UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

MARIA EDUARDA ROSA DA SILVA

SELEÇÃO DE ATRIBUTOS EM APRENDIZADO DE MÁQUINA PARA IDENTIFICAÇÃO DE FALHAS EM MOTORES DE COMBUSTÃO INTERNA

Joinville
2022

MARIA EDUARDA ROSA DA SILVA

SELEÇÃO DE ATRIBUTOS EM APRENDIZADO DE MÁQUINA PARA IDENTIFICAÇÃO DE FALHAS EM MOTORES DE COMBUSTÃO INTERNA

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica, da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Dr. Giovani Gracioli

Coorientador: Dr. Gustavo Medeiros

Joinville
2022

Dedico este trabalho a Deus, minha mãe Flávia e meu pai Carlos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela fé e perseverança que me atribuiu.
Agradeço à minha mãe Flávia, por me ensinar a nunca desistir, acreditar em mim mesma e nos meus sonhos e pelo apoio incondicional.
Agradeço ao meu pai Carlos, por todo o suporte e conhecimento transmitido.
Sou muito grata ao meu namorado Vitor, que me apoiou em todos os momentos que precisei. Obrigada por tudo que me ensinou, você transformou minha graduação.
Um agradecimento especial ao meu colega Thiago, por todas as horas e esforços me ajudando para que este trabalho fosse desenvolvido.
Obrigada aos professores Dr. Giovani Gracioli e Dr. Gustavo Medeiros, pelas oportunidades e por tudo que me ensinaram.
Agradeço ao LISHA e a Fundação de Desenvolvimento da Pesquisa – Fundep Rota 2030/Linha V (projeto número 27192.02.01/2020.09-00), que me proporcionaram trabalhar em um projeto tão importante para o desenvolvimento da tecnologia automotiva no país. Agradeço a Renault por todo o suporte para que fosse possível o desenvolvimento da pesquisa com qualidade. Fazer parte de um projeto com tanto potencial foi de muito orgulho.

Minha eterna gratidão a todos!

"Se você não sabe aonde quer ir, qualquer caminho serve."
(Lewis Carroll).

RESUMO

A busca por métodos eficazes para obter uma detecção precisa de falhas, cresce constantemente, sendo o uso de sensores de medição uma das principais técnicas, porém custosos. Nesse sentido, a aplicação de algoritmos de Aprendizagem de Máquina para identificar falhas ganhou popularidade e adesão, devido ao alto desempenho e baixo custo comparado a outras técnicas. Para melhorar o desempenho dos algoritmos e ter maior precisão para identificação algumas estratégias podem ser abordadas, como selecionar os atributos que melhor descrevem a falha. Para isso, a Seleção de Atributos é realizada para identificar atributos significativos em um conjunto de dados. A seleção alinhada ao aprendizado contribuem. significativamente, para o ambiente industrial e automotivo, em virtude da capacidade de explorar grandes quantidades de dados para identificação e predição de falhas. O trabalho aborda as etapas de aquisição e processamento de dados do processo de Aprendizado de Máquina, procurando selecionar os dados antecedentes e causadores mais significantes, para identificar as falhas de ignição e detonação. Primeiramente, gerando uma grande quantidade de dados para aplicação dos métodos de Seleção de Atributos, utilizando o software INCA e o hardware desenvolvido para o projeto que este trabalho está inserido, com o desenvolvimento de um software para geração de experimentos para as simulações, em seguida, aplicando os métodos e utilizando os resultados para comparação do desempenho dos algoritmos de aprendizagem para identificação das falhas.

Palavras-chave: Seleção de Atributos. Falha de ignição. Detonação. Árvore de causas de falha. ECU.

ABSTRACT

The search for effective methods to obtain an accurate detection of faults grows constantly, being the use of measurement sensors one of the main techniques, however, very expensive. In this sense, the application of Machine Learning algorithms to identify faults has gained popularity and acceptance due to the high performance and low cost compared to other techniques. To improve the performance of the algorithms and have greater accuracy for identification, some strategies can be addressed, such as selecting the attributes that best describe the failure. For this, Features Selection is performed to identify significant attributes in a dataset. Machine Learning Aligned Features Selection contributes significantly to the industrial and automotive environment, due to the ability to exploit large amounts of data for failure identification and prediction. The work addresses the stages of data acquisition and processing of the Machine Learning process, seeking to select the most significant antecedent and causative data to identify misfire and knocking noise. First, generating a large amount of data for the application of the Features Selection methods using the INCA software and the hardware developed for the project that this work is inserted in with the development of a software for generating experiments for the simulations, then applying the methods and using the results to compare the performance of the learning algorithms to identify the failures.

Keywords: Features Selection. Misfire. Knocking Noise. Failure cause tree. ECU.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AM Aprendizado de Máquina

CAN Controller Area Network

CARB California Air Resources Board

DAQ Data Acquisition

ECU Engine Control Unit

EGR Exhaust-gas Recirculation

INCA Integrated Calibration and Application Tool

LISHA Laboratório de Integração de Hardware e Software

MD Mineração de Dados

MFD Misfire Fault Diagnosis

MLP Multilayer Perceptron

NaN Not a number

OBDII On-Board Diagnostics II

ODT Object Description Table

RFECV Recursive Feature Elimination with Cross-Validation

RFE Recursive Feature Elimination

SA Seleção de Atributos

SFS Sequential Feature Selector

VIF Variance Inlfation Factor

SUMÁRIO

1 INTRODUÇÃO

Com o avanço tecnológico surgiu a preocupação com ações preditivas, a fim de preservar a eficiência e o desempenho de equipamentos e máquinas. No ambiente industrial o monitoramento preditivo é uma das principais ferramentas para apontar com antecedência os possíveis problemas e impedir falhas, evitando danos e prejuízos por ocorrências inesperadas.

O monitoramento das condições de motores é importante para garantir seu desempenho e vida útil, tendo em vista que, são potencialmente expostos a falhas. Falha de ignição é a falta de combustão no cilindro, devido à ausência de faísca ou outra causa e é uma das mais comuns em motores multicilindros, podendo causar uma série de problemas como destruição do equilíbrio do motor e o aumento da emissão de gases tóxicos (BINGAMIL; ALSYOUF; CHEAITOU, 2017). A detecção de falha de ignição do motor é um requisito da norma *On-Board Diagnostics* II (OBDII), presente na legislação de vários países, incluindo o Brasil, assim, entende-se que é necessário predizer para reduzir a ocorrência de falhas.

Assim como a falha de ignição, a detonação é considerada uma das falhas mecânicas mais comuns em motores de combustão interna e é um fenômeno que limita a eficiência de motores. Muitas décadas de pesquisas indicaram que os mecanismos de combustão associados à detonação, não foram totalmente compreendidos devido à alta complexidade do fenômeno. A falha é causada principalmente por autoignição da mistura ar-combustível e, dependendo, da evolução de pressão e temperatura durante o ciclo de combustão (STEURS; BLOMBERG; BOULOUCHOS, 2014).

Nesse contexto, novos sistemas e processos estão sendo pesquisados e desenvolvidos com o intuito de garantir tecnologias cada vez mais seguras e confiáveis. A difusão de sistemas computacionais nas diferentes áreas do conhecimento, tem contribuído para geração de uma quantidade crescente de dados a cada dia. Como consequência, surge o desafio de encontrar uma forma de armazenar e extrair informação útil de uma grande quantidade de dados (ALMEIDA et al., 2018).

A fim de obter informações úteis desse conjunto de dados, pode-se utilizar técnicas de Aprendizado de Máquina (AM) (MITCHELL, 1997). Aprendizado de Máquina é um conjunto de métodos que exploram e analisam grandes quantidades de dados com o intuito de automatizar modelos analíticos e entender a relevância das informações com o objetivo de auxiliar no processo de tomada de decisão (DANTAS, 2017).

Desse modo, uma alternativa para o desafio de lidar com a grande quantidade de dados é a utilização de técnicas de Mineração de Dados (MD) no Aprendizado de Máquina, entretanto, é possível que alguns dados não estejam adequados ao método de mineração que será utilizado ou para a saída dos algoritmos de AM desejada. Para adequar tais dados, faz-se uma etapa de pré-processamento em que, inicialmente, se realiza uma análise dos dados para identificar os

relevantes e os que não são importantes ou adicionem pouca informação para o objetivo proposto (AZUAJE et al., 2006).

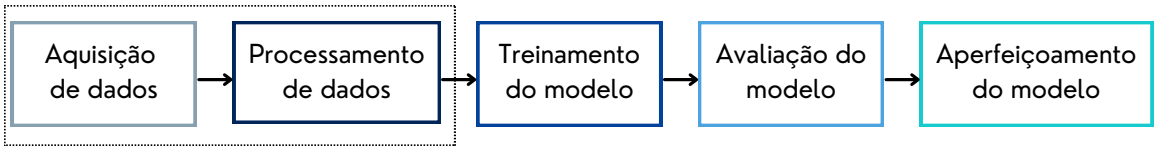
Técnicas de pré-processamento em Mineração de Dados são de fundamental importância para assegurar a qualidade dos dados fornecidos como entrada nos algoritmos. Para isso, a Seleção de Atributos é realizada para combinar e identificar atributos significativos em um conjunto de dados. A aplicação requer grande cuidado, pois demanda muito processamento e leva uma boa parte do tempo necessário para melhorar os dados que serão utilizados (LIU; MOTODA, 1998). A Seleção de Atributos (SA) é uma das técnicas que pode ser usada para redução de dimensionalidade de dados e aumento da acurácia de algoritmos sem afetar o desempenho, ou até mesmo melhorá-lo, para assim aumentar o poder preditivo do classificador (DANTAS, 2017).

Existem três abordagens para a Seleção de Atributo, Embutida, Filtro e *Wrapper*. A Abordagem Filtro faz a seleção com base na sua importância para a previsão da variável em estudo, independente do algoritmo de aprendizagem. A abordagem *Wrapper* realiza a seleção de subconjuntos de atributos durante o processo de treinamento por meio do algoritmo de aprendizagem com maiores custos de processamento. A abordagem Embutida é uma combinação das anteriores em que procura um conjunto ótimo de acordo com o algoritmo durante o treinamento (APOLÓNIA, 2019).

Portanto, as técnicas de Seleção de Atributos, alinhadas com o Aprendizado de Máquina, contribuem significativamente para o ambiente industrial e automotivo, em virtude de sua capacidade de explorar grandes quantidades de dados, afim de gerar informações identificando padrões, regras e aspectos relevantes que possam ser utilizados em predições e decisões estratégicas (SCHUCH et al., 2010).

De maneira simplificada, é possível observar na Figura 1 as etapas necessárias para o desenvolvimento do AM.

Figura 1 – Etapas processo de Aprendizagem de Máquina



Fonte: Autora (2022).

Nesse sentido, o trabalho aborda os passos iniciais de aquisição de dados e processamento de dados, tendo em vista que, ocorreu desenvolvimento para aquisição dos conjuntos de dados e a SA é uma forma de pré-processamento e tratamento dos dados antes que estes sejam utilizados no modelo dos algoritmos.

1.1 OBJETIVO

O objetivo deste trabalho é aplicar diferentes técnicas de Seleção de Atributos de forma a gerar conjuntos de variáveis relevantes relacionadas as causas da falha de ignição e detonação retiradas da *Engine Control Unit* (ECU), com o intuito de permitir a aplicação de algoritmos de Aprendizado de Máquina para identificar as falhas. A partir disso, comparar os resultados dos conjuntos analisando o desempenho dos algoritmos. Por fim, a partir dos resultados, encontrar o melhor conjunto de atributos que possa identificar uma falha antes que ocorra, levando em consideração condições mecânicas e elétricas.

1.1.1 Objetivos Específicos

- Gerar uma grande quantidade de dados para aplicação das técnicas Seleção de Atributos a partir de simulações da falha de ignição e detonação no carro;
- Criar um software para a geração de experimentos para as simulações;
- Aplicar diferentes técnicas de Seleção de Atributos aos dados gerados;
- Formar conjuntos de variáveis relacionados as causas das falhas;
- Comparar o desempenho dos algoritmos de aprendizagem com cada conjunto.

2 MINERAÇÃO DE DADOS E O CONTEXTO DAS FALHAS

Neste capítulo foram introduzidos conceitos essenciais para a compreensão do trabalho e dos resultados alcançados, dividindo em dois principais assuntos: o funcionamento do motor com falhas de ignição e detonação, e, os processos de mineração de dados. Essa divisão foi necessária para compreender o objeto de estudo e a escolha das estratégias que foram aplicadas.

A proposta do trabalho é constante validação do cotidiano da indústria, observando todos os fatores que podem levar à ocorrência das falhas, incluindo fatores ambientais, elétricos e mecânicos. Assim, conhecer o funcionamento do motor, das falhas e quais as possíveis causas, se faz necessário para compreender as escolhas e decisões tanto com os dados quanto com os métodos e estratégias ao longo do texto.

Neste capítulo, inicia-se apresentando o funcionamento básico dos motores de combustão interna, da ECU, das falhas de ignição e detonação, e, suas principais causas. Em seguida, apresenta-se os processos de mineração de dados e aprendizagem, tratamento e análise de dados que foram a base para o desenvolvimento do trabalho.

2.1 FALHAS EM MOTORES DE COMBUSTÃO INTERNA

Nesta seção são introduzidos os conceitos relativos ao objetivo de estudo, isto é, o motor de combustão interna, incluindo as fases de funcionamento e quais componentes estão envolvidos nesse processo; a falha de ignição e suas definições; a detonação e suas definições; as possíveis causas que levam a ocorrência das falhas, explicando os subsistemas e os componentes que influenciam nesse processo, com apresentação da árvore de falha; a *Engine Control Unit* responsável pela coleta e envio de dados relativos ao motor, de onde é possível retirar informações para análises; o arquivo A2L sendo uma descrição padronizada para formatar armazenagem de dados, fundamental para que este trabalho pudesse ser realizado.

2.1.1 Motores de Combustão Interna

Motores são dispositivos com a função de converter energia de diferentes tipos em energia mecânica. Os motores de combustão interna são definidos como um sistema mecânico que tem por finalidade transformar energia calorifica advinda da combustão da mistura ar e combustível, realizada por meio da sua compressão em energia capaz de realizar trabalho (MAHLE, 2016). Motores de combustão interna são empregados em inúmeras aplicações, principalmente como fonte de potência com a finalidade de locomoção de veículos (RODRIGUES, 2018).

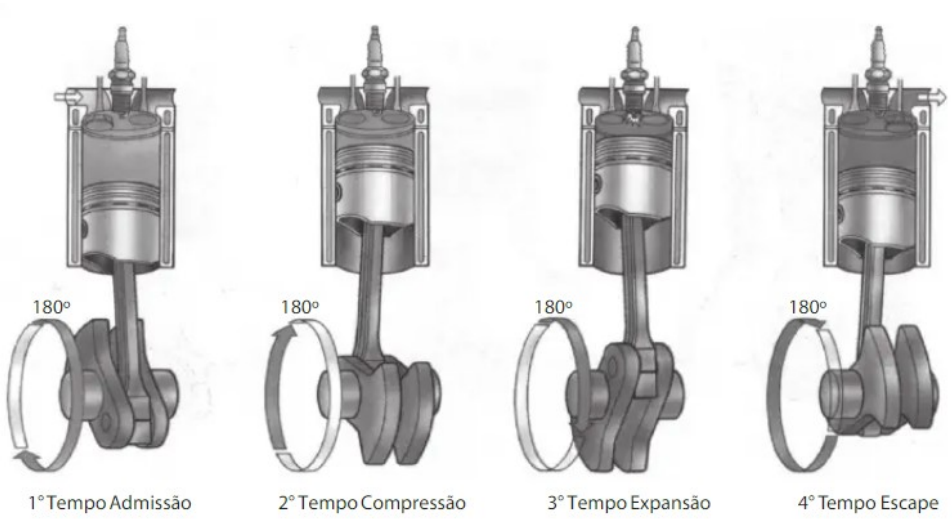
Inicialmente, acontece a mistura combustível e ar que será comprimida na câmara de combustão, nessa fase ocorre uma rápida inflamação proveniente da combinação do combustível

com qualquer material inflamável (gasolina, álcool, diesel), em cada cilindro. Em seguida, há a queima, que exerce uma compressão contra a parte superior do pistão, resultando no desloque em direção ao virabrequim (eixo de manivelas). A biela é o elemento de conexão entre o pistão e o virabrequim, e, tem a função de transmitir força que faz com que o pistão se desloque na fase de expansão dos gases, portanto, é responsável pela conversão do movimento retilíneo do pistão para o rotativo do virabrequim (MAHLE, 2016).

Os motores de combustão interna são classificados em relação ao princípio de operação em dois tipos: do ciclo otto e ciclo diesel. O ciclo de operação é o conjunto de transformações na massa gasosa no interior da câmara que se repetem periodicamente com a função de obter trabalho mecânico (RODRIGUES, 2018). O estudo foi desenvolvido utilizando um motor otto de 4 cilindro, portanto, este foi o foco desta fundamentação teórica.

Os motores de ignição por faísca ou otto classificam o motor de combustão interna como aquele qual a mistura combustível e ar inflama-se por meio de uma centelha elétrica, ocorrendo a queima da mistura e a expansão dos gases (MAHLE, 2016). Nesse caso, define-se o tempo de motor por um curso de pistão, que percorre quatro cursos para que seja completado um ciclo de trabalho (BRUNETTI, 2012). Na Figura 2 é possível observar os quatro tempos de motores são descritos baseado em Brunetti (2012).

Figura 2 – Os quatro tempos do ciclo otto.



Fonte: Brunetti (2012).

1. **Admissão**: fase em que o pistão se desloca do ponto morto superior ao ponto morto inferior, a válvula de escape está fechada, a válvula de admissão se abre e a mistura de combustível e ar é vaporizado ao interior do cilindro. Ocorre movimento de 180° do virabrequim;
2. **Compressão**: nessa fase, a válvula de admissão e escape estão fechadas, o pistão se desloca para o ponto morto superior e comprime a mistura combustível e ar no interior da câmara. Completa a primeira volta, somando um movimento de 360° do virabrequim;
3. **Combustão**: pouco antes de encerrar a fase de compressão, o sistema de ignição é acionado fazendo a corrente elétrica ser transmitida à vela, produzindo uma centelha entre os

eletrodos. Os gases da mistura são inflamados e entram em expansão, reação química dos gases de combustão. Esta expansão é transmitida pelo pistão, ocasionando o movimento de rotação do moto. Ocorre novamente o movimento do virabrequim;

4. **Escape**: depois de ocorrer a expansão dos gases, consequentemente, a explosão. Os gases saem do cilindro à medida que o pistão se movimenta para o ponto morto superior. O virabrequim completa a segunda volta.

2.1.2 Falha de Ignição

De acordo com os regulamentos do *California Air Resources Board* (CARB)(BOARD, 1991) , a falha de ignição do motor significa "[...]falta de combustão no cilindro devido à ausência de faísca, medição de combustível ruim, compressão ruim ou qualquer outra causa.". A falha de ignição é uma das falhas comuns dos motores multicilindros, que afeta seriamente a potência, podendo ser reduzida em 25% (SHARMA; SUGUMARAN; DEVASENAPATI, 2014), afetando também a segurança operacional dos motores reduzindo sua vida útil.

A falha acarreta problemas de aceleração fraca, aumento da temperatura de componentes, falha no sistema de lubrificação, ruído e vibração, corrosão e/ou desgaste por abrasão, aumento do consumo de combustível e emissões de poluentes, devido à maior concentração de hidrocarbonetos não utilizados presentes nos gases de escape (BINGAMIL; ALSYOUF; CHEAITOU, 2017; QIN et al., 2021)

O diagnóstico de falha de ignição (*Misfire Fault Diagnosis* - MFD) do motor é parte importante do sistema *On Board Diagnostics* II (OBDII) introduzido pelo Estados Unidos em 1996. A principal função do sistema OBD é o monitoramento contínuo dos parâmetros básicos do sistema, seu objetivo é reduzir as emissões de poluentes e prevenir danos ao conversor catalítico (BOGUS; MERKISZ, 2005). A detecção eficaz da falha de ignição promove a redução e a prevenção, além da conservação de energia e da diminuição das perdas econômicas causadas pela combustão incompleta.

Consequentemente, impulsionado pelo OBDII, muitos pesquisadores têm procurado apresentar métodos eficazes para obter detecção precisa da falha de ignição. Uma das técnicas para se analisar essa falha no motor é através de sensores de medição como velocidade angular do virabrequim, torque de freio e pressão da câmara de combustão, no entanto, esses sensores são caros pois precisam suportar temperaturas e pressões muito altas. Dessa forma, a aplicação de algoritmos de Aprendizado de Máquina para detectar a falha ganhou ampla adesão e popularidade, principalmente devido à alta precisão para previsão da falha (SINGH; POTALA; MOHANTY, 2018).

A detecção da falha de ignição apresenta um problema devido a variação no comportamento do sinal, já que o motor trabalha em uma ampla faixa de operação, afetando as medições. Outro fator complicador são as oscilações do virabrequim, que após a ocorrência de uma falha de ignição pode causar alarmes falsos (JUNG; FRISK; KRYSANDER, 2016). Devido à essas razões e a variedade de fatores que podem causar a falha, o processo de MFD inclui três

fases: julgar se ocorreu a falha; se ocorreu, identificar as causas; estimar a gravidade da falha de ignição (ZHENG et al., 2019).

Para que seja possível predizer a falha de ignição, é preciso conhecer as causas elétricas e mecânicas que acarretam essa condição no motor, portanto, apresenta-se com destaque as causas da falha estudada.

2.1.2.1 Causas

A falha de ignição pode ocorrer constantemente ou intermitentemente devido a falha no sistema de ignição, relação ar-combustível desequilibrada devido a sensor de oxigênio defeituoso, vela de ignição defeituosa, tampa do distribuidor rachada, sensor de fluxo de massa de ar defeituoso, junta do cabeçote queimada, temperaturas muito altas, falta de compressão ou mesmo recirculação dos gases de escape (SHARMA; SUGUMARAN; DEVASENAPATI, 2014).

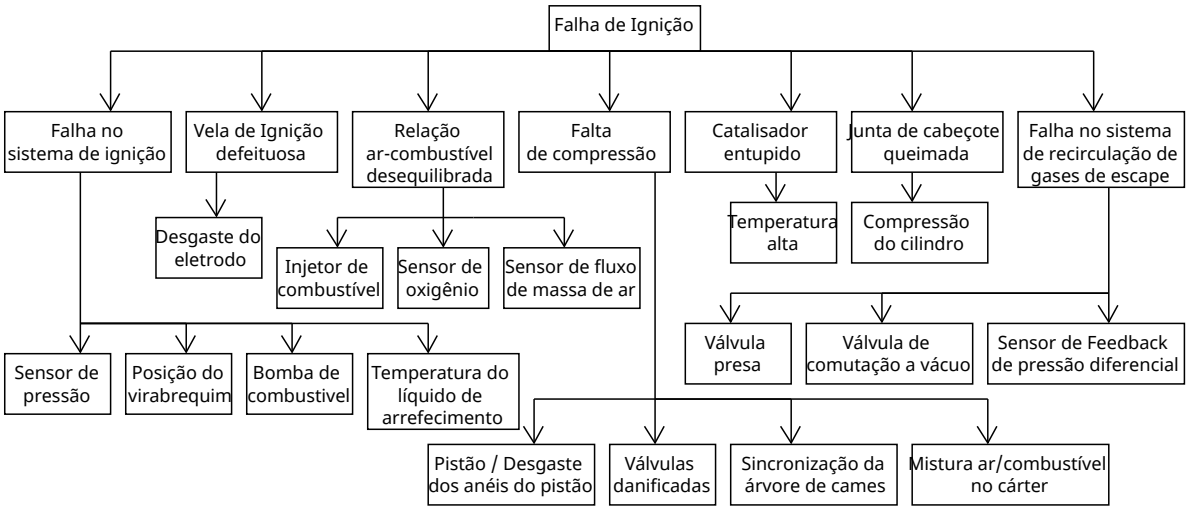
1. **Vela de ignição com defeito**. Para um desempenho ideal do motor, as velas de ignição do motor devem estar em condições limpas sem destruir os eletrodos, caso contrário, quando as velas de ignição ficarem sujas, isso mudará o funcionamento do motor. Uma vela de ignição suja ou ruim é revestida com um material como alcatrão, gasolina ou carbono. Por exemplo, a falha de ignição causada por pré-ignição leva a falha de folga entre dois eletrodos na vela de ignição. O aumento do aumento da folga da vela de ignição resulta em tensões de ignição mais altas, ocasionando perda de potência do veículo (AZRIN et al., 2021);
2. **Junta de cabeçote queimada**. Ocasionalmente, um vazamento na junta e, consequentemente, uma junta do cabeçote queimada pode ser causada pela compressão no cilindro devido ao uso de cabeçotes de alumínio em vez de ferro. Embora mais leve que o ferro, o alumínio tem uma taxa de expansão térmica muito maior, o que, por sua vez, causa mais estresse na junta do cabeçote. Uma junta de cabeçote danificada pode causar vazamento de compressão entre os cilindros. É possível investigar a junta do cabeçote verificando a pressão de compressão com manômetro ou um teste de vazamento (KOMORSKA, 2011);
3. **Relação ar-combustível desequilibrada**. Um desequilíbrio do cilindro da relação ar-combustível significa que a relação ar-combustível em um ou mais cilindros é diferente da dos outros cilindros devido a um mau funcionamento específico do cilindro. Esse mau funcionamento pode ser devido a um problema no injetor de combustível, um vazamento de admissão em um cilindro específico, um problema de fluxo do corredor de recirculação de gases de escape de um cilindro individual (NAKAGAWA; FUKUCHI; NUMATA, 2012). Um dos métodos para detectar a presença de condição de desbalanceamento do cilindro da razão combustível do ar, depende da medição do sensor de oxigênio do gás de escape universal a montante do catalisador (JAMMOUSSI et al., 201);
4. **Falha no sistema de recirculação de gases de escape (EGR)**. Quando a passagem EGR está obstruída, restrita devido válvulas presas ou caso as válvulas EGR fiquem carbonizadas

pela fuligem presente nos gases de escape, o fluxo de gás de escape é insuficiente. Além disso, uma falha no sensor de feedback de pressão diferencial ou em uma válvula de comutação de vácuo podem gerar uma falha no sistema EGR (WEI et al., 2012). Ainda que as válvulas EGR tenham sido concebidas para resistir às temperaturas elevadas, é possível que ocorram danos devido ao calor;

5. **Conversor catalítico entupido**. O motor, sob alta velocidade em condições normais sem falhas de ignição, atinge cerca de 950°C. Entretanto, ao atingir uma temperatura superior, geralmente acime de 1000°C, ocorre o superaquecimento do escape que fará que o conversor catalítico comece a derreter e acabe entupindo (MARTIN et al., 1993);
6. **Falta de compressão**. Quando a junta do cabeçote está amassada ou mal vedada, pode haver folgas, fazendo com o que os gases vazem ocasionando a perca da compressão. Além disso, pistão e anéis de pistão desgastados por deficiência de lubrificação, vedações de válvula danificadas por superaquecimento, acionador hidráulico defeituoso, sincronização incorreta do eixo de comando ou quando a mistura ar/combustível atinge o cárter (TING; MAYER, 1974);
7. **Falha no sistema de ignição**. O motor pode vir a falhar por uma falha em um sensor ou em um componente do sistema de ignição, como: bobinas, cabos e velas. Nas bobinas o defeito ocorre entre o distribuidor e as velas de ignição, pois uma ou mais velas não recebem a quantidade necessária de carga. Velas de ignição defeituosas, bobinas e cabos de ignição de alta tensão danificados ou uma má configuração podem provocar que a ignição se dê cedo demais ou tarde mais. Exemplos de sensores são: sensor de pressão no coletor de sucção, sensor de posição do virabrequim, sensor de temperatura do líquido refrigerante, entre outros (DZIUBINSKI et al., 2017).

Com as informações acima, foi construída uma árvore de causas da falha de ignição. Como pode ser visto na Figura 3.

Figura 3 – Esquemático de árvore de causas da Falha de Ignição



Fonte: Autora (2022).

A árvore foi organizada de forma que cada nó seja uma causa que leva a ocorrência do fator anterior. A raiz do problema, a falha de ignição, associa-se com os primeiros galhos, as principais causas que podem ocasionar a ocorrência da falha, que, por conseguinte, associa-se com as folhas contendo as causas que podem provocar os fatores para ocorrência da falha. Exemplificando, a falha de ignição pode ocorrer devido uma vela de ignição defeituosa que, por sua vez, pode ser ocasionada caso aconteça o desgaste do eletrodo.

2.1.3 Detonação

A detonação é caracterizada por oscilações de pressão e temperatura de alta frequência resultantes da autoignição da mistura não queimada na câmara de combustão em motores de ignição, podendo danificar gravemente os pistões além de, acabar criando desagradáveis ruídos que podem incomodar os motoristas. Quando as condições na região de não queima são favoráveis, a mistura não queimada pode se auto-inflamar. Isso ocorre tipicamente nos chamados pontos quentes no gás final. Na prática, a detonação é suprimida retardando o ponto de ignição e, em condições reais de condução, enriquecendo a mistura ar-combustível (DOORNBOS; DENBRATT; DAHL, 2018; STEURS; BLOMBERG; BOULOUCHOS, 2014).

Segundo Zhen et al. (2012), a principal causa da detonação do motor é o resultado da auto-ignição no gás final antes de ser atingido pela centelha emanado da vela de ignição. A auto-ignição raramente é homogênea, geralmente ocorre de forma aleatória em centros localizados a partir de propagações de reação, em relação aos componentes eletrônicos, mecânicos e térmicos. A autoignição no gás final aumenta a taxa de liberação de calor e desencadeia um pulso de pressão fazendo com que a estrutura do motor vibre (KALGHATGI; ALGUNAIBET; MORGANTI, 2017).

Em geral, a gravidade da detonação depende do ponto do ciclo de combustão em que ocorre a autoignição: um atraso de ignição prolongado deve reduzir a intensidade da detonação e vice-versa. Em condições pobres, a massa adicional na câmara de combustão reduz a temperatura média e a concentração de combustível. Medições de tubos de choque com combustíveis substitutos da gasolina mostraram que contrações e temperaturas mais baixas de combustível estendem independentemente o tempo de autoignição. Por outro lado, a menor capacidade de calor específico da mistura mais pobre, traz o momento de autoignição para frente porque uma menor capacidade de calor específico permite menores perdas de calor de processos de formação de radicais antes da autoignição (DOORNBOS; DENBRATT; DAHL, 2018).

Dependendo do histórico da falha e o período que ocorre, pode ocorrer inúmeros efeitos desfavoráveis como: ruptura de anéis do pistão, erosão da cabeça do cilindro, fusão do pistão, limitação da taxa de compressão do motor ou aceleração do veículo, aumento da poluição do ar, possibilidade de danos estruturais a longo prazo etc. (ZHEN et al., 2012).

2.1.3.1 Causas

A detonação pode ocorrer constantemente ou intermitentemente devido ao aumento de pressão, aumento de temperatura, falta de recirculação dos gases de escape, válvulas mal configuradas ou danificadas, relação ar-combustível desequilibrada tornado a mistura pobre, combustível com baixa octanagem, falha no sensor de detonação ou falha no sistema de ignição.

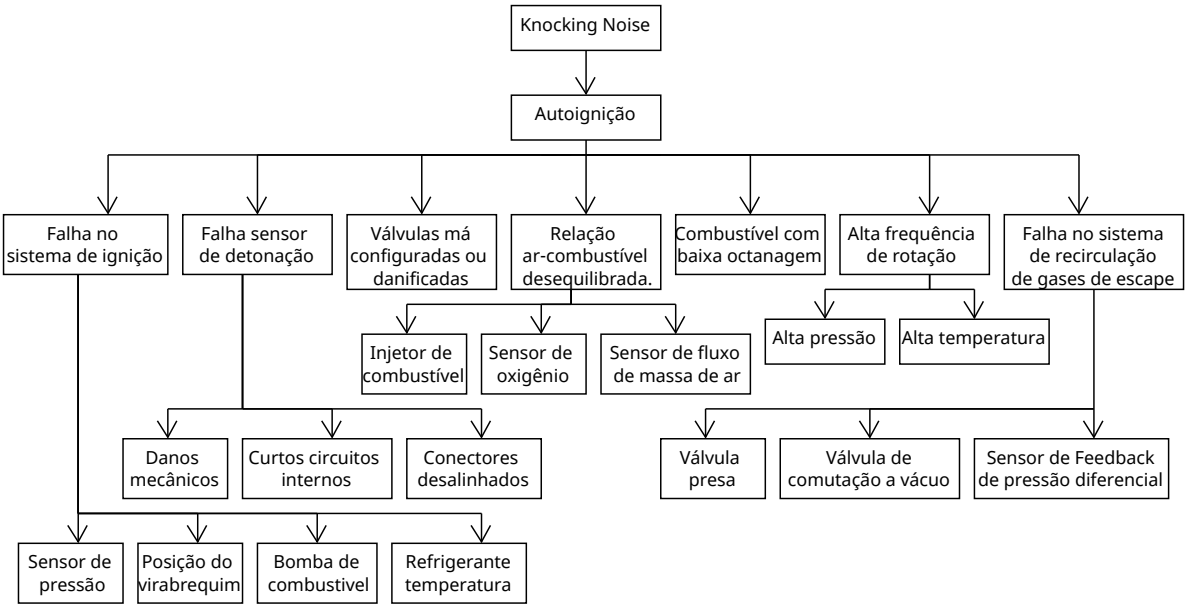
1. **Alta pressão**. A pressão afeta o tempo de autoignição. Sob pressões mais altas, independente da temperatura, o tempo de autoignição é diminuído. Uma pressão de entrada mais alta também é conhecida por aumentar a magnitude das oscilações de pressão sob condições de detonação (DOORNBOS; DENBRATT; DAHL, 2018);
2. **Alta temperatura**. O aumento nas velocidades convectivas causadas pela rápida expansão da massa da mistura na autoignição, fortes transientes de pressão na presença de reações e variações transitórias de densidade que influenciam localmente a liberação de calor, são fatores que potencializam o aumento da taxa de transferência de calor e, consequentemente, o aumento da temperatura para a detonação (CORTI; FORTE, 2009);
3. **Relação ar-combustível desequilibrada**. Um desequilíbrio do cilindro da relação ar-combustível significa que a relação ar-combustível em um ou mais cilindros é diferente da dos outros cilindros, devido a um mau funcionamento específico do cilindro. Sem combustível suficiente em cada cilindro, a mistura não vai queimar rápido o suficiente, permitindo várias detonações (NAKAGAWA; FUKUCHI; NUMATA, 2012);
4. **Combustível com baixa octanagem**. Dependendo do nível da octanagem que o motor estiver ajustado, a detonação pode ocorrer se colocado combustível com uma classificação de octanagem muito baixa. Combustíveis de alta octanagem queimam de forma mais uniforme e resistem à detonação (TOYOTA, 2019);
5. **Falha no sistema de recirculação de gases de escape (EGR)**. O EGR pode potencialmente reduzir ou eliminar o aumento de detonação associado a motores de baixa velocidade e permite que o ponto de ignição seja ajustado mais próximo do torque de freio máximo. Devido a essas características, motores que operam com EGR podem operar em temperatura com eficiências de até 40 $\%$ (DOORNBOS; DENBRATT; DAHL, 2018);
6. **Válvulas mal configuradas ou danificadas**. As válvulas configuradas inadequadamente ou danificadas afetam a concentração de gases residuais na câmara de combustão e influenciam diretamente na intensidade de detonação (DOORNBOS; DENBRATT; DAHL, 2018);
7. **Falha no sensor de detonação**. A maioria dos motores está equipada com um ou mais sensores de detonação. Esses sensores de detonação monitoram a quantidade de detonação que o motor está produzindo e, se tornar excessiva, ajustará o tempo de acordo para resolver o problema. Se o sensor de detonação não estiver funcionando corretamente, ele pode não detectar a detonação, o que significa que a ECU não saberá ajustar o tempo. A

maioria das falhas no sensor se deve ao mau uso, curto-circuito internos, danos mecânicos e desalinhamento dos conectores (TOYOTA, 2019; KAJI et al., 1986);

8. **Falha no sistema de ignição**. O motor pode vir a falhar por uma defeito em um sensor ou em um componente do sistema de ignição, como: bobinas, cabos e velas.Quando estes em alta tensão são danificados ou estão em uma má configuração pode-se alterar o tempo de ignição e ocasionar a detonação. (DZIUBINSKI et al., 2017).

Com as informações acima, foi construído uma árvore de causas da detonação. Como pode ser visto na Figura 4.

Figura 4 – Esquemático de árvore de causas da Detonação



Fonte: Autora (2022).

2.1.4 Engine Control Unit

Desde a introdução do regulador de tensão elétrica, ignição e microprocessadores em automóveis, na década de 1970, o uso de eletrônica automotiva tem aumentado constantemente e sendo implementado em grande escala. A eletrônica automotiva inclui controles e transmissão do motor, módulos de medição e diagnóstico, sistemas de segurança, entre outros. Os controles eletrônicos do motor presente em carros, regulam as emissões dos veículos, melhoram a economia média do combustível, aumentam a confiabilidade e reduzem os custos. Neste contexto, a *Engine Control Unit* (ECU) coordena a ignição por faísca, relações ar-combustível, velocidade de marcha lenta e comando de válvulas, entre outras funções (GEORGE; PECHT, 2014).

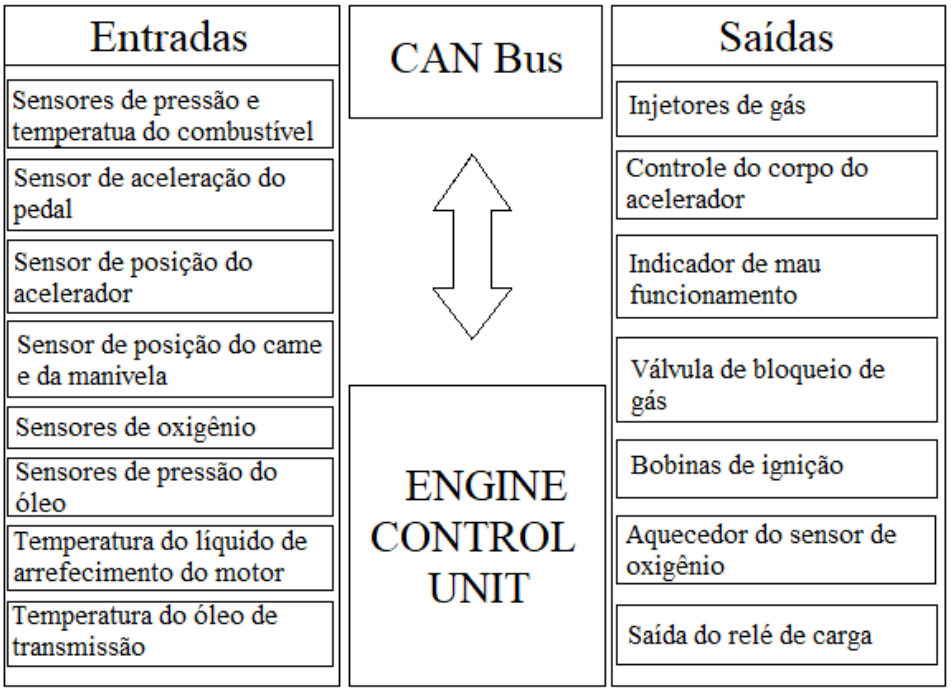
Uma ECU é um microprocessador responsável por controlar os atuadores do motor, isto é, um sistema de tempo real crítico, no qual muitos parâmetros são calculados, otimizando todo o processo de injeção e de ignição e o desempenho do motor de forma geral (CUATTO et al., 1998). A ECU é composta por um conjunto de sensores que medem periodicamente o estado

do motor e comunica a outras ECUs por meio de um barramento de *Controller Area Network* (CAN) (GEORGE; PECHT, 2014).

O processamento da ECU consiste em três etapas: coleta de dados, em que recebe sinais enviados por sensores (transdutores de entrada), processamento destes sinais e o envio do resultado de processamento para os atuadores (transdutores de saída), que executam os comandos recebidos. A ECU executa diferentes tarefas que são acionadas por eventos. Sempre que um determinado evento ocorre, uma instância de tarefa é criada e preparada para ser executada. As tarefas podem ser acionadas pelo tempo ou por um mecanismo, como os cilindros do motor (CUATTO et al., 1998).

Na Figura 5 é possível verificar a representação esquemática de uma ECU, adaptada por George e Pecht (2014).

Figura 5 – Representação Esquemática de uma ECU



Fonte: Adaptada de George e Pecht (2014).

2.1.5 A2L

O arquivo formato A2L é uma descrição padronizada, usada para declarar o formato de armazenamento de dados e as relações de conversão de diferentes tipos de ECU. Usando as tecnologias de comunicação da ECU, o sistema de calibração realiza a comunicação entre a ECU e as ferramentas de calibração, o que requer um protocolo de comunicação para fins de padronização. Algumas ferramentas de calibração comerciais compatíveis surgiram como o INCA que foi utilizado para realizar este trabalho (ZHUANG; YAN; ZHU, 2013).

A Associação para Padronização de Sistemas de Automação e Medição (ASAM) definiu o protocolo e a interface de comunicação padronizados e os padrões ASAP1, ASAP2 e ASAP3

que forma a arquitetura padrão do ASAP. O arquivo A2L é o arquivo descritivo de formato que atende ao padrão ASAP2.

O A2L descreve e define variáveis do controlador em uma linguagem padronizada como: nome da variável, descrição da variável, tipo de dados da variável, endereço inicial da variável, comprimento dos dados da variável, intervalo da variável, método de conversão. Com o arquivo A2L, a ferramenta de calibração consegue obter todas as informações de restrição das variáveis de medição e calibração definidas na ECU de forma segura e o usuário consegue acessar variáveis internas a ECU por nomes simbólicos (ZHUANG; YAN; ZHU, 2013).

No Listing 2.1 pode-se observar um exemplo da definição de uma variável de medição em um arquivo A2L (ASAM, 2022).

Listing 2.1 – Exemplo variável de medição

```
/begin MEASUREMENT ASAM.M.SCALAR.SWORD.IDENTICAL
    "Scalar measurement"
    SWORD CM.IDENTICAL 0 0 -32268 32267
    ECU_ADDRESS 0x13A04
    FORMAT "%5.0"
    /begin IF_DATA ETK  KP_BLOB 0x13A04 INTERN 1 RASTER 1 /end
   IF_DATA
/end MEASUREMENT
```

O arquivo A2L é fundamental para o trabalho, sendo usado como base para o desenvolvimento de uma ferramenta que auxilia na aquisição de dados e como suporte para a seleção dos atributos, dos quais as variáveis de medição são as principais utilizadas, relacionados ao controlador manipulado.

2.2 MINERAÇÃO DE DADOS PARA SELEÇÃO DE ATRIBUTOS

A proposta desta seção foi apresentar um resumo teórico, utilizado para fundamentar as estratégias e análises de Seleção de Atributos baseando-se em conceitos da área denominada Descoberta de Conhecimento em Banco de Dados, incluindo o Aprendizado de Máquina.

2.2.1 Mineração de dados

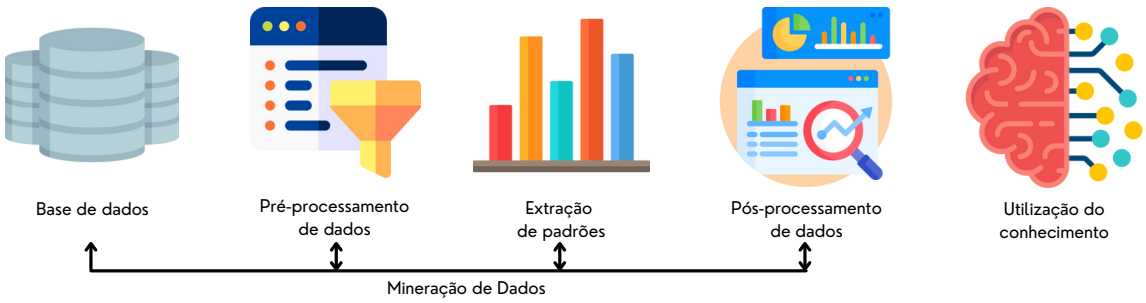
Levando em consideração a demanda de busca por soluções em torno de processos de otimização de dados, a fim de armazenar e extrair conhecimento útil de uma grande quantidade de dados, surgiu a Mineração de Dados. Esta é uma técnica relevante e a principal etapa da área denominada Descoberta de Conhecimento em Banco de Dados (*Knowledge Discovery in Databases* - KDD) (SOUZA, 2017). O KDD é o processo de identificação de padrões, a partir de dados, que sejam válidos e potencialmente úteis, e consiste em etapas de seleção, pré-processamento, transformação, mineração de dados e interpretação (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

Mineração de Dados (*Data mining* - MD) compreende em processos e métodos de seleção, tratamento, análise e interpretação de dados, procurando modelos, padrões e relacionamento entre variáveis para assimilar o conhecimento e transformá-lo em útil (APOLÓNIA, 2019). A MD engloba conhecimentos de análise estatística de dados, aprendizagem de máquina, reconhecimento de padrões e visualização de dados (CABENA et al., 1998).

A MD pode ser dividida em 3 fases: pré-processamento, extração de padrões e pós-processamento. Na Figura 6 é ilustrado o processo de Mineração de Dados, segundo Parmezan et al. (2012).

- A primeira fase de pré-processamento pretende conhecer o domínio da aplicação e a base de dados e prepará-los para a próxima fase. Geralmente, realizam-se as tarefas de preparação dos dados, limpeza dos dados, transformação de dados e Seleção de atributos. Essa fase é considerada a fase mais custosa, podendo consumir aproximadamente 80% do processamento, por isso é fundamental assegurar a qualidade de redução da dimensionalidade (LEE, 2005);
- A segunda fase de extração de padrões objetiva construir modelos que possam representar as informações embutidas nos dados, principalmente, utilizando de algoritmos de Aprendizado de Máquina (AZUAJE et al., 2006);
- A terceira fase de pós-processamento de dados tem por objetivo avaliar, validar e consolidar o conhecimento extraído na fase anterior. A avaliação ocorre por meio de interpretação dos resultados e visualização de padrões úteis e remoção dos irrelevantes. Nesta fase é necessário auxílio de especialistas da área estudada para validar os resultados alcançados (LEE, 2005).

Figura 6 – Mineração de dados



Fonte: Autora (2022).

Uma das principais metas da mineração é a descoberta relacionada com a predição e a descrição de dados. As metas de predição e descrição são alcançadas abordando os métodos de mineração de dados: classificação, regressão, agrupamento, sumarização, modelagem de dependência e identificação de mudanças e desvios (COSTA et al., 2013).

Assim, uma das aplicações da MD, com grande importância, é redução de dimensionalidade de dados, ao utilizar os métodos é possível reduzir o volume de informações, atributos, eliminando dados irrelevantes ao objetivo proposto e tornando o processo de

aprendizagem mais eficiente (ALMEIDA et al., 2018). Nesse sentido, os métodos de Seleção de Atributos (Features Selection) são utilizados para obter um conjunto mínimo que permita alcançar resultados mais próximos possíveis dos resultados do conjunto inicial, aumentando a acurácia de algoritmos e melhorando predições, sendo os objetivos propostos por esse trabalho (APOLÓNIA, 2019).

2.2.2 Aprendizado de máquina

Inteligência artificial (IA) é um ramo da ciência e tecnologia que cria máquinas e programas inteligentes para executar tarefas que requerem inteligência humana. Os sistemas de IA são regidos por algoritmos usando técnicas de Aprendizado de Máquina para demonstrar um comportamento inteligente e usam dados externos para obter excelente desempenho desses algoritmos (ANJILA, 2021).

O AM é o processo computacional que usa dados de entradas para alcançar uma tarefa desejada sem ser literalmente programado para produzir um determinado resultado, e consequentemente, desenvolver a capacidade de reconhecer padrões, aprender repetidamente e prever dados (NAQA; MURPHY, 2015). O objetivo de AM é a construção de sistemas capazes de adquirir conhecimento de forma automática, a partir disso, tomar decisões baseados em experiências acumuladas (MONARD; BARANAUSKAS, 2003).

Os algoritmos de AM automaticamente alteram ou adaptam sua arquitetura por meio de repetição para se tornarem cada vez melhores na realização da tarefa desejada. O processo de adaptação é chamado de treinamento, sendo parte da aprendizagem do AM. Existem algumas maneiras que um algoritmo pode se adaptar em resposta ao treinamento, como: o algoritmo pode ter parâmetros variáveis que são ajustados atrás de otimização iterativa, pode ter uma rede de possíveis caminhos ou pode determinar distribuições de probabilidade a partir dos dados de entrada (NAQA; MURPHY, 2015).

Assim, Ludermir (2021) descreve três tipos principais de Aprendizado de Máquina: Supervisionado, Não Supervisionado e Por Reforço.

- Aprendizado Supervisionado: para cada conjunto apresentado ao algoritmo é necessário apresentar uma resposta desejada, isto é, um rótulo informando a que classe o conjunto pertence. O objetivo é construir um classificador que possa determinar corretamente a classe de novos conjuntos de dados ainda não rotulado;
- Aprendizado Não Supervisionado: os conjuntos são fornecidos ao algoritmo sem rótulo e o algoritmo os agrupa por similaridades de atributos. O algoritmo analisa os atributos e tenta agrupar de forma a verificar o impacto dos atributos no contexto que estão sendo analisados;
- Aprendizado Por Reforço: o algoritmo não recebe a resposta correta, mas um sinal de reforço, recompensa ou punição, e faz uma hipótese baseado nos conjuntos.

2.2.2.1 Scikit-learn

A linguagem de programação Python está ganhando grande popularidade na área da computação, isso devido a sua natureza interativa de alto nível e grande disponibilidade de bibliotecas, tornando-se uma opção atraente para desenvolvimento algorítmico e análise de exploratória de dados (PEDREGOSA et al., 2011).

O Scikit-learn (Sklearn) é o pacote de aprendizado de máquina mais abrangente e de código aberto em Python. O pacote abrange quatro tópicos principais relacionados ao AM: transformação de dados, aprendizado supervisionado, aprendizado não supervisionado, avaliação e seleção de modelos. O Sklearn possui muitos recursos que se destacam entre os softwares de AM, fornece várias funções convenientes para executar esses métodos para transformação e pré-processamento de dados que foram utilizadas para o desenvolvimento deste trabalho (HAO; HO, 2019).

O objeto central do Sklearn é um estimador, que implementa um método de ajuste, aceitando como argumentos uma entrada matriz de dados e, opcionalmente, uma matriz de rótulos para problemas supervisionados (PEDREGOSA et al., 2011). Além disso, o pacote apresenta um módulo destinado a seleção de atributos e redução de dimensionalidade em conjuntos de dados, o *sklearn.feature_selection*, que apresenta seis abordagens diferentes (SCIKIT-LEARN, 2022b).

2.2.3 Seleção de atributos

Nos últimos trinta anos, a dimensionalidade de dados envolvidos em tarefas de Aprendizado de Máquina e Mineração de Dados aumentou expressivamente. Esse aumento ocasionou desafios para os métodos de aprendizagem já existentes tendo em vista que, com muitos dados um modelo de aprendizagem tende a *over-fitting*, diminuindo seu desempenho e aumentando a taxa de erro, assim, a qualidade dos dados diminui gradualmente. Para resolver esse problema de dimensionalidade aplica-se a Seleção de Atributos (TANG; ALELYANI; LIU, 2014; VENKATESH; ANURADHA, 2019).

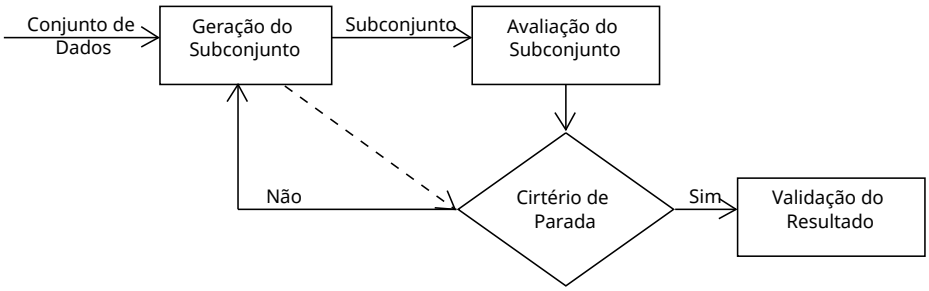
O principal objetivo da AS é construir um subconjunto de recursos menor possível, mas que represente todos os atributos vitais dos dados de entrada (ZEBARI et al., 2020). A SA busca escolher um subconjunto de atributos mais relevantes e limpar dados ruidosos e redundantes dos iniciais de acordo com determinado critério de avaliação de relevância, gerando um melhor desempenho do algoritmo de aprendizado, melhor interpretabilidade do modelo, aumento da escalabilidade, redução da complexidade computacional, armazenamento e custo (TANG; ALELYANI; LIU, 2014; VENKATESH; ANURADHA, 2019).

A seleção pode ser realizada segundo duas filosofias diferentes: avaliar um conjunto de atributos selecionados obtidos iterativamente segundo um critério de avaliação ou avaliar, individualmente os atributos, onde cada atributo tem um peso de acordo com sua relevância (APOLÓNIA, 2019). Assim, em SA o subconjunto de atributos é selecionado do conjunto com base em redundância e relevância dos atributos. Yu e Liu, 2004 classificaram o subconjunto

de atributos em quatro tipos: Barulhento e irrelevante; Redundante e Fracamente relevante; Fracamente relevante e não redundante; Fortemente relevante.

O processo tradicional de SA consiste em quatro etapas, conforme ilustrado na Figura 7: geração do subconjunto, avaliação do subconjunto, critério de parada e validação. A geração de subconjunto é um processo de busca que produz subconjuntos candidatos para avaliação com base em uma estratégia. Cada subconjunto é avaliado e comparado com o melhor anterior. Se o novo é melhor do que o anterior, substituirá este. Este processo é repetido até uma determinada condição de parada. As estratégias de classificação e avaliação os subconjuntos são baseados em estatísticas (RAMASWAMI; BHASKARAN, 2009).

Figura 7 – Processo de Seleção de Atributos



Fonte: Dash e Liu (1997)

Os algoritmos de classificação podem ser divididos em algumas categorias que serão apresentadas a seguir.

2.2.4 Abordagens para a Seleção de atributos

Conforme as informações disponíveis nos conjuntos de dados, os métodos de seleção de atributos podem ser classificados como: supervisionados, semi-supervisionados e não supervisionados. Os supervisionados precisam de um conjunto de dados rotulados para identificar e selecionas atributos relevantes, esse rótulo pode ser uma categoria, valor ordenado ou valor real. Os semi-supervisionados requerem que apenas alguns sejam rotulados. Os não supervisionados não requerem nenhum rótulo.

Os métodos não supervisionados podem ser categorizados em três abordagens de acordo com as estratégias escolhida para selecionar os atributos: abordagem Filtro, abordagem *Wrapped* e abordagem Embutida (SOLORIO-FERNáNDEZ; CARRASCO-OCHOA; MARTíNEZ-TRINIDAD, 2020).

A. Filtro

   A abordagem filtro é conhecida como método de malha aberta e é o mais antigo. Os métodos que seguem esta abordagem fazem as seleções de atributos com base na sua importância para a previsão da variável no conjunto. A seleção de atributos é realizada independente do algoritmo de mineração de dados, como consequência, reduz o tempo de agrupamento e a complexidade do algoritmo. Além disso, a abordagem apresenta um bom

desempenho e alta eficiência de computação (ZEBARI et al., 2020).

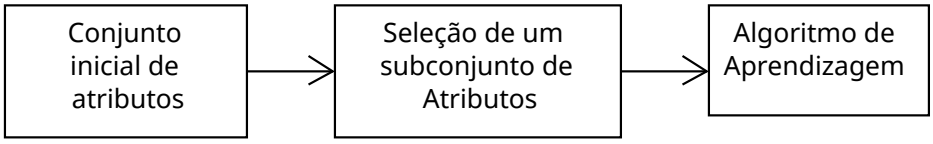
O método verifica os recursos com base em características intrínsecas antes do algoritmo de AM, medindo as características do recurso dependendo de quatro critérios de medição: informação, dependência, consistência e distância. A abordagem utiliza de padrões estatísticos para avaliar o subconjunto e filtrar atributos irrelevantes, sendo a seleção destes independentes da aplicação real dos recursos selecionados (LEE, 2005). Por estes motivos, a abordagem é mais eficiente do que a Wrapper mas é inferior no desempenho de agrupamento do subconjunto. Além de ser altamente versátil e adequado para conjuntos de dados de grande escala (CAI et al., 2018).

Nem todos os recursos de filtro podem ser usados para todas as classes de tarefas de mineração de dados. Portanto, os filtros também são classificados de acordo com a tarefa: classificação, regressão ou agrupamento (JOVIć; BRKIć; BOGUNOVIć, 2015). Dentre os métodos que usam esta abordagem pode-se citar: FOCUS, ABB, Relief, CFS, INTERACT, Consistency-based Filter, InfoGain, ReliefF, mRMR e Md (APOLÓNIA, 2019).

A desvantagem dessa abordagem é a de selecionar atributos redundantes e não selecionar atributos que, embora, não sejam importantes por si, ao serem combinados com outros sejam de grande importância. E assim, ao serem aplicados posteriormente no algoritmo de aprendizagem podem não otimizar o modelo (APOLÓNIA, 2019).

A Figura 8 demonstra a abordagem Filtro.

Figura 8 – Abordagem Filtro



Fonte: Autora (2022).

B. *Wrapper*

A abordagem *Wrapper*, ou abordagem de malha fechada, considera subconjuntos de atributos pela precisão do desempenho ou a taxa de erro do processo de classificação em um algoritmo de aprendizagem para realizar a seleção de atributos. Esta seleciona o recurso mais ideal para o algoritmo de previsão, consequentemente obtendo melhor desempenho e maior precisão (ZEBARI et al., 2020). Os métodos aplicados conforme a abordagem geram um subconjunto candidato de atributos e executam o algoritmo de aprendizagem considerando os dados representados pelo subconjunto selecionado do conjunto de treinamento iterativamente até que o critério de parada seja satisfeito (PARMEZAN et al., 2012).

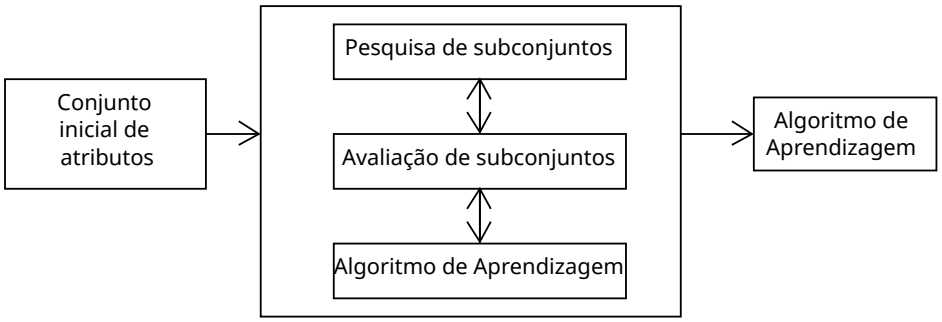
Para tarefas de classificação, o Wrapper avaliará um subconjunto conforme o desempenho do classificador, por exemplo Naive Bayes, C4.5 ou SVM, e da mesma forma para tarefas de agrupamento. Assim como nos Filtros a geração de subconjuntos depende da estratégia

adotada para avaliação.

Comparado a abordagem Filtro, os Wrappers são mais lentos para encontrar subconjuntos suficientemente bons pois dependem das demandas de recurso do algoritmo de aprendizagem. Nesse sentido, os subconjuntos de atributos são tendenciosos ao algoritmo de modelagem no qual foram avaliados. Assim para uma estimativa de erro de generalização confiável, é necessário que uma amostra de validação independente e outro algoritmo sejam usados após o subconjunto final ser encontrado (JOVIć; BRKIć; BOGUNOVIć, 2015).

O método obtém conjuntos com melhor desempenho do que os Filtros porque os subconjuntos são avaliados usando algoritmos de modelagem real. Desse modo, qualquer combinação de seleção de atributos e algoritmo de aprendizagem pode ser usado como Wrapper (JOVIć; BRKIć; BOGUNOVIć, 2015). Entretanto, a abordagem é de alta complexidade computacional e está sujeita a mais exposição ao *over-fitting*. Além disso, apresenta dificuldade em lidar com uma grande quantidade de dados (CAI et al., 2018). Na Figura 9 é possível observar uma ilustração da abordagem.

Figura 9 – Abordagem *Wrapper*



Fonte: Autora (2022).

C. Embutida

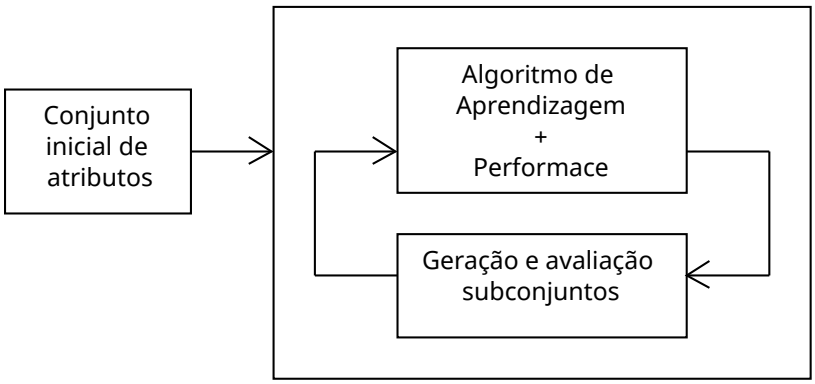
A abordagem Embutida é um método de seleção de atributos integrados que incorpora a seleção ao algoritmo de aprendizado e usa suas propriedades para orientar a avaliação de recurso (ZEBARI et al., 2020). Os métodos aplicados de acordo com a abordagem, selecionam o subconjunto de atributos no processo de construção do modelo de classificação, durante a fase de treinamento, e podem ser específicos para um dado algoritmo (PARMEZAN et al., 2012).

Esta abordagem incorpora as qualidades das abordagens Filtro e Wrapper. Mantendo o desempenho similar, a abordagem é mais eficaz e tratável computacionalmente do que o Wrapper. Efeito da execução repetida do classificador e o exame de cada subconjunto de recursos durante a implantação do algoritmo de mineração o que diminui o custo computacional e possibilita detectar dependências entre os atributos (ZEBARI et al., 2020).

O algoritmo de AM utiliza do próprio processo de seleção para realizar simultaneamente a

seleção de atributos e a avalição deste. O FS-Perceptron e o SVM-RFE são exemplos dessa abordagem (APOLÓNIA, 2019). Na Figura 10 é possível observar as etapas da abordagem.

Figura 10 – Abordagem Embutida



Fonte: Autora (2022).

Na Tabela 1 apresenta-se as vantagens e desvantagens de cada abordagem a partir do levantado nos tópicos anteriores.

Tabela 1 – Vantagens e desvantagens das abordagens de Seleção de Atributos

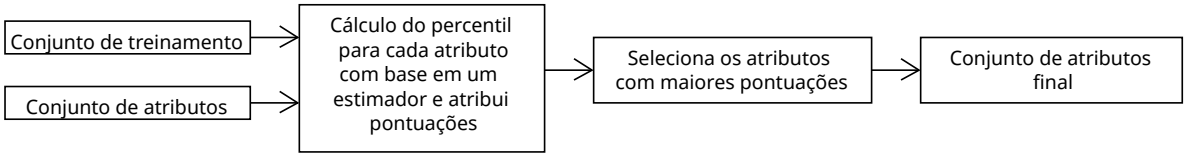
Abordagem	**Vantagem**	**Desvantagem**
Filtro	Classificador independente; Adequado para dados com baixa dimensão; Escalável; Computacionalmente mais rápido que o *Wrapper* e o Embutido; Melhor propriedade de generalização.	Não considera a correlação entre os classificadores e entre os atributos; Falha em reconhecer apropriadamente padrões o processo de aprendizagem.
Wrapper	Considera correlação entre atributos e rótulos de classe; Considera dependências entre os atributos; Maior acurácia do que o Filtro.	Avalia iterativamente os subconjuntos de atributos; Mais complexo computacionalmente; Alguns atributos não são considerados se descartados na etapa inicial; Mais propenso a over-fitting.
Embutida	Considera a dependência entre os atributos; Menor propensão a over-fitting do que o Wrapper; Mais eficiente computacionalmente e maior acurácia do que o *Wrapper* e o Filtro.	Generalidade pobre; Não é adequado para dados de alta dimensão; Computacionalmente mais custoso do que o Filtro.

Fonte: Autora (2022).

2.2.5 Métodos de Seleção de Atributos

SelectPercentile. O SelectPercentile é um método filtro e um dos métodos de SA univariados que utilizam como base testes estatísticos univariados. O método seleciona atributos de acordo com um percentil das pontuações mais altas, calculado para todos os atributos de um conjunto. Percentis são medidas que dividem o conjunto (por ordem crescente dos dados) em 100 partes, cada uma com uma porcentagem de dados aproximadamente igual. O método utiliza uma função que retorna pontuações para o conjunto, podendo ser uma função de regressão ou classificação (SCIKIT-LEARN, 2022h). Na Figura 11 observa-se o diagrama de blocos do funcionamento do método.

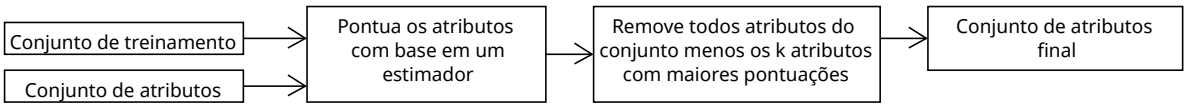
Figura 11 – Diagrama de blocos do método SelectPercentil



Fonte: Autora (2022).

SelectKBest. O SelectKBest é um método filtro e, da mesma forma que o SelectPercentile, é um dos métodos de SA univariados. O método pontua os atributos em relação à variável de destino usando uma função e, em seguida, seleciona atributos de acordo com as K pontuações mais altas com base do resultado estatístico (SCIKIT-LEARN, 2022g), como pode ser visto na Figura 12.

Figura 12 – Diagrama de blocos do método SelectKBest

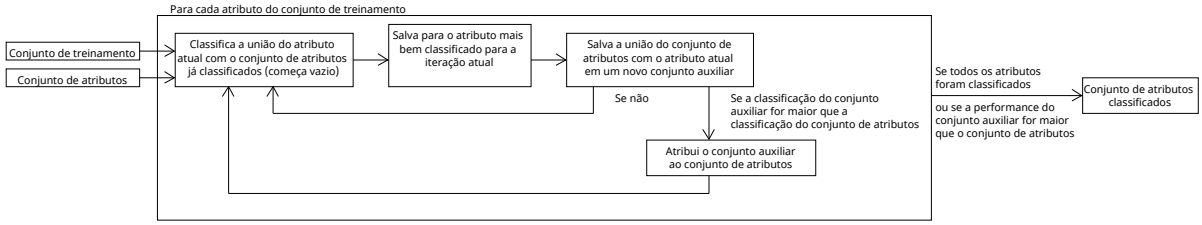


Fonte: Autora (2022).

Sequential Feature Selector (SFS). O SFS é um método *wrapper* que seleciona os atributos avançando ou retrocedendo com base na pontuação de validação cruzada de um estimador. O *Forward* realiza a seleção avançando e encontra iterativamente o melhor novo atributo para adicionar ao conjunto de atributos selecionados. Inicialmente, começa-se com nenhum atributo e, em seguida, encontra-se o único recurso que melhora o desempenho quando um estimador é treinado nesse único atributo. Assim que um atributo é selecionado, repete-se o processo classificando a união do atributo com o conjunto de atributos já classificados. A partir disso, salva-se o atributo e a união do conjunto de atributos em um novo conjunto auxiliar e, se o desempenho do conjunto auxiliar for melhor que a do conjunto de atributos, atribui-se o conjunto auxiliar ao conjunto de atributos. O processo ocorre recursivamente para cada atributo do conjunto original e para quando o número de atributos selecionados é alcançado conforme o especificado (GARCíA-TORRES et al.,

2015). Na Figura 13 pode-se observar o diagrama descritivo do método.

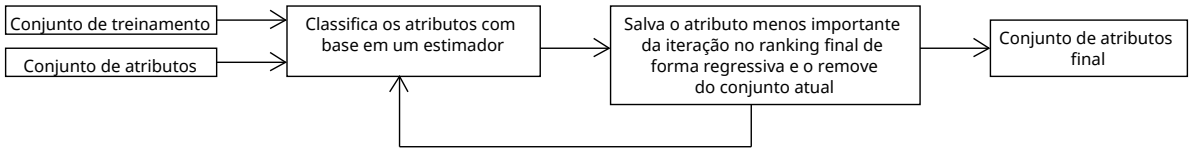
Figura 13 – Diagrama de blocos do método SFS



Fonte: Autora (2022).

Recursive Feature Elimination (RFE). O RFE é um método *wrapper*, que utiliza de um modelo de AM para selecionar os atributos, eliminando o atributo menos importante após o treinamento recursivo, como pode ser observado na Figura 14. De acordo com Scikit-learn (2022i), o RFE é um método para selecionar atributos considerando recursivamente conjuntos cada vez menores. Primeiro, o estimador é treinado no conjunto inicial de atributos e obtêm-se a importância de cada um dos atributos. Em seguida, os atributos menos importantes são removidos do conjunto atual. Esse processo ocorre recursivamente até obter-se a quantidade desejada de atributos. Por padrão, o número de atributos selecionados é a mediana do total (GRANITTO et al., 2006).

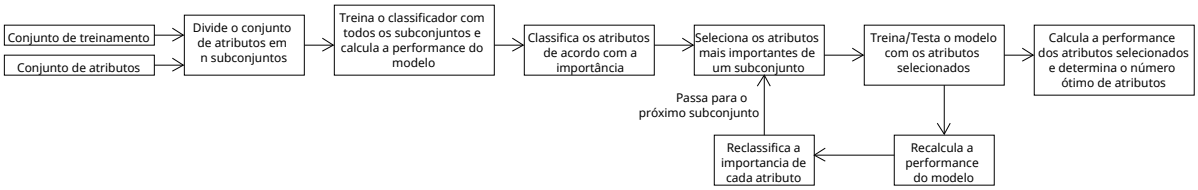
Figura 14 – Diagrama de blocos do método RFE



Fonte: Autora (2022).

Recursive Feature Elimination with Cross-Validation (RFECV). O RFECV é um método *wrapper* que executa o RFE em um loop de validação cruzada para encontrar o número ideal de atributos (SCIKIT-LEARN, 2022i) . O método remove atributos redundantes e fracos, cuja exclusão afeta menos o erro de treinamento e mantém o recurso independente, para melhorar o desempenho de generalização do modelo. Da mesma forma que o RFE, este usa o procedimento iterativo para classificação dos atributos. Inicialmente, divide o conjunto de atributos em n subconjuntos e treina o classificador com todos os subconjuntos, calculando o desempenho do modelo. A partir disso, seleciona recursivamente os atributos mais importantes e treina o modelo com estes, novamente calculando o desempenho. A cada nova iteração, o método armazena um valor sequencial e os atributos com melhor classificação nos quais o reajuste e o desempenho do modelo são acessados. O valor com o melhor desempenho é calculado e os atributos de melhor desempenho se encaixam no modelo final (MISRA; SINGH, 2020), como descrito no diagrama da Figura 15.

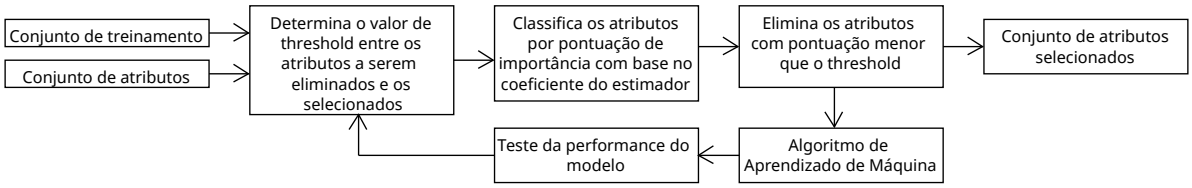
Figura 15 – Diagrama de blocos do método RFECV



Fonte: Autora (2022).

SelectFromModel. O SelectFromModel é um método embutido baseado em uma estimativa do Modelo de AM para selecionar atributos. O algoritmo é um metatransformador que pode ser usado juntamente com qualquer estimador que atribua importância a cada atributo por meio de um atributo específico. Inicialmente, este determina um valor de threshold que será usado para limitar os atributos selecionados e os eliminados. Em seguida, os atributos são classificados por pontuação de importância usando como base um estimador. Os atributos com menor importância que o valor do threshold são removidos. Por padrão, o threshold é a média (HULJANAH et al., 2019; SCIKIT-LEARN, 2022i). Na Figura 16 detalha-se o funcionamento do método.

Figura 16 – Diagrama de blocos do método SelectFromModel



Fonte: Autora (2022).

A seguir, apresenta-se a Tabela 2 com um resumo dos métodos utilizados no trabalho juntamente com suas classificações e principais características.

Tabela 2 – Resumo métodos de Seleção de Atributos

Classificação	**Método**	**Características**
Filtro	SelectPercentile	Seleção com base em percentil das pontuações mais altas.
Filtro	SelectKBest	Seleção com base nas K pontuações mais altas.
Wrapper	SFS	Seleção avançando ou retrocedendo com base na pontuação de validação cruzada do estimador com um atributo recursivamente.
Wrapper	RFE	Seleção com base em *ranking* de atributos eliminando o menos importante recursivamente.
Wrapper	RFECV	Seleção recursiva com validação cruzada com base em *ranking* de atributos mais importantes para o treinamento.
Embutida	SelectFromModel	Seleção eliminando recursivamente os valores, classificados por pontuação de importância, menores que o *threshold* estabelecido.

Fonte: Autora (2022).

Com os métodos levantados, é possível realizar as devidas implementações e ajustes de parâmetros.

3 PROJETO E IMPLEMENTAÇÃO DOS ALGORITMOS

A abordagem para as próximas sessões do capítulo é apresentar o desenvolvimento do projeto e implementações dos algoritmos de SA, desde a metodologia para selecionar os atributos, realizar os testes forçando falha e criação de uma interface para obtenção de conjuntos de dados e, então, as estratégias para tratar dados e implementar os algoritmos.

3.1 SELEÇÃO DE VARIÁVEIS

Para que fosse possível obter um banco de dados, a fim de treinar os algoritmos de SA e, consequentemente, alcançar o objetivo do trabalho, isto é, gerar um subconjunto relevante para identificação das falhas, foi necessário ter as possíveis causas das falhas e suas devidas variáveis relacionadas na ECU. Como mostrado no capítulo anterior, após uma revisão da literatura conseguiu-se encontrar uma árvore de causas de falha para cada uma das duas falhas pesquisadas neste trabalho.

Para obter as variáveis relacionadas na ECU, utilizou-se como base o arquivo A2L da ECU do carro utilizado no Projeto Rota2030, um Renault Sandero ano 2016. O A2L contém inúmeras informações de comunicação, características, sistemas e componentes da unidade que se refere, incluindo as variáveis das causas levantadas. As variáveis do A2L usado podem ser divididas em *measurements*, variáveis de medição, e *characteristics*, variáveis de calibração. Como o propósito do trabalho é avaliar o comportamento dos sistemas e correlacionar com a ocorrência da falha, apenas usou-se as variáveis de medição. Além disso, o A2L citado apresenta um formato de difícil uso e manipulação, por isso, foi preciso transformar o arquivo usando de uma analisadora sintática. Como, o desenvolvimento de uma analisadora sintática é um processo extenso e que demandaria muitas horas, durante reuniões do projeto foi decidido que seria utilizado uma analisadora sintática *open source* disponível e que poderia dividir o máximo de informação possível do A2L (LENZEN, 2022). Entretanto, o A2L apresenta um formato diferente do padrão logo, foi necessário analisar algumas informações manualmente, como o campo *ifDatas* no Listing 3.1.

Listing 3.1 – Exemplo de campos de uma variável de medição

```
"name": "name_speed",
"longIdentifier": "Engine speed",
"datatype": "UWORD",
"conversion": "name_of_the_equation",
"resolution": 1,
"accuracy": 0.0,
"lowerLimit": 0.0,
```

```
"upperLimit": 1700.0,
"annotations": [],
"arraySize": null,
"bitMask": null,
"bitOperation": null,
"byteorder": null,
"discrete": false,
"displayIdentifier": "name_speed",
"ecuAddress": 123456789,
"ecuAddressExtension": null,
"ifDatas": [{
        "content": "/begin IF_DATA xxxxxxx  /end IF_DATA"
    }
],
```


Todas as variáveis possuem um campo de identificação longa ou *"longIdentifier"* como apresentado no Listing 3.1, isto é, uma descrição do que a variável mede e que auxilia no reconhecimento e entendimento de cada uma, pois, as variáveis são denominadas conforme um padrão de códigos estabelecido pela Renault. Como na primeira fase de seleção das variáveis a ideia era ter um primeiro contato com os dados e retirar as variáveis iniciais, o mais coerente era avaliar as identificações longas e o que elas dizem sobre cada variável. Utilizando um *script* simples de pesquisa de palavra contida no campo, conseguiu-se obter algumas variáveis, porém, não abrangeu todas as causas levantadas nas árvores.

Para a falha de ignição foram encontradas 11 categorias relacionadas, totalizando 3200 variáveis.

1. Bomba de combustível
2. Gás de escape
3. Fluxo de ar em massa
4. Injetor
5. Ignição
6. Sensor de oxigênio
7. Sensor de pressão
8. Virabrequim
9. Temperatura refrigerante
10. Vela de ignição
11. Compressor

Embora, muitas variáveis tenham sido encontradas, era inviável realizar uma aquisição com todas no carro, devido à restrição de largura de banda imposta pelo protocolo CCP em cima do barramento CAN. Para tanto, foi realizada uma seleção manual das principais variáveis relevantes. Para fazer essa seleção, utilizou-se como base toda a revisão teórica realizada até o

momento e reuniões com os engenheiros especialistas da Renault. Ao final da seleção manual, a lista de variáveis foi reduzida a 383. Sendo do total de variáveis: 35 do compressor; 64 de temperatura do refrigerante; 12 de vela de ignição; 17 de gases de escape; 36 de sensor de oxigênio; 11 de fluxo de ar em massa; 43 do virabrequim; 73 de ignição; 40 de injetores; 26 da bomba de combustível; 26 de sensores de pressão.

Para detonação, o procedimento foi realizado da mesma forma que para a falha de ignição. Desse modo, encontrou-se 9 categorias relacionadas e um total de 2025 variáveis. Como pode ser visto na lista abaixo.

1. Velocidade do motor
2. Ignição
3. Gás de escape
4. Sensor de detonação
5. Curto circuito
6. Sensor de oxigênio
7. Válvulas
8. Alta temperatura
9. Alta pressão

Ao observar o diagrama da árvore de falhas da detonação é possível perceber que não está escrito a velocidade do motor. Entretanto, ao se pesquisar sobre frequência de rotação não foi encontrado nenhum resultado no identificador longo, e como a velocidade do motor é altamente correlacionada a frequência de rotação do mesmo, optou-se por utilizá-la como parâmetro para substituir e observar a frequência. Da mesma forma como realizado com a falha de ignição, após a seleção manual, conseguiu-se reduzir para 197 variáveis. Sendo do total de variáveis: 55 de velocidade do motor; 49 de ignição; 11 de gás de escape; 5 de sensor de detonação; 33 de curto-circuito; 4 de alta pressão; 2 de alta temperatura; 22 de sensor de oxigênio; 16 de válvulas.

Ademais, vale acrescentar que foi encaminhado pela equipe da Renault uma lista de 70 variáveis básicas do motor, que foram acrescentadas em todos os experimentos de ambas as falhas. A inclusão das variáveis desta lista ocorreu devido a observação por parte da equipe do projeto de que elas ajudaram nos resultados de seus algoritmos de identificação mesmo sem o acréscimo do processo de seleção de atributos. Portanto, decidiu-se que as variáveis seriam importantes para observar correlações e melhorar os resultados para este trabalho, sendo adicionadas aos conjuntos de dados a ser aquisitados em todos as simulações realizadas.

Na lista de variáveis básicas utilizadas encontram-se as variáveis relacionadas :

- Motor: *status*, temperatura do líquido de arrefecimento, velocidade, posição do pedal do acelerador, pedal de freio, pedal da embreagem, contador de ponto morto superior e estado da chave;
- Controle de ar do motor: carga de ar do motor, pressão atmosférica, pressão do coletor de admissão, temperatura do ar de admissão e temperatura do coletor de admissão;
- Controle de combustível e riqueza: valor do contador de fases de computação adaptativa

ao álcool, álcool final adaptável, tempo da primeira injeção em cada cilindro, relação ar-combustível, *status* de regulação de riqueza, fator de adaptação de riqueza, critérios de diagnóstico do catalisador e tensão do sensor de oxigênio do gás de exaustão do catalisador;
- Avanço de ignição: ponto de ajuste da posição do *phaser* da árvore de cames de admissão, posição do *phaser* da árvore de cames de admissão e avanço de ignição;
- Torque do motor: toque sem solicitação da caixa de câmbio, torque efetivo estimado, *setpoint* de torque efetivo solicitado por *drivers* reais e virtuais, perdas de toque, alvo de torque final e ponto de ajuste de torque final indicado;
- Ar-condicionado: pedido, estado e pressão relativa;
- Gerenciamento do alternador e bateria: carga do alternador, voltagem da bateria, potência do alternador, estado de carga da bateria, ponto de ajuste da tensão do alternador e corrente filtrada do rotor do alternador.

3.2 AQUISIÇÃO DE DADOS

Levando em consideração que um dos objetivos deste trabalho é gerar uma grande quantidade de dados para aplicação das técnicas de Seleção de Atributos a partir de simulações da falha de ignição e detonação no carro do projeto, após encontrar e selecionar variáveis relevantes, a próxima etapa foi aquisitar os dados.

A aquisição de dados da ECU, verificando a ocorrência das falhas, ocorreu de duas formas diferentes. Nos primeiros experimentos utilizou-se do software INCA (Integrated Calibration and Application Tool) da empresa ETAS. Posteriormente, os experimentos foram feitos utilizando o sistema desenvolvido no projeto Rota 2030. Em ambos, usou-se de um software em que é possível configurar um experimento com quais variáveis deseja-se medir durante um período de tempo em que o carro está em funcionamento. As configurações de variáveis são feitas individualmente e em eventos periódicos ou não periódicos, escolhidos a critério do usuário.

No caso, foram realizadas várias simulações diferentes, variando o tempo de experimento e as variáveis contidas, para falha de ignição e para detonação. Os experimentos foram feitos com as variáveis relacionadas as falhas citadas na seção anterior e as que descrevem componentes e sistemas básicos do carro, repassadas pela equipe da Renault. Outro fator importante foi adicionar variáveis de contadores de falha para que os algoritmos de identificação e de seleção de atributos pudessem ser treinados, com as variáveis que representam a ocorrência da falha ao longo do tempo. Assim, após o processo de selecionar as variáveis e simula-las forçando o carro a falhas, conseguiu-se obter conjuntos grandes de dados para análises dos algoritmos.

Ao utilizar o sistema desenvolvido no projeto Rota 2030, os dados foram enviados para o servidor próprio do LISHA (Laboratório de Integração de Hardware e Software). Entretanto, para que fosse possível retirar esses dados corretamente do servidor era preciso que as variáveis

do experimento estivessem mapeadas em um arquivo em formato csv, incluindo dados de nome, identificação longa, limites, unidade no A2L, unidade de conversão, unidade característica entre outros. O arquivo, denominado *SmartDataModel*, foi desenvolvido retirando informações do A2L, isto é, o mesmo A2L utilizado para o desenvolvimento da ferramenta de geração de experimento e configuração e desenvolvimento do sistema do projeto. Devido a esse fato, o arquivo *SmarDataModel* só pode ser utilizado em testes para o modelo de ECU especifico do carro em que este trabalho foi desenvolvido, pois diferentes ECUs variam o A2L. Ao final do trabalho, conseguiu-se mapear 377 variáveis distintas no arquivo contido no servidor.

3.2.1 Ferramenta para geração do experimento

O objetivo do projeto Rota 2030 em parceria com a Renault foi o desenvolvimento de um componente eletrônico inteligente de aquisição e análise de dados para controladores automotivos. Dentro desse objetivo está incluso a substituição do INCA no processo de aquisição e análise de dados por um sistema desenvolvido pelos integrantes do projeto. Entretanto, para que ocorresse a substituição foi necessário haver algumas das mesmas funcionalidades do INCA e, para isto foi preciso implementar uma ferramenta que possibilita a configuração de experimentos que seja compatível com o componente sendo desenvolvido no escopo do projeto.

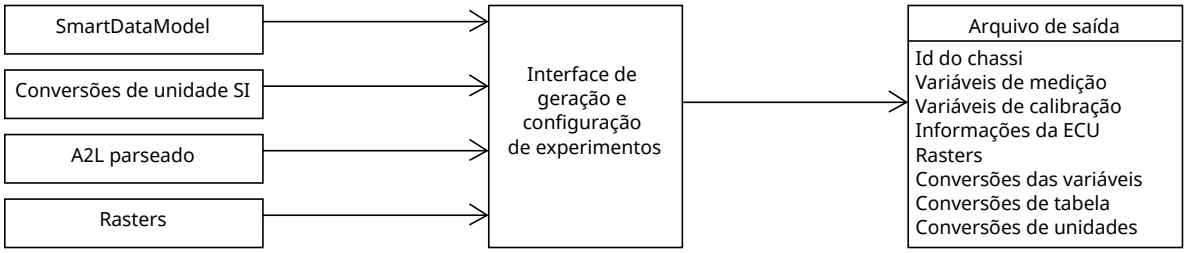
Desse modo, o objetivo da criação do software é configurar e gerar experimentos, baseados em informações da ECU, para que o componente eletrônico inteligente seja capaz de capturar dados de medição e calibração.

O software foi desenvolvido no Linux usando a linguagem Python e a biblioteca PyQt5, para concepção da interface gráfica, entre outras bibliotecas auxiliares que precisam estar instaladas no sistema operacional para que o software opere. Para realização dos testes e desenvolvimentos, o software foi utilizado pelos integrantes do projeto no sistema operacional Linux, porém, como a Renault utiliza o Windows Microsoft, a ferramenta apresenta duas versões para ser operada em ambos os sistemas.

Como entrada do software tem-se as informações sobre *rasters*, isto é, eventos periódicos e não periódicos, que são associados as variáveis e que foram analisadas manualmente, o próprio A2L em outro formato, um banco de dados contendo as informações das unidades das variáveis e suas devidas informações de conversões para sistema internacional, que é necessário para o servidor na nuvem e o *SmartDataModel*, isto é, um banco de dados contendo as informações das variáveis mapeadas que é utilizada no servidor para a aquisição. Todos os arquivos de entradas são em formato JSON, assim como o arquivo de saída.

Do mesmo modo, o arquivo de saída da ferramenta contém os dados de identificação do carro, variáveis de medição, variáveis de calibração, informações sobre a ECU necessárias para configurar o sistema de aquisição, os *rasters*, as identificações de conversão de valores internos da ECU para valores físicos, coeficientes de tabela de conversão e conversão das unidades. O fluxo de informações e arquivos de entrada e saída podem ser vistos na Figura 17.

Figura 17 – Diagrama de fluxo do software desenvolvido.



Fonte: Autora (2022).

Tendo os arquivos de entrada e saída, contendo as informações que o componente eletrônico precisa para realizar aquisição e devidas conversões, foi utilizado da ferramenta para gerar os experimentos contendo as variáveis necessárias para identificar cada falha. Da mesma forma, para as simulações utilizando o INCA os experimentos foram gerados utilizando a ferramenta, pois implementou-se internamente o cálculo dos *rasters* associado a largura de banda no barramento da CAN.

Os *rasters* são eventos periódicos ou não periódicos, sendo periódicos medidos por tempo e não periódicos medidos por posição do ponto morto superior do virabrequim. Cada *raster* é uma lista DAQ (*Data Acquisition*) e cada DAQ possui um *trigger* de quando acontece o evento, quando ocorre o *trigger* envia-se todas as variáveis contidas no DAQ para o dispositivo conectado. Os *rasters* contém um campo de tamanho que corresponde ao número de ODTs (*Object Description Table*s) que podem ser contidas em cada *raster*, assim limitando a largura de banda no barramento. O cálculo implementado soma o número de ODTs de todas as DAQs e multiplica pelo número de bytes de dados da CAN.

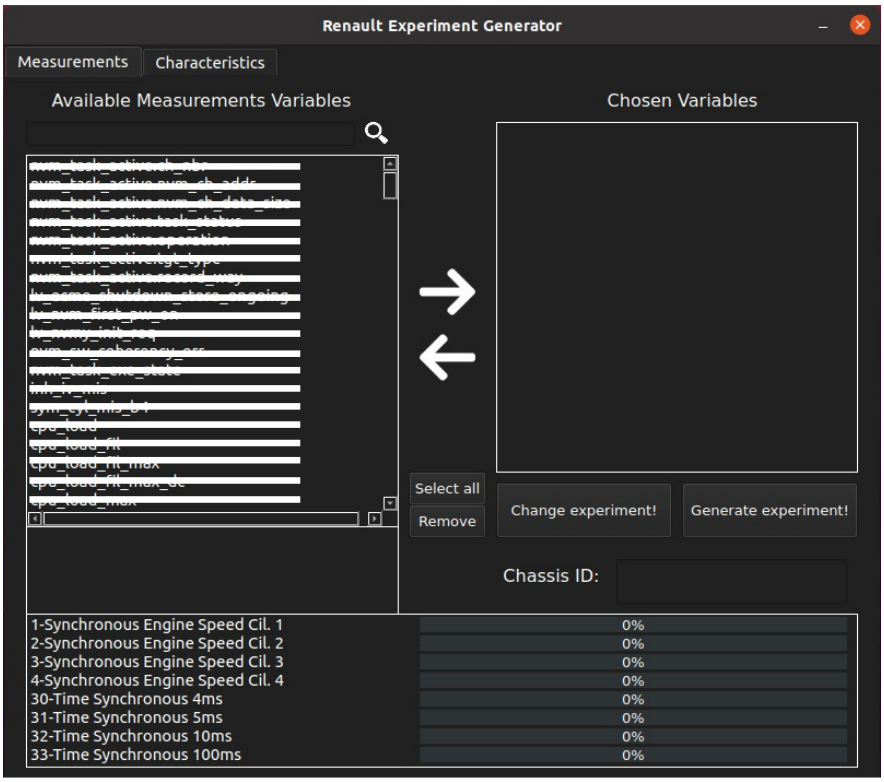
A escolha de qual *raster* usar para cada variável de medição ocorreu com o critério do tamanho de bits da variável, sendo que algumas poderiam ser UBYTE, contendo 8 bits, ou ULONG, contendo 32 bits, por exemplo. Variáveis com maiores tamanhos foram configuradas em *rasters* com maior capacidade, proporcionalmente a escolha ocorreu para as variáveis com menores tamanhos.

Na ferramenta existem duas abas, uma aba destinada a variáveis de medição e outra para variáveis de calibração. As variáveis de calibração não precisam ser configuradas em um *raster*, apenas as variáveis de medição, logo, o layout das abas muda já que essa parte não está inclusa na aba de calibração.

Na Figura 18 apresenta-se a aba do software destinada a configuração de variáveis de medição. Por questões de confidencialidade, os nomes das variáveis foram suprimidos da imagem. Ao lado esquerdo tem-se as variáveis de medição disponíveis a serem escolhidas e ao lado direito tem-se uma coluna com as variáveis que foram escolhidas pelo usuário. Para realizar a seleção ou remoção das variáveis, o software apresenta alguns botões para selecionar como o *Select all* e a seta para direita e botões para remover como o *Remove* e a seta para esquerda. Na seção logo abaixo da lista de variáveis disponíveis, são mostradas as identificações longas

cada vez que uma for selecionada. Subsequentemente, uma lista com os *rasters* disponíveis e barras com o percentual de ocupação de cada um. Além disso, tem-se um campo para pesquisa na lista de variáveis disponíveis que podem ser pesquisadas pressionando no *Enter* ou na lupa, e um campo para ser adicionado a identificação do carro. Por fim, dois botões para mudar o experimento e gerar o experimento.

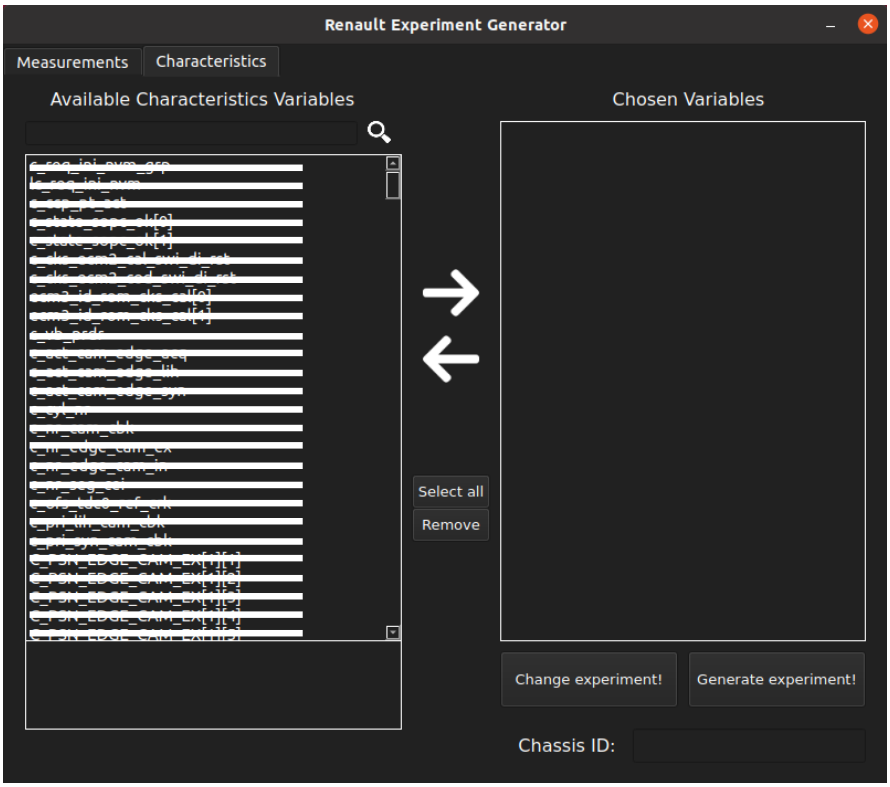
Figura 18 – Aba de seleção de variáveis de medição



Fonte: Autora (2022).

Na Figura 19 apresenta-se a aba destinada a configuração de variáveis de calibração. Assim como a aba de medição, a aba de calibração apresenta as funcionalidades de pesquisa, adicionar a identidade do carro, mostrar a identificação longa, selecionar e remover variáveis, mudar o experimento e gerar o experimento. Vale acrescentar que as variáveis não se misturam na lista de escolhidas, isto é, as variáveis de calibração só serão mostradas na aba referente. Assim é possível ter melhor visualização e consciência dos atributos escolhidos.

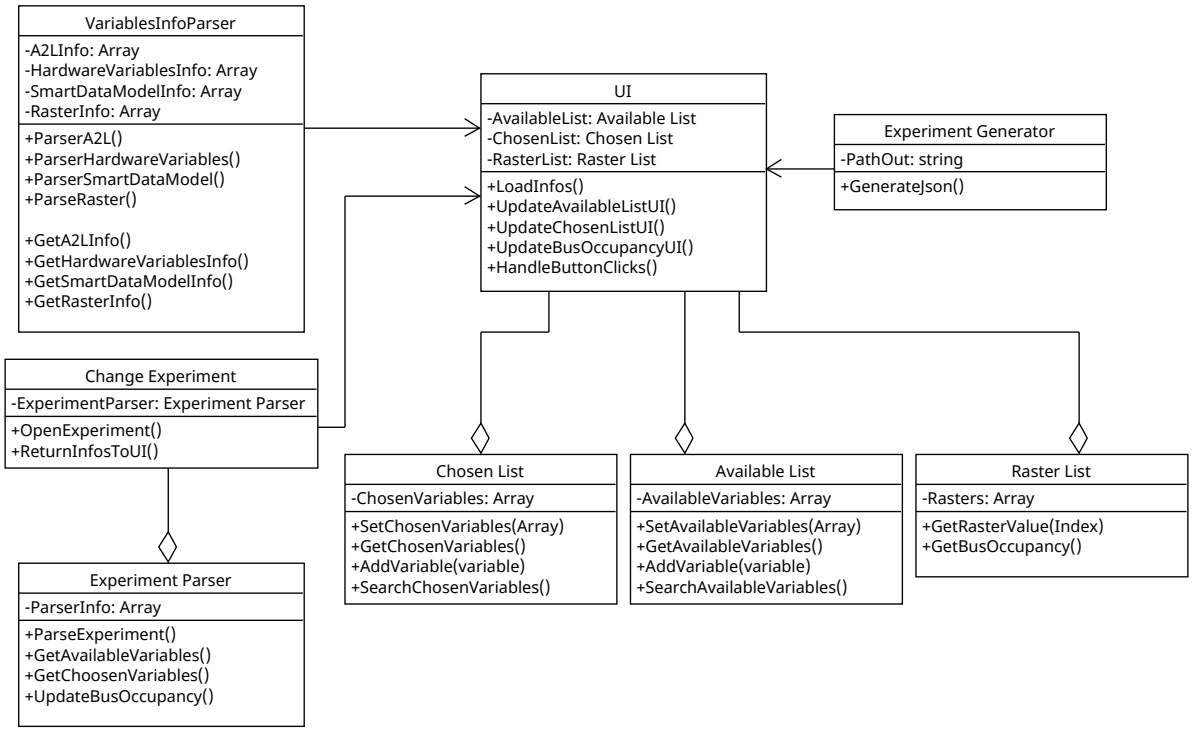
Figura 19 – Aba de seleção de variáveis de calibração



Fonte: Autora (2022).

Na Figura 20, pode-se observar o diagrama de classes do software contendo suas classes e métodos genericamente.

Figura 20 – Diagrama de classe do software de geração de experimentos



Fonte: Autora (2022).

A ferramenta pode gerar experimentos de duas formas diferentes: criando um novo experimento ou alterando um já existente no formato correto. A classe *Change Experiment* é atribuída para abrir o experimento já existente e retornar as informações para a classe da interface. Uma classe auxiliar, a *Experiment Parser*, realiza as funções necessárias para mudar o experimento, como *parsear* as variáveis escolhidas diretamente do arquivo de entrada, atualizar os *rasters* conforme a entrada tinha sido previamente configurada e retirar do A2L as variáveis disponíveis. Por conseguinte, a ferramenta possui uma classe UI central de configuração da própria interface, atualizando constantemente as listas de variáveis escolhidas, disponíveis e responsável todos os botões presentes na ferramenta. A classe *Variables Info Parser* realiza a análise sintática dos arquivos de entrada, citados anteriormente, para que as informações possam ser correlacionadas e utilizadas durante o uso da interface. As classes *Chosen List* e *Available List* são responsáveis por configurar na interface as variáveis, adicionar e remover, e realizar a busca caso solicitado pelo usuário. A classe *Raster List* faz o cálculo do *raster* a cada novo acréscimo de variáveis e, em seguida, atualiza a interface. Ao fim, a classe *Experiment Generator* gera um arquivo JSON contendo todas as variáveis configuradas com devidos *rasters* e as informações do A2L referentes a essas variáveis.

3.3 SIMULAÇÃO DAS FALHAS

Como explicado na seção anterior, os experimentos para simulação das falhas ocorreram de duas formas diferentes que estão detalhadas nesta seção, apresentando a forma de simulação de cada falha separadamente.

Embora muitos artigos citados neste trabalho tenham realizado as simulações das falhas em bancadas de testes, todas as simulações das falhas realizadas no projeto ocorreram no próprio carro, com a equipe do projeto dirigindo o veículo dentro do perímetro do complexo onde a universidade está localizada.

Para que fosse possível obter conjuntos de dados com e sem falha para treinamento dos algoritmos, as simulações das falhas foram realizadas mais de uma vez por dia, sendo de 2 a 3 experimentos de 10 minutos por falha. A ideia inicial era que os experimentos pudessem ser mais longos, para ser possível obter conjuntos maiores ao longo do tempo. Entretanto, o carro de teste apresentava sinais de pane ao ser forçado por mais tempo, principalmente, tendo em vista, que algumas falhas podem prejudicar consideravelmente o motor.

Assim, em um primeiro momento as simulações de falhas foram realizadas utilizando o INCA, em que as escolhas das variáveis e do período de aquisição de cada uma foi escolhida manualmente. O computador rodando o INCA ao ser conectado com a ECU via CAN, aquisita os dados das variáveis configuradas, mostrando em tela os valores atualizados instantaneamente de cada variável. Com a utilização do INCA é possível também configurar algumas opções para forçar falha como é apresentado a seguir na falha de ignição. Ao final da rodagem do experimento, forçando o carro como desejado, o INCA gera um arquivo com os resultados das variáveis ao

longo do experimento em formato DAT. Esse arquivo pode ser salvo e usado diretamente nos algoritmos, sem a necessidade de ser retirado do servidor e do mapeamento das variáveis.

Os experimentos, utilizando o sistema desenvolvido no projeto, usam como entrada o arquivo de saída da ferramenta de geração de experimentos em formato JSON citada anteriormente, sem ter a necessidade de ser setada manualmente de outra forma. Para realizar as simulações das falhas nesse formato, é necessário que o arquivo seja carregado no computador do responsável pela simulação, pois este envia ao hardware o arquivo que será simulado no momento, caso seja necessário alguma alteração ou mudança de arquivo. Após todas as conexões necessárias serem feitas entre o hardware e o carro, ao ser iniciada a simulação e colocar o carro para andar, é necessário que o sistema tenha um resfriamento externo, no caso utilizou-se o próprio ar-condicionado do carro. Desse modo, nessa fase do projeto é preciso de pelo menos duas pessoas para realizar as simulações das falhas com o sistema.

A falha de ignição só pode ser forçada propositalmente e conscientemente utilizando o INCA. O software apresenta uma configuração em que se pode escolher iniciar ou não a injeção da falha durante o experimento. A configuração é setada para 1, para iniciar a falha, e setada para 0, para parar a falha, e também possibilita escolher a cada quantas rotações pode ocorrer uma falha. Essa configuração não permite forçar falhas instantâneas no carro porém, escolher um período do experimento em que pode ocorrer ou não a falha. Caso essa configuração não seja setada a probabilidade de não ocorrer a falha é alta. Pode-se citar, por exemplo, em um experimento com o sistema desenvolvido do projeto em que ocorreu apenas uma falha, prejudicando as análises e utilização do conjunto de dados gerado nesse experimento.

A detonação ocorre com o aumento da carga do motor, para isso a ideia era utilizar de todas as ferramentas que pudessem realizar isso. Inicialmente, a equipe da Renault apresentou algumas formas de conseguir aumentar a carga, como ligar o ar-condicionado, ligar o desembaçador do vidro traseiro, manter em luz alta entre outros enquanto ocorria a rodagem do carro nas simulações. Entretanto, após os testes iniciais, percebeu-se que apenas manter baixa velocidade e alta marcha era o suficiente para ter um número considerável de falhas por experimento. Assim, os testes de detonação ocorriam de forma a parar o carro, colocar em 4ª ou 5ª marcha e acelerar o carro rapidamente ou andar com o carro a 20 km/h.

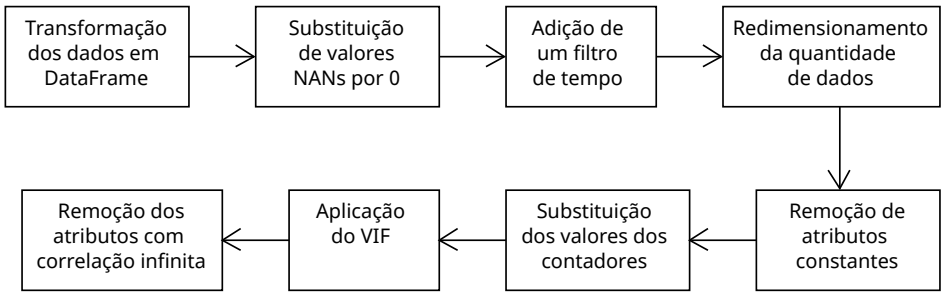
3.4 APLICAÇÃO DOS MÉTODOS

Os algoritmos de SA escolhidos para serem utilizados neste trabalho, citados no capítulo anterior, foram implementados usando da linguagem de programação Python e com a biblioteca Sklearn. Como o trabalho foi desenvolvido em uma parceria e em equipe, escolheu-se utilizar do ambiente Google Colaboratory "Colab"em que seus *notebooks* possuem a característica de seus códigos podem ser compartilhados de maneira colaborativa, para que os outros estudantes pudessem ter acesso as alterações, rodar códigos e modificar suas próprias versões. O Colab é um serviço de nuvem gratuito hospedado pelo próprio Google e é especialmente indicado

para aprendizado de máquina e análise de dados, apresentando uma interface intuitiva, permite a misture de código fonte e texto rico com imagens e resultados, entre outros suportes. A ferramenta trabalha com a linguagem Python, mas com alguns ajustes é possível rodar código em R, Julia, Swift e outras linguagens.

Como esperado, os dados obtidos após realizar as simulações apresentavam algumas inconsistências, como valores NaN, isto é, valores ausentes no conjunto de dados, e valores que permaneceram constantes ao longo de todo o experimento, e algumas informações não relevantes para as análises. Portanto, foi necessário realizar um tratamento nos dados como pode ser visto na Figura 21.

Figura 21 – Diagrama do processo de tratamento do dados



Fonte: Autora (2022).

Primeiramente, os arquivos de saída de experimento do INCA são em formato DAT e, para conseguir usar as funções do Python e do Sklearn, utilizou-se um módulo, desenvolvido no projeto, que transformasse os dados em *DataFrame*. Da mesma forma, os dados coletados do sistema desenvolvido ao serem retirados do servidor, apresentavam formato Parquet, sendo necessário transforma-los em formato *DataFrame* para serem aplicados nos algoritmos.

Após transformar os dados em um formato viável, a próxima etapa foi substituir NaNs por zeros, utilizando a função fillna, e adicionar um filtro de tempo, que retirou informações de data e hora do index para gerar uma coluna com tempo em segundos de cada ponto do conjunto de dados. Para fazer essa etapa de tratamento, usou-se uma *string* contendo todos os nomes de todas as variáveis do arquivo A2L, que é utilizada para criação de um dicionário contendo o nome e a identificação longa de todas as variáveis para ser utilizado na transformação do tipo de arquivo.

Como as simulações ocorreram em um período considerável de tempo e a aquisição acontecia em questão de milissegundos, a quantidade de dados foi expressiva. Para melhor processamento, escolheu-se diminuir o conjunto de dados e realizar um *resample* de 10 milissegundos nos dados. Esse valor foi obtido, após alguns testes dos outros estudantes, em que os resultados conseguiram diminuir o conjunto de dados consideravelmente sem perder informações relevantes dos dados.

Inicialmente, apenas esse tratamento foi realizado nos dados porém, percebeu-se após resultados iniciais dos algoritmos citados a seguir que algumas variáveis estavam interferindo consideravelmente nos resultados, isto pois se uma variável não era relevante mas com alta

correlação com outra não relevante o algoritmo estava detectando estas como importante, embora não representassem atributos significativos para a identificação das falhas. Assim, foi implementado outro filtro no conjunto de dados que removia as variáveis constantes do conjunto.

Para realizar o treinamento dos algoritmos, o conjunto de referência para identificação selecionado foi o do contador de falha porém, na aquisição da ECU a conta da falha ocorre de em soma crescente e isso poderia interferir no treinamento do modelo. Assim, substituiu-se todo valor diferente de zero por um, para transformar o contador em valores binários, isto é, configurar a lógica de ocorreu ou não falha.

Por fim, foi aplicado o VIF (*Variance Inflation Factor*) em todo o conjunto de dados restante e eliminou-se as variáveis com resultado infinito, pois isso significava que a multicolinearidade entre as variáveis independentes era tão expressiva que elas estavam muito correlacionadas e poderiam ser eliminadas, e, utilizada de outras para melhorar o resultado do modelo.

O VIF é usado para detectar a presença de multicolinearidade e determina a força da correlação entre variáveis independentes. O VIF é obtido regredindo cada variável independente e como resultado gera uma lista de variáveis em que as com valores mais altos de VIF são consideradas colineares, isto significa que uma variável independente pode ser prevista a partir de outra variável independente em um modelo de regressão. O valor R^2 é determinado para descobrir quão bem uma variável independente é descrita por outras. A fórmula do VIF pode ser vista na equação 1.

$$VIF = \frac{1}{1 - R^2} \tag{1}$$

A proposta de realizar a seleção de atributos e implementação dos algoritmos é que os subconjuntos gerados pudessem ser validados pelos algoritmos de identificação de falha de outros alunos do projeto, em que cada conjunto seria avaliado no algoritmo a partir da pontuação f1 e, com base nisso observar quais categorias de atributos melhor detectaram a falha e quais os algoritmos de SA conseguiram encontrar esse subconjunto.

A pontuação f1 é a média ponderada da precisão e do *recall*, levando em consideração tanto os faltos positivos quanto os falsos negativos. A precisão quantifica o número de previsões de classe positiva que realmente pertence a classe positiva. O *recall* é a quantifica o número de previsões de classe positivas feitas de todos os exemplos positivos no conjunto de dados. Na Equação 2 pode-se observar o cálculo da pontuação f1.

$$Pontuação f1 = 2 * \frac{Recall * Precisão}{Recall + Precisão} \tag{2}$$

Assim, a pontuação f1 se torna interessante pois resume o desempenho preditivo de um modelo combinando duas métricas concorrentes.

3.4.1 SelectPercentile

Para implementação de um algoritmo usando o método de SelectPercentile, primeiramente faz-se a leitura do *dataset* em formato csv para DataFrame e, em seguida, utiliza-se a função de dividir matrizes em subconjuntos aleatórios de treinamento e teste do Sklearn com o conjunto de dados do contador de falha como alvo e com tamanho de teste de 30%.

Listing 3.2 – Divisão de subconjuntos de treinamento e teste

```
X_train,X_test,y_train,y_test=train_test_split(df.drop(labels=['
    contador'], axis=1),df['contador'], test_size=0.3, random_state
    =0)
```

Para seleção, usou-se a função SelectPercentile da biblioteca feature_selection do Sklearn com o regressor *Mutual Information Regression* e alterando o um percentil do total a ser mantido. Como a proposta da SA é redução da dimensionalidade, observou-se diferentes quantidades de dados selecionados pelo algoritmo, sendo escolhido analisar com 20%, 15% e 10%. Como entrada da função temos os subconjuntos de treinamento.

Listing 3.3 – Função SelectPercentile

```
selected_percent = SelectPercentile(mutual_info_regression,
    percentile=15)
```

O mutual_info_regression estima informações mútuas para uma variável de alvo contínua. A informação mútua entre duas variáveis aleatórias é um valor não negativo, que mede a dependência entre as variáveis. A função baseia-se em métodos de estimativa de entropia de distâncias de k-vizinhos mais próximos (SCIKIT-LEARN, 2022e).

Embora, o contador de falha não ocorria a todo momento, escolheu-se utilizar a função do mutual_info_regression pois, os dados coletados eram contínuos no tempo, inclusive, o contador. Assim, acredita-se que a função teria uma melhor abordagem nos dados trabalhados.

3.4.2 SelectKBest

Para implementação do algoritmo usando o método de SelectKBest, realizou-se o mesmo procedimento de leitura e divisão de dados como mostrando no Listing 3.2. Para selecionar os melhores atributos, usou-se a função SelectKBest da biblioteca feature_selection do Sklearn com o regressor *R Regression* e com parâmetro de selecionar os melhores 15, 10 e 5 atributos. Esses valores de quantidade de atributos a ser analisado foram escolhidos com base no algoritmo dos outros alunos, sendo que um encontrou 9 atributos e o outro 8 atributos que melhor identificação a falha e melhoravam o resultado do algoritmo. Portanto, acredita-se que as quantidades abrangiam uma variação coerente.

Listing 3.4 – Função SelectKBest

```
selected_best = SelectKBest(r_regression, k=10)
```

O r_regression calcula o r de Pearson, coeficiente de correlação de Pearson, para cada atributo e o alvo (SCIKIT-LEARN, 2022f). A correlação entre cada atributo e o alvo é calculada pela Equação 3.

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{3}$$

O coeficiente de correlação de Pearson mede a relação estatística entre duas variáveis contínuas, sendo assim, compatível com os dados sendo analisados. Além disso, uma vantagem da utilização do coeficiente é com relação ao tamanho da amostra, quanto maior a amostra mais provável é a precisão da estimativa. Assim, tornando a função do coeficiente adequada para o SelectKBest.

3.4.3 Sequential Feature Selector

O procedimento inicial de leitura e divisão de dados foi realizado da mesma forma que os algoritmos anteriores e mostrado no Listing 3.2.

Em seguida, para implementar a função SequentialFeatureSelector da biblioteca feature_selection do Sklearn fez-se a utilização de dois estimadores para comparação de resultado: o LogisticRegression e o LassoCV (SCIKIT-LEARN, 2022d; SCIKIT-LEARN, 2022c). O LogisticRegression foi implementado com um número máximo de iterações de 1000, fazendo a seleção de 15, 10 e 5 atributos, pelo motivo citado anteriormente, e a seleção para frente, pontuando os atributos pela função métrica de perda de regressão de erro quadrático médio e com estratégia de divisão de validação cruzada configurada para 10 pastas.

Listing 3.5 – Função SequentialFeatureSelector com LogisticRegression

```
selected_sfs = SequentialFeatureSelector(LogisticRegression(
   max_iter=1000), n_features_to_select=5, scoring='
   neg_mean_squared_error', direction='forward', cv = 10)
```

O LassoCV foi configurado com número máximo de iterações em 5000, fazendo a seleção dos atributos e pontuando com as mesmas configurações do LogisticRegression.

Listing 3.6 – Função SequentialFeatureSelector com LassoCV

```
sfs = SequentialFeatureSelector(LassoCV(max_iter=5000),
   n_features_to_select=5, scoring='neg_mean_squared_error',
   direction='forward', cv = 10)
```

O número máximo de iterações foi encontrado ao realizar testes com os dados e percebeu-se que a quantidade de iterações padrão da função não era o suficiente para realizar a seleção. Assim, os outros parâmetros que foram configurados conforme encontrado o melhor resultado de atributos selecionados.

O LogisticRegression implementa a regressão logística regularizada usando a biblioteca liblinear e solucionadores newton-cg, sag, saga, lbfgs. A regularização L2 é aplicada por padrão,

que resulta em valores de peso menores, o que estabiliza os pesos quando há alta correlação entre os atributos. O LassoCV implementa o modelo linear de laço com ajuste iterativo ao longo de um caminho de regularização, selecionado o melhor modelo por validação cruzada.

Os estimadores citados são utilizados para realizar previsão da possibilidade de um evento, utilizando diferentes técnicas de regularização. Assim, como o SequentialFeatureSelector é um método *wrapper*, escolheu-se aplicar ambas para que fosse possível comparar os resultados.

3.4.4 Recursive Feature Elimination

Iniciou-se o algoritmo da mesma forma como citado no Listing 3.2 para leitura e divisão do conjunto de dados. Em seguida, implementou-se a função RFE da biblioteca feature_selection do SKlearn com um estimador. Para efeito de encontrar o melhor resultado possível para este algoritmo, foi implementado dois estimadores juntamente ao modelo: o RandomForestRegressor e o LogisticRegression (SCIKIT-LEARN, 2022d; SCIKIT-LEARN, 2022a).

Para implementar a função com o LogisticRegression configurou-se para um número máximo de 1000 iterações, com 15, 10 e 5 atributos a serem selecionados, passo de 3 e verbosidade de 5.

Listing 3.7 – Função RFE com LogisticRegression

```
selected_rfe = RFE(estimator=LogisticRegression(max_iter=1000),
    n_features_to_select=10, step=3, verbose=5)
```

Na implementação do RandomForestRegressor foram utilizados os mesmos parâmetros de número máximo de atributos, passo e verbosidade. O número de estimadores manteve-se o configurado como padrão e alterou-se o critério de medida da qualidade da divisão para a *poisson*, após observar que os outros critérios não apresentavam bons resultados para os dados trabalhados. Um exemplo da configuração desse parâmetro pode ser visto no Listing 3.8.

Listing 3.8 – Função RFE com RandomForestRegressor

```
selected_rfe =  RFE(estimator=RandomForestRegressor(n_estimators
    =100, criterion='poisson'), n_features_to_select=10, step=3,
    verbose=5)
```

O RandomForest é um estimador de floresta aleatória, isto é, um meta estimador que ajusta vários classificadores de árvore de decisão em várias subamostras do conjunto de dados e usa a média para melhor a precisão preditiva e controlar o ajuste excessivo, podendo ser de regressão ou classificação.

O estimador de floresta aleatória apresenta melhor desempenho ao ser utilizado com um grande conjunto de dados, tornando-o efetivo ao trabalhar com os dados das simulações de falha. Além de que, o estimador não apresenta problemas com *overfitting*, o que é o correto para problemas de identificação e predição.

3.4.5 Recursive Feature Elimination with Cross-Validation

A implementação do RFECV seguiu a mesma linha de raciocínio do RFE, primeiramente realizou-se a leitura e divisão base para todos os algoritmos, em seguida, foi implementado a função RFECV, da biblioteca feature_selection do Sklearn, com os dois estimadores. No modelo configurou-se o mínimo de atributos selecionados em 15, 10 e 5 e a estratégia de divisão de validação cruzada configurada para 10 pastas, como pode ser visto no Listing 3.9.

Listing 3.9 – Função RFECV com RandomForestRegressor

```
selected_rfecv= RFECV(RandomForestRegressor(n_estimators=100,
    criterion='possion'), min_features_to_select=5, cv =10)
```

A escolha de qual estimador utilizar com o método foi baseado nos mesmos critérios citados para o método RFE, tendo em vista que, a proposta de se implementar os dois estimadores era a comparação dos resultados de cada um com os dados das falhas a fim de encontrar o melhor entre eles.

3.4.6 SelectFromModel

Para implementar o algoritmo com o método SelectFromModel manteve-se o procedimento inicial assim como anteriormente. Como estimador para o modelo usou-se o RandomForestRegressor (SCIKIT-LEARN, 2022a) com o critério de erro quadrado, sendo o padrão da função, e o modelo configurado para número máximo de 15, 10 e 5 atributos, em duas configurações diferentes de *threshold* de 0.1 e nenhum. O número de árvores da floresta manteve-se o padrão após observar que com os dados trabalhados não haviam mudanças nos atributos selecionados.

Listing 3.10 – Função SelectFromModel

```
selected_classifier = SelectFromModel(RandomForestRegressor(
    criterion='squared_error'),max_features=5, threshold=0.1)
```

O método SelectFromModel é um método embutido que necessita de um estimador para atribuir importância a cada atributo e como o RandomForest realiza uma divisão de árvore com atributos diferentes, a atribuição de importância ocorre sem apresentar problemas de *overfitting*. Essa característica foi fundamental para a escolha do estimador para o método. Além disso, métodos baseados em regressão analisam dados contínuos o que torna o estimador compatível com os dados trabalhados.

3.5 CONSIDERAÇÕES

Neste capítulo foram apresentados os passos aplicados para aquisição dos dados e simulações das falhas, iniciando com uma descrição detalhada de quais atributos do carro foram

base para gerar os conjuntos de dados e, em seguida, como foi realizado a aquisição dos dados, o desenvolvimento de uma ferramenta para dar suporte e realizar a aquisição e o modo como as simulações de cada falha tomaram forma. Logo após, os passos para o tratamento dos dados de tal forma que estivessem em formato satisfatório para ser utilizado nos algoritmos, juntamente com os parâmetros configurados para cada método de SA.

Como o objetivo do trabalho foi realizar a SA para as duas falhas, os passos do desenvolvimento que englobam a escolha e a aquisição dos dados, assim como, a simulação das falhas foi fundamental para ter uma grande quantidade de dados relevantes para análise com os algoritmos. Do mesmo modo, a decisão das configurações dos parâmetros dos métodos foi apoiado em testes constantes junto com os algoritmos de identificação de falha dos outros estudantes e, consequentemente, gerou resultados que serão apresentados posteriormente.

Portanto, percebe-se que parte dos objetivos propostos no início do trabalho foram alcançados, finalizando com a comparação do desempenho dos algoritmos de AM com cada conjunto no próximo capítulo.

4 RESULTADOS

Neste capítulo estão apresentados os resultados obtidos após as simulações das falhas, aquisição dos dados e implementação dos algoritmos com os conjuntos de dados gerados. Além disso, estão realizadas comparações entre os métodos e conjuntos finais que obtiveram os melhores resultados, levando como métrica de análise a pontuação f1.

4.0.1 Falha de ignição

Para realizar a análise e comparação entre os conjuntos gerados pelos algoritmos de SA para falha de ignição, foi desenvolvido um algoritmo MLP (*Multilayer perceptron*) com quatro camadas de neurônios. Duas dessas camadas são a camada de entrada, uma camada de saída e duas camadas escondidas. O número de neurônios da camada de entrada é igual ao número de variáveis e o número de neurônios da camada de saída é igual ao número de saída esperadas, sendo nesse caso apenas 1 saída. Para as camadas escondidas, o número de neurônios para a primeira camada escondida foi definido como sendo igual ao número de variáveis de entrada e, dessa forma, o número de neurônios para a segunda camada escondida foi definido como a metade do número de variáveis de entrada.

Os testes foram executados para 10 épocas com um número variável de passos de acordo com o número de amostras de cada conjunto de dados. O tempo de treinamento para cada subgrupo de variáveis foi de aproximadamente 10 minutos utilizando um processador de 64 núcleos.A rede foi criada utilizando as bibliotecas SKlearn, Tensorflow e Keras - ambas *open source* - e o código foi implementado em Python 3.10.

As simulações de falha de ignição foram realizadas com três conjuntos de dados e quantidades distintas. Primeiramente, foram simuladas todas as variáveis selecionadas manualmente, como citado na seção 3.1, e as variáveis básicas do motor, totalizando 290 variáveis, sendo que foi preciso dividir o experimento em 2 devido a quantidade de atributos total e a capacidade permitida pelo barramento da CAN. Assim, foi simulado os 2 experimentos por 4 vezes. A partir dessas simulações iniciais, com intuito de ter apenas um experimento, utilizou-se dos resultados obtidos pelos algoritmos de SA para gerar um experimento diferente. O segundo tipo de experimento apresentava 76 variáveis resultantes do anterior e 40 variáveis básicas do motor, totalizando 116 atributos, que foram simulados por 4 vezes.Os resultados apresentados a seguir usam como base o segundo tipo de experimento. Por fim, para realizar uma comparação simulou-se uma única vez um último tipo de experimento, sendo criado da mesma forma que o anterior, com 21 variáveis dos resultados dos algoritmos de SA com o conjunto anterior e 16 variáveis básicas do motor, totalizando 37.

A MLP construída foi treinada para todos os subconjuntos de atributos de falha de ignição gerados nas simulações e os resultados obtidos pode ser vistos nas Tabelas dessa seção.

Na Tabela 3 são apresentados um conjunto geral somando os atributos selecionados por todos os algoritmos de SA para a Falha de Ignição. Os resultados em seguidas estão apresentados baseados nessa lista.

Tabela 3 – Conjunto geral de atributos selecionados pelos algoritmos de SA para Falha de Ignição

Nº de atributos	**Categoria do atributo**
18	Ignição
17	Velocidade do motor
13	Detonação
8	Injeção
7	Sensor de pressão
6	Posição virabrequim
5	Sensor de oxigênio
5	Torque
4	Alternador
4	Gás de escape
4	Temperatura do líquido de arrefecimento
2	Alta temperatura
2	Relação ar-combustível
2	Compressor
2	Falha de ignição
1	Distância total do veículo
1	Tensão da bateria

Fonte: Autora (2022).

Primeiramente, a Tabela 4 apresenta os resultados alcançados pelo método SelectPercentile com diferentes porcentagens de atributos escolhidos.

Tabela 4 – Resultado SelectPercentile para Falha de Ignição

Porcentagem de atributos	**Nº de atributos**	**Pontuação f1**
20%	13	98.18%
15%	10	98.44%
10%	7	97.41%

Fonte: Autora (2022).

O método SelectPercentile escolheu, principalmente, atributos relacionados a ignição, temperatura do líquido de arrefecimento, sensor de pressão, distância total do veículo e tensão da bateria, variando a quantidade de cada categoria selecionada entre os subconjuntos. Observa-se que maior diferença do resultado entre a variação de atributos ocorreu entre 10% e 15%, sendo que no conjunto de 15% o método selecionou a tensão da bateria e o sensor de pressão, que também estão contidos no conjunto de 20%.

A Tabela 5 exibe os resultados obtidos pelo método SelectKBest com diferentes números de atributos selecionados.

Tabela 5 – Resultado SelectKBest para Falha de Ignição

Nº de atributos	Pontuação f1
15	98.96%
10	97.95%
5	97.17%

Fonte: Autora (2022).

Os principais atributos selecionados pelo método SelectKBest estão relacionados a ignição, porém vale ressaltar, uma vez que esse conjunto apresentou o melhor resultado mesmo com um número maior de atributos, que apenas o conjunto de 15 atributos apresentava uma variável de detonação e uma de fluxo de massa de ar.

Na Tabela 6 mostra-se os resultados alcançados com o método SelectFromModel alterando o *threshold*.

Tabela 6 – Resultado SelectFromModel para Falha de Ignição

Threshold	Nº de atributos	Pontuação f1
0.1	2	0%
None	1	0%

Fonte: Autora (2022).

Os resultados alcançados para o método ocorreram devido ao número de atributos selecionados por este, mesmo com uma grande quantidade de dados presentes no conjunto aplicado, não sendo possível treinar o modelo de AM com apenas 1 e 2 atributos.

A Tabela 7 apresenta os resultados encontrados usando o método RFE com estimador RandomForestRegressor.

Tabela 7 – Resultado RFE RandomForestRegressor para Falha de Ignição

Nº de atributos	Pontuação f1
15	98.11%
10	98.44%
5	96.11%

Fonte: Autora (2022).

Todos os conjuntos selecionados pelo método RFE com estimador RandomForestRegressor, os atributos relacionados com ignição, posição do virabrequim e distância total do veículo estavam presentes. Contudo, para o conjunto com 10 atributos, variáveis de temperatura do gás de escape e injeção foram escolhidas, da mesma forma, o conjunto de 15 atributos incluía os mesmos atributos além de variável de torque e de fluxo de massa de ar.

Na Tabela 8 apresenta-se os resultados encontrados usando o método RFE com estimador LogisticRegression.

Tabela 8 – Resultado RFE LogisticRegression para Falha de Ignição

Nº de atributos	Pontuação f1
15	98.46%
10	98.57%
5	98.18%

Fonte: Autora (2022).

O resultado do método RFE com estimador LogisticRegression para diferentes números de atributos foi semelhante, não obtendo significativa discrepância. Assim, vale destacar que os principais atributos selecionados estão relacionados à ignição, alternador, detonação, sensor de oxigênio, velocidade do motor e temperatura do líquido de arrefecimento.

A Tabela 9 apresenta os resultados obtidos pelo método SFS com estimador LogisticRegression.

Tabela 9 – Resultado SFS LogisticRegression para Falha de Ignição

Nº de atributos	Pontuação f1
15	98.16%
10	97.77%
5	98.44%

Fonte: Autora (2022).

Ao comparar os atributos escolhidos pelo método SFS com estimador LogisticRegression, constatou-se que o conjunto com 5 atributos não tem nenhuma variável de detonação, apenas sensor de pressão, temperatura do líquido de arrefecimento e ignição. Em contrapartida, de 10 e 15 atributos apresentam as mesmas variáveis que 5, com acréscimo de atributos de detonação.

A Tabela 10 apresenta os resultados obtidos pelo método SFS com estimador LassoCV.

Tabela 10 – Resultado SFS LassoCV para Falha de Ignição

Nº de atributos	Pontuação f1
15	98.14%
10	98.08%
5	98.44%

Fonte: Autora (2022).

Os principais atributos selecionados pelo método SFS com estimador LassoCV estão relacionados com sensor de pressão e temperatura do líquido de arrefecimento. Do mesmo que com o estimador LogisticRegression, apenas o conjunto com 5 atributos não apresentava variáveis de detonação, sendo o conjunto com melhor resultado entre os conjuntos do método.

Na Tabela 11 mostra-se os resultados alcançados usando o método RFECV com estimador LogisticRegression.

Tabela 11 – Resultado RFECV LogisticRegression para Falha de Ignição

Nº de atributos	Pontuação f1
15	98.25%
10	98.25%
5	98.71%

Fonte: Autora (2022).

O método RFECV com estimador LogisticRegression selecionou atributos de ignição, sensor de pressão, temperatura do líquido de arrefecimento, alternador e posição do virabrequim. Embora, o conjunto de 5 atributos apresente uma diferença do conjunto de 15 e 10 atributos, não observou-se uma desigualdade entre as categorias selecionadas, apenas com a relação a quantidade de cada categoria no conjunto.

Na Tabela 12 mostra-se os resultados alcançados usando o método RFECV com estimador LogisticRegression.

Tabela 12 – Resultado RFECV RandomForestRegressor para Falha de Ignição

Nº de atributos	Pontuação f1
15	98.42%
10	98.25%
5	98.66%

Fonte: Autora (2022).

Os conjuntos selecionados pelo método RFECV com estimador RandomForestRegressor incluíam os atributos relacionados com fluxo de massa de ar, alternador, torque, ignição, gás de escape, detonação e sensor de oxigênio. Pode-se concluir que o método obteve resultado semelhante com diferentes quantidades de atributos, não sendo observado diferença relevante entre as categorias de atributos.

Por fim, constatou-se que os resultados obtidos com os algoritmos de identificação de falha não apresentavam discrepâncias significativas para cada subconjunto trabalho. Acredita-se que esta falta de diferença ocorreu devido à qualidade dos conjuntos de dados obtidos a partir das simulações com a quantidade de falhas somado aos parâmetros configurados no algoritmo de Aprendizado de Máquina. Entretanto, três diferentes métodos alcançaram os melhores resultados, conforme a Tabela 13.

Tabela 13 – Melhores resultados para Falha de Ignição

Método	Nº de atributos	Precisão	Recall	Pontuação f1
SelectKBest	15	98%	99.59%	98.96%
RFECV RandomForestRegressor	5	97.69%	99.69%	98.71%
RFECV LogisticRegression	5	98%	99.44%	98.66%

Fonte: Autora (2022).

Como citado anteriormente, foi realizado um terceiro experimento com redução

significativa no número de atributos para avaliar se os resultados dos algoritmos eram alterados com a redução do conjunto de teste e treinamento. Para fins de comparação, escolheu-se os três melhores resultados para falha de ignição, como pode ser visto na Tabela 14.

Tabela 14 – Melhores resultados em conjunto reduzido para Falha de Ignição

Método	**Nº de atributos**	**Precisão**	**Recall**	**Pontuação f1**
SelectKBest	15	99.43%	99.78%	99.60%
RFECV RandomForestRegressor	5	99.5%	98.5%	98.99%
RFECV LogisticRegression	5	99.10%	99.63%	99.37%

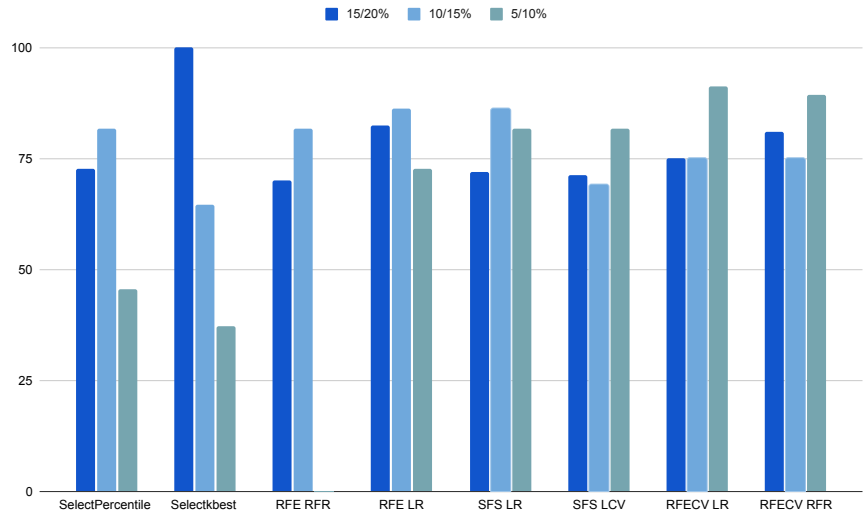
Fonte: Autora (2022).

Conclui-se que com a redução do número de atributos no experimento houve uma melhoria os resultados obtidos pelos algoritmos, no caso o método RFECV com estimador LogisticRegression e o SelectKBest obtiveram os maiores percentuais de melhora.

Para todos os resultados do treinamento, nota-se uma precisão, *recall* e pontuação f1 muito semelhantes e elevados, em parte devido ao fato de que os conjuntos de dados obtidos após as simulações apresentavam aproximadamente 50% do tempo com falha. Apesar disso, observa-se um equilíbrio entre os valores das métricas de análise. Devido a essa proximidade entre os resultados das métricas de avaliação da MLP, para uma melhor visualização, implementou-se um algoritmo que classifica os resultados alcançados e apresentados anteriormente, realizando uma pontuação para os subconjuntos de variáveis em uma escala de 0 a 100, sendo 100 o subconjunto com melhor resultado.

O algoritmo foi implementado de forma em que subtraiu-se de todos os resultados o menor valor entre eles, em seguida, foi encontrada uma relação de 100 dividido pelo maior valor dos resultados após a subtração e multiplicou-se todos os valores após a subtração pela relação encontrada anteriormente. Dessa forma, todos os resultados da pontuação f1 para cada subconjunto foi classificado de 0 a 100. O resultado da classificação dos resultados dos métodos pelo número de atributos em relação a todos os conjuntos pode ser visto na Figura 22.

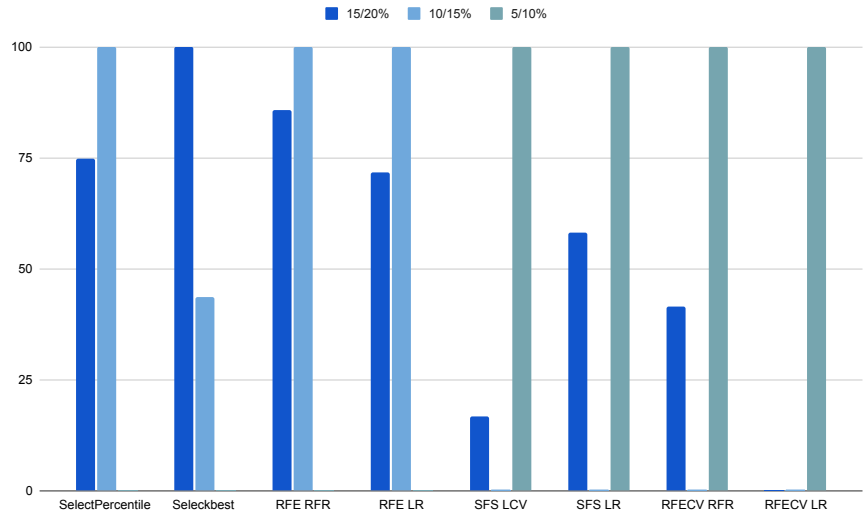
Figura 22 – Classificação dos resultados dos métodos por nº de atributos selecionados



Fonte: Autora (2022).

A partir do gráfico acima, pode-se observar que de forma geral, conjuntos com 10 e 5 atributos alcançaram uma quantidade de classificação próxima em relação a todos os conjuntos. Na Figura 23, trouxe um gráfico da relação dos atributos entre os conjuntos do próprio método.

Figura 23 – Classificação dos nº de atributos selecionados por método



Fonte: Autora (2022).

Os resultados nulos significam que os conjuntos apresentam a menor classificação entre os conjuntos. Nota-se que os conjuntos de 5 atributos obtiveram os melhores resultados.

Após análise e comparação dos resultados obtidos pelos métodos de SA implementados, nota-se que cada os métodos apresentaram diferentes conjuntos, porém os principais atributos levantados incluem ignição, sensor de pressão, temperatura do líquido de arrefecimento e distância total do veículo. Segundo a Figura 3, é interessante pontuar que os atributos estão estritamente correlacionados a falha no sistema de ignição.

4.0.2 Detonação

As simulações da falha de detonação foram realizadas com três conjuntos de dados diferentes. Os primeiros experimentos continham as 197 variáveis citadas na seção 3.1 e mais algumas básicas do motor, totalizando 230 variáveis. Com essa quantidade foi necessário dividir o experimento em dois, simulou-se os dois experimentos 4 vezes cada com 10 minutos. Usou-se da grande quantidade de dados gerados pelas 8 simulações para os algoritmos de SA e os resultados foram usados para gerar uma lista geral para reduzir os experimentos em apenas 1. Portanto, em seguida, obteve-se um experimento com 93 variáveis, sendo que 56 foram da lista gerada pelas simulações anteriores e 37 variáveis básicas do motor, que foi simulado 4 vezes para obter os resultados apresentados nessa seção. Para comparação, realizou-se um único experimento com 50 variáveis com os resultados obtidos com o conjunto anterior, sendo 27 dos experimentos anteriores.

Na Tabela 15 apresenta-se o conjunto geral de atributos selecionados pelos algoritmos, sendo que os resultados apresentados posteriormente escolheram os atributos a partir da lista.

Tabela 15 – Conjunto geral de atributos selecionados pelos algoritmos de SA para Detonação

Nº de atributos	**Categoria do atributo**
16	Velocidade do motor
9	Ignição
9	Detonação
4	Gás de escape
3	Sensor de oxigênio
2	Injeção
2	Fluxo de ar
2	Relação ar-combustível
2	Alta temperatura
2	Alta pressão
2	Posição virabrequim
2	Relação ar-combustível
1	Temperatura do líquido de arrefecimento
1	Distância total do veiculo
1	Posição do acelerador
1	Alternador

Fonte: Autora (2022).

A seguir estão apresentados os resultados da redução de atributos por cada método de SA no algoritmo de identificação de falha do aluno Victor Elizio Pierozan em seu trabalho de conclusão em Engenharia Mecatrônica, intitulado "Inteligência artificial aplicada para detecção de Knock em motores automotivos", com o conjunto de dados gerados pelas simulações com o segundo tipo de experimento com 93 variáveis.

Primeiramente, a Tabela 16 apresenta os resultados do método SelectPercentile com diferentes porcentagens de atributos selecionados.

Tabela 16 – Resultado SelectPercentile para Detonação

Porcentagem de atributos	Nº de atributos	Pontuação f1
20%	15	66%
15%	12	64%
10%	8	62%

Fonte: Autora (2022).

O resultado do método SelectPercentile com a redução de atributos foi muito similar, não obtendo resultados com significativa diferença. Além disso, nos três casos redução vale ressaltar que os principais atributos selecionados estão relacionados à detonação e velocidade do motor, além de em menor quantidade à temperatura do gás de escape, injeção e ignição.

A Tabela 17 apresenta os resultados obtidos com o método SelectKbest.

Tabela 17 – Resultado SelectKBest para Detonação

Nº de atributos	Pontuação f1
15	69%
10	78%
5	49%

Fonte: Autora (2022).

Ao comparar os atributos escolhidos pelo algoritmo configurado para selecionar 10 variáveis e com 15 e 5 variáveis, constatou-se que com 15 atributos se tem as categorias de injeção e engrenagem que não estão contidos na lista com 10, e com 5 atributos não se tem nenhuma de categoria de detonação e menos atributos de velocidade do motor.

A Tabela 18 apresenta os resultados obtidos pelo método SelectFromModel.

Tabela 18 – Resultado SelectFromModel para Detonação

Threshold	Nº de atributos	Pontuação f1
0.1	4	22%
None	5	36%

Fonte: Autora (2022).

Os principais atributos selecionados pelo método SelectFromModel estão relacionados com detonação, velocidade do motor e posição do acelerador. Pode-se concluir que o método não obteve resultado satisfatório para identificação da detonação.

Na Tabela 19 mostra-se os resultados encontrados usando o método RFE com estimador RandomForestRegressor.

Tabela 19 – Resultado RFE RandomForestRegressor para Detonação

Nº de atributos	**Pontuação f1**
15	67%
10	64%
5	69%

Fonte: Autora (2022).

Assim, como o resultado do método SelectPercentile, o método RFE com estimador RandomForestRegressor não apresentou significativa diferença com a mudança do número de atributos. Ademais, vale mencionar que não apenas os resultados ficaram similares como os atributos escolhidos foram os mesmos, exceto que o método RFE selecionou atributos de sensor de oxigênio. Assim, pressupõe-se que a diferença entre os resultados dos métodos possa estar relacionado a escolha dessa categoria de atributo.

Na Tabela 20 apresenta-se os resultados encontrados usando o método RFE com estimador LogisticRegression.

Tabela 20 – Resultado RFE LogisticRegression para Detonação

Nº de atributos	**Pontuação f1**
15	0%
10	0%
5	73%

Fonte: Autora (2022).

Todos os conjuntos selecionados pelo método RFE com estimador LogisticRegression, os atributos relacionados com velocidade do motor e sensor de oxigênio estavam presentes. Contudo, nota-se que os conjuntos com 15 e 10 atributos apresentaram uma pontuação f1 zerada, com um percentual baixo em *recall* e precisão 0. Acredita-se que esse resultado ocorreu devido a junção de outras categorias de variáveis aos dois tipos de atributos presentes no conjunto de 5, das quais para o conjunto com 10 atributos, variáveis de temperatura do gás de escape foram escolhidas, da mesma forma, o conjunto de 15 atributos incluía variáveis de injeção.

A Tabela 21 apresenta os resultados obtidos pelo método SFS com estimador LogisticRegression.

Tabela 21 – Resultado SFS LogisticRegression para Detonação

Nº de atributos	**Pontuação f1**
15	0%
10	0%
5	0%

Fonte: Autora (2022).

A Tabela 22 apresenta os resultados obtidos pelo método SFS LassoCV.

Tabela 22 – Resultado SFS LassoCV para Detonação

Nº de atributos	Pontuação f1
15	0%
10	0%
5	0%

Fonte: Autora (2022).

Pode-se constatar que o algoritmo de AM usando método SFS com ambos os estimadores foi incapaz de identificar a falha de detonação e o método de selecionar conjuntos de atributos adequados, pois para todos os conjuntos o resultado da pontuação f1 foi 0, com mesmo resultado para precisão. Acredita-se que esse resultado possa ter sido influenciado pela combinação dos atributos, já que diferente dos outros métodos, o SFS tendeu a selecionar mais atributos relacionados a detonação em conjunto com ignição.

Na Tabela 23 apresenta-se os resultados encontrados usando o método RFECV com estimador LogisticRegression.

Tabela 23 – Resultado RFECV LogisticRegression para Detonação

Nº de atributos	Pontuação f1
15	0%
10	0%
5	76%

Fonte: Autora (2022).

Todos os conjuntos selecionados para o método RFECV com estimador LogisticRegression selecionaram atributos de velocidade do motor e sensor de oxigênio. Entretanto, o conjunto com 10 atributos apresentou variáveis de injeção e temperatura de gás de escape, e o conjunto com 15 atributos obteve ainda mais variáveis de injeção e temperatura do líquido de arrefecimento. Observou-se que adicionar variáveis de injeção, temperatura do líquido de arrefecimento e de gás de escape juntamente com variáveis de velocidade do motor, o algoritmo foi incapaz de detectar a falha de detonação.

Na Tabela 24 apresenta-se os resultados encontrados usando o método RFECV com estimador RandomForestRegressor.

Tabela 24 – Resultado RFECV RandomForestRegressor para Detonação

Nº de atributos	Pontuação f1
15	0%
10	0%
5	0%

Fonte: Autora (2022).

Os conjuntos selecionados pelo método RFECV com estimador RandomForestRegressor contém atributos de posição angular do virabrequim para admissão,

que não foram selecionados pelos outros métodos, e não contém atributos de sensor de oxigênio.

Por fim, observou-se que dos resultados apresentados anteriormente, três métodos diferentes com números de atributos diferentes conseguiram obter os melhores resultados dentre os vários testes realizados de diferentes conjuntos de dados. A tabela com os métodos e outras informações podem ser vistos na Tabela 25.

Tabela 25 – Melhores resultados para Detonação

Método	**Nº de atributos**	**Precisão**	**Recall**	**Pontuação f1**
SelectKBest	10	97%	66%	78%
RFE Logistic Regression	5	98%	59%	73%
RFECV Logistic Regression	5	97%	62%	76%

Fonte: Autora (2022).

Pode-se observar que os três métodos citados como melhores resultados tiveram uma tendência a escolher atributos relacionados a velocidade do motor e sensor de oxigênio, que por sua vez, como citado na Figura 4, estão associados a frequência de rotação, pressão e temperatura do motor. Acredita-se que o equilíbrio entre os parâmetros poderiam melhorar, mantendo os atributos, com alguns ajustes no algoritmo de identificação de falha .

Como mencionado, foram realizados várias simulações com diferentes conjuntos de dados para fim de testes. O último experimento realizado possuía 50 variáveis, sendo que 27 foram os resultados dos experimentos anteriores e 23 da lista de variáveis básicas do motor. Realizou-se todo o processo com os algoritmos com os três melhores métodos e o último conjunto de dados reduzido, como pode ser visto na Tabela 26.

Tabela 26 – Melhores resultados em conjunto reduzido para Detonação

Método	**Nº de atributos**	**Precisão**	**Recall**	**Pontuação f1**
SelectKBest	10	98%	66%	78%
RFE Logistic Regression	5	99%	53%	69%
RFECV Logistic Regression	5	98%	54 %	69%

Fonte: Autora (2022).

Concluiu-se que a redução do experimento mantém o mesmo resultado ou piora a identificação da detonação pelos algoritmos utilizados.

Na Tabela 27 compara-se o resultado obtido pelo estudante Victor Elizio Pierozan em seu algoritmo e o resultado alcançado após aplicação da SA.

Tabela 27 – Resultado comparativo para Detonação

Método	Nº de atributos	Precisão	Recall	Pontuação f1
Seleção de atributos RFECV Logistic Regression	5	97%	62%	76%
Extração de atributos com classificador	9	74%	77%	76%

Fonte: Autora (2022).

O conjunto encontrado pela SA inclui 4 atributos de velocidade do motor e 1 atributo de sensor de oxigênio. Em comparação, o conjunto encontrado pelo algoritmo de identificação de falha inclui 4 atributos de sinal de detonação por cilindro, 1 atributo de válvula de pressão do coletor, 1 atributo de temperatura de ar do coletor, 1 atributo de velocidade do motor, 1 atributo de carga de ar do motor e 1 atributo de ruído de detonação médio.

É importante ressaltar que se conseguiu um resultado bem aproximado com menos atributos, tendo em vista que esse é o objetivo da SA, e utilizando os atributos levantados na árvore de causas de falha de detonação com alta frequência de rotação e sensor de oxigênio.

Em conclusão, com intuito de aplicação em algoritmos de aprendizado de máquina para identificação das falhas de motores a combustão interna estudados nesse trabalho, os métodos de seleção de atributos que melhor apresentaram resultados foram o SelectKBest, um método filtro, e o RFECV com o estimador LogisticRegression, um método *wrapper*. De modo que, o método filtro atingiu melhor resultado com 15 e 10 atributos e o método *wrapper* com 5 atributos.

5 CONCLUSÕES

Falhas em motores de combustão interna podem vir a ser muito danosas aos veículos, prejudicando a vida útil e desempenho. Um solução para evitar as falhas é o monitoramento das condições e sistemas do veículo, e para isso algumas soluções estão presentes no mercado. A Seleção de Atributos, utilizada em grandes quantidades de dados para observar os atributos mais relevantes no conjunto e reduzir a dimensionalidade, é uma estratégia de Aprendizado de Máquina que possui menor custo comparada a outras abordagens de identificação, predição e tratamento de falhas. Nesse sentido, o objetivo do trabalho foi conseguir quais atributos melhor identificam as falhas trabalhadas para que os sistemas correlacionados possam ser monitorados.

O trabalho desenvolvido baseou-se em diversas áreas do conhecimento da engenharia, como descrito detalhadamente no Capitulo 2, com o intuito de alcançar resultados relevantes no âmbito da Seleção de Atributos aplicada em Aprendizado de Máquina para falhas de motores a combustão interna, focando em falha de ignição e detonação.

Posteriormente a uma pesquisa aprofundada nas causas das falhas e aquisição dos dados, aplicou-se de métodos de Seleção de Atributos gerando subconjuntos de atributos. Esse subconjuntos foram aplicados em algoritmos de Aprendizado de Máquina, alcançando resultados consideráveis e, assim, validando as árvores de causas de falhas criadas. Deste modo, consegue-se afirmar que os objetivos propostos do trabalho foram alcançados.

A partir dos resultados obtidos para as falhas, a utilização de algoritmos de Seleção de Atributos pode impactar consideravelmente na identificação de falhas de motores. Acredita-se que os resultados apresentados poderiam ter uma melhoria caso houvesse novas simulações, com um conjunto de dados maior, e ajustes nos modelos dos algoritmos de Aprendizado de Máquina.

Portanto, conclui-se que observar resultados com outros algoritmos de identificação de falha, realizando alguns ajustes do modelo, poderia auxiliar a obtenção de um conjunto específico de atributos e um método mais preciso. Recomenda-se para pesquisas futuras esta abordagem comparativa para alcançar o conjunto ideal de atributos para identificação de falha em motores de combustão interna.

REFERÊNCIAS

ALMEIDA, T. B. et al. Seleção de atributos usando abordagem wrapper para classificação multirrótulo. **Universidade Tecnológica Federal do Paraná**, 2018.

ANJILA, F. **Artificial intelligence**. Learning Outcomes of Classroom research, 2021. Disponível em: https://www.researchgate.net/publication/358119068.

APOLÓNIA, J. **Seleção de atributos de dados inconsistentes**. 2019. Universidade Abert. Disponível em: http://hdl.handle.net/10400.2/8066.

ASAM. **ASAM MCD-2 MC**. 2022. Disponível em: https://www.asam.net/standards/detail/mcd-2-mc/wiki/#:~:text=An\%20A2L\%20\%2Dfile\%20contains\%20one,or\%20more\%20modules\%20(MODULE).

AZRIN, A. et al. An overview of the spark plug engine profile in a spark ignition engine. **IOP Conference Series: Materials Science and Engineering**, v. 1092, p. 012030, mar 2021.

AZUAJE, F. et al. Data mining: Practical machine learning tools and techniques. **Biomedical Engineering Online**, v. 5, p. 1–2, 01 2006.

BINGAMIL, A.; ALSYOUF, I.; CHEAITOU, A. Condition monitoring technologies, parameters and data processing techniques for fault detection of internal combustion engines: A literature review. In: **2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)**. [S.l.: s.n.], 2017. p. 1–5.

BOARD, C. A. R. Technical status update and proposed revisions to malfunction and diagnostic system requirements applicable to 1994 and subsequent california passenger cars, light-duty trucks, and medium - duty vehicles – (obdii). **CARB Staff Report**, 1991.

BOGUS, P.; MERKISZ, J. Misfire detection of locomotive diesel engine by non-linear analysis. **Mechanical Systems and Signal Processing**, v. 19, n. 4, p. 881–899, 2005. ISSN 0888-3270.

BRUNETTI, F. **Motores de Combustão Interna**. São Paulo: Blucher, 2012. v. 1.

CABENA, P. et al. **Discovering data mining: from concept to implementation**. [S.l.]: Prentice-Hall, Inc., 1998.

CAI, J. et al. Feature selection in machine learning: A new perspective. **Neurocomputing**, v. 300, p. 70–79, 2018. ISSN 0925-2312. Disponível em: https://www.sciencedirect.com/science/article/pii/S0925231218302911.

CORTI, E.; FORTE, C. Statistical analysis of indicating parameters for knock detection purposes. 04 2009.

COSTA, E. et al. Mineração de dados educacionais: conceitos, técnicas, ferramentas e aplicações. **Jornada de Atualização em Informática na Educação**, v. 1, n. 1, p. 1–29, 2013.

CUATTO, T. et al. A case study in embedded system design: an engine control unit. In: **Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)**. [S.l.: s.n.], 1998. p. 804–807.

DANTAS, C. A. **Seleção de atributos baseado em algoritmos de agrupamento para tarefas de classificação**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2017.

DASH, M.; LIU, H. Feature selection for classification. **Intelligent Data Analysis**, v. 1, n. 1, p. 131–156, 1997. ISSN 1088-467X. Disponível em: https://www.sciencedirect.com/science/article/pii/S1088467X97000085.

DOORNBOS, G.; DENBRATT, I.; DAHL, D. Knock phenomena under very lean conditions in gasoline powered si-engines. **SAE International Journal of Engines**, v. 11, 03 2018.

DZIUBINSKI, M. et al. Modelling characteristics of spark ignition engine injection system. **Advances in Science and Technology Research Journal**, v. 11, p. 103–117, 06 2017.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI Magazine**, v. 17, n. 3, p. 37, Mar. 1996. Disponível em: https://ojs.aaai.org/index.php/aimagazine/article/view/1230.

GARCíA-TORRES, M. et al. Feature grouping and selection on high-dimensional microarray data. In: **2015 International Workshop on Data Mining with Industrial Applications (DMIA)**. [S.l.: s.n.], 2015. p. 30–37.

GEORGE, E.; PECHT, M. Tin whisker analysis of an automotive engine control unit. **Microelectronics Reliability**, v. 54, n. 1, p. 214–219, 2014. ISSN 0026-2714. Disponível em: https://www.sciencedirect.com/science/article/pii/S0026271413003107.

GRANITTO, P. M. et al. Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products. **Chemometrics and Intelligent Laboratory Systems**, v. 83, n. 2, p. 83–90, 2006. ISSN 0169-7439. Disponível em: https://www.sciencedirect.com/science/article/pii/S0169743906000232.

HAO, J.; HO, T. K. Machine learning made easy: A review of scikit-learn package in python programming language. **Journal of Educational and Behavioral Statistics**, v. 44, n. 3, p. 348–361, 2019.

HULJANAH, M. et al. Feature selection using random forest classifier for predicting prostate cancer. **IOP Conference Series: Materials Science and Engineering**, v. 546, p. 052031, 06 2019.

JAMMOUSSI, H. et al. Diagnostics of individual air fuel ratio cylinder imbalance. **SAE International Journal of Passenger Cars - Electronic and Electrical Systems**, v. 126, n. 7, 201. ISSN 1946-4614. Disponível em: https://www.sae.org/publications/technical-papers/content/2017-01-1684/.

JOVIć, A.; BRKIć, K.; BOGUNOVIć, N. A review of feature selection methods with applications. In: **2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. [S.l.: s.n.], 2015. p. 1200–1205.

JUNG, D.; FRISK, E.; KRYSANDER, M. A flywheel error compensation algorithm for engine misfire detection. **Control Engineering Practice**, v. 47, p. 37–47, 2016. ISSN 0967-0661. Disponível em: https://www.sciencedirect.com/science/article/pii/S0967066115300575.

KAJI, K. et al. Development of knock senso. **SAE transactions**, JSTOR, p. 315–323, 1986.

KALGHATGI, G.; ALGUNAIBET, I.; MORGANTI, K. On knock intensity and superknock in si engines. **SAE International Journal of Engines**, v. 10, 03 2017.

KOMORSKA, I. Detection of the engine head gasket defects on the basis of vibration signal. **Silniki Spalinowe**, 2011. ISSN 0138-0346. Disponível em: https://www.infona.pl/resource/bwmeta1.element.baztech-article-LOD6-0030-0017.

LEE, H. D. **Seleção de atributos importantes para a extração de conhecimento de bases de dados**. 2005.

LENZEN, A. **A2LParser**. 2022. Disponível em: https://github.com/Luncher91/A2LParser.

LIU, H.; MOTODA, H. **Feature extraction, construction and selection: A data mining perspective**. [S.l.]: Springer Science & Business Media, 1998. v. 453.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. **Estudos Avançados**, Universidade de São Paulo, v. 35, n. 101, p. 85–94, 2021. ISSN 1806-9592.

MAHLE. **Manual Técnico do Curso Mahle Leve- Motores de Combustão Interna**. São Paulo: [s.n.], 2016. v. 1.

MARTIN, K. et al. Misfire detection by evaluating crankshaft speed - a means to comply with obdii. **Journal of engines SAE Transactions**, v. 102, 1993. Disponível em: c.

MISRA, P.; SINGH, A. Improving the classification accuracy using recursive feature elimination with cross-validation. v. 11, p. 659–665, 05 2020.

MITCHELL, T. **Machine Learning**. [S.l.]: McGraw-Hill Science/Engineering/Math, 1997. v. 432.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. In: **Sistemas Inteligentes Fundamentos e Aplicações**. 1. ed. Barueri-SP: Manole Ltda, 2003. p. 89–114. ISBN 85-204-168.

NAKAGAWA, S.; FUKUCHI, E.; NUMATA, A. A new diagnosis method for an air-fuel ratio cylinder imbalance. **SAE 2012 World Congress & Exhibition**, 2012. ISSN 0148-7191.

EL NAQA, I.; MURPHY, M. What is machine learning? In: ____. [S.l.: s.n.], 2015. p. 3–11. ISBN 978-3-319-18304-6.

PARMEZAN, A. et al. Avaliação de métodos para seleção de atributos importantes para aprendizado de máquina supervisionado no processo de mineração de dados. 12 2012.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. **the Journal of machine Learning research**, JMLR. org, v. 12, p. 2825–2830, 2011.

QIN, C. et al. Dtcnnmi: A deep twin convolutional neural networks with multi-domain inputs for strongly noisy diesel engine misfire detection. **Measurement**, v. 180, p. 109548, 2021. ISSN 0263-2241. Disponível em: https://www.sciencedirect.com/science/article/pii/S026322412100525X.

RAMASWAMI, M.; BHASKARAN, R. A study on feature selection techniques in educational data mining. **arXiv preprint arXiv:0912.3924**, 2009.

RODRIGUES, N. F. Diagnóstico de falha de ignição em veículos automotivos através de vibração de smartphone. **Universidade Federal da Paraíba**, 2018.

SCHUCH, R. et al. Mineração de dados em uma subestação de energia elétrica. In: **Proceedings of the 9th Brazilian Conference on Dynamics, Control and Their Applications–dincon**. [S.l.: s.n.], 2010. v. 10, p. 804–810.

SCIKIT-LEARN. **Ensemble RandomForestRegressor**. 2022. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=randomforestregressor#sklearn.ensemble.RandomForestRegressor.

SCIKIT-LEARN. **Feature selection**. 2022. Disponível em: https://scikit-learn.org/stable/modules/feature_selection.html.

SCIKIT-LEARN. **Linear Model LassoCV**. 2022. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoCV.html?highlight=lassocv#sklearn.linear_model.LassoCV.

SCIKIT-LEARN. **Linear Model LogisticRegression**. 2022. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic\%20regression#sklearn.linear_model.LogisticRegression.

SCIKIT-LEARN. **Mutual Info Regression**. 2022. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html.

SCIKIT-LEARN. **R Regression**. 2022. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.r_regression.html?highlight=r_regression#sklearn.feature_selection.r_regression.

SCIKIT-LEARN. **SelectKBest**. 2022. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest.

SCIKIT-LEARN. **SelectPercentile**. 2022. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html.

SCIKIT-LEARN. **Univariate Feature selection**. 2022. https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection.

SHARMA, A.; SUGUMARAN, V.; DEVASENAPATI, S. B. Misfire detection in an ic engine using vibration signal and decision tree algorithms. **Measurement**, v. 50, p. 370–380, 2014. ISSN 0263-2241.

SINGH, S.; POTALA, S.; MOHANTY, A. An improved method of detecting engine misfire by sound quality metrics of radiated sound. **Proceedings of the Institution of Mechanical Engineers Part D Journal of Automobile Engineering**, v. 233, 2018.

SOLORIO-FERNáNDEZ, S.; CARRASCO-OCHOA, J. A.; MARTíNEZ-TRINIDAD, J. F. A review of unsupervised feature selection methods. **Artificial Intelligence Review**, v. 53, p. 907–948, 2020. Disponível em: https://link.springer.com/article/10.1007/s10462-019-09682-y#citeas.

SOUZA, J. T. d. Métodos de seleção de atributos e análise de componentes principais: um estudo comparativo. **Universidade Tecnológica Federal do Paraná**, 2017. Disponível em: https://repositorio.utfpr.edu.br/jspui/handle/1/2387.

STEURS, K.; BLOMBERG, C. K.; BOULOUCHOS, K. Knock in an ethanol fueled spark ignition engine: Detection methods with cycle-statistical analysis and predictions using different auto-ignition models. **SAE Int. J. Engines**, v. 7, 2014. ISSN 1946-3936.

TANG, J.; ALELYANI, S.; LIU, H. Feature selection for classification: A review. **Data classification: Algorithms and applications**, CRC press, p. 37, 2014.

TING, L. L.; J. E. MAYER, J. Piston ring lubrication and cylinder bore wear analysis, part i—theory. **Journal of Lubrication Technology**, v. 96, p. 305–313, 1974. Disponível em: https://asmedigitalcollection.asme.org/tribology/article-abstract/96/3/305/418449/Piston-Ring-Lubrication-and-Cylinder-Bore-Wear.

TOYOTA, C. **6 Things That Can Cause Engine Knock In Your Car**. 2019. https://www.captoyota.com/service/information/learn-the-common-causes-for-engine-noise-knocking-salem-or.htm.

VENKATESH, B.; ANURADHA, J. A review of feature selection and its methods. **Cybernetics and Information Technologies**, v. 19, p. 3, 03 2019.

WEI, H. et al. Gasoline engine exhaust gas recirculation – a review. **Applied Energy**, v. 99, p. 534–544, 2012. ISSN 0306-2619. Disponível em: https://www.sciencedirect.com/science/article/pii/S0306261912003595.

ZEBARI, R. et al. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. **Journal of Applied Science and Technology Trends**, v. 1, n. 2, p. 56–70, 2020.

ZHEN, X. et al. The engine knock analysis - an overview. **Applied Energy**, v. 92, 04 2012.

ZHENG, T. et al. Real-time combustion torque estimation and dynamic misfire fault diagnosis in gasoline engine. **Mechanical Systems and Signal Processing**, v. 126, p. 521–535, 2019. ISSN 0888-3270. Disponível em: https://www.sciencedirect.com/science/article/pii/S0888327019301396.

ZHUANG, J.; YAN, Y.; ZHU, Z. Ecu calibration system based on asap. **Applied Mechanics and Materials**, v. 336-338, p. 1902–1906, 07 2013.