Aplicação de AutoML em Técnicas de Aprendizado de Máquina para Classificação de Motoristas

Francisco Érbio Dias
Programa de pós-graduação em engenharia elétrica
Universidade Federal do Piauí – UFPI
Email: erbiodias@gmail.com

José Maria Pires Menezes Júnior
Programa de pós-graduação em engenharia elétrica
Universidade Federal do Piauí – UFPI
Email: josemenezesjr@ufpi.edu.br

Resumo- Um sistema inteligente de telemetria é uma ferramenta valiosa para coletar e monitorar dados em tempo real de uma variedade de dispositivos e sensores, onde a otimização de parâmetros do modelo de classificação de motoristas é um processo importante para melhorar o desempenho do sistema. Desta forma, em vez de definir manualmente esses parâmetros, a otimização busca encontrar os valores ideais automaticamente, minimizando uma métrica de desempenho. No entanto, em um sistema de telemetria a grande quantidade de dados gerados pode tornar a análise manual difícil, exigindo soluções automatizadas. Nesse contexto, o uso de técnicas de aprendizagem de máquina com AutoML (Automated Machine Learning) tem se mostrado uma opção eficiente para otimização de modelos e análise de um grande volume de dados de forma automatizada. Nesse estudo é utilizado quatro técnicas de aprendizagem de máquinas: K-NN, Redes Neurais Artificiais tipo MLP, algoritmo Random Forest e o GradientBoostingClassifier com o método validação cruzada e a técnica do k-fold, com folds igual a 10. Os resultados dos algoritmos de otimização de hiperparâmetros mostraram desempenho superior quando comparado com os resultados sem a utilização de técnicas de AutoML, tanto nas taxas de acerto como na precisão em todos os modelos de técnicas de aprendizagem de máquinas aplicados.

Palavras-chaves: Telemetria, AutoML, Hiperparâmetro

I. INTRODUÇÃO

A otimização de parâmetros e hiperparâmetros é um processo importante em *machine learning*. Esses parâmetros e hiperparâmetros podem ter um grande impacto no desempenho do modelo, por isso é importante encontrar os valores ideais para maximizar a precisão do modelo. Neste trabalho, é discutido como realizar a otimização de parâmetros e hiperparâmetros usando a linguagem Python como apoio para desenvolvimento dos modelos.

É importante distinguir entre parâmetros e hiperparâmetros. Os parâmetros são valores internos do modelo que são aprendidos durante o treinamento, enquanto os hiperparâmetros são valores definidos antes do treinamento e que afetam o comportamento do modelo. Existem várias abordagens para otimização de parâmetros e hiperparâmetros em aprendizado de máquina, como busca em grade, busca aleatória, otimização bayesiana, otimização evolutiva e aprendizado por reforço [1].

O uso de técnicas de aprendizagem de máquina com AutoML (*Automated Machine Learning*) tem se mostrado promissor para a análise de dados de telemetria, melhorando significativamente os hiperparâmetros dos algoritmos de aprendizagem de máquinas. Por exemplo, em [1], foi desenvolvido um modelo de previsão de falhas em motores elétricos com o uso de técnicas de AutoML. Já em um estudo recente de [2], os autores exploraram o uso de técnicas de AutoML para a detecção de anomalias em sistemas de telemetria de veículos autônomos. Os resultados obtidos demonstraram que os modelos desenvolvidos apresentaram uma precisão maior do que outros modelos tradicionais e que as técnicas de AutoML podem detectar anomalias com alta precisão e eficiência, tornando-se uma solução viável para a detecção de falhas em tempo real em sistemas de telemetria.

Segundo [3], o uso de técnicas de AutoML em sistemas de telemetria pode trazer benefícios significativos em termos de eficiência, precisão e escalabilidade. Os autores propuseram um *framework* para a implementação de um sistema de telemetria com AutoML, no qual os dados são coletados, pré-processados e utilizados para treinar modelos de aprendizagem de máquina com o objetivo de realizar predições em tempo real. Os resultados mostraram que o sistema proposto foi capaz de obter uma precisão elevada e se mostrou eficiente em termos de tempo de processamento.

Através da utilização de técnicas de aprendizado de máquina, este trabalho propõem a aplicação das principais bibliotecas Python de AutoML para otimização de hiperparâmetros em um sistema de telemetria, uma vez que relevância desse estudo tem se ampliado significativamente nos últimos anos. Estudos realizados mostram que sistemas de telemetria são usados em hidrômetros residenciais [4], para coletar dados de um minifoguete [5], em aeronaves radio controladas [6], com técnicas de aprendizagem de máquina para predição do consumo de combustível em veículos [7] e para monitoramento de área ambiental [8].

As pesquisas já realizadas sobre o uso de técnicas de aprendizagem de máquinas com AutoML motivaram a realização desse estudo em busca da otimização de hiperparâmetros que apresentem uma melhor taxa de acerto para tarefas de identificação e classificação de condutores de veículos utilizando o sistema OBD II (*On Board Diagnostic*). Desta forma, esse trabalho traz também como contribuição o estudo das principais bibliotecas de AutoML disponíveis no Python.

Essas ferramentas para a otimização de hiperparâmetros de modelos de aprendizagem de máquina trazem benefícios como eficiência na seleção de recursos e otimização de modelos, redução de erros humanos, adaptação dinâmica e acesso facilitado a profissionais de diferentes áreas. Essas contribuições ajudam a melhorar a precisão, a confiabilidade e a eficácia da análise de dados de telemetria, permitindo tomadas de decisão mais informadas e eficientes.

Em relação a construção desse artigo, na seção II é mostrado o funcionamento do sistema OBD utilizado para a aquisição dos dados; na seção III são apresentadas as bibliotecas de AutoML para Python mais usadas; na seção IV aparece a metodologia empregada nesse estudo e na seção V são apresentados os resultados obtidos.

II. SISTEMA OBD II

Para que se possa receber informações do veículo, este deve possuir uma rede de comunicação interna capaz de monitorar seus sistemas eletromecânicos e entender as requisições realizadas por um agente externo. As especificações do OBD II incluem a padronização do conector e da função de seus pinos. Como é informado na norma SAE J1962, o conector, que deve estar posicionado até 0,61m do volante, é composto por duas colunas de oito pinos.

Além do sistema interno de controle do automóvel ser capaz de identificar as requisições externas é necessário um sistema que possa enviá-las pelo terminal. Para requerer quaisquer dados, o dispositivo compatível conectado ao terminal OBDII deve enviar comandos de diagnóstico ou parâmetros de informações, conhecidos como Parameter Identification (PID), padronizados pela SAE J1979 [9] e ISO15031-5 [10].

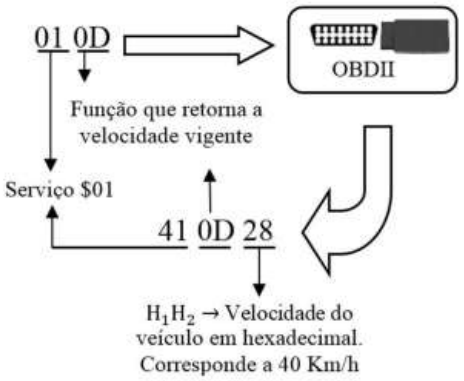
Os PIDs são divididos em um total de 10 serviços. Cada um possui uma série de códigos que podem ser utilizados para atividades específicas [11]. Neste trabalho, utiliza-se majoritariamente o serviço $01, que contém os PIDs relacionados às informações de funcionamento. Como o suporte à diferentes parâmetros dependem do fabricante, também é disponibilizado neste serviço um responsável pela identificação dos PIDs implementados.

Como informado na SAE J1979 [11], os hexadecimais que compõem cada parâmetro e sua resposta são uma forma de codificar as informações enviadas. Dessa forma, no PID 010D, por exemplo, 01 indica o serviço ao qual pertence o parâmetro e 0D a função específica do serviço escolhido. Assim, 010D é um parâmetro que realiza uma captura da velocidade do veículo em tempo real.

As respostas obtidas para cada PID, extraídas do sistema OBD II, são sequências de hexadecimais, as quais mudam para cada um dos parâmetros. A Figura 1 mostra as formas de cálculo para as informações coletadas e utilizadas neste trabalho. Note que os símbolos H1, H2, H3 e H4 representam a sequência de hexadecimais recebida como resposta útil para cada tipo de requisição.

As informações chave para captura, foram selecionadas com o intuito de garantir uma avaliação da situação do automóvel e do perfil de condução do motorista, pois são capazes de modelar o funcionamento do automóvel. Além disso, tais dados podem ser utilizados para o cálculo de informações não disponibilizadas pelo OBD II, como o consumo de combustível e tempo de parada.

Figura 1. Exemplo de comunicação utilizando um PID.



Fonte:[13]

III. TÉCNICAS DE APRENDIZAGEM DE MÁQUINA AUTOMATIZADA

Aprendizagem de Máquina Automatizada (AutoML) é uma abordagem que visa automatizar as etapas do processo envolvido nas técnicas de aprendizagem de máquina, tais como, pré-processamento dos dados, seleção e ajuste de algoritmos, ajuste de hiperparâmetros e até mesmo a geração automática de código. Mais especificamente, ela usa técnicas de automação e de busca para explorar e otimizar automaticamente diferentes opções e configurações do modelo, visando encontrar a melhor combinação possível para um determinado problema de aprendizado de máquina.

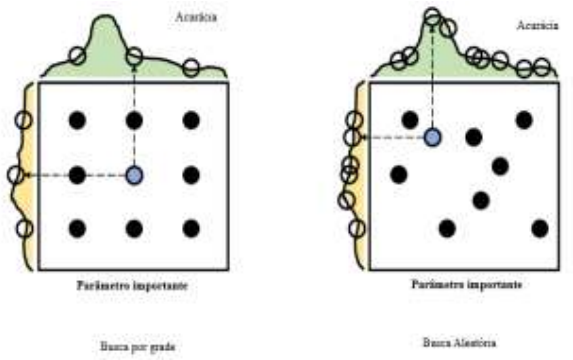
A otimização se mostra essencial e busca garantir que as técnicas de aprendizagem de máquinas operem com boa exatidão e eficiência. Portanto o uso de AutoML para a busca por hiperparâmetros tem se mostrado uma abordagem eficiente e promissora no desenvolvimento de modelos de aprendizado de máquinas. Existem vários métodos automáticos de otimização de hiperparâmetros usados na literatura, entre eles: busca em grade [12], busca aleatória [13], otimização baseada em gradiente [14], computação evolucionária [15] e otimização bayesiana [16].

Na visão de [17] a técnica de busca por grade (*Grid Search*) é uma abordagem sistemática que explora todas as combinações possíveis de valores para os hiperparâmetros especificados. A ideia é criar uma grade ou tabela com todas as combinações possíveis de valores dos hiperparâmetros e, em seguida, treinar e avaliar um modelo para cada

combinação. É como se fosse uma busca exaustiva por todas as combinações no espaço de hiperparâmetros definido.

A técnica de busca aleatória (*Random Search)*, por outro lado, é uma abordagem estocástica que amostra aleatoriamente um subconjunto dos valores possíveis para cada hiperparâmetro. Em vez de explorar todas as combinações, a Random Search seleciona aleatoriamente uma combinação de valores de hiperparâmetros e treina e avalia um modelo com essa combinação. Isso é repetido várias vezes com diferentes combinações aleatórias [18].

Figura 2. Comparação dos modelos *Grid Search* e *Random Search.*



Fonte: [18]

De acordo com [19] a otimização de hiperparâmetros utilizando a técnica de otimização bayesiana geralmente envolve os seguintes passos:

1) Definição do espaço de busca: Primeiro, é necessário definir o espaço de busca dos hiperparâmetros. Cada hiperparâmetro possui um intervalo ou um conjunto de valores possíveis.

2) Escolha de uma função de aquisição: Uma função de aquisição é usada para determinar qual configuração de hiperparâmetros deve ser avaliada em cada iteração. Uma função comum é a *Upper Confidence Bound* (UCB), que considera tanto a média estimada do desempenho quanto a incerteza associada a essa estimativa.

3) Construção de um modelo probabilístico inicial: Um modelo probabilístico inicial é construído para descrever a relação entre os hiperparâmetros e o desempenho do algoritmo. Geralmente, é utilizado um modelo de regressão chamado Processo Gaussiano (*Gaussian Process*) para modelar essa relação.

4) Avaliação das configurações de hiperparâmetros: Com base no modelo probabilístico inicial, é selecionada uma configuração de hiperparâmetros para avaliação. O algoritmo de aprendizado de máquina é executado com essa configuração e o desempenho é medido usando uma métrica pré-definida.

5) Atualização do modelo probabilístico: Com o desempenho da configuração avaliada, o modelo probabilístico é atualizado para refletir essa nova informação. A posteriori do modelo é recalculada para incorporar os resultados da avaliação.

6) Repetição dos passos 4 e 5: Os passos de avaliação e atualização do modelo são repetidos até que um número desejado de iterações seja alcançado ou até que uma configuração satisfatória dos hiperparâmetros seja encontrada.

Dentre as bibliotecas de AutoML para Python, pode-se destacar as seguintes, por serem as mais utilizadas e difundidas:

- Pycaret - uma biblioteca de machine learning de código aberto que automatiza todo o fluxo de trabalho de machine learning, desde a preparação dos dados até a seleção do modelo final.
- Auto-sklearn - uma biblioteca de AutoML baseada em aprendizado de máquina automatizado que seleciona e ajusta os hiperparâmetros do modelo.
- Autokeras - uma biblioteca de aprendizado profundo de código aberto que automatiza o processo de criação de modelos de redes neurais.
- H2O.ai: É uma plataforma de inteligência artificial que fornece várias ferramentas, incluindo o H2O AutoML. Essa biblioteca oferece automação completa do processo de aprendizado de máquina, desde pré-processamento de dados até a seleção e ajuste de modelos.

Cada uma dessas bibliotecas tem suas próprias vantagens e limitações, e a escolha de uma biblioteca depende das necessidades específicas do projeto.

O Pycaret permite que os usuários escolham entre o *Grid Search* e o *Random Search* para otimização de hiperparâmetros através do parâmetro "*search_method*" em sua função de treinamento de modelo. Por padrão, o Pycaret utiliza o *Grid Search*, mas os usuários podem especificar "*random*" para realizar a otimização usando a Random Search.

O AutoKeras é baseado em pesquisas e algoritmos existentes no campo de otimização de hiperparâmetros e busca automatizada. Sua abordagem geral é inspirada por algoritmos de otimização, como otimização bayesiana [18].

A biblioteca H2O AutoML realiza a busca por hiperparâmetros de forma automatizada. Ela também utiliza técnicas como a busca em espaço de configuração ou otimização bayesiana para encontrar a combinação de hiperparâmetros que maximize o desempenho do modelo com base na métrica de avaliação escolhida.

Para a realização dessa pesquisa está sendo utilizada a linguagem de programação Python na plataforma Google Colab, com as bibliotecas pycaret, pandas, *sklearn*, *matplotlib*, *numpy*, *matplotlib* e *seaborn*. Para aprendizagem automático de máquina foram utilizadas a pycaret, H2O.ai e Autokeras auxiliando na escolha do melhor modelo de classificação, através da comparação entre os modelos disponíveis na biblioteca.

Uma das funcionalidades da biblioteca Pycaret é que ela permite comparar automaticamente vários modelos de aprendizado de máquina para problemas de classificação de acordo com o problema a ser resolvido. No caso do banco de dados desse estudo está sendo utilizado o classificador *Gradient Boosting Classifier*, que gera um modelo aditivo de maneira progressiva, através de uma estrutura de dados que usa escolhas árvores paralelas, onde cada árvore é construída a partir do ajuste dos resíduos da árvore anterior. Escolhidos ainda três outros modelos: o *Multilayer*

Perceptron, modelo de rede neural artificial que se caracterizam por usar uma ou mais camadas de neurônios ocultas, o *Random Forest*, que é um algoritmo que combina várias árvores de decisão individuais, onde a previsão de uma nova amostra é feita através de uma votação majoritária das previsões individuais das árvores componentes, e o K-NN *(k Nearest Neighbor*), um algoritmo não-paramétrico para aprendizagem supervisionada, que consiste em atribuir a um exemplo de teste **x** a classe do seu vizinho mais próximo, classificando **x** atribuindo a ele o rótulo representado mais frequentemente dentre as *k* amostras mais próximas [20][21].

IV. METODOLOGIA

Para a realização dessa pesquisa está sendo utilizada a linguagem de programação Python na plataforma Google Colab, com as bibliotecas pycaret, pandas, sklearn, numpy, matplotlib e seaborn. Para aprendizagem automático de máquina são utilizadas a pycaret, H2O.ai e Autokeras auxiliando na escolha do melhor modelo de classificação, através da comparação entre os modelos disponíveis na biblioteca.

Uma das funcionalidades das bibliotecas PyCaret, e H2O é que elas permitem comparar automaticamente vários modelos de aprendizado de máquina para problemas de classificação de acordo com o problema a ser resolvido. No caso do banco de dados desse estudo está sendo utilizados quatro modelos de classificadores: Extreme Gradient Boosting (Xgboost), o Multilayer Perceptron, modelo de rede neural artificial que se caracterizam por usar uma ou mais camadas de neurônios ocultas, o Random Forest, e o K-NN (k Nearest Neighbor).

A. Banco de dados Utilizado

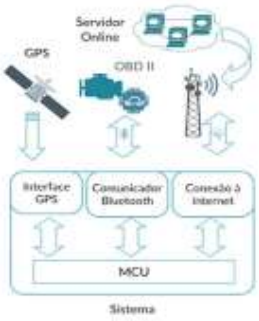
O conjunto de dados aqui utilizado é construído quanto aos motoristas, diferenciando quem está dirigindo o veículo. Para a aquisição do conjunto de dados são selecionados cinco motoristas distintos para conduzir o veículo no percurso instruídos a realizarem o percurso naturalmente, como se estivessem dirigindo em virtude de suas atividades diárias, com objetivo de a partir da análise desse banco classificar o tipo de motorista que está conduzindo o veículo.

Nesse estudo, como dados de entrada, é utilizada a base de o banco de dados citado acima criado e implementado em sua totalidade por [22], com 7582 amostras e 8 atributos, os quais são: Consumo (km/l), Carga no Motor, Velocidade do motor (Rotações por minuto – RPM), Fluxo de Massa (MAF), Velocidade do veículo (km/h), Temperatura do ar de Admissão (*Load*), Posição do Acelerador, e perfil de condução. Este último classificando cinco tipos de motoristas diferentes.

A captura e aquisição de dados é composta por três hardwares distintos. O primeiro é a placa com microcontrolador ESP32, utilizada neste trabalho por disponibilizar interfaces de comunicação integradas em sua estrutura. Os outros correspondem ao módulo Bluetooth HC-05 e ao scanner ELM327, padrão OBD-II. A Figura 3 detalha o sistema de telemetria utilizado para a aquisição dos dados.

As informações disponibilizadas pelo servidor, após capturadas do veículo, são seis: Velocidade do veículo, Rotações por minuto, Pressão do ar de admissão – MAP, aceleração, temperatura e carga no motor. Destas, acrescentou-se o cálculo do consumo de combustível.

Figura 3. Sistema de telemetria automotiva.



Fonte:[22]

Uma descrição acerca dos atributos é apresentada a seguir:

- Velocidade do veículo: representada pela grandeza medida pelo sensor de velocidade. Diferentemente da velocidade calculada via GPS, de acordo com o comentado em um bom número de trabalhos com OBD-II [9];
- Pressão do ar de admissão – MAP: Corresponde a grandeza medida por sensores do sistema de controle eletrônico, que transmite informações instantâneas da pressão do coletor. Esses dados são usados no cálculo da densidade do ar a fim de determinar a vazão mássica do ar do motor, que, por sua vez, determina a quantidade de combustível necessária para a combustão ideal (estequiometria), influenciando diretamente no comportamento e desempenho do motor, consumo de combustível e aspectos das emissões de gases [23];
- Posição do acelerador: Controla a entrada de ar no motor, ou seja, quando é aberto aumenta a entrada de ar; quando está quase fechado, pouco ar entra, e de modo geral está localizado no eixo da borboleta. A posição do acelerador e a rapidez com que ele é pressionado são transmitidas ao módulo de controle do motor, sendo utilizadas para injeção de combustível no motor, refletindo diretamente na velocidade do veículo [24];
- Carga do Motor: Corresponde a porcentagem do pico de torque disponível. Sofre influência da temperatura do ar de admissão (IAT), da pressão absoluta do coletor (MAP), da posição do acelerador (TPS) e da temperatura do líquido de arrefecimento do motor [25];
- Temperatura do ar de admissão: Representa a temperatura do ar que entra no motor. É fornecida por um sensor com termistor, significando que as mudanças de temperatura alterarão a resistência elétrica do sensor. O defeito nesse sensor acarretaria

a falha de ignição, fato que prejudica a economia de combustível e aumenta as emissões de gases [26]

• Rotações do Motor: Representa o cálculo da velocidade de rotação do eixo de manivela do motor na ocorrência do movimento rotacional do mesmo em termos de RPM (rotações por minuto) [27].

B. Bibliotecas e Métricas utilizadas

O estudo está dividido em duas etapas, a primeira feita de forma manual, usando o treinamento dos modelos de técnicas de aprendizagem de máquina, através de algoritmos das bibliotecas keras e sklearn, e a segunda etapa usando as bibliotecas de AutoML Pycaret, H2O e Autokeras.

Na etapa de pré-processamento, os dados são convertidos automaticamente em informações categóricas presentes no *dataset* em dados numéricos, através da função *preprocessing.LabelEncoder*() da biblioteca *sklearn*, para melhor aplicação dos algoritmos de aprendizagem de máquina pelas ferramentas disponíveis no pycaret.

Durante a etapa de treinamento dos modelos de *Machine Learning* é utilizado o método validação cruzada (*cross validation*) e a técnica do *k-fold*, com *folds* igual a 10. Para a rede MLP são utilizados os seguintes parâmetros: três camadas escondidas, uma com 100 neurônios e as outras duas com 50 cada.

Primeiramente é feita a classificação das técnicas sem o AutoML Depois de aplicados os algoritmos K-NN, Xgboost, Random Forest e Rede MLP da biblioteca Pycaret são utilizados também no mesmo bando de dados, os algoritmos da biblioteca H2O.ai para se otimizar os hiperparâmetros das técnicas de aprendizagem de máquina utilizadas no banco de dados original (exceto o K-NN, por não estar disponível nessa biblioteca). Por fim a biblioteca Autokeras é utilizada para otimizar os hipeparâmetros da rede neural artificial escolhida neste estudo. Em seguida, os resultados obtidos são analisados.

E por fim, analisa-se as seguintes métricas: I. Acurácia, que diz respeito a taxa de acertos do classificador, II. Precisão, que é a proporção dos exemplos que realmente possuem a classe entre todos aqueles que são classificados como pertencentes a ela, III. *Recall* (percentual de amostras positivas classificadas corretamente sobre o total de amostras positivas), e IV. F1-*score* (média ponderada de Precisão e *Recall*, conforme Tabela 1.

O Verdadeiro Positivo (VP) e Verdadeiro Negativo (VN) ocorrem quando a classe buscada está correta. O Falso Positivo (FP) e o Falso Negativo (FN) ocorrem quando a classe buscada é classificada incorretamente.

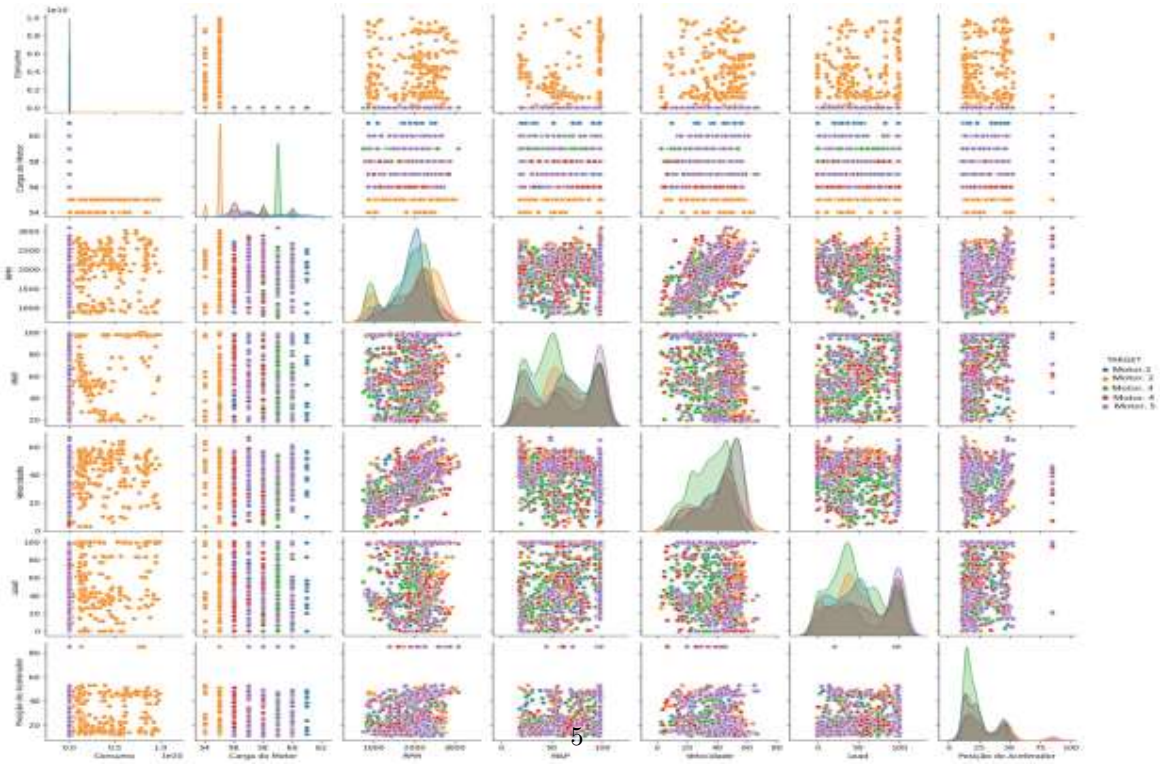
Tabela 1. Fórmula para o cálculo das métricas de avaliação utilizadas.

Métrica	**Fórmula**
Acurácia	$\frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}}$
Precisão	$\frac{\text{VP}}{VP + FP}$
Recall	$\frac{\text{VP}}{VP + FN}$
F1-score	$\frac{2 * \text{Precisão} * \text{Recall}}{Precisão + Recall}$

V. RESULTADOS

Nesse estudo são consideradas as relações de correlação das variáveis envolvidas, conforme mostrado na Figura 4.

Figura 4: Relação de correlação entre as variáveis envolvidas.



Percebe-se que, dentre os atributos deste banco de dados, o "Consumo", o "MAP" e o "Carga do Motor" são importantes, por estarem entre os atributos que mais permitem a separação entre as classes, proporcionando uma melhor caracterização do tipo de motorista, que está conduzindo o veículo.

Na primeira parte do experimento, sem uso de AutoML, o classificador K-NN apresenta melhor acurácia média e melhor precisão que os demais algoritmos, 88,53% e 88,56% respectivamente. Mas, após a aplicação da biblioteca AutoML Pycaret, observa-se que o classificador Xgboost obtém melhores resultados, tanto na acurácia média, quanto na precisão, chegando ao valor de 99,38%.

Com aplicação da biblioteca H20.ai o classificador Xgboost também obtém melhores resultados, tanto na acurácia média, quanto na precisão, chegando ao valor de 99,62%. Resultado similar ao encontrado Pycaret, demostrando que as bibliotecas de AutoML possuem desempenhos parecidos, principalmente por conta dos processos de otimização utilizados.

Para a rede neural MLP é utilizado para a otimização dos hiperparâmetros a biblioteca Pycaret, que gera os seguintes valores como os melhores: duas camadas escondidas, a primeira com 100 neurônios e a outra com 50 neurônios, 200 épocas, taxa de aprendizagem igual a 0,001 e acurácia de 58,9%. Para H20.ai a acurácia é de 79,5%, e já com a biblioteca Autokeras é melhorado a acurácia da Rede Neural Artificial para 96%, conforme pode ser visto na Tabela 2.

Tabela 2: Comparativo das métricas obtidas por cada classificador com as bibliotecas de automl utilizadas.

Classificador/ Biblioteca	**Acurácia**	**Precisão**	***Recall***	**F1-*score***
K-NN				
Sklearn	88,53%	88,56%	87,65%	87,98%
Pycaret (AutoML)	**97%**	**97,03%**	**97%**	**97%**
XGBOOST				
Sklearn	87,25%	86,92%	85,73%	86,15%
Pycaret (AutoML)	**99,38%**	**99,38%**	**99,38%**	**99,38%**
H2O.ai (AutoML)	**99,62%**			
Random Forest				
Sklearn	57,58%	67,67%	50,64%	47,06%
Pycaret (AutoML)	76,82%	80,16%	76,82%	76,37%
H2O.ai (AutoML)	**99,43%**			
Rede MLP				
Sklearn	58,55%	69,34%	52,18%	49,59%
Pycaret (AutoML)	58,9%	56,39%	58,9%	55,56%
H2O.ai (AutoML)	79,56%			
Autokeras (AutoML)	**96%**			

É importante destacar na Tabela 2 que na linha onde aparece o nome da biblioteca Sklearn, esta representa os resultados obtidos antes de se utilizar as bibliotecas de AutoML para se otimizar os hiperparâmetros dos algoritmos utilizados nesse estudo.

Observa-se ainda que as técnicas de aprendizagem de máquina Xgboost e Random Forest, alcançam uma taxa de acerto com valores muito próximos com a aplicação da Biblioteca H2O.ai, de onde vê-se que constituem ferramentas eficazes para a solução de problema de classificação de dados obtidos através de um sistema de telemetria.

Os estudos realizados indicam que a aplicação das bibliotecas de AutoML nas técnicas de aprendizagem de máquina têm sido utilizadas com sucesso na otimização de hiperparâmetros, melhorando os resultados das métricas de avaliação e favorecendo melhores taxas de acerto na resolução de problemas de classificação.

VI. CONCLUSÕES

Devido ao aumento da coleta de dados em tempo real em diversos setores, é necessário o uso de tecnologia eficaz para analisar esses dados e tomar decisões oportunas. Nesses casos, combinar técnicas de aprendizado de máquina com AutoML tem se mostrado uma solução eficaz, pois permite selecionar e ajustar automaticamente o modelo de aprendizado de máquina mais adequado para o problema em questão.

A pesquisa mostra que combinar técnicas de aprendizado de máquina com AutoML em sistemas de telemetria pode trazer benefícios significativos em termos de eficiência, precisão e escalabilidade. Automatizar o processo de seleção e ajuste de modelos de aprendizado de máquina permite uma análise mais rápida e eficiente dos dados coletados, permitindo que decisões oportunas sejam tomadas. Pode-se inferir que o uso de técnicas de aprendizado de máquina e autoML em sistemas de telemetria é uma área de estudo promissora.

Chega-se à conclusão de que o uso das bibliotecas de AutoML Pycaret, H2O e Autokeras se mostram ferramentas eficazes para otimizar hiperparâmetros de modelos para técnicas de aprendizado de máquina, melhorando a taxa de acerto na classificação de padrões, além de permitir uma eficiente análise de dados veiculares obtidos através de um sistema de telemetria.

REFERÊCIAS BIBLIOGRÁFICAS

[1]. YANG, J.; ZHANG, H.; LI, H.; LI, Y.; LI, X.; LI, X. Automatic machine learning-based fault prediction for electric motors. IEEE Access, 8, 81999-82010. (2020).

[2]. SINGH, A.; SHUKLA, R.; PATEL, M. Anomaly detection in autonomous vehicle telematics data using AutoML. International Journal of Advanced Intelligence Paradigms, 18(1), 45-62. (2022).

[3]. ALOMARI, M. A.; KHADER, A. T.; AL-BASHABSHEH, A. A scalable end-to-end telemetric system with automated machine learning models selection. Measurement, 166, 108138. (2020).

[4]. PASSOS, Ivan dos; QUADROS, Murilo Augusto Castagnoli de; AVEIRO, Tiago Gregio de. Sistema de telemetria de hidrômetro residencial. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná. (2015).

[5]. MEDEIROS, J. G. Sistemas de telemetria embarcada com Arduino para coleta de dados em um minifoguete experimental. Programa de Pós-Graduação. Universidade Federal do Paraná, Paraná, Brasil, (2015).

[6]. GOMES, Pedro Henrique de Oliveira. Modelamento de sistema de telemetria para aeronaves remotamente controladas. 2016.

[7]. LI, H.; Ota, K.; DONG, M. Learning iot in edge: Deep learning for the internet of things with edge computing.IEEENetwork, v. 32, n. 1, p. 96–101, (2018).

[8]. CABRAL, Samuel. Um sistema de telemetria com tecnologias GSM/GPRS para a área ambiental. 2023.

[9]. DESREVEAUX, A. et al. Impact of the velocity profile on energy consumption of electric vehicles. IEEE Transactions on Vehicular Technology, IEEE, v. 68, n. 12, p. 11420–11426, 2019.

[10]. International Standard Office (2015) ISO 15031: Road vehicles -- Communication between vehicle and external equipment for emissions-related diagnostics -- Part 5: Emissions-related diagnostic services.

[11]. Society of Automotive Engineers (2012) SAEJ1979: E/E Diagnostic Test Modes.

[12]. LERMAN, E.; Otimização estocástica: uma visão geral. In: Controle e sistemas dinâmicos, v. 18, p. 293-336, 1980.

[13]. SOLIS, Francisco Javier; WETS, Willem H."Minimização por técnicas de busca aleatória." Matemática da pesquisa operacional 6.1 (1981): 19-30.

[14]. BENGIO, Yoshua. "Otimização baseada em gradiente de hiperparâmetros." Computação neural 12.8 (2000): 1889-1900.

[15]. FRIEDRICHS, Jens e Christian Igel. "Estratégias evolutivas para otimização de parâmetros em máquinas de vetores de suporte." Conferência Europeia sobre Computação Evolutiva em Otimização Combinatória. Springer, Berlim, Heidelberg, 2005.

[16]. PELIKAN, Martin.; GOLDBERG, Dirk E.; CANTÚ-PAZ, Erick. "Algoritmos de otimização bayesiana para ajuste de hiperparâmetros SVM." Conferência Internacional sobre Algoritmos Computacionais Adaptativos e Naturais. Springer, Berlim, Heidelberg, 1999.

[17]. KUHN, M.; JOHNSON, K. (2013). Applied Predictive Modeling. Springer.

[18]. BERGSTRA, J.; BENGIO, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13, 281-305.

[19]. BROCHU, E.; CORA, V. M.; DE FREITAS, N. (2010). A Tutorial on Bayesian Optimization. Proceedings of the IEEE, 98(6), 1022-1039. doi: 10.1109/JPROC.2009.2034765.

[20]. CHEN, T.Q.; GUESTRIN, C. (2016) XGBoost: A Scalable Tree Boosting System. arXiv:1603.02754v3.

[21]. FACELI, Katti et al. Inteligência artificial: uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, (2011).

[22]. RIBEIRO, Pedro H. A. Sistema inteligente de telemetria para classificação de dados veiculares. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal do Piauí. Teresina, 2020.

[23]. ABDULLAH, N. R. et al. Effects of air intake pressure to the fuel economy and exhaust emissions on a small si engine. Procedia Engineering, Elsevier, v. 68, p. 278–284, 2013.

[24]. LIU, Y.; SUN, B.; YANG, J. The position estimation of electronic throttle based on kalman-like estimator. In: IEEE. 2019 Chinese Automation Congress (CAC). [S.l.], 2019. p. 3342–3346.

[25]. MOUSAVI, M. S. R.; ALIZADEH, H. V.; BOULET, B. Estimation of synchromesh frictional torque and output torque in a clutchless automated manual transmission of a parallel hybrid electric vehicle. IEEE Transactions on Vehicular Technology, IEEE, v. 66, n. 7, p. 5531–5539, 2016.

[26]. CHEN, W.; LI, H.; LIU, S.; ZHANG, L. Fault prediction of telemetric systems using Auto-Keras. IEEE Access, 7, 78953-78960. (2019).

[27]. NA, J. et al. Vehicle engine torque estimation via unknown input observer and adaptive parameter estimation. IEEE Transactions on Vehicular Technology, IEEE, v. 67, n. 1, p. 409–422, 2017.