para separar os dados, o SVM cria hiperplanos com o objetivo de encontrar o maior espaço que pode ser colocado entre diferentes vetores, esse espaço é chamado de margem, que é a distância entre dois hiperplanos, e os vetores são as amostras que estão mais próximas de cada hiperplano que são chamados de vetores de suporte (RUSSEL *et al.*, 2013; RASCHKA; MIRJALILI, 2017). Com o auxílio da Figura 9 o processo descrito pode ser entendido mais facilmente, onde é representada uma classificação binária, uma classe é representada pelo círculo e a outra pelo símbolo '+', os vetores de suporte, os hiperplanos e a margem estão indicados na figura.

Figura 9 – Classificação binária com SVM



Fonte: Adaptado de (RASCHKA; MIRJALILI, 2017)

Embora a Figura 9 ilustre um exemplo em duas dimensões, no caso de uma entrada de dados não ser linearmente separável, o SVM consegue trabalhar com esses dados em uma dimensão superior, e assim trabalhar com os processos já descritos.

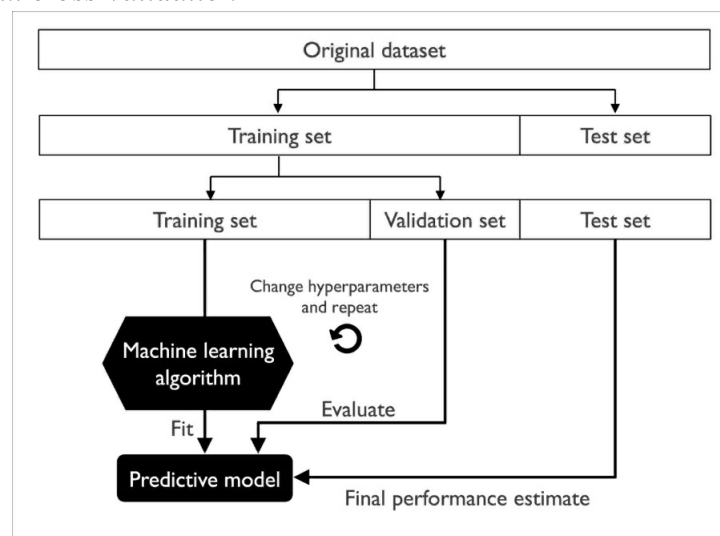
2.2.5 Validação e avaliação de modelos

Quando se fala em modelos de aprendizagem de máquina, mais especificamente, modelos de classificação, é preciso ter uma forma de avaliar e mensurar o quão bom está aquele modelo, existem muitas métricas para avaliação de modelos de classificação no aprendizado de máquina, nesta seção serão apresentados os métodos usados no trabalho.

Um modelo de aprendizagem de máquina precisa de dados para ser treinado e avaliado, e segundo Raschka e Mirjalili (2017) um método muito comum de divisão de dados é o método *holdout* onde separamos os dados disponíveis em uma porcentagem para treino e outra para teste do modelo, mas ao selecionarmos um modelo para o nosso problema é comum querermos encontrar os melhores parâmetros para ele afim de melhorar seu funcionamento, parâmetros esses que são comumente chamados de hiperparâmetros, mas ao fazermos isso, acabamos usando os mesmos dados de treino para avaliar todos os possíveis modelos e hiperparâmetros, essa não é uma boa prática, apesar de ser bem comum.

Dito isso, existe um método de *holdout* onde além dos dados de treino e teste, teremos também os dados de validação chamado *holdout cross-validation*, usando apenas os dados de treino e validação para treinamento e *tunning* dos hiper-parâmetros do modelo, dessa forma os dados de teste ficarão isolados para que sejam usados para avaliar o modelo apenas quando a versão final for selecionada. A Figura 10 ilustra os procedimentos descritos.

Figura 10 – *Holdout cross-validation*



Fonte: (RASCHKA; MIRJALILI, 2017)

Apesar de ser um método significativamente melhor para divisão dos dados, esse

tipo de procedimento nos prende a mesma parcela dos dados, independentemente de quantas vezes treine, altere os hiperparâmetros, podendo essa divisão não ser a melhor possível. Com base nisso existem métodos de validação cruzada onde a divisão dos dados é feita k vezes em k divisões dos dados.

Dessa forma o modelo é treinado e testado k vezes obtendo k valores como resultado, então para valorarmos a performance desse modelo, podemos fazer a média dos k valores obtidos após o teste. A Figura 11 ilustra um exemplo de validação cruzada com $k = 3$.

Figura 11 – Validação cruzada com $k = 3$

Teste	Treino	Treino
Treino	Teste	Treino
Treino	Treino	Teste

Fonte: Autoria Própria

Agora que sabemos a divisão dos dados, precisamos avaliar nosso modelo, e um método usado para avaliar como o modelo está classificando os dados é a matriz de confusão, essa técnica não nos dá uma pontuação baseada nos erros e acertos como nas métricas que serão descritas logo após, mas através dela podemos avaliar o modelo com base nos erros e acertos, comparando as classes preditas pelo algoritmo e as classes reais (RASCHKA; MIRJALILI, 2017). Uma ilustração geral de uma matriz de confusão está na Figura 12.

Figura 12 – Forma geral de uma matriz de confusão

		Classe predita	
		Positivo	Negativo
Classe real	Positivo	Verdadeiros positivos	Falsos negativos
	Negativo	Falsos positivos	Verdadeiros negativos

Fonte: Adaptado de (RASCHKA; MIRJALILI, 2017)

Como exemplo de representação para uma matriz de confusão, considere um exemplo

de classificação de gatos e cachorros como ilustrado na Figura 13, onde temos um total de 7 gatos e 5 cachorros, e o nosso modelo fictício classificou de forma correta 5 gatos e 4 cachorros, e classificou de forma errada 3 exemplos, classificando 1 cachorro como sendo gato e 2 gatos como sendo cachorros. Onde poderíamos assumir gato como sendo nossa classe 1 e cachorro como classe 0 por exemplo, mas os nomes das classes foram mantidos para um melhor entendimento.

Figura 13 – Exemplo de uma matriz de confusão

		Classe predita	
		Gato	Cachorro
Classe real	Gato	5	2
	Cachorro	1	4

Fonte: Autoria própria

De acordo com Raschka e Mirjalili (2017) uma métrica de classificação muito comum é o *accuracy*, onde é obtida uma pontuação obtida através da proporção de acertos. A fórmula de cálculo é feita da seguinte forma:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Sendo, *TP* os verdadeiros positivos, *FP* os falsos positivos, *TN* os verdadeiros negativos e *FN* os falsos negativos. Lembrando que quando se fala em falso são os dados classificados de maneira errada, e verdadeiro os dados classificados de maneira correta.

Existe uma métrica que é calculada de forma similar ao *accuracy* que é chamada *Error Rate* onde ao invés de usarmos *TP + TN* no numerador, usamos *FP + FN*.

Outras duas métricas muito comuns quando se fala em modelos de classificação são *Precision* e *Recall* que se aplicam muito bem em conjunto de dados desbalanceados.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{FN + TP}$$

Outra métrica que é obtida através da combinação de *Precision* e *Recall* chamada *F1-score*.

$$F1 - score = \frac{Precision \times Recall}{Precision + Recall}$$

Os métodos apresentados são usados somente em classificação e apenas alguns foram listados aqui, existindo outras métricas. Para modelos de regressão as métricas serão diferentes, alguns exemplos são *Mean Absolute Error (MAE)*, *Mean Squared Error (MSE)* entre outros (SCIKIT-LEARN, 2021).

2.3 Injeção Eletrônica

Motores de combustão interna são caracterizados pela queima da mistura ar-combustível que desencadeia a transformação de energia química em calor através da combustão, assim aumentando a pressão dentro de um meio e por consequência gerando trabalho mecânico (BOSCH, 2005). Existem outros tipos de motores de combustão interna no mercado, porém neste trabalho para um melhor entendimento, será apresentada apenas uma pequena explicação sobre motores de ignição a centelha de quatro tempos, também chamados de ciclo otto. Motores de ignição a centelha são caracterizados pelo fato do início da queima da mistura ar-combustível ser dado por uma centelha (BOSCH, 2005).

Em motores de quatro tempos, duas voltas das manivelas são necessárias para que seja completado um ciclo de trabalho. Os pistões se movem nos cilindros indo do PMI (ponto morto inferior) ao PMS (ponto morto superior). Um conjunto de válvulas são responsáveis pela admissão e exaustão, sendo elas, as válvulas de admissão e escape (MILHOR, 2002). A Figura 14 ilustra esse sistema.

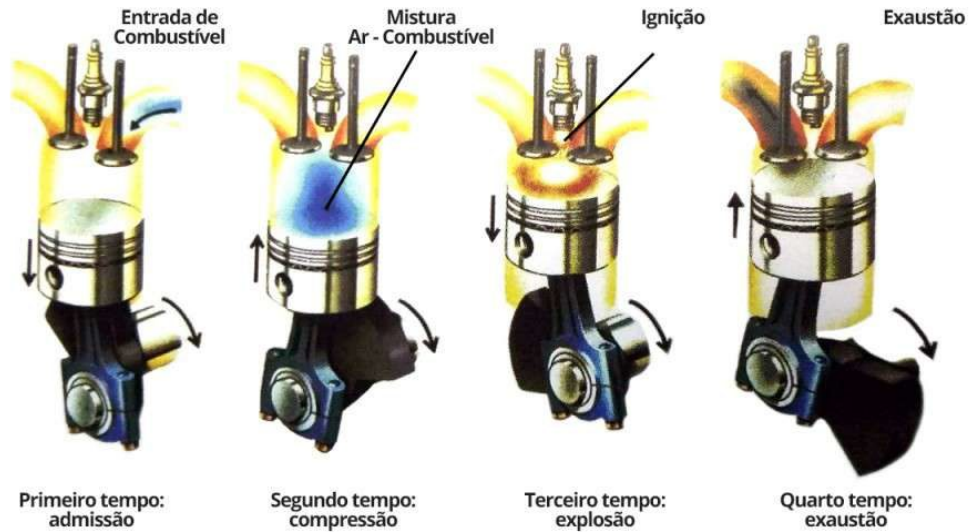
Primeiro tempo (Admissão): com a válvula de escape fechada e a de admissão aberta e o pistão indo do PMS ao PMI, permitindo a entrada da mistura ar-combustível.

Segundo tempo (Compressão): nessa etapa, o pistão está indo do PMI ao PMS, com ambas as válvulas fechadas. Fazendo a compressão da mistura, então a centelha é gerada, queimando a mistura e aumentando a pressão no cilindro.

Terceiro tempo (Expansão): ambas as válvulas permanecem fechadas, e a pressão gerada pela queima faz com que o pistão vá do PMS ao PMI, transformando a energia gerada pela queima em movimento.

Quarto tempo (Exaustão): agora o pistão vai do PMI ao PMS, então a válvula de exaustão é aberta, permitindo a saída dos gases resultantes da queima.

Figura 14 – Motor 4 tempos



Fonte: (SPORT, 2020)

Em veículos puramente mecânicos, os carburadores têm a função de disponibilizar a melhor relação ar-combustível dentro de um sistema. Nos carburadores, o que dosa a quantidade de ar admitida é a posição da borboleta, que é o mecanismo responsável pela entrada de ar. A posição irá variar de acordo com a posição do pedal de aceleração, e a quantidade de combustível irá variar de acordo com a quantidade de ar admitido (BOSCH, 2005). Porém, existem algumas variáveis que o carburador não pode controlar em tempo real. Nos dias de hoje o controle é feito através de uma ECU EFI (*Eletronic Control Unit Eletronic Fuel Injection*), ou injeção eletrônica de combustível (SOCORRO, 2017).

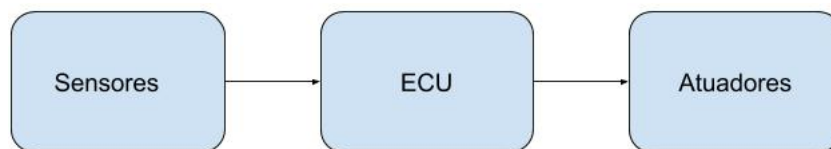
Assim como os carburadores, sistemas de injeção tem a função de disponibilizar a melhor relação ar-combustível dentro de um sistema, principalmente sistemas de injeção eletrônica, que tem uma melhor capacidade de seguir limites predeterminados. O resultado disso é um melhor consumo de combustível, melhor dirigibilidade e potência (BOSCH, 2005).

A injeção eletrônica funciona de modo a coletar informações de sensores, como posição do pedal do acelerador, ângulo da borboleta e etc. Esses dados são enviados para uma ECU, que faz cálculos baseados em mapas armazenados em memória. Através desses cálculos são estimados tempo de injeção e momento de ignição, por exemplo. Deste modo, após os cálculos serem feitos, os dados são enviados para os atuadores, visando uma melhor eficiência, definindo a melhor mistura ar-combustível (BOSCH, 2005). Considerando as limitações dos sistemas carburados e as normas rigorosas já estabelecidas para os poluentes gerados pela

queima de combustível dos automóveis regidos pelas leis de trânsito, esses sistemas vêm sendo substituídos por sistemas de injeção eletrônica (SOCORRO, 2017; BOSCH, 2005).

Um diagrama básico do funcionamento da injeção eletrônica pode ser visto na Figura 15. Os dados dos sensores são enviados para a ECU, os cálculos necessários são feitos, e então as informações são enviadas da ECU para os atuadores. Sensores são dispositivos capazes de captar comportamentos e grandezas do meio físico e transformá-los em sinais elétricos que serão enviados ao ECU. E atuadores são dispositivos que através de informações de sinais elétricos, são capazes de coordenar e ajustar outros dispositivos através de informações enviadas pela ECU (BOSCH, 2005).

Figura 15 – Diagrama injeção eletrônica



Fonte: Autoria própria

Os ECUs se conectam através da Rede CAN (*Control Area Network*), que é um barramento serial que fornece um meio de comunicação confiável entre sensores, ECUs e atuadores (SHAFI *et al.*, 2018; KOWALIK, 2018).

Existem diversos sensores ligados à injeção eletrônica, como por exemplo sensor TPS, usado para medir a posição de abertura da válvula borboleta. Sensor de fase, responsável por indicar à ECU em qual fase o cilindro está. Sensor MAP, responsável por medir a pressão no coletor de admissão. Sensores de temperatura, responsáveis por medir a temperatura do ar admitido pelo motor e temperatura do líquido de arrefecimento. E sensores massivos de ar, responsáveis por medir a quantidade massiva de ar admitida pelo motor (SOCORRO, 2017; ALVES JUNIOR *et al.*, 2016; MILHOR, 2002). Esses são alguns exemplos, destes um dos sensores mais importantes para a injeção eletrônica é a sonda lambda.

O sensor sonda lambda é um componente capaz de medir a quantidade de oxigênio presente nos gases de escape. As informações são enviadas a ECU para que o controle da concentração ar-combustível seja feito (ENTLER; ECU, 2018). Para entendermos melhor como é feita a mistura ar-combustível, precisaremos entender o que seria a mistura ideal. Chamada mistura estequiométrica, a mistura ideal em motores de combustão interna é 14.7:1, ou seja, 14,7 kg de ar para 1 kg de massa de combustível (BOSCH, 2005).

O cálculo para sabermos o quanto a mistura desvia da mistura ideal é feito da seguinte forma

(SOCORRO, 2017):

$$\lambda = \frac{\text{Ar/Combustível (Real)}}{\text{Ar/Combustível (Ideal)}}$$

$\lambda = 1$: Mistura estequiométrica.

$\lambda < 1$: Mistura rica, ou seja, pouco ar para muito combustível.

$\lambda > 1$: Mistura pobre, ou seja, pouco combustível e muito ar.

Também é importante saber que na mistura estequiométrica todo o combustível é consumido na queima, enquanto na mistura rica nem todo o combustível é consumido, isso acaba por gerar mais poluentes (SOCORRO, 2017). Já nas misturas pobres, é obtido um menor consumo de combustível e potência, limitado pelo fato da mistura não ser mais facilmente inflamável, ocorrendo falhas na combustão (BOSCH, 2005).

Sensores de temperatura são responsáveis por medir a temperatura do ar que está sendo admitido e a temperatura do líquido de arrefecimento. No caso da temperatura do ar a importância dessa medição se deve pelo fato de que a temperatura do ar admitido é usada para determinar a densidade de massa do ar, enquanto no caso da temperatura do líquido de refrigeração para que estratégias de resfriamento sejam tomadas. Os tipos mais comuns de sensores são os sensores resistivos: *NTC* (*Negative Temperature Sensor*) que geram uma maior resistência para temperaturas menores e *PTC* (*Positive Temperature Sensor*) que geram maior resistência para temperaturas maiores (MILHOR, 2002).

O sensor *MAP* (*Manifold Absolute Pressure*) é usado para medir a pressão no coletor de admissão. Essa leitura juntamente com a informação de temperatura do ar feita pelo *NTC*, são usadas pela *ECU* para avaliar a quantidade de ar admitido e mensurar a massa de combustível que será injetado (SOCORRO, 2017).

E o sensor *TPS* (*Throttle Position Sensor*) é usado para indicar a posição angular de abertura da válvula borboleta que é uma peça que faz ações de fechamento e abertura para passagem de ar, essas ações são feitas de acordo com os movimentos do motorista no acelerador (ALVES JUNIOR *et al.*, 2016).

Sensores de massa de ar são usados para medir a quantidade massiva de ar admitida pelo motor, esses sensores não oferecem resistência a passagem de ar e seu funcionamento não é alterado pela densidade do ar. As medições desses equipamentos são feitas se baseando

na temperatura de uma superfície, e conforme a variação dessa temperatura, a massa de ar é calculada (MILHOR, 2002).

2.3.1 Engine Control Unit (ECU)

ECU (*Engine Control Unit*) é um microcomputador responsável por controle, gerenciamento e processamento de informações em quase todo o carro. Em um carro existem muitos ECUs, sendo compostos por vários sensores distribuídos em diferentes partes do carro. Os dados desses sensores chegam em forma de sinal elétrico (SHAFI *et al.*, 2018). A ECU converte os dados recebidos de analógicos para digitais, faz o processamento das informações, e calcula quais ajustes devem ser feitos (OSS, 2018). Os ajustes são feitos baseados no mapa de características armazenado na unidade de memória (SHAFI *et al.*, 2018). Com base nisso, são enviados sinais de comando aos atuadores para que o sistema se adapte (CAVALCANTE, 2018).

O mapa de injeção e curvas características ficam armazenadas em memória. Embora possam possuir diferentes arquiteturas, esses sistemas podem ser classificados das seguintes formas. Processamento, onde toda a parte lógica e tratamento de dados é feita. Memória, onde todas as informações são armazenadas. Entradas, são os dados recebidos dos sensores. Saídas, são as informações que saem da ECU e vão para os atuadores. Rede de comunicação, são as informações trocadas entre os diferentes módulos (OSS, 2018).

As ECUs estão atuando em quase todo o automóvel, em sistemas mecânicos e não mecânicos. Além do controle do motor, também controlam sistemas críticos como ABS e *Air-bags*, ou até mesmo sistemas ligados ao conforto, como ar-condicionado e multimídia (OSS, 2018).

Quando a ECU detecta algum erro, ela armazena esse erro em forma de código, chamado de DTC (*Diagnostic trouble code*) (CAVALCANTE, 2018). Caso a ECU detecte que um sensor está com mau funcionamento, esta possui um modo chamado de *Recovery* onde ignora os valores do sensor com defeito, e prioriza valores preestabelecidos para as leituras (CAVALCANTE, 2018).

2.3.2 On Board Diagnostics (OBD)

O OBD (*On Board Diagnostics*), diagnóstico de bordo em português, surgiu com o intuito de diminuir o nível de poluição gerada pelos automóveis na década de 80. Onde o auto-diagnóstico era feito por meio de sensores distribuídos pelo carro, com isso era possível

verificar se o nível de poluição gerada estava dentro dos padrões estabelecidos. Além de controlar emissões de poluentes, uma das ideias por trás da OBD era padronizar o modo como o diagnóstico veicular era feito, pois como existem diferentes montadoras, caso elas escolhessem normas diferentes para fazer o diagnóstico, o preço com mão de obra e equipamento necessário para fazer a manutenção seria muito elevado para o consumidor final (OSS, 2018).

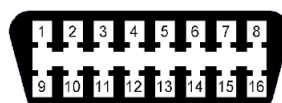
Apesar de ter começado com a ideia de monitoramento de poluentes, com o tempo também passou a fornecer dados de diagnóstico (SILVA NETO, 2019). O diagnóstico é feito de modo que a ECU pode indicar um mau funcionamento no carro (CAVALCANTE, 2018).

Alguns padrões foram surgindo com o passar do tempo até chegarmos ao OBD2 (*On Board Diagnostics 2*) que é o padrão atual (OSS, 2018). O primeiro sistema de diagnóstico foi chamado de ALDL, desenvolvido pela GM (*General Motors*), este é visto com o precursor dos sistemas de diagnóstico (OSS, 2018). Em 1988 na Califórnia, surgiu o OBD1, o primeiro estágio da legislação da CARB (*California Air Resources Board*), que estabelecia regras, como monitoramento dos gases de escapamento e armazenamento de falhas na ECU. Visualização das falhas detectadas ao motorista por meio de uma luz MIL (Lâmpada de Indicação de Falhas). Fornecer um meio para leitura do componente que apresentou falha (BOSCH, 2005). Essa foi uma tentativa falha de padronização (OSS, 2018). Então, em 1994 na Califórnia foi implantado o segundo estágio chamado OBD2 (BOSCH, 2005).

Além do que já estava presente no OBD1, o OBD2 exige que todos os componentes relacionados ao gás de escapamento sejam monitorados (BOSCH, 2005). Também estabelece como padrões o conector J1962, formato da mensagem e os sinais elétricos. Este padrão começou a ser produzido no Brasil em 2010 (OSS, 2018).

Uma das principais funções do OBD2 é a obtenção de códigos de erro chamados de DTCs (*Diagnostics trouble codes*) (VIANA, 2019). Através do OBD2 também é possível ter acesso aos dados de sensores, e leitura desses dados em tempo real, a partir de uma requisição por PID (*Parameter ID*) (ENGINEERS, 2002b). O formato e a pinagem do conector pode ser visto na Figura 16.

Figura 16 – Conector J1962



Fonte: OBDII

1. Reservado ao fabricante

2. Barramento positivo SAE J1850 PWM e VPW
3. Reservado ao fabricante
4. Chassi GND SAE J1978
5. Sinal GND SAE J1978
6. CAN HIGH ISO 15765-4
7. K Line ISO 9141-2 e ISO 14230-4
8. Reservado ao fabricante
9. Reservado ao fabricante
10. Barramento negativo SAE J1850 PWM
11. Reservado ao fabricante
12. Reservado ao fabricante
13. Reservado ao fabricante
14. CAN LOW ISO 15765-4
15. L Line ISO 9141-2 e ISO 14230-4
16. Sinal de tensão positiva

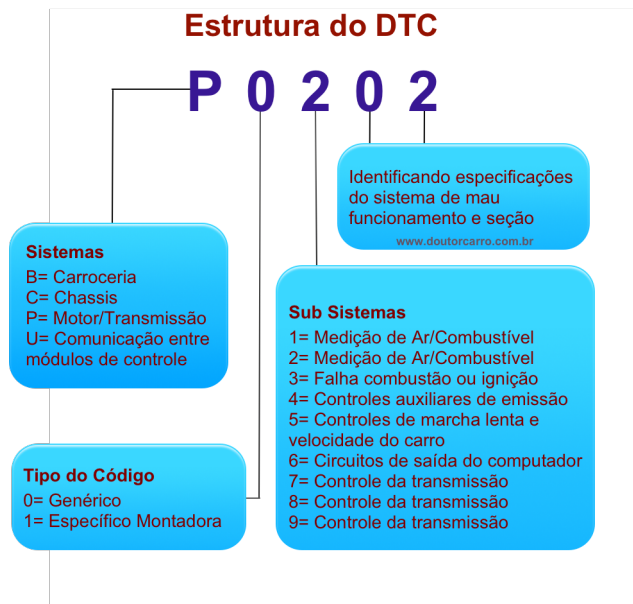
As interfaces padrões são referentes aos pinos não listados como sendo reservados ao fabricante. Começando pelo ISO9141-2, que era o protocolo mais usado antes da introdução do CAN (*Control Area Network*). Esse protocolo com linhas K-Line e L-Line enquanto a K faz toda a comunicação, a linha L é opcional, apenas é usada para início de conexão. O ISO 14230-2 é semelhante ao ISO 9141-2, com mecanismo de inicialização rápida. Já no SAE J1850, o barramento pode assumir dois tipos de abordagens: PWM (Modulação por largura de pulso) que é uma abordagem diferencial de dois fios; ou VPW (Largura de pulso variável) com fio único. Ambos são variantes do protocolo CSMA. O ISO 15765-4 se refere a rede CAN (*Control Area Network*) que trata o suporte a comunicação e se baseia em um mecanismo de prioridade, onde o transmite sempre a mensagem com maior prioridade. Esse barramento é capaz de enviar *frames* com dados de 1 a 8 bytes (ENGINEERS, 2002a; OBDDIAG, 2020).

Existem inúmeros dispositivos capazes de ler informações a partir da porta OBD2 do carro. O dispositivo conectado pode ser usado tanto para ler os DTCs quanto para ler informações em tempo real (AZEEZ; BANDARA, 2015). Muitos desses dispositivos são baseados no microcontrolador ELM327, que nos permite acessar os dados através da conexão com outros dispositivos, seja via cabo, *Bluetooth* ou Wi-Fi (OSS, 2018).

2.3.3 Diagnostic Trouble Codes (DTCs)

Sempre que um erro é detectado, um DTC (*Diagnostic Trouble Code*) é armazenado, contendo as informações do estado do carro, e dados dos sensores (KRIEBE *et al.*, 2019). A maioria desses códigos ficam armazenados em memória e podem ser lidos através de um equipamento de leitura apropriado, o condutor pode ser alertado de alguns desses erros através de uma luz acesa no painel (VIANA, 2019). DTCs consistem de um código alfanumérico, sendo umas das letras, B, C, P ou U seguida por 3 dígitos. Sendo B referente a códigos ligados a carroceria, iniciado com C correspondem ao chassi, P se referem a códigos ligados ao motor, e U conexões dos módulos. A especificação do dígito alfanumérico correspondente é atribuída pela área mais adequada. Para cada código é atribuído uma descrição para indicar onde a falha ocorreu, que pode ser de um sistema, componente ou circuito específico. Os códigos podem ser genéricos ou proprietários, que são códigos implementados pelas montadoras. Um DTC é usado para indicar um problema em uma área que precisa de atenção, mas não deve ser usado para indicar que um sistema não está com problemas, ou mesmo para indicar o estado de funcionamento dele (ENGINEERS, 2002c). A Figura 17 ilustra a estrutura do código.

Figura 17 – Estrutura do código do DTC



Fonte: (DOUTORCARRO, 2020)

2.3.4 *Parameter IDs (PIDs)*

Parameter IDs (PIDs), são códigos usados para fazer requisições ao veículo (AZEEZ; BANDARA, 2015). Como resposta a essa requisição, são enviados da ECU para o dispositivo externo uma série de dados correspondentes à requisição feita (ENGINEERS, 2002b). Os 9 serviços são listados e descritos no Quadro 2.

Quadro 2 – Serviços disponíveis por PIDs

Serviço	Descrição
01	Solicita as leituras atuais dos sensores
02	Solicita valores de leitura armazenados
03	Solicita todos os DTCs armazenados
04	Limpa os DTCs armazenados
05	Requisita o resultado do teste do sensor de oxigênio
06	Requisita o resultado de testes de sistemas específicos
07	Solicita os DTCs detectados recentemente
08	Solicita controle de uma parte do sistema
09	Requisita informações do veículo

Fonte: Adaptado de (ENGINEERS, 2002b)

Algumas literaturas listam um décimo serviço, listado como serviço 0A, responsável por ler os DTCs que foram apagados.

Para cada serviço disponível, são definidos os PIDs suportados, onde para cada PID solicitado, são retornados os valores referentes aquela solicitação. Cabe ao dispositivo que está conectado fazer a interpretação dos dados (ENGINEERS, 2002b).

3 MOMENT

Neste capítulo é apresentado o processo usado para o desenvolvimento e avaliação do protótipo de um Sistema de Aprendizado de Máquina para Manutenção Preditiva chamado *MOMENT*. Na coleta de dados e seleção dos sensores usados, é descrito o método usado para coletar os dados e a metodologia usada para selecionar as *features* mais relevantes para o problema. Já na etapa de pré-processamento e divisão dos dados, são descritas as estratégias usadas para tratar valores faltantes, informações ruidosas, a estratégia de normalização dos dados usada e qual a divisão feita nos dados para que fosse possível treinar, validar e testar os modelos de aprendizado de máquina e nosso protótipo. Na seleção dos modelos e *tunning* de hiperparâmetros é apresentada a estratégia usada para selecionar o modelo com melhor desempenho para o conjunto de dados juntamente com o conjunto de parâmetros que possibilitam alcançar o melhor desempenho. Por fim, em desenvolvimento e avaliação do protótipo, é descrito o funcionamento do protótipo, juntamente com a avaliação do que é proposto e os casos de erro.

1. Coleta de dados e seleção dos sensores usados

A coleta dos dados foi feita usando os recursos do aplicativo Torque¹, que após conectado ao *scanner* plugado à porta OBD2 do carro nos permite fazer a leitura dos dados em tempo real e fazer o *upload* desses dados para serem acessados posteriormente no site.

O conjunto de dados foi coletado de um Suzuki Grand Vitara 2014 AWD durante viagens e passeios feitos no automóvel. O conjunto de dados possui um total de 32167 amostras sendo dividido em 13867 amostras do filtro em estado de troca (*label* atribuída como 1 no *dataset*), e 18300 amostras com um filtro em bom estado (*label* atribuída como 0 no *dataset*) caracterizando então uma divisão de cerca de 54.2% e 45.8% dos dados para cada tipo de amostra respectivamente. A rotulação foi feita de acordo com a avaliação de um mecânico, que indicou que o filtro estava em estado de troca e deveria ser substituído. Antes que fosse efetuada a troca, foi feita a coleta dos dados com rotulação 1, e para serem coletados os dados com rotulação 0, o filtro velho foi substituído por um novo.

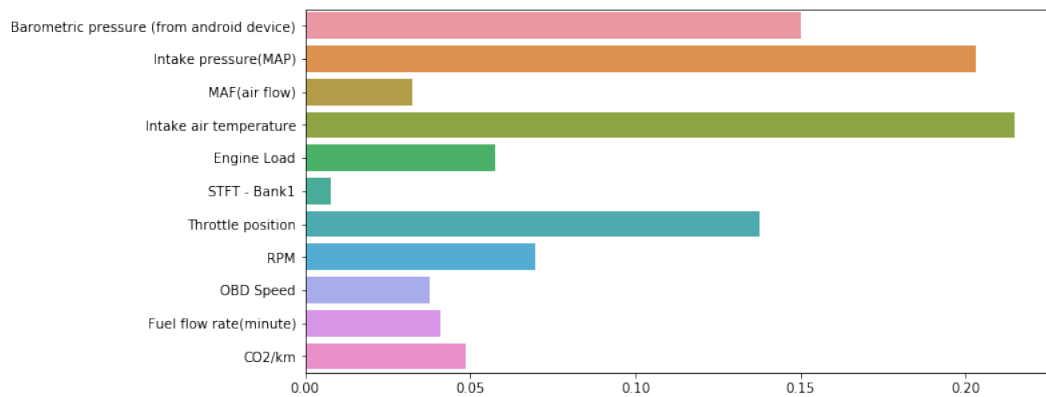
A seleção dos sensores foi feita em duas etapas, de forma que na primeira etapa foi feita uma análise manual dos dados e todas as informações redundantes ou irrelevantes para o modelo foram descartadas, restando apenas as *features* com maior relação com o filtro ar e/ou funcionamento do motor do carro, onde um filtro em mau estado seria detectado mais facilmente.

Na segunda etapa que aconteceu após o pré-processamento dos dados, foi usada a

¹ https://play.google.com/store/apps/details?id=org.prowl.torque&hl=pt_BR&gl=US

implementação de *VarianceThreshold*², com o intuito de avaliar os dados e eliminar *features* com pouca variância, após essa análise mais algumas colunas do *dataset* foram descartadas. Em seguida um modelo de *RandomForestRegressor (RF)* foi treinado para que sua função *feature_importance_* fosse usada para fazer outro refinamento no *dataset* com o intuito de descartar qualquer informação irrelevante, porém todas as *features* pré selecionadas se mostraram com importância acima de 0, ou seja, segundo o modelo de RF todas possuem relevância, e nenhuma foi descartada. O gráfico de barras com a importância atribuída pelo modelo de RF para cada uma das *features* está representado na Figura 18.

Figura 18 – Importância das *features* selecionadas para o modelo



Fonte: Autoria Própria

Por fim, as *features* selecionadas para serem usadas no modelo foram: *Barometric pressure (from android device)*, *Intake pressure(MAP)*, *MAF(air flow)*, *Intake air temperature*, *Engine Load*, *STFT - Bank1*, *Throttle position*, *RPM*, *OBD Speed*, *Fuel Flow Rate(minute)* e *CO2/km*. Logo abaixo encontra-se a descrição do *dataset* usado.

- *Barometric pressure(do dispositivo android)*, é a pressão barométrica lida a partir de um dispositivo *android*. Essa variável não está diretamente ligada ao funcionamento do carro, mas é importante pelo fato da pressão atmosférica interferir na admissão de ar.
- *Intake pressure(MAP)*, esse sensor fornece informações sobre a pressão no coletor de admissão, os valores normais de leitura são 0kPa para valor mínimo e 255 kPa para valor máximo.
- *MAF(air flow)*, essa variável nos diz qual o fluxo de massa de ar que está entrando no motor, com os valores normais lidos vão 0 g/s como mínimo valor e 655.35 g/s de máximo valor, em caso de não ser possível realizar a leitura o valor 0 g/s é mostrado.

² https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html

- *Intake air temperature*, é responsável por medir a temperatura do ar admitido no coletor de admissão, podendo ser obtido através de um sensor ou de outra estratégia de medição, os valores normais de leitura estão entre -40C como mínimo e +215C como máximo.
- *Engine Load*, essa variável se refere a carga do motor, com uma função de cálculo que segundo Engineers (2002b):

LOAD_PCT = [fluxo de ar atual]/[(pico de fluxo de ar em WOT@STP em função do rpm) * (BARO/29.92) * SQRT(298/(AAT+273))]

Alternativamente, os motores de ignição por centelha e de ignição por compressão podem usar a seguinte definição para valor calculado de carga do motor:

LOAD_PCT = [torque atual do motor]/[(pico de torque do motor @STP em função do rpm) * (BARO/29.92) * SQRT(298/(AAT+273))]

– STP = Temperatura e pressão padrão = 25 °C, 29.92 in Hg BARO

– SQRT = Raiz quadrada;

– WOT = Borboleta totalmente aberta;

– AAT = Temperatura ambiente em C°;

- *STFT - Bank1*, deve indicar o ajuste feito na mistura ar-combustível, com os valores normais de leitura indo de -100% para misturas pobres até +99.22% para misturas ricas.
- *Throttle position*, se refere a posição do pedal do acelerador, definido em porcentagem de 0% a 100%.
- *RPM*, essa variável mede a quantidade de rotações por minuto.
- *OBD Speed*, se refere a velocidade do automóvel medida pelo ECU.
- *Fuel Flow Rate(minute)*, nos diz a quantidade de combustível consumido na unidade de litros por minuto.
- *CO2/km*, nos diz a quantidade de CO2 por quilômetro rodado gerado durante a queima de combustível.

2. Pré-processamento e divisão dos dados

As informações de valores máximos e mínimos, para cada coluna do *dataset* serviram como auxílio no tratamento de *Outliers*, que são informações ruidosas no conjunto de dados, dessa forma, qualquer informação que estava fora do padrão descrito, foi descartada. Além de alguns dados ruidosos, o *dataset* também apresentou alguns dados faltantes, e o tratamento escolhido para os dados faltantes foi de descarte, dado que a substituição pela média, moda, mediana ou alguma das estratégias comuns de substituição de dados era de difícil aplicação, pois o carro segue padrões muito específicos de funcionamento. Após esse processo o total de amostras ficou em 28226, sendo 15291(54.2%) para classe 0 e 12935(45.8%) para classe 1.

Para a divisão dos dados em treino e teste foi escolhido um *holdout* de 80% para treino, 10% para validação do modelo e 10% para teste do protótipo, essa foi a divisão escolhida

visando os procedimentos usados para seleção do modelo e *tunning* de hiperparâmetros, deixando o máximo possível de dados para treinamento e validação dos modelos, já que os procedimentos de *tunning* escolhidos usam validação cruzada, e esses procedimentos são feitos usando apenas os dados de treino, com os dados de validação e teste sendo apresentados ao modelo apenas nas respectivas avaliações.

Para normalização dos dados foi escolhido o método de *Standardization* através do *StandardScaler*³, mas não foi usado na etapa de pre-processamento, sendo aplicado nas etapas de seleção do modelo através de um *pipeline* e na etapa de treinamento dos modelos.

3. Seleção dos modelos e *tunning* de hiperparâmetros

Usando os trabalhos de Pistorius *et al.* (2020), Bhavsar e Ganatra (2012) e Osisanwo *et al.* (2017) como auxílio para escolha dos melhores modelos a serem testados no trabalho, foram selecionados os algoritmos *Random Forest (RF)*, *Support Vector Machine (SVM)* e *K-Nearest Neighbors (KNN)*. Usando os classificadores para as respectivas estratégias escolhidas a serem testadas teremos então, *RandomForestClassifier*⁴, *SVC*⁵ e *KNeighborsClassifier*⁶. A seguir estão as instâncias para os modelos sem hiperparâmetros, pois os mesmo serão definidos no procedimento seguinte.

```

1 RF = RandomForestClassifier()
2 SVC = svm.SVC()
3 KNN = KNeighborsClassifier()

```

Para escolha dos hiperparâmetros para cada um dos modelos especificados, foi o usado um método chamado *Grid-search* mais especificamente *gridsearchCV*⁷, que segundo Raschka e Mirjalili (2017) é um algoritmo de força bruta, onde é possível listar e testar vários hiperparâmetros para um modelo, e para cada combinação de parâmetros o desempenho será avaliado, até que todas as combinações sejam testadas e o conjunto ideal de parâmetros seja encontrado. Além disso, o método escolhido possui validação cruzada embutida no procedimento de *tunning*, isso permite que os melhores hiperparâmetros sejam selecionados considerando a generalização dos dados feita pelo modelo. Nos procedimentos também é usado um processo

³ <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

⁴ <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

⁵ <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

⁶ <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

⁷ https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

de *Pipeline*⁸ usado para fazer a normalização dos dados durante o processo de treinamento dos modelos nas etapas do *gridsearchCV*.

Por fim, o código escrito de forma genérica ficou da seguinte maneira:

```

1 pipe = Pipeline([('scaler', scaler), ('estimator', model)])
2 gs = GridSearchCV(pipe, cv=cv, param_grid=params, scoring=
    scoring)
3 gs.fit(X, y)

```

Sendo *model* o modelo que será usado, *scaler* o método de normalização, *params* são todos os parâmetros que vão ser testados para aquele modelo específico, *scoring* é a métrica de avaliação escolhida para estimar o desempenho do modelo, *cv* se refere aos parâmetros de validação cruzada, neste caso o método usado foi o *Kfold*⁹ com os seguintes parâmetros:

```

1 KFold(n_splits=3, shuffle=True, random_state=42)

```

Onde, *n_splits* é o número de divisões para validação cruzada, *shuffle* é usado para embaralhar os dados, e *random_state* controla a aleatoriedade dos dados. O método *fit* é usado para treinar o modelo, e os parâmetros *X* e *y* são os dados de treino e as respectivas respostas. E ao final desse processo é possível ter acesso às informações referentes aquele modelo, como melhores parâmetros que no caso nosso código genérico pode ser acessado com `gs.best_params_`, ou melhor resultado obtido com `gs.best_score_`, entre outros. Lembrando que esse procedimento foi aplicado nos três algoritmos pré-selecionados. Um dos principais motivos para a escolha do método de seleção do modelo foi evitar problemas posteriores a seleção, apesar de ser um método custoso computacionalmente, ao final a escolha tem uma grande chance de ser ótima, evitando que o modelo passe por uma nova análise com o intuito de investigar o problema e melhorá-lo com um novo *tunning* de hiperparâmetros e estratégias afins.

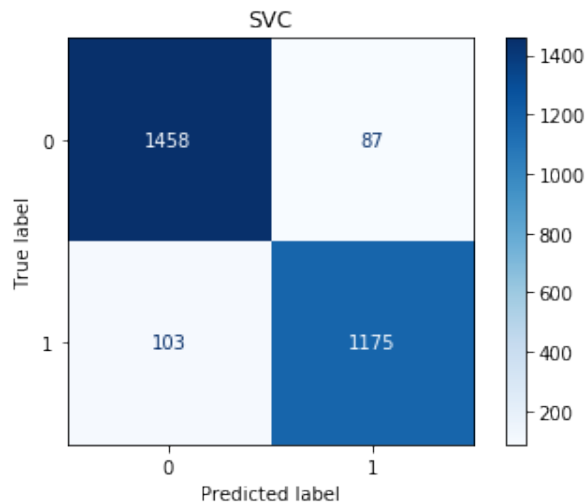
Por fim, para selecionar apenas um dos modelos para ser usado no protótipo, todos os modelos foram instanciados novamente com os hiperparâmetros selecionados, e treinados com os dados de treino já normalizados. Seus desempenhos foram avaliados usando a métrica *F1-score*, os resultados obtidos para cada um dos modelos para os dados de validação foi: SVC = 0.9251968503937007, RF = 0.9792332268370607, KNN = 0.9826771653543307. Para uma melhor avaliação, também foi observada a matriz de confusão para cada um dos modelos.

⁸ <https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>

⁹ https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

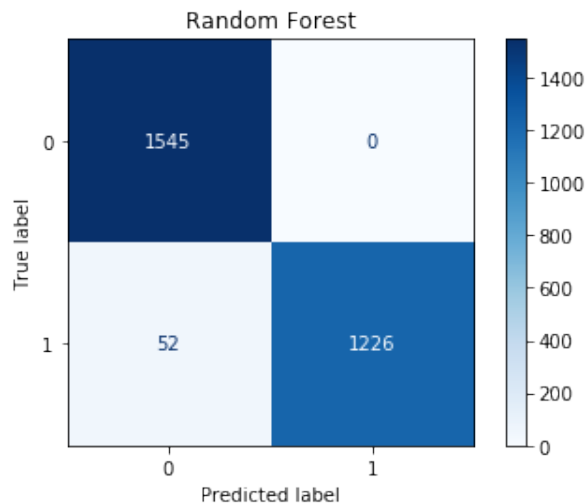
Analisando a matriz de confusão para o SVC, presente Figura 19, podemos ver 103 falsos positivos e 87 falsos negativos. Da mesma forma para a matriz de confusão do RF na Figura 20, temos um total de 52 falsos positivos e 0 falsos negativos. Por fim, para a matriz de confusão do KNN na Figura 21, temos um total de 14 falsos positivos e 30 falsos negativos.

Figura 19 – Matriz de confusão para os dados de validação do algoritmo SVC



Fonte:Autoria própria

Figura 20 – Matriz de confusão para os dados de validação do algoritmo RF

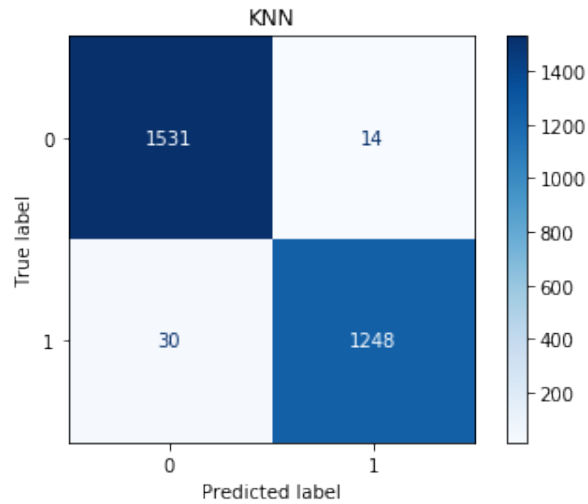


Fonte:Autoria própria

Ambos os modelos obtiveram ótimos desempenhos, mas após analisar as informações de *F1-score* e matriz de confusão para cada um deles, o KNN foi selecionado, por ter o melhor desempenho nos dados de validação.

4. Desenvolvimento e avaliação do protótipo

Figura 21 – Matriz de confusão para os dados de validação do algoritmo KNN



Fonte:Autoria própria

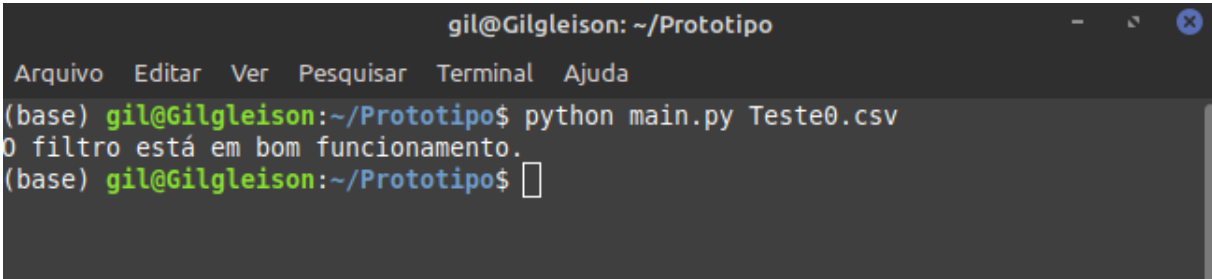
A próxima etapa é o desenvolvimento do protótipo, e para isso após modelo treinado e devidamente avaliado, foi salvo com os recursos do *joblib*¹⁰ para ser usado em um programa *python*, que inicialmente recebe os dados via linha de comando no terminal, faz toda a etapa de pré-processamento, descarte de informações irrelevantes e tratamento de valores nulos, tudo de forma automatizada, dado que a análise manual já foi concluída. Com intuito de aproveitar o recurso de coleta de dados do aplicativo Torque, o protótipo da aplicação desenvolvida roda localmente no computador do usuário, onde os dados coletados são usados como entrada para a aplicação, então os novos dados são avaliados para classificar o estado do filtro e retornará se o filtro deve ou não ser substituído.

Os dados de teste separados para avaliar o protótipo foram divididos, de modo que os dados da classe 1 ficaram em um *dataset* e salvo sem a *label* com o nome de *Teste1* e os dados classe 0 em outro também sem a *label* com o nome de *Teste0*. No caso de um filtro em bom estado a mensagem devolvida ao usuário é "O filtro está em bom funcionamento", caso contrário a mensagem é "O filtro precisa de manutenção". O retorno do protótipo para o *dataset* *Teste0* pode ser visto na Figura 22, e o retorno para o *Teste1* na Figura 23.

Para os casos em que o algoritmo não consegue classificar os dados são retornados estados de erro, os estados previstos até o momento são: caso do *dataset* estar vazio e caso em que os dados passados ao algoritmo não possuem as colunas necessárias para a classificação (que são as *features* usadas no trabalho). Para testar os casos de erro, foi criado um *dataset* apenas com valores *null* que serão descartados pelo algoritmo durante o pré processamento e ficará

¹⁰ <https://joblib.readthedocs.io/en/latest/>

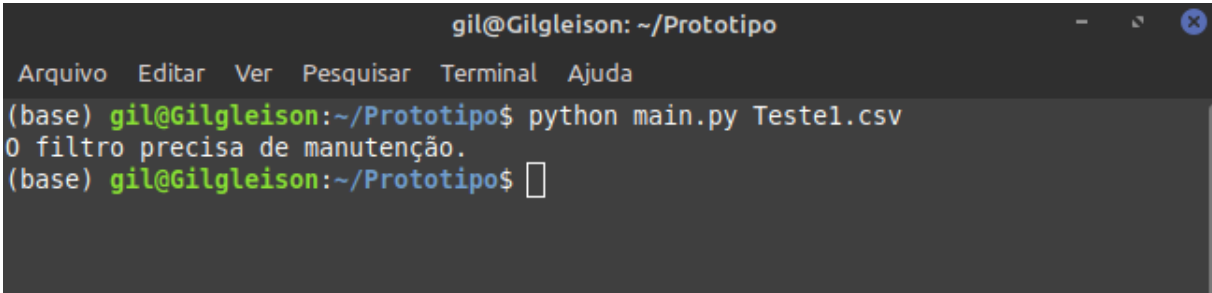
Figura 22 – Primeira avaliação do protótipo

A terminal window titled 'gil@Gilgleison: ~/Prototipo' with a menu bar containing 'Arquivo', 'Editar', 'Ver', 'Pesquisar', 'Terminal', and 'Ajuda'. The terminal shows the command 'python main.py Teste0.csv' being executed, followed by the output '0 filtro está em bom funcionamento.' and a prompt for the next command.

```
gil@Gilgleison: ~/Prototipo
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
(base) gil@Gilgleison:~/Prototipo$ python main.py Teste0.csv
0 filtro está em bom funcionamento.
(base) gil@Gilgleison:~/Prototipo$
```

Fonte: Autoria própria

Figura 23 – Segunda avaliação do protótipo

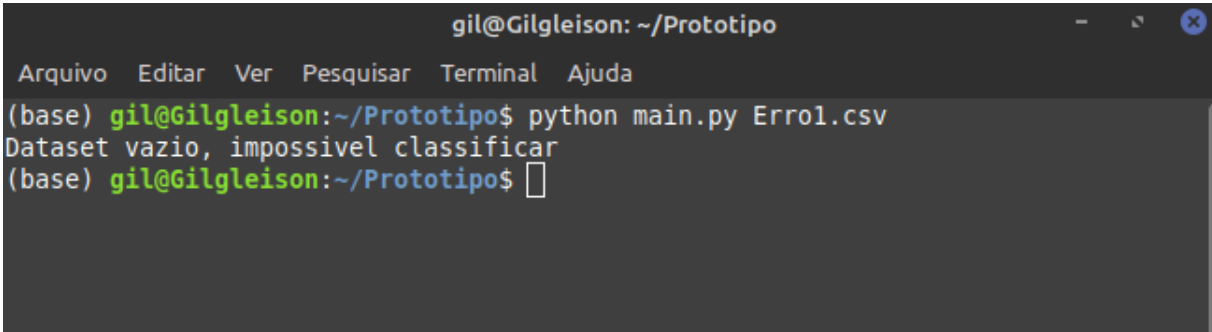
A terminal window titled 'gil@Gilgleison: ~/Prototipo' with a menu bar containing 'Arquivo', 'Editar', 'Ver', 'Pesquisar', 'Terminal', and 'Ajuda'. The terminal shows the command 'python main.py Teste1.csv' being executed, followed by the output '0 filtro precisa de manutenção.' and a prompt for the next command.

```
gil@Gilgleison: ~/Prototipo
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
(base) gil@Gilgleison:~/Prototipo$ python main.py Teste1.csv
0 filtro precisa de manutenção.
(base) gil@Gilgleison:~/Prototipo$
```

Fonte: Autoria própria

vazio, e outro sem uma das colunas necessárias, a resposta do protótipo para o primeiro caso pode ser visto na Figura 24, e para o segundo na Figura 25.

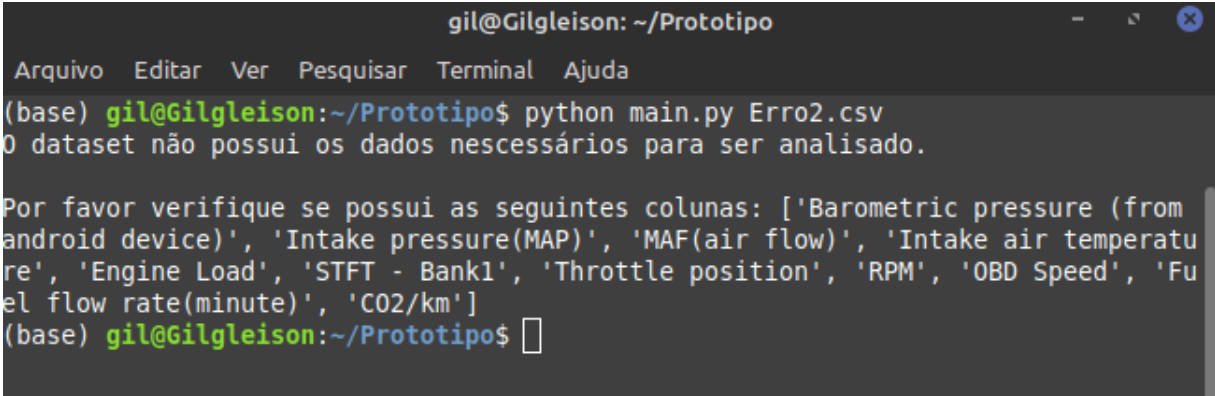
Figura 24 – Primeiro caso de erro

A terminal window titled 'gil@Gilgleison: ~/Prototipo' with a menu bar containing 'Arquivo', 'Editar', 'Ver', 'Pesquisar', 'Terminal', and 'Ajuda'. The terminal shows the command 'python main.py Erro1.csv' being executed, followed by the output 'Dataset vazio, impossivel classificar' and a prompt for the next command.

```
gil@Gilgleison: ~/Prototipo
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
(base) gil@Gilgleison:~/Prototipo$ python main.py Erro1.csv
Dataset vazio, impossivel classificar
(base) gil@Gilgleison:~/Prototipo$
```

Fonte: Autoria própria

Figura 25 – Segundo caso de erro

A terminal window titled 'gil@Gilgleison: ~/Prototipo' with a menu bar containing 'Arquivo', 'Editar', 'Ver', 'Pesquisar', 'Terminal', and 'Ajuda'. The terminal shows a command prompt where the user has run 'python main.py Erro2.csv'. The output is '0 dataset não possui os dados necessários para ser analisado.' followed by a list of required columns: ['Barometric pressure (from android device)', 'Intake pressure(MAP)', 'MAF(air flow)', 'Intake air temperature', 'Engine Load', 'STFT - Bank1', 'Throttle position', 'RPM', 'OBD Speed', 'Fuel flow rate(minute)', 'CO2/km'].

```
gil@Gilgleison: ~/Prototipo
Arquivo  Editar  Ver     Pesquisar  Terminal  Ajuda
(base) gil@Gilgleison:~/Prototipo$ python main.py Erro2.csv
0 dataset não possui os dados necessários para ser analisado.

Por favor verifique se possui as seguintes colunas: ['Barometric pressure (from
android device)', 'Intake pressure(MAP)', 'MAF(air flow)', 'Intake air temperatu
re', 'Engine Load', 'STFT - Bank1', 'Throttle position', 'RPM', 'OBD Speed', 'Fu
el flow rate(minute)', 'CO2/km']
(base) gil@Gilgleison:~/Prototipo$
```

Fonte: Autoria própria

4 CONCLUSÕES E TRABALHOS FUTUROS

Foi apresentado neste trabalho o desenvolvimento do MOMENT, uma aplicação de aprendizado de máquina na manutenção preditiva de veículos, um sistema capaz de fornecer um método confiável para avaliar a substituição do filtro de ar do motor, identificando se a peça está em bom estado ou precisa de manutenção. Foram descritos os diversos processos usados, desde a coleta dos dados até chegarmos ao protótipo final da aplicação.

O desenvolvimento do trabalho foi focado no desempenho dos modelos, buscando sempre obter o melhor resultado, já que o objetivo é fornecer um meio eficiente para avaliar se uma peça deve ou não ser substituída. Os modelos alcançaram resultados satisfatórios, com um bom desempenho nas classificações avaliadas, com isso podemos dizer que o protótipo produzido é preciso nas avaliações. O trabalho foi complexo, com muitas variáveis a considerar e também diversas possibilidades para leituras. O conjunto de sensores selecionados contribuíram muito para o resultado, juntamente com a estratégia usada para *tunning* de hiperparâmetros dos modelos e a diversidade de leituras do conjunto de dados. Por fim, conclui-se que os resultados obtidos com o trabalho superaram as expectativas, e o protótipo desenvolvido atende seu propósito de fornecer um método confiável para avaliar a substituição do filtro de ar do motor.

A validação do sistema está limitada ao Suzuki Grand Vitara 2014 AWD, para ser expandido a novos carros é necessário a coleta de um novo conjunto de dados para treinamento do modelo referente ao automóvel que será adicionado.

Como limitação, é entendido que seja o tempo, pois os carros elétricos estão em ascensão, e apesar de não ser em um futuro tão próximo, é possível que os carros com motores de ignição a centelha sejam substituídos gradativamente.

4.1 Trabalhos futuros

Para trabalhos futuros são previstas 4 abordagens, a primeira é evoluir de uma classificação para uma regressão, onde além de recomendar a substituição da peça, é possível prever o nível de desgaste do filtro. A segunda é a adição de novas peças e partes do carro, de modo que seja possível analisar diversos componentes em simultâneo. A terceira é o monitoramento e aprendizado contínuo. A quarta abordagem prevista é adicionar uma função de cálculo de custo esperado, relacionando o valor da despesa com a substituição ou não da peça. Essas abordagens ainda estão em estudo e podem ser implementadas separadamente ou de forma conjunta.

REFERÊNCIAS

- ALVES JUNIOR, E. I.; JATO, F.; HIROKI, G. B. **Desenvolvimento de uma unidade de gerenciamento eletrônico para motor volkswagen 1.6 l**. São Paulo Fatec Santo André, 2016.
- AZEEZ, M. A.; BANDARA, H. D. **Cloud-Based Driver Monitoring and Vehicle Diagnostic with OBD2 Telematics**. [S.l: s.n], 2015.
- BHAVSAR, H.; GANATRA, A. A comparative study of training algorithms for supervised machine learning. **International Journal of Soft Computing and Engineering (IJSCE)**, Citeseer, v. 2, n. 4, p. 2231–2307, 2012.
- BOSCH, R. **Manual de tecnologia automotiva**. [S. l.]: Editora Blucher, 2005.
- CAVALCANTE, L. H. **Sistema de monitoramento automotivo via rede Can**. [S.l: s.n], 2018.
- CORCOVIA, L. O.; ALVES, R. S. Aprendizagem de máquina e mineração de dados. **Revista Interface Tecnológica**, [S.l.], v. 16, n. 1, p. 90–101, 2019.
- DOUTORCARRO. **Códigos de Erro OBD-II Motor/Cambio – P0001 a P3493**. [S. l.], 2020. Disponível em: <https://www.doutorcarro.com.br/tabela-de-codigos-de-erro-dtcs-obd2-significados/>. Acesso em: 04 out.2020.
- ENGINEERS, S. O. A. **SAE J1962: Diagnostic Connector Equivalent to ISO/DIS 15031**. [S. l.], 2002. Disponível em: <https://law.resource.org/pub/us/cfr/ibr/005/sae.j1962.2002.pdf>. Acesso em: 04 out.2020.
- ENGINEERS, S. O. A. **SAE J1979: E/E Diagnostic Test Modes**. [S. l.], 2002. Disponível em: <https://law.resource.org/pub/us/cfr/ibr/005/sae.j1979.2002.pdf>. Acesso em: 04 out.2020.
- ENGINEERS, S. O. A. **SAE J2012: Diagnostic Trouble Code Definitions**. [S. l.], 2002. Disponível em: <https://law.resource.org/pub/us/cfr/ibr/005/sae.j2012.2002.pdf>. Acesso em: 04 out.2020.
- ENTLER, W. J.; ECU, G. D. T. P. **Centro paula de souza faculdade de tecnologia fatec santo andre tecnologia em eletrônica automotiva**. [S.l:s.n], 2018.
- GOYAL, N.; GOEL, V.; ANAND, M.; GARG, S. Smart vehicle: Online prognosis for vehicle health monitoring. **Journal of Innovation in Computer Science and Engineering**, Guru Nanak Publications, v. 9, n. 2, p. 12–22, 2020.
- KOWALIK, B. Introduction to car failure detection system based on diagnostic interface. In: IEEE. **2018 International Interdisciplinary PhD Workshop (IIPhDW)**. [S. l.], 2018. p. 4–7.
- KRIEBE, S.; KUSMENKO, E.; RUMPE, B.; SHUMEIKO, I. Learning error patterns from diagnosis trouble codes. In: IEEE. **2019 IEEE Intelligent Vehicles Symposium (IV)**. [S. l.], 2019. p. 179–184.
- MARCORIN, W. R.; LIMA, C. R. C. Análise dos custos de manutenção e de não-manutenção de equipamentos produtivos. **Revista de ciência & tecnologia**, [S.l.], v. 11, n. 22, p. 35–42, 2003.
- MASSARO, A.; SELICATO, S.; GALIANO, A. Predictive maintenance of bus fleet by intelligent smart electronic board implementing artificial intelligence. **IoT, Multidisciplinary Digital Publishing Institute**, [S.l.], v. 1, n. 2, p. 180–197, 2020.

- MILHOR, C. E. **Sistema de desenvolvimento para controle eletrônico dos motores de combustão interna ciclo Otto**. São Carlos. 72p. Dissertação de Mestrado-Escola de Engenharia de São Carlos, Universidade de São Paulo, 2002.
- MONARD, M. C.; BARANAUSKAS, J. A. **Conceitos sobre aprendizado de máquina: Sistemas inteligentes-fundamentos e aplicações**. Manole Ltda, [S.l.], v. 1, n. 1, p. 32, 2003.
- NASSIF, A. B.; SHAHIN, I.; ATTILI, I.; AZZEH, M.; SHAALAN, K. Speech recognition using deep neural networks: A systematic review. **IEEE Access**, IEEE, [S.l.], v. 7, p. 19143–19165, 2019.
- OBDDIAG. **Vehicle On-board Diagnostic**. [S. l.], 2020. Disponível em: <http://www.obddiag.net/adapter.html>. Acesso em: 07 dez. 2020.
- OBDII. **Does My Car Have OBD-II?** [S. l.], 2020. Disponível em: <http://www.obdii.com/connector.html>. Acesso em: 04 dez. 2020.
- OSISANWO, F.; AKINSOLA, J.; AWODELE, O.; HINMIKAIYE, J.; OLAKANMI, O.; AKINJOBI, J. Supervised machine learning algorithms: classification and comparison. **International Journal of Computer Trends and Technology (IJCTT)**, [S.l.], v. 48, n. 3, p. 128–138, 2017.
- OSS, M. **Leitura OBD2 através de smartphone**. Paraná: Universidade Tecnológica Federal do Paraná, 2018.
- PISTORIUS, F.; GRIMM, D.; ERDÖSI, F.; SAX, E. Evaluation matrix for smart machine-learning algorithm choice. In: IEEE. **2020 1st International Conference on Big Data Analytics and Practices (IBDAP)**. [S. l.], 2020. p. 1–6.
- RASCHKA, S.; MIRJALILI, V. **Python machine learning**. [S. l.]: Packt Publishing Ltd, 2017.
- RUSSEL, S.; NORVIG, P. *et al.* **Artificial intelligence: a modern approach**. [S. l.]: Pearson Education Limited London, 2013.
- SCIKIT-LEARN. **scikit-learn**. [S. l.], 2021. Disponível em: <https://scikit-learn.org>. Acesso em: 19 jul. 2021.
- SHAFI, U.; SAFI, A.; SHAHID, A. R.; ZIAUDDIN, S.; SALEEM, M. Q. Vehicle remote health monitoring and prognostic maintenance system. **Journal of advanced transportation**, Hindawi, v. 2018, 2018.
- SILVA NETO, E. F. d. **Um sistema de detecção de anomalias em sensor veicular baseado em classificadores one-class**. Dissertação (Mestrado) – Universidade Federal de Pernambuco, 2019.
- SILVA NETO, J. C.; LIMA, A. Gonçalves de. Implantação do controle de manutenção. **Revista Club de Mantenimiento**, [S.l.], n. 10, 2002.
- SOCORRO, A. F. **Desenvolvimento de uma unidade de gerenciamento eletrônico flex para motor volkswagen 1.6 l**. [S.l.: s.n], 2017.
- SPORT, R. M. **Preparação de Motores**. [S. l.], 2020. Disponível em: <http://www.garagemroyal.com.br/preparacao-de-motores/>. Acesso em: 13 dez. 2020.

VIANA, L. B. D. **Desenvolvimento e teste de soluções de comunicação para diagnóstico eletrônico**: estágio na empresa stratio. Tese (Doutorado), [S.l.], 2019.

XAVIER, J. N. **Manutenção–tipos e tendências**. Relatório Técnico. Minas Gerais: Tecém, 2005.

XAVIER, J. N. **Manutenção Preditiva Caminho para a excelência**. Minas Gerais: Tecém, 2005.