UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Leonardo Tomasi Francis

Anomaly Detection of Engine Knock in Automotive ECU using Machine Learning Algorithms

Florianópolis
2023

Leonardo Tomasi Francis

Anomaly Detection of Engine Knock in Automotive ECU using Machine Learning Algorithms

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Ciência da Computação.
Supervisor:: Prof. Giovani Gracioli, Dr.
Co-supervisor:: Prof. Gustavo Medeiros, Dr.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Francis, Leonardo Tomasi
Anomaly Detection of Engine Knock in Automotive ECU using Machine Learning Algorithms / Leonardo Tomasi Francis ; orientador, Giovani Gracioli, coorientador, Gustavo Medeiros, 2023.
70 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2023.

Inclui referências.

1. Ciência da Computação. 2. Engine Failure Detection. 3. Machine Learning. 4. Engine Knock. 5. Fault Detection. I. Gracioli, Giovani . II. Medeiros, Gustavo. III. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Ciência da Computação. IV. Título.

Leonardo Tomasi Francis

Anomaly Detection of Engine Knock in Automotive ECU using Machine Learning Algorithms

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof.(a) Patricia Della Mea Plentz, Dr(a).
INE/CTC/UFSC

Prof.(a) Vinicius Faria Culmant Ramos, Dr.
EGC/CTC/UFSC

Prof. Cristian Cechinel, Dr.
CIT/CTS/ARA/UFSC

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Ciência da Computação.

Documento assinado digitalmente
Marcio Bastos Castro
Data: 29/11/2023 11:36:57-0300
CPF: ***.707.720-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de
Pós-Graduação

Documento assinado digitalmente
Giovani Gracioli
Data: 29/11/2023 11:32:19-0300
CPF: ***.838.060-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Giovani Gracioli, Dr.
Supervisor:

Florianópolis, 2023.

Este trabalho é dedicado aos meus pais e amigos, pelo apoio constante e amor inabalável. Suas presenças fizeram esta jornada valer a pena.

ACKNOWLEDGEMENTS

The author acknowledges the financial support of the Fundação de Ensino e Engenharia de Santa Catarina - FEESC.

RESUMO

Esta dissertação avalia múltiplos algoritmos de aprendizado de máquina para detecção de *engine knock* em veículos com motores de combustão interna. O objetivo é utilizar dados de baixa frequência extraídos diretamente da Unidade de Controle do Motor (ECU), aplicar uma abordagem baseada em dados para detectar a falha e melhorar a compreensão de quais fatores aplicados nos dados podem impactar esse processo de detecção. Os objetivos do estudo incluem identificar e selecionar os algoritmos mais prevalentes utilizados para detecção de falhas em sistemas automotivos. Isto será seguido pela criação de conjuntos distintos de variáveis, permitindo uma avaliação sistemática de suas contribuições individuais para os resultados da detecção de falhas. Além disso, a investigação abrangerá vários aspectos do pré-processamento de dados, analisando as suas influências no desempenho global dos métodos de detecção de falhas. Em última análise, o estudo visa avaliar de forma abrangente o desempenho de diversos modelos, facilitando a seleção daquele que demonstra o maior f1-score na detecção de falhas no contexto de sistemas automotivos.

Ao analisar os dados coletados da ECU de um carro Renault Sandero, vários algoritmos de aprendizado de máquina foram explorados, incluindo Classificador, Autoencoders Densos e Convolucionais, SVM e Floresta Isolada. Esses algoritmos foram treinados e avaliados usando um conjunto de dados abrangente compreendendo 19 variáveis de 32 experimentos. Apesar da limitação de taxas de amostragem mais baixas, resultados promissores foram alcançados, com uma taxa máxima de detecção de *engine knock* de f1-score 81%. Melhorias futuras são propostas, como a incorporação de dados de detecção adicionais e o desenvolvimento de um sistema de detecção abrangente que vai além de depender apenas do sensor piezoelétrico.

No geral, este estudo demonstra o potencial das técnicas de aprendizado de máquina para detecção de *engine knock* em motores de veículos automotivos. Enfatiza a importância de abordagens baseadas em dados para melhorar a segurança e o desempenho dos motores de combustão interna. As descobertas contribuem para os esforços contínuos de pesquisa na indústria automotiva e inspiram novos avanços nos sistemas de detecção de falhas.

Palavras-chave: Detecção de falha do motor. Aprendizado de Máquina. Problema de classificação. Batida de Pino. Detecção de anomalia.

ABSTRACT

This dissertation evaluates multiple machine learning algorithms for engine knock detection in vehicles with internal combustion engines. The goal is to utilize low-frequency data extracted directly from the Engine Control Unit (ECU), apply a data-driven approach to detect the fault and improve the understanding of which factors applied in the data can impact this detection process. The objectives of the study include identifying and selecting the most prevalent algorithms utilized for fault detection within automotive systems. This will be followed by creating distinct sets of variables, allowing for a systematic assessment of their individual contributions to the outcomes of fault detection. Furthermore, the investigation will encompass various aspects of data preprocessing, analyzing their influences on the overall performance of fault detection methods. Ultimately, the study aims to comprehensively evaluate the performance of diverse models, facilitating the selection of the one that demonstrates the utmost f1-score in detecting faults within the context of automotive systems.

Various machine learning algorithms were explored by analyzing data collected from the ECU of a Renault Sandero car, including Classifier, Dense and Convolutional Autoencoders, SVM, and Isolated Forest. These algorithms were trained and evaluated using a comprehensive dataset comprising 19 variables from 32 experiments. The lower sampling rates were effective in detecting engine knock at a high rate of 81%. Future improvements are proposed, such as incorporating additional sensing data and developing a comprehensive detection system beyond relying solely on the piezoelectric sensor.

Overall, this study demonstrates the potential of machine learning techniques for engine knock detection in automotive vehicles. It emphasizes the importance of data-driven approaches in enhancing the safety and performance of internal combustion engines. The findings contribute to ongoing research efforts in the automotive industry and inspire further advancements in fault detection systems.

Keywords: Engine Failure Detection. Machine Learning. Classification Problem. Engine Knock. Fault Detection.

RESUMO EXPANDIDO

Introdução

Profissionais de diversos setores da indústria automotiva têm conduzido extensas pesquisas sobre Motores de Combustão Interna (MCI) devido à sua importância econômica para melhorar a eficiência de combustível e cumprir regulamentações de emissões de carbono. Apesar dos esforços para promover energias renováveis, os MCI ainda desempenham um papel significativo no consumo de petróleo no transporte, representando cerca de 60% do total em 2021. O desenvolvimento da diretiva 2018/2001/EC reflete a busca global por energia renovável. A evolução da indústria automotiva desde os anos 1970 incluiu avanços nas chamadas "diagnósticos a bordo"(OBD), inicialmente sistemas simples, agora evoluídos para autodiagnóstico veicular. Com o aumento das unidades de controle eletrônico (ECUs) nos veículos (30 a 150 por carro), a coleta de dados dessas ECUs tornou-se viável, gerando grandes volumes de dados. Um desafio crítico enfrentado pelos fabricantes automotivos é a detecção de detonação ou *engine knock*, um fenômeno prejudicial em MCI modernos. A detecção deste fenômeno pode ocorrer dentro ou fora da câmara de combustão, sendo a detecção interna ideal com sensores de pressão, embora o custo seja um obstáculo. A detecção externa, via sensores de vibração conectados à ECU, é mais comum, onde a ECU analisa o sinal de vibração e ajusta a calibração do motor. Métodos para detectar *engine knock* incluem escutar o som, uso de sensores de detonação e análise de dados do motor. A implementação de aprendizado de máquina, como Redes Neurais Artificiais (ANNs) e Convolutional Neural Networks (CNNs), destaca-se para detecção de falhas. A indústria automotiva tem adotado sensores de vibração, permitindo o uso de técnicas de processamento de sinal, como Fast Fourier Transform (FFT). Ao explorar uma abordagem orientada por dados e aproveitar as variáveis internas controladas pela ECU, é possível monitorar uma variedade de dados do sensor, incluindo o sensor de detonação. A coleta e análise dessas leituras permitem identificar falhas ou anomalias relacionadas ao *engine knock*, facilitando a detecção precoce de falhas. O uso de técnicas de aprendizado de máquina desempenha um papel crucial na implementação de técnicas detecção de falhas que trazem mais segurança aos veículos, e por consequência, aos usuários e a sociedade.

Objetivos

Este trabalho tem como objetivo principal aplicar modelos de aprendizado de máquina para detectar *engine knock* em veículos automotivos por meio de dados coletados pela ECU do motor, utilizando dados de baixa frequência. Objetivos específicos incluem a seleção de algoritmos, formação de conjuntos de variáveis, análise de preparação de dados e avaliação do desempenho do modelo.

Metodologia

ECU significa Unidade de Controle Eletrônico, que gerencia vários sistemas elétricos em um veículo. A ECU recebe dados de vários sensores no veículo e utiliza esses dados para controlar várias funções, incluindo injeção de combustível, temporização de ignição, controle de velocidade ociosa e mais. O hardware Sistema Inteligente de Aquisição e Análise para ECU (IASE) foi desenvolvido pela LISHA e Renault e permite acesso às variáveis monitoradas pela ECU. Este hardware específico se conecta à ECU por meio dos protocolos XCP ou CCP para extrair dados em baixa frequência a serem analisados. Dividimos o processo em quatro etapas para facilitar a compreensão. Na Etapa 1, criamos um Modelo de Dados que associa cada sinal da ECU a uma estrutura Smartdata, definindo as características do sinal. Para extrair os dados de cada sinal da ECU, usamos o Modelo de Dados para gerar o arquivo de experimento. Esse arquivo de experimento é responsável por selecionar um conjunto de sinais e definir a taxa de amostragem de cada sinal. Cada taxa de amostragem tem uma quantidade máxima de sinais. A menor taxa de amostragem (4 ms) tem uma quantidade limitada de sinais que podem ser selecionados. O hardware IASE deve ser carregado com o arquivo de experimento para iniciar o processo de aquisição. A Etapa 2 envolve a interação entre o Dispositivo IoT e o Servidor IoT. Criar uma série de Smartdata é necessário antes de iniciar o processo de aquisição de dados. Cada sinal possui características únicas, e essas definições distinguem cada sinal na plataforma. O método de criação de uma série de Smartdata é então transmitido ao Servidor IoT, que interpreta o método para determinar quando a recepção dos conjuntos de dados deve começar e se os dados devem passar por um processo de pré-ingestão conhecido como Fluxo de Trabalho. Fluxos de trabalho podem ser utilizados para modificar, acumular e analisar dados, mas têm benefícios específicos no campo de aprendizado de máquina. Uma vez que cada série de Smardata foi criada, o Servidor IoT aguarda a chegada dos dados do sinal correspondente. Na Etapa 3, exportamos e preparamos os dados brutos armazenados no Servidor IoT para torná-los adequados à aplicação de técnicas de aprendizado de máquina. O conjunto de dados é analisado em um processo offline para garantir sua prontidão. Uma vez que obtemos resultados satisfatórios por meio de algoritmos de aprendizado de máquina para a detecção de falhas por batida de motor, integramos o modelo resultante ao Fluxo de Trabalho da plataforma. Na Etapa 4, utilizamos o modelo de aprendizado de máquina para detectar a ocorrência de falhas em tempo real, possibilitando ações proativas. Nesse contexto serão utilizados os algoritmos de aprendizado de maquina chamados Floresta Isolada, Máquinas de Vetores de Suporte, Autoencoder e Classificadores.

Resultados e Discussão

A Floresta Isolada é um algoritmo de detecção de anomalias projetado para identificar

outliers em conjuntos de dados, baseando-se no conceito de particionar aleatoriamente os dados em conjuntos menores. Os parâmetros chave incluem a contaminação, o número de estimadores, max features e max samples. Os experimentos resultaram em diferentes f1-scores, destacando a sensibilidade à configuração dos parâmetros. A contaminação otimizada para o conjunto B resultou no melhor desempenho, alcançando um f1-score de 0.31 (31%). Máquinas de Vetores de Suporte (SVMs) são usadas para classificação e regressão. Diversos experimentos foram conduzidos, variando o kernel. Resultados indicaram que kernels lineares e sigmoides tiveram desempenho insatisfatório, enquanto o kernel polinomial alcançou um f1-score de 0.48 (48%). Notavelmente, o kernel RBF apresentou dificuldades na detecção de padrões, com um f1-score de 0.07 (7%). Autoencoders densos e convolucionais foram explorados. Diferentes arquiteturas e limiares foram testados, com destaque para o Autoencoder Convolucional com tamanho de janela de 256, que alcançou um f1-score de 0.55 (55%). Os resultados sugerem que autoencoders são eficazes na extração de características, especialmente quando associados a janelas maiores. Um modelo de classificação (Arquitetura I) alcançou resultados promissores, especialmente com o conjunto D, obtendo um f1-score de 0.81 (81%). A escolha do conjunto de dados, tamanho do lote e comprimento da sequência afetou significativamente o desempenho. A precisão variou entre 3% e 80%, indicando um equilíbrio entre identificação precisa de instâncias positivas e minimização de falsos positivos. Os resultados foram resumidos, evidenciando que o classificador obteve o melhor desempenho, destacando a importância de variáveis baseadas no conhecimento. O Autoencoder Convolucional superou outros algoritmos, enquanto SVM e Floresta Isolada apresentaram desempenhos inferiores. As conclusões enfatizam a eficácia do classificador na detecção de *engine knock* e a necessidade de cuidadosa seleção de conjuntos de dados e configuração de parâmetros.

Considerações Finais

No geral, este estudo demonstra o potencial das técnicas de aprendizado de máquina para detecção de *engine knock* em motores de veículos automotivos. Enfatiza a importância de abordagens baseadas em dados para melhorar a segurança e o desempenho dos motores de combustão interna. As descobertas contribuem para os esforços contínuos de pesquisa na indústria automotiva e inspiram novos avanços nos sistemas de detecção de falhas.

Palavras-chave: Detecção de falha do motor. Aprendizado de Máquina. Problema de classificação. Batida de Pino. Detecção de anomalia.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

A2L	ASAM MCD-2 MC Language
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Networks
BPANN	BackPropagation Artificial Neural Network
CAD	Crank Angle Degree
CCP	CAN Calibration Protocol
CNN	Convolutional Neural Network
CWT	Continuous Wavelet Transformation
DCNN	Deep Convolutional Neural Network
DRNN	Deep Recurrent Neural Network
ECU	Engine Control Unit
EU	European Union
FDM	Fault Detection Method
GB	Gigabyte
IASE	Intelligent Acquisition and Analysis System for ECUs
ICE	Internal Combustion Engines
IoT	Internet of Things
KNN	K Nearest Neighbor
LISHA	Software/Hardware Integration Lab
LSTM	Long-Short Term Memory
MSCNN	MultiScale Convolutional Neural Network
MSE	Mean Squared Error
OBD	On-Board Diagnostics
RBF	Radial Basis Function
RF	Random Forest
SAE	Stacked Autoencoder
SI	Spark-Ignited
SVM	Support Vector Machines
XCP	Universal Measurement and Calibration Protocol

CONTENTS

1 INTRODUCTION

Numerous professionals in various fields of the automotive industry have conducted extensive research on internal combustion engines (ICE) due to their economic importance for improving fuel efficiency and complying with carbon emission regulations (KIRSANOVS; BARISA; SAFRONOVA, 2020; KOSENOK; BALYAKIN, 2020). Since its creation, the directive 2018/2001/EC has set a comprehensive policy to promote and increase the utilization of renewable energy sources in the European Union (EU) (EU PARLIAMENT, 2018). This initiative has gained acceptance with net-zero, reflecting a global movement towards renewable energy (EU PARLIAMENT, 2018). However, ICE still plays a significant role in today's transportation oil consumption, including both gasoline and diesel fuel used by cars, trucks, buses, and other vehicles. It accounted for approximately 60% of total oil consumption in 2021, and the consumption of oil for transportation is expected to continue to grow in the coming years, particularly in emerging economies ((IEA), 2021; KALGHATGI, 2018).

The automotive industry has come a long way since the early 1970s when the term "on-board diagnostics" (OBD) first emerged due to government regulations. Initially, it referred to simple diagnostic systems, but over time, it has evolved to include the vehicle's ability to self-diagnose and report issues (SANGHA et al., 2005). Nowadays, the industry has made significant efforts to increase self-diagnosis systems in vehicles, allowing the detection of faults, and efficient methods for acquiring data from electronic control units (ECUs) to conduct data-driven fault analyses.

With technological advancements, monitoring numerous mechanical and electrical systems in a vehicle's ECU has become possible. As a result, it is now possible to identify any anomaly or fault that might occur in various vehicle components, such as the engine, brakes, battery, and more. Nowadays, most cars have between 30 to 150 ECUs, so considering 150 ECUs with 100 signals, sampled at a rate of 4 milliseconds over 60 minutes, generates 22.5 million data points (NAIR; KOUSTUBH, 2017). It might not seem much, but consider analyzing a car during 10 hours of running tests. It would generate 1.8 GB per car and day. While it is crucial to analyze this massive amount of data to ensure the safety-critical systems of the vehicle, normal vehicle testing may take hours, making it challenging to process and analyze the data efficiently.

Among the faults that can occur in modern combustion engines, one of the most common is engine knock. This phenomenon, also known as detonation or knocking noise, is an abnormal combustion process in an internal combustion engine. It happens when the air/fuel mixture in the engine's cylinder explodes instead of burning smoothly, causing a knocking or pinging sound. This can cause damage to the engine if left untreated. One of the critical challenges faced by automobile manufacturers dealing with the engine knock is to reduce maintenance costs and prolong engine life while

simultaneously increasing efficiency (SAMIMY; RIZZONI, 1996).

As a result of the severity of this event, its detection is approached in two different manners: inside or outside the combustion chamber. The most effective way to measure the detection inside the combustion chamber is by using a pressure sensor, which provides the best signal for analysis. However, the cost of the sensor is a significant obstacle to using this approach on a large scale (HORNER, 1995). On the other hand, measurements can be taken outside of the combustion chamber, typically through the use of a vibration sensor attached to the engine block. This sensor is often integrated into the engine's design and connected directly to the engine ECU. In this approach, applied in most of today's vehicles, the ECU detects the knocking noise phenomenon by analyzing the vibration signal and makes corrections in the engine calibration to retard the spark advance (HEINZ; HANS-ULRICH, 1985).

Some of the methods to detect engine knock in vehicles include listening for the knocking sound, using a knock sensor, and analyzing engine data (HEINZ; HANS-ULRICH, 1985). While listening for the knocking sound is simple, it may not always be reliable. The use of a knock sensor is highly reliable and accurate while analyzing engine data is highly accurate but requires specialized equipment and expertise. The use of machine learning for fault detection, including engine knock, have been highlighted due to its data-driven approaches (ZHENG et al., 2020). Among the various machine learning techniques, Artificial Neural Networks (ANNs), particularly Convolutional Neural Networks (CNNs), have been highlighted for their effectiveness (AHMED et al., 2015; LUJÁN et al., 2017; YAN; YU, 2019).

The automotive industry has mainly adopted vibration sensors in this scenario, which enables the usage of signal processing applications like Fast Fourier Transform (FFT), a mathematical algorithm with excellent detection results. However, any noise in the same frequency can affect it, leading to the spark advance process of the engine being triggered. If the detection is incorrect, the engine's spark timing may become excessively advanced, resulting in engine knock. This highlights the importance of properly detecting this fault (HEINZ; HANS-ULRICH, 1985).

By harnessing the power of a data-driven approach and leveraging the internal variables controlled by the ECU, it becomes feasible to monitor a wide range of sensor data. These sensor measurements can provide valuable insights into the occurrence of various phenomena, including engine knock. The ECU collects data from sensors positioned throughout the vehicle, such as the knock sensor, which is specifically designed to detect abnormal combustion processes. By extracting and analyzing these sensor readings, it becomes possible to identify potential faults or anomalies related to engine knock. Using ECU data for analysis enables a thorough examination of critical variables and facilitates the early detection of engine faults.

Applying machine learning techniques is crucial for implementing security mea-

sures and containing problems by detecting fault events. Therefore, it plays a significant role in enhancing the safety of vehicle users and society. This dissertation aims to develop an appropriate process using machine learning techniques, to detect engine knocks in an automotive vehicle through the data collected from the engine ECU. Considering other aspects that may affect the fault, such as pressure, temperature, speed, and vibration signal processed by the engine' ECU with its results detection to ensure it is an actual occurrence.

1.1 OBJECTIVES

The main objective of this work is to develop machine learning models and evaluate them using a data-driven method for achieving the highest performance on engine knock detection in vehicles with internal combustion engines using low-frequency data extracted directly from the Engine Control Unit (ECU).

To achieve the main objective, the following specific objectives were defined:

1. **Algorithm Selection:** Identify and choose the most prevalent algorithms for fault detection in automotive systems.
2. **Variable Set Formation:** Construct distinct sets of variables to systematically assess their contributions to fault detection results.
3. **Data Preparation Analysis:** Investigate various facets of data preprocessing and their impacts on fault detection performance.
4. **Model Performance Evaluation:** Evaluate the performance of different models and select the one exhibiting the highest level of accuracy in fault detection.

2 BASIC CONCEPTS

In the following sections, we present some basic concepts about fault detection and diagnosis, as well as the engine knock phenomenon.

2.1 FAULT DETECTION AND DIAGNOSIS

To effectively discuss methods for detecting and diagnosing faults, it is essential to understand what constitutes a fault in a computing system. A fault is any deviation in a system's behavior from its expected or intended performance. Various factors, including component failures, shifts in operating conditions, or environmental disturbances, can cause such deviations. It is also important to distinguish between faults and anomalies, as anomalies refer to deviations from normal behavior that are not necessarily indicative of a fault (GAO; CECATI; DING, Steven X, 2015a). An error is a human action or decision that produces an incorrect or unexpected result. Thus, an error made during the design of a system might introduce a fault in its functioning, causing the system to fail.

The field of study concerned with identifying and diagnosing problems or faults in complex systems, such as industrial plants (QIN, 2009), aircraft (LU et al., 2016), and automotive vehicles (JAFARIAN; DARJANI; HONARKAR, 2016), is called fault detection and diagnosis (FDD). The idea is to ensure that a system operates safely and efficiently by detecting faults early and accurately. It involves two main steps: fault detection and fault diagnosis. According to Park, Fan, and Hsu (2020), fault detection is the process of identifying when a fault has occurred, while fault diagnosis is the process of identifying the root cause of the fault.

Various techniques are used in FDD, including the traditional model-based and signal-based approachs (GAO; CECATI; DING, Steven X, 2015a), as well as data-driven (YIN et al., 2014), and hybrid methods (GAO; CECATI; DING, Steven X., 2015b). Model-based methods use mathematical models of the system to simulate its behavior and detect faults by comparing the model output to the actual system output. The signal-based methods, on the other hand, rely on the analysis of sensor signals to detect and diagnose faults. This method uses signal processing techniques such as Fourier analysis, wavelet analysis, and statistical methods to analyze the time-domain or frequency-domain characteristics of the signals (ABID; KHAN; IQBAL, 2021).

Data-driven models use statistical and machine learning algorithms to analyze historical data from the monitored system and learn patterns associated with different fault conditions. These approaches do not rely on knowledge of the system's physical behavior and model. They can handle intricate and nonlinear connections between input and output variables. Data-driven models can also identify hidden relationships and patterns that may not be immediately apparent. According to Qin (2009), combining

methods from different approaches, known as hybrid methods, is a way to achieve better performance and efficiency.

Multiple models can be applied depending on the application and its complexity. As previously mentioned, the automotive industry has adopted some of those techniques to increase the safety and reliability of vehicles by detecting and diagnosing faults in various components and systems. These processes of detection are crucial because early detection of faults can help prevent serious and costly problems and improve vehicle safety and reliability.

According to Saibannavar, Math, and Kulkarni (2021), the standard methods to fault detection in vehicles is through the use of OnBoard Diagnostics (OBD) systems. OBD systems are designed to continuously monitor various vehicle parameters such as engine speed, coolant temperature, oxygen sensor readings, and other relevant signals. These systems use data collected from sensors and other sources to detect and diagnose faults in real-time. The OBD system generates Diagnostic Trouble Codes (DTCs) that are used to identify the source of the fault. These codes are then read by a technician using a diagnostic scan tool, which provides information about the nature of the fault and its location in the vehicle system.

The use of OBD systems has significantly improved the reliability and safety of vehicles by detecting faults at an early stage and providing a warning to the driver to take necessary corrective action. However, this process is still manual and dependent on tools to check for fault occurrence. Advanced systems are currently in development, using machine learning algorithms to detect faults more accurately. These advanced systems have been highlighted by improving fault detection efficiency and accuracy, enhancing vehicle safety (JAFARIAN; DARJANI; HONARKAR, 2016).

While using OBD systems has significantly enhanced the reliability and safety of vehicles by detecting faults in real time, it is limited in its ability to read data from the ECU at high frequencies. Our proposed approach aims to overcome this limitation by studying different architectures and models to evaluate engine fault detection using low-frequency data from the ECU. By employing a data-driven method, we leverage the power of statistical and machine learning algorithms to analyze historical data and identify patterns associated with different fault conditions. This approach offers several advantages, including handling intricate and nonlinear relationships between variables, uncovering hidden patterns, and improving fault detection efficiency and accuracy. By developing and implementing this method, we aim to ultimately contribute to advancing fault detection and diagnosis techniques in the automotive industry.

2.1.1 Machine Learning in Automotive Applications

Fault detection on automotive vehicles using machine learning has gained significant attention in recent years due to its ability to detect faults automatically in real

time, improving vehicle reliability. Machine learning algorithms such as artificial neural networks, deep learning, and support vector machines have been applied to various vehicle subsystems, including the engine, transmission, suspension, and braking system, to detect faults (INCE et al., 2016; VIJAYAN et al., 2022).

According to Mitchell (1997), machine learning is a subset of artificial intelligence that enables systems to learn and improve from experience without being explicitly programmed. It can be broadly classified into three paradigms: supervised, unsupervised, and reinforcement learning (ALPAYDIN, 2010). However, in this context, the focus is on the first two paradigms - supervised and unsupervised learning.

Supervised learning involves using labeled data to train a model to classify new, unlabeled data (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). In this paradigm, the model learns to map input data to output labels based on training examples. A common problem that can be solved by supervised learning is classification. For example, given a dataset of labeled images of animals, a supervised learning algorithm can be trained to classify new images of animals as either dogs, cats, or horses based on patterns and features learned from the training data. Common algorithms used in supervised learning include decision trees, support vector machines (SVM), and artificial neural networks (ANNs).

On the other hand, unsupervised learning involves using unlabeled data to identify patterns and relationships within the data (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). In this paradigm, the model learns to cluster (group) similar data points or discover underlying structures in the data. A common problem that can be solved by unsupervised learning is anomaly detection. For example, given a dataset with mostly normal instances, a small number of anomalous instances may differ significantly from the normal ones. Unsupervised learning algorithms can detect these anomalies by identifying patterns or structures in the data. Common algorithms used in unsupervised learning include k-means clustering, principal component analysis, and self-organizing maps.

One of the main advantages of using machine learning for fault detection is its ability to learn from past data and adapt to new situations, making it effective in detecting faults that may not be easily identified through traditional methods. For example, in the case of engine knock, an abnormal combustion process that occurs in an internal combustion engine, machine learning algorithms can be trained to detect anomaly patterns that can be associated with knock and alert the driver or initiate corrective actions before significant damage occurs.

Moreover, machine learning-based fault detection systems can also analyze large amounts of data in real-time, allowing for faster and more accurate detection and diagnosis of faults. These systems can also reduce the need for manual inspections and routine maintenance, leading to cost savings for vehicle owners and manufacturers.

In summary, to deploy our proposed approach, both supervised learning and unsupervised algorithms will be applied. The supervised algorithms will learn from labeled data to classify instances, while the unsupervised techniques will help identify anomalous patterns that may indicate the presence of engine knock, which later will be evaluated with labeled data. This comprehensive approach will enhance the robustness and effectiveness of our proposed method for engine knock detection.

2.1.2 Machine Learning Algorithm Details

This subsection provides an overview of both supervised and unsupervised machine learning techniques. The strengths and weaknesses of these techniques are highlighted, along with their practical applications. Decision trees, SVM, and ANNs are popular supervised learning techniques extensively studied and applied in various fields. Nevertheless, autoencoders and isolation forests are the ones popularly known for unsupervised learning. These techniques are used for anomaly detection and data compression (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Decision Trees are a type of supervised machine learning technique that utilizes a tree-like model to arrive at decisions. They can handle categorical and continuous data, making them simple and easy to interpret. The advantage of Decision Trees lies in their ability to handle both linear and nonlinear data while remaining robust to noisy data and outliers. They have found applications in various fields, including medicine, finance, and engineering. However, overfitting is a common challenge with Decision Trees, which occurs when the model becomes too complex and memorizes the training data instead of generalizing it to new data. The model can be pruned to combat overfitting, or ensemble methods such as Random Forests can be utilized (QUINLAN, 1986; BREIMAN et al., 1984).

Support Vector Machines (SVMs) is another popular algorithm for supervised learning. The use of this algorithm is common for classification and regression tasks on supervised machine learning techniques. The primary objective of SVMs is to find the optimal hyperplane that can divide the data points into different classes with maximum margin. SVMs can handle both linear and nonlinear data by transforming the data into a higher-dimensional space using kernel functions. SVMs have been widely used in various fields, such as biology, finance, and image recognition, for tasks such as predicting protein-protein interactions, classifying credit card transactions, and recognizing handwritten digits. However, SVMs can be sensitive to the choice of hyperparameters, such as the kernel function and regularization parameter, which can affect the model's performance. Additionally, SVMs can be computationally expensive, especially for large datasets.

Artificial Neural Networks (ANNs) are a supervised machine learning technique that mimics the structure and function of biological neural networks. ANNs consist

of interconnected nodes organized in multiple layers. Each node performs a weighted sum of its inputs, applies an activation function, and passes the output to the next layer. ANNs can handle linear and nonlinear data and learn complex patterns and relationships. They have found applications in diverse fields such as image recognition, natural language processing, and robotics, such as recognizing faces, translating languages, and controlling robots (LECUN; BENGIO; HINTON, 2015; CHO et al., 2014; LEVINE et al., 2016). However, ANNs are prone to overfitting and require large datasets for training. They can also be challenging to interpret, and it may not be evident how the model reached its decision.

Autoencoders are unsupervised machine learning models that aim to learn a compressed representation of the input data. The model consists of an encoder that maps the input data to a compressed representation and a decoder that maps the compressed representation back to the original input data. The goal of the autoencoder is to minimize the difference between the input data and the reconstructed data, known as the reconstruction error. Autoencoders can be used for anomaly detection because anomalies will likely have a higher reconstruction error than normal data. They have been applied in various fields, such as computer vision, natural language processing, and cybersecurity. However, autoencoders can suffer from overfitting, require a large amount of data for training, and can be difficult to interpret.

Isolation Forests, similarly is a technique used for anomaly detection that utilizes random forests to isolate anomalies from normal data (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). It generates a tree structure for each data point in which each node signifies a feature or attribute, and each branch represents a possible value or outcome. The isolation forest then measures the path length that is required to separate the data point from the rest of the data, with anomalies being likely to have a shorter path length compared to normal data. Isolation Forests have been widely applied in different fields, including finance, cybersecurity, and fraud detection. It has been applied to identify fraudulent credit card transactions, predict stock prices, and detect network intrusions. However, Isolation Forests can be affected by the curse of dimensionality, and their performance might decrease as the number of dimensions increases. Additionally, the choice of hyperparameters, such as the number of trees in the forest and the sub-sampling size, may impact the accuracy of the technique.

2.1.3 Machine Learning Algorithms Evaluation

Several metrics are commonly used for evaluating machine learning algorithms. Prior to understanding the metrics, it is essential to understand the confusion matrix, a widely used tool for evaluating the performance of a machine learning algorithm in a classification problem. The matrix presents *TP* (true positive) is correctly classified as positive samples, *FP* (false positive) is misclassified as positive samples, *TN* (true

negative) is correctly classified as negative samples, and *FN* (false negative) is misclassified as negative samples as depicted in Figure 1. It provides a detailed breakdown of the model's predictions, allowing us to analyze its strengths and weaknesses.

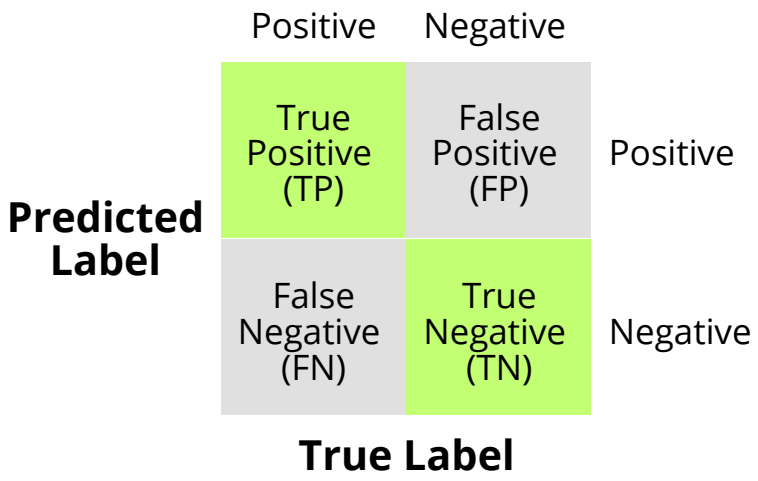

Figure 1 – Confusion Matrix.

In the context of a classification problem and its associated confusion matrix, the term 'predicted label' can sometimes lead to confusion. In a classification problem, we aim to assign instances to specific classes or categories based on the features or attributes of those instances. For instance, in a classification problem to detect if an image has a cat or a dog, the 'predicted label' is the model's result for choosing a cat or dog. The confusion matrix will evaluate the model's performance for detecting cats as cats and dogs as dogs. So, the predicted labels represent the results generated by ML model itself, the model's predictions. It does not refer to the process of foreseeing actions, at least not in these cases presented.

The confusion matrix can be used to compute various performance metrics, such as accuracy, precision, recall, and f1 score, as well as to visualize the performance of the algorithm. Models are evaluated using these metrics mentioned has generally range from zero to one or from zero to one hundred percent. Higher values of the metrics indicate better performance of the model in classifying the target variable.

Accuracy is the proportion of correct classification out of the total classifications made. It is a simple and commonly used metric for classification problems, and it is computed using the Equation (1). Precision is the proportion of true positives (correctly classified positive instances) out of total classified positives, and it is computed using the Equation (2). It is useful when the focus is on minimizing false positives.

$$accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{1}$$

$$precision = \frac{TP}{(TP + FP)} \tag{2}$$

Recall is the proportion of true positives out of total actual positives, calculated using the Equation (3). It is useful when the focus is on minimizing false negatives. Finally, the f1-score is the harmonic mean of precision and recall, which balances both metrics. It is useful when both false positives and false negatives are important, calculated using the Equation (4).

$$recall = \frac{TP}{(TP + FN)} \tag{3}$$

$$f1score = 2 * \frac{precision * recall}{precision + recall} \tag{4}$$

Mean Squared Error (MSE) is another technique frequently employed to assess the effectiveness of machine learning algorithms, particularly in the context of regression problems. It is calculated by taking the average of the squared differences between the predicted and actual values across all data points. Equation (5) presents the formula where n is the number of data points, y_i is the actual value of the i-th data point, and $\hat{y}_i$ is the predicted value of the i-th data point. Lower values of MSE indicate better performance of the model in predicting the target variable.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

The choice of evaluation metrics is crucial in assessing the performance of a machine learning model, as it helps to identify whether the model is overfitting or underfitting the data. Commonly used evaluation metrics, such as accuracy, precision, recall, F1 score, and mean squared error, can help to identify overfitting and underfitting.

Overfitting occurs when a model is too complex and is trained too well on the training data, resulting in a low error rate on the training data but a high error rate on the testing data (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). This happens because the model is too specific to the training data and captures noise and randomness in the data, leading to poor generalization to new data. Overfitting can be avoided by simplifying the model, reducing the number of features, or adding regularization techniques such as Lasso or Ridge regression.

Underfitting, on the other hand, occurs when a model is too simple and cannot capture the underlying patterns in the data (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). This results in a high error rate on both the training and testing data. Underfitting can be avoided by increasing the complexity of the model, adding more features, or using a different algorithm that can better capture the patterns in the data.

To determine if a model is overfitting or underfitting, it is important to evaluate its performance on a separate testing dataset. For instance, high accuracy and precision on the training data but lower accuracy and precision on the test data may indicate overfitting, while low accuracy and precision on both training and test data may indicate underfitting. Cross-validation is another technique used to evaluate the performance of a model and can help identify if the model is overfitting or underfitting.

In the context of this study, the primary goal is to achieve a high f1-score for evaluating the employed algorithms. Maximizing the f1-score allows for assessing the algorithms' effectiveness in accurately identifying and classifying instances, providing valuable insights into their overall performance and classifying capabilities. Furthermore, application of cross-validation techniques will be employed to ensure the robustness and generalizability of the algorithms' performance.

2.2 INTELLIGENT ACQUISITION AND ANALYSIS SYSTEM FOR ECU

LISHA (Software/Hardware Integration Lab), a research lab focused on innovative techniques and tools for developing embedded systems, has collaborated with Renault to develop the Intelligent Acquisition and Analysis System for ECUs (IASE) in the realm of Program Rota 2030. The foundation of a data acquisition system that collects data from a car's ECU is connected to the concept of the Internet of Things (IoT). IoT refers to the network of physical devices, vehicles, buildings, and other objects embedded with sensors, software, and network connectivity that enable them to collect and exchange data. The concept of IoT is the idea that everyday objects can be connected to the internet and communicate to create a smart, interconnected world that improves efficiency, convenience, and quality of life.

The IASE project explored the feasibility of utilizing AI techniques in the context of the IoT to enhance the functioning of Internal Combustion Engines. The primary focus is optimizing controller parameter calibration and detecting anomalies. IASE hardware, depicted in Figure 2, is attached to the vehicle's ECU and can communicate to the IoT Server, storing its datapoints. More details about the communication models (SmartData) and its process is presented in the following subsection.

The hardware components of this system include a CAN interface for data acquisition, a memory capacity of 16GB, and data upload via 4G cellular network. There are LED status indicators, a Bluetooth Low Energy connection, Ethernet and USB connections, and a power conditioning circuit. Additionally, the system has 4GB RAM memory, a Zynq Ultrascale processor with a Quadcore ARM Cortex A53 and Dualcore ARM Cortex R5 (BEDRETCHUK et al., 2023). This robust hardware demonstrates remarkable performance efficiency, operating consistently at low CPU percentages. This highlights the system's power and minimizes the overhead, ensuring optimal utilization of computational resources.

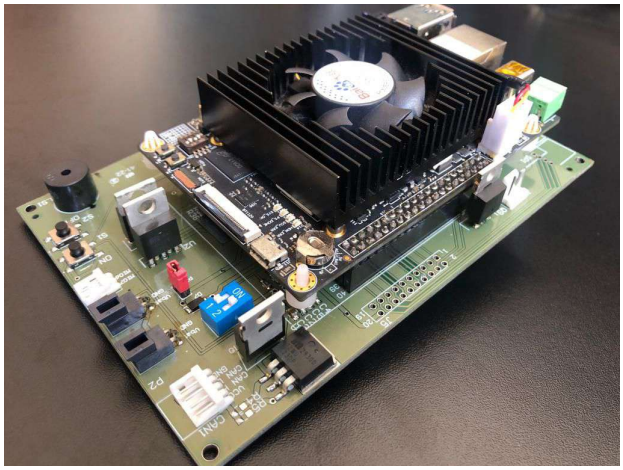

Figure 2 – IASE Hardware

Source: https://lisha.ufsc.br/IASE-public

On the software side, the system allows for CAN protocol selection (XCP / CCP) and a customizable list of variables. Communication is possible over 4G and offline operation is also available. The system also has more than 1 day data backup, an Android UI via Bluetooth, live status updates in LEDs and UI, and safe communication with servers using SSL certificates. Overall, this system offers a range of hardware and software features for data acquisition, communication, and processing, making it a powerful tool for a variety of applications.

2.2.1 SmartData Communication

An overview map of the interactions of IoT Device and IoT Server is presented in the Figure 3. The communication from the IASE Hardware (IoT Device) and the LISHA's IoT Platform (IoT Server) is based on SmartData. SmartData is a high-level API for wireless sensor networks (WSN) that provides a common way to access sensed data while incorporating metadata such as semantics, spatial location, timing, and trustfulness (FRÖHLICH, 2018). The data is marked with a 32-bit type identifier that indicates whether it represents an SI Physical Quantity or Digital data. From the communication protocols perspective the SmartData has this caracteristcs presented in Table 1, which allow the identification of geolocalization, unit and value of each datapoint.

SmartData acts as the sole application-visible entity in the platform, serving as a mediator for all system services, including communication, synchronization, and interaction with transducers and actuators. This method is a key component of the Internet of Things (IoT) ecosystem, where devices and sensors generate large amounts of data that need to be processed and analyzed in real-time to extract meaningful insights.

This architecture was designed to be able to execute some tasks in the ingestion process, and it was defined as Workflow. Each Workflow can be implemented according

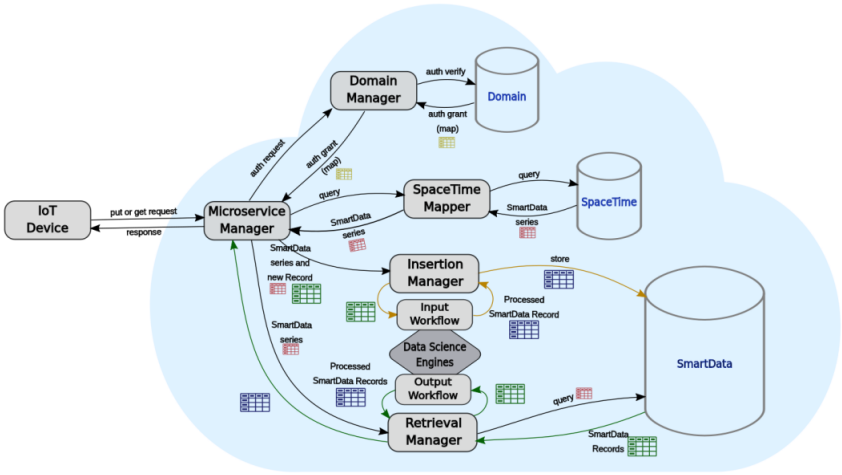

Figure 3 – Overview of LISHA's IoT Platform.

Table 1 – SmartData encapsulated in a network packet

Origin (x, y, z, t)	Unit	Value	Expiry	MAC

Source: (FRÖHLICH, 2018).

to its necessity, and it runs before the storage (input Workflow) of the data or after storage (output Workflow). This process allows the use of advanced techniques such as machine learning, data mining, and artificial intelligence to analyze large data sets in real time. The fault detection in real-time of the proposed approach has been applied with input Workflow.

2.3 ENGINE KNOCK

The engine knock or knock noise is a severe issue that car manufacturers have been addressing. This phenomenon can negatively affect your car's performance, fuel efficiency, and overall lifespan. Many researchers have studied engine knocking, focusing on developing a way to prevent engine knocking (FIOLKA, 2006; PANZANI; ÖSTMAN; ONDER, 2017). The effects of its occurrence have been monitored, aiming for the engine to have a lower maintenance cost and a longer life. Efficiency improvements, reducing noise, significantly reducing pollution emissions, and extending engine life are all goals to achieve (SAMIMY; RIZZONI, 1996).

The normal scenario of combustion in an internal combustion engine involves a controlled and timed ignition of the air-fuel mixture in the combustion chamber. During this process, the spark plug ignites the air-fuel mixture, causing it to burn smoothly and release energy that pushes the piston down and rotates the crankshaft. This process is repeated in each cylinder of the engine, resulting in the smooth operation of the engine. However, if the air-fuel mixture explodes instead of burning smoothly, as in the case of engine knock, it can cause high-pressure waves which they are reflected inside the

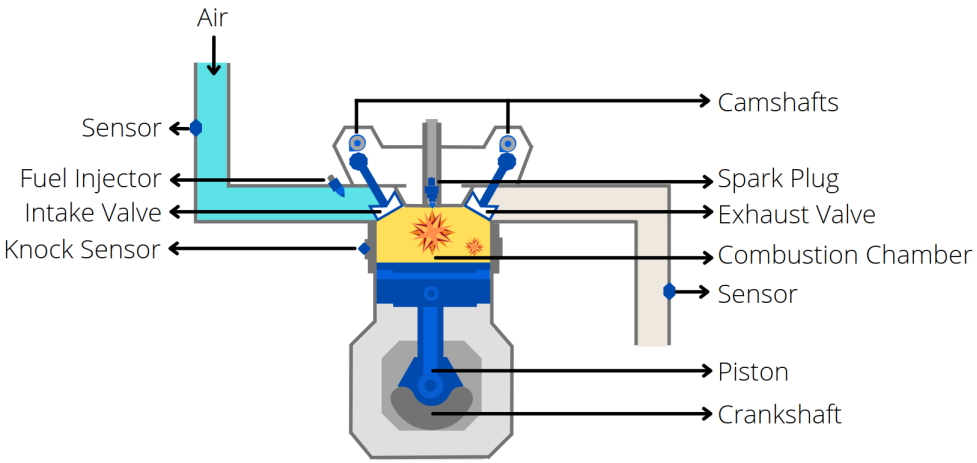

Figure 4 – Engine Knock Occurrency.

chamber causes the engine knock, commonly known as *knocking noise* (DOORNBOS et al., 2018).

Engine knock is a phenomenon that occurs when the air-fuel mixture in an internal combustion engine detonates spontaneously and uncontrollably, causing vibrations and knocking sounds and generating a high-frequency shock wave as depicted in Figure 4 (FIOLKA, 2006; BOUBAI, 2000). Overall, knock is associated with high-frequency pressure oscillations, acoustic signature, and mechanical vibrations in the range of 6-13 kHz, with most studies finding significant frequency content in the range of 6-8 kHz (SAMIMY; RIZZONI, 1996; DOORNBOS et al., 2018; HORNER, 1995).

To prevent the occurrence of the knock is necessary to reduce pressure in-cylinder. To reduce the pressure, the spark timing is postponed. However, the spark timing delay reduce the expansion work, responsible for moving the piston down, leading to lower engine efficiency (MAHENDAR, 2021).

2.3.1 Causes

Engine knock, also known as detonation, can have several causes that can be harmful to the engine if not addressed. There are several common causes of engine knock, including using low octane fuel, a lean air/fuel mixture, bad timing, a faulty knock sensor, high intake temperature, and high loads on the engine.

The first aspect that affects the occurrence of engine knock is using low-octane fuel. This type of fuel is less resistant to engine knock, while high-octane fuel is better suited for high-performance engines that operate under higher compression ratios, such as those found in sports cars or racing vehicles. Moreover, in lean conditions, the air/fuel mixture has a higher proportion of air relative to fuel, which can increase the likelihood of knocking or misfiring (DOORNBOS et al., 2018).

Any malfunctions on the electronic control unit (ECU) can lead to several problems on the vehicle. Thus, bad timing or problems with sensors that affect the combus-

tion process can potentially lead to engine knock, as it can cause the air-fuel mixture to ignite too early or too late, creating pressure waves that result in knock.

According to Motahari and Chitsaz (2019), high temperature can intensify engine knock, which leads to a reduction in engine efficiency. As the intake air temperature increases, the probability of engine knock also increases. To prevent knock, the spark time is retarded, which further reduces engine efficiency. Additionally, high temperatures can also affect the thermal efficiency of the engine, which can result in decreased fuel economy and power output. To prevent engine knock and keep the engine running smoothly, it is important to address any of these common causes promptly.

Another aspect that leads to engine knock is high loads on the engine (MAHENDAR, 2021). This is because high loads require higher pressure and temperature in the combustion chamber, which can cause auto-ignition of the fuel mixture before the spark plug fires, resulting in knocking noise.

2.3.2 Symptoms

Engine knock is characterized by a distinct knocking or ticking sound emanating from the engine. This sound is caused by the collision of the flame front with the end gases in the combustion chamber, leading to high-frequency pressure oscillations. These oscillations create a distinct high-pitched metallic knocking or ticking sound that can be heard from the engine. In some cases, the sound can be mistaken for valve train noise, but the characteristic frequency and pattern of the sound are different from valve train noise (HEYWOOD, 1988).

In addition to the noise, the vehicle may experience a decrease in power and acceleration and an increase in fuel consumption (HEYWOOD, 1988). Furthermore, engine knock is also known to result in a substantial increase in carbon dioxide emissions (BOUBAI, 2000). According to Heywood (1988), engine knock can lead to a range of engine problems, including piston damage, connecting rod damage, head gasket failure, and valve damage. Therefore, it is important to address engine knock as soon as possible to prevent further damage to the engine.

2.3.3 Detection

Detecting engine knock is an essential task in the field of internal combustion engines, as it can provide vital information about the health and performance of the engine. Engine knock occurs when the air-fuel mixture in the combustion chamber detonates instead of burning smoothly, as mentioned in the previous subsections, causing a sudden pressure spike that can damage engine components over time.

Various methods have been used to detect engine knock, including in-cylinder pressure measurements, vibration analysis, and machine learning techniques. In-cylinder pressure measurements involve analyzing the pressure waveform to determine the

knock intensity (PANZANI; ÖSTMAN; ONDER, 2017). Vibration analysis is a non-intrusive technique that involves attaching a vibration sensor to the engine block to detect knock events (SAMIMY; RIZZONI, 1996). Machine learning techniques involve analyzing (PETRUCCI et al., 2020). These methods have different levels of invasiveness and require different types of signal processing devices.

In-cylinder pressure measurements involve using a sensor installed inside the engine to directly measure the pressure inside the combustion chamber during operation (KEFALAS et al., 2021). This method provides a highly accurate measurement of knock intensity and is considered the most direct method for detecting knock. Nevertheless, it also requires an invasive installation of a pressure sensor in the engine, which can be expensive and challenging to implement on a large scale. Therefore, in-cylinder pressure measurements are mostly limited to research and development purposes or high-performance engines where the cost and complexity of installation can be justified.

Vibration analysis is a non-intrusive method that can detect knock events by analyzing the vibration signal of an engine block (HORNER, 1995). The advantage of this method is that it does not require intrusive modifications and is relatively cheaper, and then, broadly used. These method use sensors that in a higher speed scenario can get difficulties detect the occurrence, because engine knock frequency range runs below $20kHz$. However, compared to in-cylinder pressure measurements, vibration analysis may not provide precise measurements of knock intensity. Furthermore, the accuracy of the vibration signal may be impacted by background noise and other sources of vibration, making it more challenging to differentiate knock events from other sources of vibrations.

However, recent advances in machine learning have shown promise in detecting engine knock using more straightforward, non-intrusive methods. A recent study by Petrucci et al. (2020) and Falcão, Barros, and Melo (2019) have highlighted the benefits of using machine learning techniques for engine knock. Specifically, they emphasize that machine learning can provide accurate and fast engine knock detection, allowing for effective knock control in real time. This effective knock detection is achieved by training the machine learning models using large datasets of engine signals and corresponding knock occurrences, allowing the models to detect knock patterns that are difficult for humans to identify.

Modern engines use various techniques to prevent knock, such as adjusting the fuel injection timing, modifying the air-fuel ratio, and increasing the compression ratio. While each method has its strengths and weaknesses, many combinations have been studied to allow for detecting and addressing engine knock issues. With accurate and effective detection methods, engine knock can be addressed quickly and effectively, leading to improved engine performance and reduced risk of damage or failure.

In conclusion, these advancements in machine learning offer promising opportu-

nities for non-intrusive and efficient knock detection, enabling real-time knock control. By effectively addressing engine knock through precise detection and implementing preventive measures, automotive engineers can ensure improved engine performance, reduced emissions, and enhanced reliability for a smoother driving experience and extended engine life.

3 RELATED WORK

In this section, we examine the existing literature on Fault Detection, focusing on various aspects to gain a comprehensive understanding of the techniques employed and their distinguishing features. To gain a better understanding of engine faults and engine knock, it is important to explore the essential techniques utilized within these fields.

3.1 DETECTION AND CLASSIFICATION OF FAULTS

Fault detection and classification are crucial in diagnosing faults in complex systems. Vibration analysis is commonly used for fault diagnosis and involves measuring and analyzing machine vibrations. The process includes data pre-processing, feature extraction, and pattern classification. Traditional methods of selecting relevant features can be time-consuming and prone to error. However, CNNs can learn features automatically from raw data, reducing the risk of overlooking important features and identifying complex patterns. This makes CNNs a powerful tool in fault diagnosis, especially when traditional feature extraction methods are limited or poorly understood.

Pan et al. (2018) presented an improved method that combined one-directional CNN and Long-Short Term Memory (LSTM) to enhance fault detection performance in a vibrational analysis process. The CNN was designed with convolutional and pooling layers to extract relevant features from the input data. The experiment was conducted with electro-discharge machining, where signals were acquired under four conditions: normal, ball fault, inner race fault, and outer race fault. The experiment had a sampling frequency of 48 kHz, a sampling time of 10 seconds, and a rotation speed of 1725 r/min. A comparison was made between different batch sizes, where a batch size of 80 with a learning rate of 0.004 achieved 100% of accuracy. Even with a range of batch sizes from 20 to 100 and learning rates from 0.0005 to 0.006, the overall accuracy remained around 99%. The proposed approach outperformed other deep learning models such as CNN, LSTM, and DCNN with an accuracy of 100%.

Zilong and Wei (2018) aims to apply the vibration signals to detect a fault on the rolling element bearing. Considering that a convolutional neural network has proven to be the most important method operating directly with raw vibration signals, it was defined to the base of CNNs to solve this problem. The experiment acquired 12.000 samples per second, considering the rotation speed of 1797. The classification process had 10.000 samples, where 9/10 was used for training and 1/10 samples for testing. Three study cases were analyzed, considering 1-dimensional DCNN, 2-dimensional DCNN, and MSCNN. Even though they are similar, they have significantly different organizational structures. As the result of the training process, after 100 epochs achieved 98.57%, 98.25%, and 99.27%.

The key components of rotating machinery are rolling element bearings and gears, which their health states affect the performance and reliability of the machinery. Traditional methods of feature extraction are generated manually, suiting a specific issue. The benefits of using ANN is capable of learning complex non-linear relationships on faults diagnosis with the much bigger interrelationship between faults. Having that in mind, the ability to detect the fault in the machinery is a very important study by Jia et al. (2016), which proposes a method using SAE with N-hidden-layer. Considering the maximum training epochs of 100, the learning rate of 0.05, momentum of 0.05, with the activation function hyperbolic tangent. The acquisition process to get the vibrational signal had a sampling frequency of 5.12 kHz, and considering four stages of engine speed: 2100 rpm, 2400 rpm, 2700 rpm, and 3000 rpm. There were 203 signals for each experiment, leading to 2560 data points. Training and testing were done using 50% of the samples, and each one of the experiments had higher accuracy than the referenced ones. The dataset A, B, and C achieved the classification accuracies of 99.95%, 99.61%, and 99.74%.

Hongkai Jiang et al. (2018) proposed an intelligent system for fault diagnosis of rolling bearings using a deep recurrent neural network (DRNN). Using frequency spectrum sequences as input allowed a reduction of the size of the inputs. Thus, DRNN architecture uses multiple layers of recurrent hidden units, which work together to automatically extract relevant features from input spectrum sequences. The experiments were conducted under 1750 rpm and 1797 rpm captured by an accelerometer where the sampling rate was 12 kHz. There are twelve operation conditions, one normal, and eleven fault conditions, where three types of faults are defined: ball faults, inner race faults, and outer race faults. As a result of the experiment without manual feature extraction and feature selection (experiments 1 and 2), the proposed approach achieved an average testing accuracy of 94.75%.

In their study, Krummenacher et al. (2018) proposed a real-world application of using two machine learning methods to automatically detect defects on railway wagon wheels. They used a vertical force measurement technique in a multi-sensor structure and applied multiple instance learning and shifted invariant networks to classify defects. The proposed solution used a CNN with a two-dimensional representation for wheel defect classification, which was found to be more effective than a Wavelet-SVM. The study used two datasets, one for calibration and another for testing, which had unbalanced class proportions. The authors randomly over-sampled the smaller class to balance the data. The proposed model achieved an accuracy of 92% for the first dataset and between 87% for the second dataset, with each target class being accurately detected.

The vibrational analysis can be applied to many scenarios, for instance, according to Guoqian Jiang et al. (2019) who proposed a fault diagnosis method on wind turbine gearbox where no additional signal processing is applied only using a mul-

tiscale convolutional neural network (MSCNN). The measurement process is made in raw vibrational signals that are characterized by nonlinearity and nonstationarity due to speed, loads, and environmental noise. The base of the proposed solution is a traditional CNN but provides a more complex way to extract faults in a multiscale paradigm. The model was trained offline using a backpropagation algorithm to optimize its parameters. Additionally, to the measurement process, the vibrational signals were acquired using a piezoelectric with a sampling rate of 10 kHz and used a 10-fold cross-validation method. The split process was done and divided the subset of 20800 samples into 10 equal-sized subsets, where 1/10 was used for testing, and 9/10 was used for training. It was compared between traditional CNN and MSCNN, where the CNNs performed poorly compared to the proposed method, implying that happened a mismatch in noisy data because of its single scale. Increasing the number of scales presents a significantly higher accuracy showing that the proposed method can learn from robust features, but eventually leading to slightly larger training and testing phases compared to the traditional method. When compared to other traditional multiscale approaches like multiscale entropy (MSE), wavelet package decomposition (WPD), and empirical mode decomposition (EMD) applied to an SVM shows that the proposed approach achieved stable performance with an overall performance of 98.53%.

3.2 ENGINE FAULTS AND KNOCKING NOISE

Engine faults refer to any abnormal condition or malfunction that affects the performance, efficiency, or reliability of an internal combustion engine. These faults can arise from various issues, such as mechanical wear and tear, electrical or electronic failures, fuel system problems, lubrication issues, cooling system failures, and other factors. Engine faults can be detected through various methods, such as in-cylinder pressure measurements, vibration analysis, temperature monitoring, and other diagnostic techniques. Addressing engine faults promptly and accurately is crucial to ensure optimal engine performance, extending engine lifespan, and prevent costly repairs or replacements.

Firmino et al. (2021) proposed two different methods for misfire detection on an SI engine, using two acquisition systems with vibration signals and another using acoustic signals. The experiment was conducted with a four-cylinder engine, that works on the four-stroke stages. The detection of misfire using both techniques acoustic analysis was able to acquire using a microphone placed near the engine block to record the sounds from the engine, and the vibrational analysis was able using to acquire using an accelerometer in an Arduino UNO placed in the middle of the engine. The acquisition process for both processes was performed simultaneously, thus they have the same environmental conditions. The use of ANN is due to its non-linearly separable pattern, creating a multilayer perceptron. The result for the two methods performed with high

accuracy for fault detection of 100% for vibration-based and 87.5% for acoustic-based.

According to Millo and Ferraro (1998), several knock-detection methods, based both on cylinder pressure analysis and engine block vibration analysis, have been proposed. The authors concluded that all of them are reliable and with a careful selection of a proper filtering-frequency band and crank-angle window, mechanically induced noises can be minimized and vibration based methods can achieve comparable signal-to-noise ratios to those of the corresponding pressure based methods.

Liu et al. (2010) proposed a new method of detecting the knock occurrence using wavelet packet transform, where using the power spectral density estimation presents the resonant frequency of knock, allowing the decomposition later done by the wavelet package. The frequency range of knock is from 5kHz to 10kHz, with a higher frequency of 7.0313kHz. The vibration signals were captured in a gasoline engine with four cylinders, with a sensor that has a sampling rate of 100kHz, between two engine rotation speeds of 2600 rpm and 3000 rpm. The wavelet mother was defined as sym8. Even for light knock occurrences, the analysis indicates that the proposed method was effective.

Janakiraman et al. (2015) developed a machine learning approach to detect faults in an engine based on in-cylinder pressure measurements. The authors based the study on homogeneous charge compression ignition (HCCI), which utilizes a compressed mixture of fuel and oxidizer for combustion and is a type of internal combustion engine. The study was conducted at a constant engine speed of 2500 rpm, using various machine learning models, including LR, LLS, SVM, and ELM. In the training phase, the researchers trained the models using 6400 cycles of data, and in the testing phase, they conducted tests using 10200 cycles. The results indicated that the SVM model had the best overall performance, achieving an accuracy of 93.1%. The other models performed as follows: ELM had an accuracy of 90%, LR had an accuracy of 85.7%, and LLS had an accuracy of 85.2%. Overall, the study demonstrates the potential of machine learning in detecting faults in HCCI engines using in-cylinder pressure measurements.

Panzani, Östman, and Onder (2017) proposed a logistic regression application using in-cylinder pressure of an SI engine for knock detection. This engine is equipped with 2 parallel twin cylinders. The investigation relies on two topics, physical mechanisms and principal component analysis, allowing the extraction of pressure information. The researchers used an oscilloscope with a sampling frequency of up to 2.5GHz to capture in-cylinder signals during acquisition. They tested different speeds ranging from 1800 rpm to 3000 rpm. The result showed that only three features led to better performance: engine speed, intake manifold temperature, and in-cylinder pressure.

Falcão, Barros, and Melo (2019) presented a knocking engine phenomenon definition for machine learning problems. The knock occurrence is defined as a categorical

random variable which leads to defining the event as a classification problem (logistic regression). To be able to identify the knock probability, it is necessary to analyze data from in-cylinder pressure, temperature, and engine speed.

Petrucci et al. (2020) aimed to investigate the effectiveness of various machine learning algorithms in detecting engine knock events and their intensity. The study used backpropagation artificial neural networks (BPANN) with different activation functions such as Sigmoid (applied on MATLAB) and ReLu (applied on Python), and Random Forest (RF) methods. One important aspect highlighted was the ability of these methods to accurately distinguish valid knock events from no-knock events, which could potentially cause damage to the engine structure. The authors concluded that all the methods tested were able to detect knock events with an acceptable level of error. The result was measured using the root mean squared error (RMSE), which resulted in 0.00174, 0.00674, and 0.08556 for the BPANN MATLAB, BPANN Python, and RF, respectively.

Kefalas et al. (2021) analyzed the knocking noise phenomenon in three different internal combustion engines. Aiming to use a processing technique called Continuous Wavelet Transformation (CWT), which provides simultaneous analysis of the in-cylinder pressure traces in the time and frequency domains with coefficients. These coefficients were used as input for a Convolutional Neural Network (CNN) in a way to classify the combustion as non-knocking and knocking. As result, they made successful detection of knocking combustion and the classification of non-knocking and knocking combustion performed by the CWT and the CNNs yielded an overall accuracy of 92.6%.

Unlike the others mentioned before, Shahid, Ko, and Kwon (2022) proposed a real-time abnormality detector using CNN. This analysis used a magnetic pickup sensor that measures the angular speed of the engine, allowing the signal to be transformed to crank angle degree (CAD) signal. The model was designed to detect any variation in the CAD signal, which was used to monitor each internal combustion diesel engine cycle. Additionally, the signal was filtered to remove the presence of noise components. The experiment was conducted with a four-stroke marine diesel engine at a speed of 720 rpm with a sampling rate of 10 MHz through an MPU sensor. It was considered three classes: normal state, engine load change, and fault condition (Misfire). Considering a learning rate of 0.001 and a limit of 2200 epochs. Compared to methods like SVM and KNN, the proposed approach has a higher accuracy achieving more than 99.7%.

3.3 SUMMARY OF APPLIED TECHNIQUES

This section provides a wide overview of the techniques, algorithms, and specific parameters adopted in fault detection fields on machinery and engines. Vibrational analysis is a widely used technique for identifying faults in engines and machinery. Whenever a fault occurs, it generates a unique vibration pattern that can be detected and analyzed. However, it is important to note that the fault condition can also impact

other aspects of the engine. If sensors monitor these aspects, they also can be used to increase the detection performance or implement another way of detecting the fault.

Among all the fault detection methods (FDM), 2 model-based, 3 signal-based, 6 data-driven and 5 hybrid methods were reviewed can be seen in Table 2. Most machine learning applications rely on data and the technique's capability to identify patterns to detect faults. Therefore, the proposed approach to this work is to apply a data-driven method in this field of fault detection applied to engine knock.

Even though most of the related works have achieved their results based on high frequencies acquired data to detect faults, where the frequency range from 512Hz to 2.5GHz. Our approach will be based on data acquired from the vehicle's ECU, which has a bandwidth limit resulting in low frequency data, achieving a frequency range from 100Hz (10ms) to 250Hz (4ms). Further aspects like speed, temperature, and other sensors have been added to the analysis to compensate and be used in some machine learning techniques.

Regarding fault detection, extracting noise from raw data and carefully selecting features is crucial to the results. CNNs can automatically extract pertinent features from the raw data, eliminating the need for manual feature engineering. The related work achieved results that range from 84% to 100%, and some of the techniques are CNN, MSCNN, SVM, Wavelet-SVM, and many others. These results were achieved in a high frequency spectrum.

Overall, using CNNs and classifiers in fault detection provides a powerful and effective approach for automatically extracting features, capturing complex fault patterns, handling large datasets, and achieving accurate fault classification or diagnosis. Therefore, we selected neural networks (dense or convolutional), SVM, autoencoders, and a method based on trees as the machine learning techniques to be applied.

Several other applications of fault detection for engine knock that are more analytical in nature were reviewed. By examining these approaches, we can gain insights into improving fault detection methodologies and develop more effective strategies for detecting knock.

The subsequent section will delve into a more comprehensive exploration of the method employed for detecting engine knock using machine learning. It will provide an in-depth analysis of the specific techniques and algorithms utilized to identify and classify instances of engine knock effectively. The detailed description will encompass the preprocessing steps, feature extraction methods, and the specific machine learning models employed in the detection process. Additionally, the section will discuss the rationale behind the chosen approach and highlight its potential advantages and limitations based on the results. By delving into the intricacies of the methodology, a deeper understanding of the intricacies of engine knock detection using machine learning will be obtained.

Table 2 – Summary of techniques

Source	**Features**	**FDM**	**Freq.**	**Technique**	**Engine Knock**	**Metrics (%)**
(MILLO; FERRARO, 1998)	Vibration & pressure signals	Signal-based	1MHz	filtering-frequency band	Yes	NA
(PANZANI; ÖSTMAN; ONDER, 2017)	Pressure, temperature, engine speed and air-to-fuel ratio	Model-based	2.5GHz	Logistic Regression	Yes	NA
(JIANG, G. et al., 2019)	Vibration signal	Hybrid	10kHz	MSCNN	No	F1-Score: 98.53
(KRUMMENACHER et al., 2018)	Vibration signal	Hybrid	NF	CNN & SVM	No	F1-Score: 86.49, Precision: 87, Recall: 86
(LIU et al., 2010)	Vibration signal	Signal-based	100kHz	Wavelet Transform	Yes	NA
(PETRUCCI et al., 2020)	Pressure, temperature, air-fuel ratio, and engine speed	Data-driven	NF	BPANN & RF	Yes	NA
(KEFALAS et al., 2021)	Pressure signal	Hybrid	NF	CWT & CNN	Yes	Accuracy: 92.62
(FALCÃO; BARROS; MELO, 2019)	Vibration signal	Model-based	NA	Logistic Regression	Yes	NA
(JANAKIRAMAN et al., 2015)	Pressure signal	Hybrid	NF	SVM	No	Total Accuracy: 93.10
(ZILONG; WEI, 2018)	Vibration signal	Data-driven	12kHz	MSCNN	No	Accuracy: 98.57
(JIA et al., 2016)	Vibration signal	Data-driven	5.12kHz	SAE	No	Accuracy: 99.61 $\sim$ 100
(FIRMINO et al., 2021)	Vibration & acustic signals	Hybrid	512Hz	BPANN	No	Accuracy: 87.50 $\sim$ 100
(PAN et al., 2018)	Vibration signal	Data-driven	48kHz	CNN & LSTM	No	Accuracy: 100
(JIANG, H. et al., 2018)	Vibration signal	Signal-based	12kHz	DRNN	No	Accuracy: 94.75 $\sim$ 96.54
(SHAHID; KO; KWON, 2022)	CAD signal	Data-Driven	10MHz	CNN	No	Accuracy: 99.70
Proposed approach	Vibration signal, temperature, air-fuel ratio, engine speed, more details on Table 4	Data-Driven	100Hz$\sim$ 250Hz	CNN|SVM|AE	Yes	F1-score: 81%

Source: Author.

NA: Not applicable.
NF: Not found.

4 DETECTION OF ENGINE KNOCK

In the following sections, we provide a high-level description of the process used for detecting engine knock, an overview of the experiments conducted, details on the signals that were analyzed, and a summary of the results.

4.1 OVERVIEW

ECU stands for Electronic Control Unit, which manages various electrical systems in a vehicle. The ECU receives data from various sensors throughout the vehicle and uses this data to control various functions, including fuel injection, ignition timing, idle speed control, and more. It also monitors the vehicle's performance and can identify and diagnose any issues that may arise. The Intelligent Acquisition and Analysis System for ECU (IASE) hardware was developed by LISHA and Renault and allow access to the variables monitored by the ECU. This specific hardware connects to the ECU through the XCP or CCP protocols to extract data to be analysed.

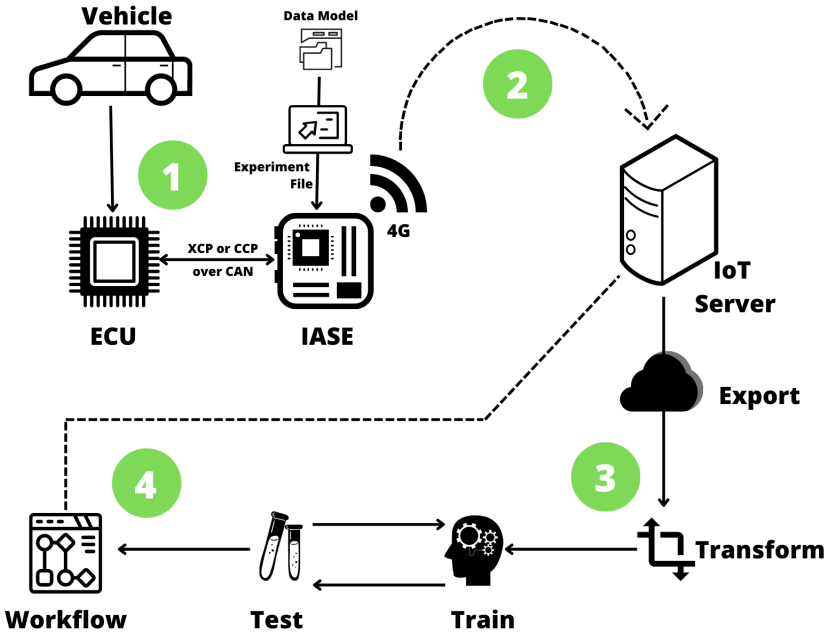

Figure 5 – Overview of the proccess.

We have defined the process into four stages to enhance comprehension, illustrated in Figure 5. In Stage 1, we create a Data Model that associates each ECU signal with a Smartdata structure, defining the signal characteristics as shown in Table 3. In order to extract the data of each signal from the ECU, we use the Data Model to generate the experiment file. This experiment file is responsible for selecting a range of signals and defining the sampling rate of each signal. Each sampling rate has a max amount of signals. The smallest sampling rate (4ms) has a limited amount of signals

which can be selected. The IASE hardware has to be loaded with the experiment file to start the acquisition process.

Stage 2 involves the interaction between the IoT Device and the IoT Server. Creating a series of Smartdata is necessary before initiating the data acquisition process. Each signal possesses unique characteristics, and these definitions distinguish each signal within the platform. The creation method for a series of Smartdata is then transmitted to the IoT Server, which interprets the method to determine when data sets reception should start and whether the data should undergo a pre-ingestion process known as Workflow. Workflows can be utilized to modify, accumulate, and analyze the data, but it has particular benefits in the realm of machine learning. Further details regarding the Workflow will be explored in subsequent sections. Once each series of Smardata has been created, the IoT Server awaits the data arrival of the corresponding signal.

In Stage 3, we export and prepare the raw data stored in the IoT Server to make it suitable for applying machine learning techniques. The data set is analyzed in an offline process to ensure its readiness. Once we achieve satisfactory results through machine learning algorithms for Engine Knock fault detection, we integrate the resulting model into the platform Workflow. In Stage 4, we utilize the machine learning model to detect the occurrence of faults in real-time, enabling proactive actions.

This overview provided a roadmap of the entire process. In the subsequent sections, a more detailed exploration of each stage will be presented, providing in-depth information and insights.

4.2 FIRST STAGE

As previously mentioned, the purpose of this section is to provide additional details regarding the creation of the Data Model, the generation of the experiment file, and the process of loading it onto the IASE hardware. In order to create the Data Model, it is necessary to comprehend the structure of signals within the ECU, which is unique to each ECU and is presented in an ASAM MCD-2 MC Language (A2L) file. This file serves as a standardized interface for accessing and manipulating an ECU's calibration and measurement parameters. It enables calibration tools and diagnostic equipment to interact with the ECU standardized and consistently. Thus, permitting extract crucial information to define each signal as a Smartdata.

Due to a non-disclosure agreement that prohibits the presentation of crucial information, each signal corresponds to a variable, but the variable names have been renamed. Thus, the next step is to define the Smartdata mode of operation. There are two operations versions, the first (1.1) addresses using space-time coordinates fixed to identify the device, and the second (1.2), which has the same data representation but with moving coordinates. To capture the moving device from the second version, a key

tag represents the device's signature, in our case, the vehicle's chassis. Each variable is associated with a date and time (timestamp) in addition to its corresponding value, and unit.

Table 3 – Data Model Example

Signals		SmartData								
Name	Variable	Version	x	y	z	t	Unit	Dev	Signature	Value
Engine Speed	var0	1.2	0	0	0	0	m/s²	0	car's chassis	10
Air Temperature In	var1	1.2	0	0	0	0	F	0	car's chassis	86
Air Temperature Out	var2	1.2	0	0	0	0	F	1	car's chassis	102

Source: Author.

Table 3 presents an example of how the correlation works in the Data Model. The definition of the Engine Speed as its origin(x, y, z, t) varies, but in version 1.2, the signature allows the identification of the data. The Smartdata structure defines each signal's uniqueness, so even though we have the same aspect of the unit, the dev method allows us to differentiate each signal from the other, as presented for Air Temperature In and Out. The Data Model mapped all the variables presented in the A2L file.

Prior to discussing the generation of the experiment file, essential to understanding that the protocols connections between IASE and ECU have an intrinsic limitation on the bandwidth imposed by the CAN (BEDRETCHUK et al., 2023). The bandwidth limitation in the system imposes an acquisition period of 4 ms, 5 ms, 10 ms, and 100 ms, or according to the synchronicity of the engine cylinder 1, 2, 3, and 4. Each acquisition period has a limited amount of signals as well.

Experiment file is generated based on the Data Model and provide the selection of range of variables to be acquired from the ECU. Depending on the analysis requirements, it is possible to select the acquisition frequency of each variable while respecting the limitations imposed by the acquisition period. Than, the IASE hardware has to be loaded with the experiment file to start the acquisition process. During the acquisition, the data will be buffered and prepared to send as series through 4G to the LISHA IoT Platform using the SmartData format.

4.3 SECOND STAGE

Preparing the IoT Server to receive the data is necessary at this stage. The complex structured system created by LISHA allows the system to identify if the Smartdata series has errors of structure because there are many methods of insertion of series, for instance, series with start and end set, start and end set by count, start and end set by flag finish, and many others. This work will use only the method that sets the start and end of the series creation. If the IASE is still running for 4 minutes, the software in

the device will generate another series creation with 5 minutes ahead, so the platform will concatenate both series as one.

Another aspect of creating the Smartdata series is defining the Workflow process that the data will go through before being stored. LISHA's IoT Server uses Workflow to execute server-side algorithms on the received series. It can be defined as input or output workflows. The input workflow is executed when the request method is PUT (inserting), while the output workflow is executed when the request method is GET (searching). These workflows allow preprocessing data, running machine learning, fixing data points, adjusting measurement errors, generating notifications, and interacting with other series already inserted. In this work, we aim to create the engine knock detection process that, after set, can be attached to the Workflow to detect engine faults in running time.

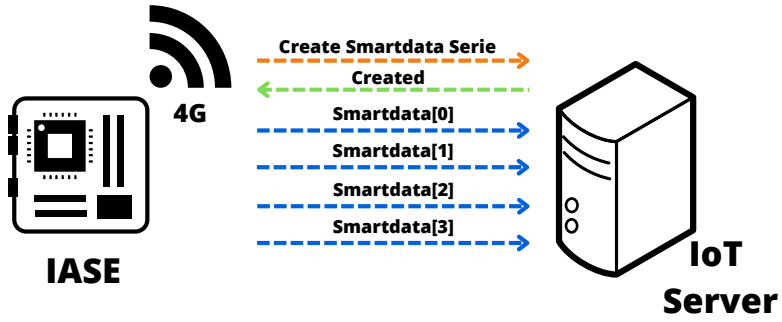

Figure 6 – IoT Device and IoT Server Communication

Source: Author.

A method for creating a Smartdata series is sent to the IoT Server and waits for the successfulness of the request, then accepts datapoints from the Smartdata created, as depicted Figure 6. After, the IASE hardware will buffer the data and be grouped uniquely and sent to the server, allowing the IoT Server to store it. Storing the data in the LISHA's IoT Platform allows it to visualize the data quickly using dashboards in real-time, which is a critical and costly process in a vehicle running test. Any analysis requiring extra engineers and companies to analyze the data before presenting it to Renault can save money and time, which is a key fact in using IASE hardware for this purpose.

4.3.1 Driving Experiment for Data Acquisition

The 'vehicle running test' we called the driving experiment, was performed on a Renault Sandero 1.0 equipped with a four-stage SI engine featuring a four-cylinder gasoline engine. The experiments were generated according to the experiment file loaded in the IASE hardware after careful selection of variables and filling each sampling

rate. Most of the main signals were defined to be 4 ms sampling rate, fetching to a 250 Hz of frequency rate.

The execution of the experiment was a difficult process because it depended on a team of researchers to do it correctly. Another fact is that the nature of the engine knock event brings difficulties in getting it in running tests. In some experiments, we have some parts with faults and others without. Renault has some historical experiments, but we could not use them because each running test was focused on one particular fault event occurrence. There would not have all variables analyzed or selected in the necessary sampling rate for our application, which would prejudice the results of machine learning techniques.

Table 4 – Selected Variables

	Sets								
-	A	B	C	D	E	F	**Description**	**Variable**	**Target**
1	x	x	x	x			Knock Signal Cylinder 0	var0_0	Input
2	x	x	x	x			Knock Signal Cylinder 1	var0_1	Input
3	x	x	x	x			Knock Signal Cylinder 2	var0_2	Input
4	x	x	x	x			Knock Signal Cylinder 3	var0_3	Input
5	x						Intake Air Temperature	var1	Input
6	x		x			x	Fuel Consumption	var2	Input
7	x			x	x		Intake Manifold Temperature	var3	Input
8	x		x	x	x	x	Intake Manifold Pressure	var4	Input
9	x						Engine Coolant Temperature	var5	Input
10	x		x			x	Torque index	var6	Input
11	x		x			x	Duration of 30° Crank	var7_0	Input
14	x		x			x	Duration of 30° Crank	var7_3	Input
12	x		x			x	Duration of 30° Crank	var7_1	Input
13	x		x			x	Duration of 30° Crank	var7_2	Input
15	x		x			x	Duration of 30° Crank	var7_4	Input
16	x		x			x	Duration of 30° Crank	var7_5	Input
17	x		x	x	x	x	Engine Speed	var8	Input
18	x		x	x	x	x	Mean Knock Noise	var9	Input
19	x		x	x	x	x	Engine Air Load	var10	Input
20	x	x	x	x	x	x	Knock Detection	var11	Output

Table 4 shows the selected variables based on the literature review and related works, where some sets (A, B, C, D, E and F) contain a subset of all the variables. A variable may contain single or many signals referred to it. The variable 'Knock Signal' has four signals representing each engine cylinder, although most of the variable represents a single signal as the 'Intake Air Temperature' and others. The number of signals each variable contains are presented in Table 4.

The set A has all 19 input variables, set B has 4 input variables, and set C has 16 input variables. Set C applied a feature selection using Random Forest to find the best subset of highly correlated features. In set D, the variables were knowledge-driven selected, and the set E is based on D, but the knock signal has been removed. The last one is set F, which is set C without the knock signals. All these different sets allow us to investigate the engine knock phenomenon in various ways and give us the possibility

to understand how it affects other variables.

We have established two scenarios for the experiments, which could be monitored in real-time: one representing normal driving conditions without knocking and another where we intentionally induce knocking in the engine. We increase the engine load by activating components such as the air conditioning, rear window defroster, and high-beam lights to provoke the knocking failure. However, in our driving experiments, we generate knocking by deliberately delaying the gear changes, resulting in higher values on the tachometer at lower speeds. ECU data was collected from multiple experiments, with varying durations ranging from 1 to 35 minutes, as outlined in Table 5. Each data set is identified numerically as 1 to 6, followed by the time duration and knock detection based on the piezoelectric sensor. It is accurate to state that the vehicle's ECU incorporates a piezoelectric sensor to detect engine knock through vibrational analysis. However, it is essential to note that high speeds and other aspects can influence this detection mechanism. Consequently, our proposed methods aim to expand the scope of detection beyond vibrations alone, incorporating additional signals to identify instances of engine knock.

Table 5 – Data Sets

Id.	Time Length	Knock Detection	Total Points
1	00:03:44.25	21	22425
2	00:14:49.43	323	88943
3	00:01:48.19	19	10819
4	00:30:11.65	672	181165
5	00:17:53.91	63	107391
6	00:34:42.95	180	208295

4.4 THIRD STAGE

The third stage is the data preparation, variable analysis, and feature selection process to apply machine learning techniques. This process is made offline, so first, it is necessary to export the data from the IoT server to have the data sets. The following sections will present the application of the datasets in machine learning algorithms, hyperparameter tuning, and more details.

4.4.1 Data Extraction and Transformation

The generation of the machine learning algorithm is based on the data acquired and stored in the IoT Platform. A script was developed to perform the data export process from the platform to prepare the data set for the next steps. As the experiment file has the specification from each signal, it was used to prepare the requests to extract from the server the data. Due to the high amount of signals defined in the experiment file, a parameter was created that divides the experiment file into batches. The batches'

value is configurable, defined as 50 signals per step. However, It is necessary to know the period of the experiment. Requests are generated based on the start time (t0) and the end time (t1) to be able to search for each Smartdata series. The IoT Platform receives batches of search requests based on the GET method, and as an asynchronous method was applied, the process keeps waiting for the return of all searched requests.

This process is repeated until finishes all the batches. Then it is necessary to adjust the data because it comes with all the details defined by the architecture of the IoT Platform. The basic information needed are unit, dev, value, timestamp, and signature, which refers to the vehicle. The Data Model has defined the tuple between (unit, dev) of the platform to signals name, then, it reorganized the data frame. The acquisition time is on nanoseconds, so a resample must be applied, as most data have a sampling rate of 4ms, 5ms, and 10ms.

Initial data exploration was performed using Python's pandas library's describe() method. It gives a quick overview of the data's central tendency, spread, and range, providing a snapshot of the dataset's distribution. It is a helpful tool for exploratory data analysis and establishing a foundation for further data processing and analysis tasks. The application of this method in knock signals as raw values can be seen at Table 6.

Table 6 – Description of Raw Knock Signals - Data Set[4]

	var0_0	**var0_1**	**var0_2**	**var0_3**
count	181166	181166	181166	181166
mean	0.878383	0.873187	0.816898	0.897222
std	0.559985	0.602291	0.611108	0.658291
min	0.268631	0.287094	0.316620	0.268478
25%	0.640335	0.610275	0.547791	0.602188
50%	0.758667	0.734177	0.654907	0.736542
75%	0.918427	0.906219	0.828400	0.933990
max	4.999924	4.999924	4.999924	4.999924

Another useful tool is the box plot, also known as a box-and-whisker plot, which is a graphical representation of the distribution of a data set. It concisely summarizes the data's central tendency, spread, and presence of outliers. The plot consists of a rectangular box and two "whiskers" extending upper and down from it. Outliers are data points that fall outside the whiskers and are plotted individually. The same values plotted in the Table 6 are presented in the box plot form in Figure 7, which allows better visualization of the data. Based on the plotted data, approximately 50% of dataset four's data falls within the box, indicating a relatively small range of values. However, the wide range of values, particularly for var7 (Duration of 30° Crank), in the dataset signals could potentially affect the performance of machine learning models when applied.

Normalization, or min-max scaling, is a data preprocessing technique that rescales numerical features to a specific range, typically between 0 and 1. It ensures that different features are on a similar scale, preventing the dominance of any single feature based on magnitude. Normalization maintains close relationships between values, han-

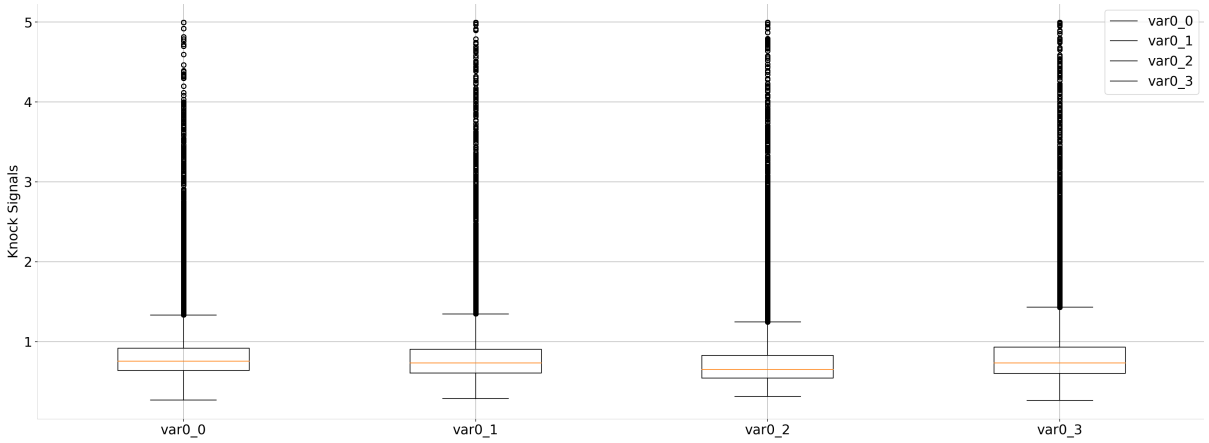

Figure 7 – Box Plot Raw Knock Signals - Data Set[4]

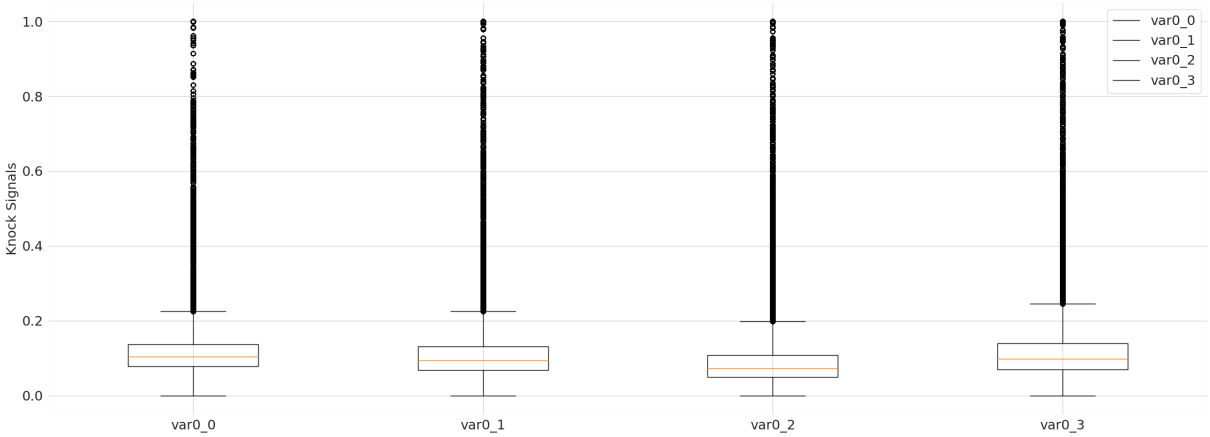

Figure 8 – Box Plot Normalized Knock Signals - Data Set[4]

dles outliers cautiously, and benefits algorithms sensitive to feature scale, such as neural networks. The normalization technique was applied to the datasets to rescale all features to the range of 0 and 1.

Figure 8 presents the normalized knock signals data depicted to contrast with the raw data previously presented in Figure 7. It can be observed that the data maintains its original distribution but in a narrower range of values. The normalized box plot representation shows that most values are under 0.2 with outliers on a higher level, which can be checked in Figure 9. Moreover, in Figure 10 is presented all signals of the dataset [4] normalized. Standardization, another alternative method for data preparation, was implemented. However, it did not yield significant improvements in our proposed methods.

4.4.2 Selection of Variables

In machine learning, feature selection plays a vital role in identifying the most important features for classifying outcomes. This helps to eliminate irrelevant or redundant features, which can enhance model performance, reduce complexity, and improve

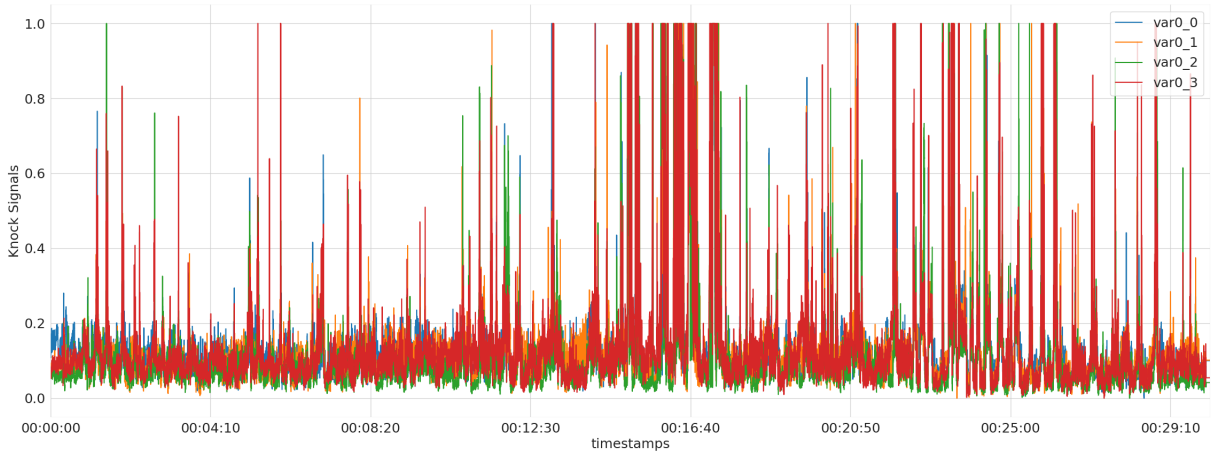

Figure 9 – Plot Normalized Knock Signals - Data Set[4]

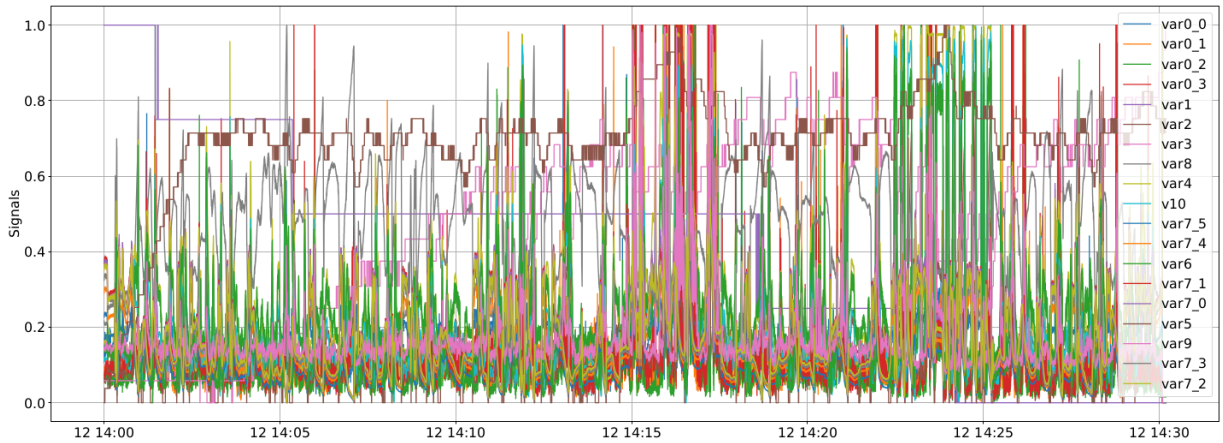

Figure 10 – Signal Normalization - Data Set[4]

interpretability. The RandomForestClassifier algorithm, available in Python's scikit-learn library, is a powerful tool for accurate classification. It uses multiple decision trees to assign feature importance scores, which aid in feature selection and understanding relationships within the dataset. In conclusion, the RandomForestClassifier algorithm is a robust and flexible method with feature importance analysis, making it valuable for classification tasks.

To identify the most relevant features in multiple datasets, the RandomForest-Classifier algorithm was applied. By using ensemble learning, the algorithm detected key features that significantly contributed to generating higher performance across different datasets. The algorithm generated feature importance scores, providing valuable insights into the relationship between each feature and the target variable. The most important features are presented in Set C's Table 4, including engine air load, torque index, etc.

4.4.3 Machine Learning Techniques

The field of fault detection in engines has experienced a significant paradigm shift with the advent of machine learning. By harnessing the power of advanced algorithms, we can now achieve unprecedented accuracy in identifying faults and anomalies, leading to enhanced performance, improved maintenance, and reduced downtime. It will be explored the merits of four popular machine learning (ML) algorithms - Isolation Forest, SVM, and Autoencoder (Dense and Convolutional), Classifier - in the context of fault detection on engines.

Throughout this section and the subsequent ones, we will delve into several experiments that incorporated an ML algorithm. Each experiment considered an architecture, specific parameters, a splitting process, and various datasets that were trained and tested based on a range of variables (sets). The experiments conducted in this study were generated, and the performance of each algorithm was evaluated using another part of the dataset not included in the training process and using the F1-score metric to evaluate the detection of faults.

Isolation Forest is a powerful anomaly detection algorithm that identifies rare and abnormal instances. This algorithm constructs a decision tree ensemble that effectively isolates anomalies in fewer splits than normal instances. In engine fault detection, Isolation Forest can quickly identify outliers and anomalies that may not conform to standard patterns. Its ability to work well with high-dimensional data and provide interpretable anomaly scores makes it a valuable addition to any fault detection system. This algorithm can be applied using the method *IsolationForest* from the sklearn python library. In this scenario, it was analyzed intrinsic aspects of properties and kept the splitting of 75/25 the same for each experiment.

SVM are highly regarded for their versatility in handling both linear and non-linear classification problems, making them invaluable for engine fault detection. The flexibility of SVMs lies in their ability to utilize different kernel functions, allowing them to capture diverse patterns of engine faults and adapt to different scenarios. Additionally, SVMs exhibit robustness by being resistant to overfitting, resulting in lower false positive rates and ensuring the reliability of fault detection in engines. In this scenario, the analysis focuses on the kernel that in this case, can be Linear, Sigmoid, Radial Basis Function (RBF), and Polynomial on various degrees starting from 3 to 6. Each experiment maintained a splitting correlation of 75/25 between the train and the test.

Dense and Convolutional Autoencoder excel in dimensionality reduction, capturing intricate relationships in engine sensor data, and reconstructing fault-free representations. They can unveil subtle abnormalities and minimize noise interference. While autoencoders are commonly used for unsupervised learning, they can also be adapted for supervised learning scenarios by incorporating labeled data during training and validation. By training an Autoencoder solely on normal data, the model learns to

capture the patterns and characteristics specific to normal engine behavior. This allows the model to become highly efficient in recognizing normal instances, as it becomes adept at reconstructing and representing normal patterns accurately.

In the context of Autoencoder, only one dataset [2] was found to have a clear separation between healthy and faulty data, while other datasets have a mixture of both healthy and faulty values. It was used the same sets of variables, but the Dense Autoencoder used the architecture presented in Table 7 which consists of changing its threshold value to evaluate the performance. While the Convolutional Autoencoder that uses this architecture presented in Table 8 involves the analysis of the window size of validation for performance evaluation.

Table 7 – Architecture of the Autoencoder

Step	**Layer**	**Activation**	**Kernel**
Encoder	Dense(len(INPUT))	ReLu	glorot_uniform
//	Dense(9)	//	//
//	Dense(4)	//	//
//	Dense(2)	//	//
Decode	Dense(4)	//	//
//	Dense(9)	//	//
//	Dense(len(INPUT))	//	//

Table 8 – Architecture of the Convolutional Autoencoder

Step	**Layer**	**Activation**	**Kernel**
Encoder	InputLayer (None, 256, 19)	LeakyReLU	glorot_uniform
//	Conv1D (None, 16, 19)	//	//
//	Dropout(None, 16, 19)	//	//
//	Conv1D(None, 8, 19)	//	//
Decode	Conv1DT(None, 16, 19)	//	//
//	Dropout(None, 16, 19)	//	//
//	Conv1DT(None, 256, 19)	//	//
//	Conv1DT(None, 256, 19)	//	//

Classifier algorithm is an excellent starting point for fault detection tasks due to its simplicity and interpretability. Its primary advantage lies in its ease of implementation and understanding. By dividing the data into distinct classes, this algorithm can accurately categorize engine behavior as normal or abnormal. Simple classifier is ideal for scenarios where interpretability and transparency are crucial, enabling engineers to comprehend the underlying factors contributing to a fault.

In the scenario using the classifier algorithm, we have proposed two architectures, each considering aspects of batch size and sequence length. The first architecture is presented in Table 9, considering the batch size of 126 and sequence length of 64 (126/64) and batch size of 64 and sequence length of 32 (64/32). This architecture

Table 9 – Architecture I - Classifier

Step	**Layer**	**Activation**	**Kernel**
//	Input(len(X))	ReLu	glorot_uniform
//	Conv1D	//	//
//	Dropout(rate=0.01)	//	//
//	Flatten()	//	//
//	Dense(X/2)	//	//
//	Dense(X/4)	//	//
//	Dense(X/6)	//	//
//	Dense(1)	//	//

consists of an input layer that takes input data with a length determined by the number of input features (X). This is followed by a 1D convolutional layer, which captures local patterns and features in the input data. To prevent overfitting, a dropout layer randomly sets a fraction of input units to zero during training. The output is then flattened into a one-dimensional vector using a flattened layer.

Subsequently, there are three fully connected dense layers, with the number of units in each layer being a fraction (x/2, x/4, x/6) of the input size (x). The rectified linear unit (ReLU) activation function is applied in each dense layer, introducing non-linearity. Finally, the output is obtained from a single-unit dense layer, which defaults to a linear activation function.

The second architecture reconds to the Table 10, considering the batch size of 64 and sequence length of 32 (64/32). The architecture begins with an Input Layer that takes input data with a shape defined by the sequence length and the number of features. This is followed by two layers of 1D convolution and 1D max pooling, which capture relevant patterns and features from the input sequence. Subsequently, the output is flattened into a one-dimensional vector.

Table 10 – Architecture II - Classifier

Step	**Layer**	**Activation**	**Kernel**
//	Input(len(X))	ReLu	glorot_uniform
//	Conv1D	//	//
//	MaxPooling1D	//	//
//	Flatten()	//	//
//	Dense(X/2)	//	//
//	Dropout(rate=0.01)	//	//
//	Dense(X/4)	//	//
//	Dropout(rate=0.01)	//	//
//	Dense(X/8)	//	//
//	Dropout(rate=0.01)	//	//
//	Dense(X/16)	//	//
//	Dropout(rate=0.01)	//	//
//	Dense(1)	//	//

After the flattening process, there are four layers of dense neural networks. Each layer has a different width (w) determined by dividing the sequence length by a factor, starting from 2 and doubling the value until reaching 16. For example, the first dense layer has a width of (sequence length // 2), the second layer has a width of (sequence length // 4), and so on. Dropout with a rate of 0.01 is applied after each dense layer to prevent overfitting. Thus, the architecture concludes with a dense layer with a single unit, which defines the classification process, producing the output results.

The experiments applied two splitting processes of the dataset for training and testing, it ranges from 80/20 or 50/50. The selection of the splitting ratios is based on their prevalence and widespread usage as commonly adopted processes in the field. Each experiment had slightly different aspects, which directly affected its performance. The analysis applied in this scenario was to identify the variance in performance based on changing datasets and sets of variables.

4.5 FOURTH STAGE

Regarding fault detection using machine learning, it's essential to achieve optimal results in model development to ensure accurate detection results. Once the necessary adjustments have been made, the model can be uploaded to the IoT Platform. Before ingesting data, it is important to execute the Smartdata method, which includes defining the workflow ID to create a custom workflow. This workflow can then be used to identify and detect faults using the model developed.

To systematically store batches of data transmitted from the IoT Device and instantly analyze them for engine knock faults, an appropriate script was utilized. This process allows for the use of multiple machine learning models to detect faults, and analysis can be developed in the same workflow asynchronously. As the IoT Device transmits data, the IoT Server continuously receives and analyzes batches, storing them locally for workflow analysis and then storing them on the server side. Once a detection is made, the local data can be deleted. However, since the data is stored on the server, it can be exported later. The script created for this purpose can export the data from the IoT server. Overall, using a customized workflow and multiple machine learning models allows for efficient and effective fault detection in IoT systems.

Having discussed the key stages of the proposed fault detection method, the next step is to delve deeper into the results obtained from each algorithm. In the following sections, we will explore and analyze the outcomes achieved by utilizing simple classifiers, isolation forests, SVM, and autoencoders (both dense and convolutional). By examining the results, we can gain valuable insights into the effectiveness and performance of each algorithm in detecting faults within engines.

5 EXPERIMENTAL RESULTS

This section presents more details on the results achieved for each ML algorithm developed, along with a comparison of the best results.

5.1 EVALUATION

Isolated Forest. The Isolation Forest is an anomaly detection algorithm designed to identify outliers or anomalies within a dataset, as previously mentioned. This algorithm is based on the concept of randomly partitioning the data into smaller subsets, making it relatively straightforward to detect anomalies. The key parameters of the Isolation Forest include contamination, the number of estimators, max features, and max samples.

The contamination parameter represents the proportion of anomalies in the dataset, which can vary depending on the specific dataset, the characteristics of the engine data, and the goals of the anomaly detection system. The primary objective of this machine learning model is to ensure the detection of actual knock events, thereby minimizing false negatives, while also reducing the number of false alarms or false positives.

In practice, it is common to start with a contamination value in the range of 0.01 to 0.05 (1% to 5%) and then fine-tune it using techniques like cross-validation. Cross-validation helps to address the issue of overfitting or underfitting by providing a more robust assessment of how well the model will generalize to new, unseen data. It is particularly useful when the available dataset is limited, as it allows for better utilization of the available data for both training and evaluation.

According to the available datasets in Table 5, it is possible to calculate the contamination values based on the following Equation (6). Thus, the average contamination value for all six datasets is 0.0019 (0.19%).

$$\text{contamination} = \frac{\text{Number of Knock Detection}}{\text{Total Number of Data Points}} \tag{6}$$

A grid-search was conducted using *GridSearchCV* to optimize the model's hyperparameters. GridSearchCV is a class in the scikit-learn library in python (imported using from sklearn.model_selection import GridSearchCV) that performs hyperparameter tuning through an exhaustive search over a specified parameter grid. The search space was systematically explored, evaluating the model's performance for each combination of hyperparameter values to identify the optimal configuration.

The grid-search parameters were defined to encompass both average contamination values and randomly selected values. Additionally, specific values within the typical range of 1% to 5%, commonly applied in generic fault detection cases, were included. The number of estimators represents the number of trees in the forest. In-

creasing the number of trees may extend the model's runtime and resource usage; typically, most problems utilize between 50 to 150 trees. Furthermore, the number of samples drawn to build each tree (max_samples) is usually set as a fraction of your dataset size

In the specific scenario highlighted in the first instance of Table 11, an f1-score of 0.22 (22%) was achieved for engine knock detection. The grid-search provided this result, which is the best local optimum. It is the best among the selected grid parameters, resulting in a contamination value of 0.0233 (2.33%) and a max samples value of 0.25. This specifies that 25% of the total samples are used to build each isolation tree. The max features parameter is set to 19 (set A), indicating that the algorithm selects all 19 features at each split, facilitating the capture of complex patterns.

Even though we had set a contamination value based on the calculated contamination, it may not represent the actual map of the engine knock fault contamination area. It is crucial in any application of machine learning based on fault detection to test and cross-validate multiple times. In all cases, the results based on calculated contamination were tested and achieved less than 5% f1-score for detecting the engine knock.

As the contamination values rise, larger areas are identified as faults. This leads to an increase in recall values, subsequently causing a reduction in precision values. This directly impacts the F1-score, resulting in its decrease. So, the optimal selecion of contamination is crutial in a fault detection system to balance the recall and precision, achieving great results on fault detection.

Table 11 – Isolation Forest Results

Id.	Algorithm	Split [Train/Test]	DS.	Sets	F1-Score	Recall	Precision
1	Isolated Forest	75%, 25%	2	A	0.22	0.37	0.15
2	Isolated Forest	75%, 25%	2	A	0.26	0.42	0.19
3	Isolated Forest	75%, 25%	2	B	0.31	0.67	0.20

Similarly, another round of random contamination values was applied in the second scenario identified as 2 in Table 11. The contamination set as 0.021, indicating that approximately 2.1% of the data are considered anomalous. The max features and max samples parameters remain the same as in the previous scenario, maintaining the same feature selection and sample usage characteristics, achieving an f1-score of 0.26 (26%).

In the third scenario, result in 3 at Table 11, the contamination parameter is set to 0.0138, suggesting a lower expected proportion of anomalies at approximately 1.38% of the data. The max features parameter is reduced to 4 (set B), indicating that only four randomly chosen features are considered at each split. This feature reduction may limit the algorithm's ability to capture complex patterns, but it can enhance interpretability and mitigate the influence of noise or irrelevant features. The max samples value

remains consistent with the previous scenarios, using 25% of the total samples for constructing each isolation tree.

These parameter configurations play a crucial role in controlling the behavior of the isolation forest algorithm, determining its sensitivity to anomalies, capturing complex patterns, and managing computational efficiency. Despite our efforts to fine-tune the contamination parameter, we couldn't surpass an f1-score of 0.26 (26%) while using set A. However, focusing solely on the knock signals (set B) with this method brought about better results, with a f1-score of 0.31 (31%), as showcased in Table 11. No related works have been found that focus specifically on the detection of engine knock faults.

SVM. As explained earlier, Support Vector Machines (SVMs) are a type of supervised machine learning algorithms that are utilized for classification and regression tasks. They are popularly used in various applications owing to their ability to handle both linear and non-linear relationships in data. SVMs is a suitable choice for fault classification tasks, like distinguishing between engine knock and non-knock conditions. The key parameters of the SVM include kernel, regularization parameter (C), kernel coefficient (gamma), degree, class weight (class_weight), and decision function shape (decision_function_shape).

In the related work by Krummenacher et al. (2018), the authors use a CNN model to create an image based on sensor frequency, and then classify this image using SVM to identify different types of faults. On the other hand, in the proposed SVM method, the approach involves adapting the signal from the ECU directly to classification, specifically between two types: knock and non-knock.

So, the key difference lies in the input representation for the SVM. In the related work, the input is an image generated by a CNN from sensor frequency data, while in your proposed method, the input is the signal directly from the ECU, with the classification focused on distinguishing between knock and non-knock conditions.

It is essential to consider the advantages and limitations of each approach. The image-based approach may capture spatial dependencies and patterns in the sensor data, while the direct signal adaptation may leverage specific characteristics of the ECU signal for effective classification between knock and non-knock conditions. Depending on the nature of the data and the problem, one approach may be more suitable than the other.

Basically, the architecture of this machine learning model depends on the choice of its kernel, which is impacted by the application. It was used the *linear*, *sigmoid*, *RBF*, and *polinomial*. The selection of the appropriate kernel often involves experimentation and cross-validation to assess the model's performance on a validation set. The kernel used by the authors in the SVM model of the related work is Gaussian, which is the Radial Basis Function (RBF). On the other hand, this model did not perform well under this dataset, which will be exploited in detail in the following experiments.

The performance of the linear and sigmoid kernels, even with hyperparameter tuning through grid-search and cross-validation, did not achieve satisfactory results. The testing phase of these models resulted in an f1-score of under 1%, and when applied to other datasets, the performance further degraded. Despite efforts to optimize the hyperparameters, fault classification remained unsatisfactory. It is important to note that the model was configured to consider unbalanced data.

The inefficacy of the linear and sigmoid kernels in detecting meaningful patterns from the given dataset is evident in the obtained results. As indicated in Table 12 under result 4, the f1-score for these kernels was notably low. This challenges the suitability of these kernel functions for the specific classification task at hand. Linear kernels are inherently limited to linear separations, while sigmoid kernels may struggle with capturing intricate nonlinear relationships.

Table 12 – SVM Results

Id.	Algorithm	Split [Train/Test]	DS.	Sets	F1-Score	Recall	Precision
4	SVM (Linear/Sigmoid)	75%, 25%	2	B	0.00	0.00	0.00
5	SVM (RBF)	75%, 25%	2	B	0.07	0.04	0.50
6	SVM (Poly nº3)	75%, 25%	2	B	0.21	0.15	0.38
7	SVM (Poly nº4)	75%, 25%	2	B	0.28	0.19	0.56
8	SVM (Poly nº5)	75%, 25%	2	B	0.36	0.27	0.56
9	SVM (Poly nº6)	75%, 25%	2	B	0.39	0.47	0.33
10	SVM (Poly nº7)	75%, 25%	2	B	0.48	0.50	0.46

The second scenario was using RBF kernel and it is presented in Table 12 under result 5. this model's overall performance, as indicated by the f1-score, is quite low 0.07 (7%). The recall, which measures the ability of the model to capture positive instances, is also low at 0.04 (4%). On the positive side, the precision is relatively higher at 0.50 (50%), indicating that when the model predicts the positive class, it tends to be correct half of the time. In summary, the model's performance, especially in terms of f1-score and recall, suggests that there may be challenges in effectively capturing the faults events in the dataset.

In contrast, superior outcomes were attained by utilizing the polynomial kernel within the range of degrees from 3 to 7, as observed in instances 6 to 10. A notable observation is the direct correlation between the degree level and the computational demand, with increased training time corresponding to higher degrees. However, the model encountered convergence challenges beyond a degree level of 7.

The most commendable performance, denoted by an f1-score of 0.48 (48%), is highlighted in Table 12 under result 10. These findings underscore the proficiency of the SVM model employing the polynomial kernel, particularly within the degree range of 3 to 7, exhibiting the most promising f1-score performance. Nevertheless, it is imperative to carefully weigh the trade-off between performance gains and the associated increase in training time when deciding on the degree level for optimal model configuration.

Dense and Convolutional Autoencoder. Autoencoders are a type of neural network architecture used for unsupervised learning, particularly in the field of feature learning and data compression as it was detailed previously. There are two common types of autoencoders: dense autoencoders and convolutional autoencoders.

Dense autoencoders are effective in learning compact and dense representations of input data. They are commonly used for applications where the data has a tabular or vector-like structure, as it was applied with vectoralized data from de ECU. Convolutional autoencoders are particularly effective for capturing spatial hierarchies and patterns in data. They are widely used in image-related tasks but can also be applied to sequential data or any data with a grid-like structure.

Both dense and convolutional autoencoders excel at learning meaningful representations or features from input data. This capability is crucial in fault detection tasks, as it helps identify patterns indicative of engine knock. Autoencoders inherently perform dimensionality reduction, transforming high-dimensional input data into a lower-dimensional representation. This can be beneficial in highlighting relevant features while reducing noise. Despite the availability of diverse sets of variables, as presented in Table 4, utilizing fewer variables in the context of this machine learning model is counterproductive, as it contradicts the intrinsic nature of the model.

Guoqian Jiang et al. (2019) and Zilong and Wei (2018) have used this model as a base to increment its solution, creating multiscale CNN. The authors had vibration signals straightforwardly to be able to classify the type of faults. The data availability was different from theirs in these proposed methods using AE. The ECU centralized the signals, giving a different aspect of the sampling rate. Kefalas et al. (2021), on the other hand, uses pressure signals to analyze faults, but this type of sensor is not available in the vehicle. However, it applied the CNN followed by CWT. Other authors have applied variations of CNN and BPANN, but they used a high-frequency sampling rate in all cases.

Table 13 – Dense and Convolutional Results

Id.	Algorithm	Split [Train/Test]	DS.	Sets	F1-Score	Recall	Precision
11	Dense Autoencoder (0.29)	Healthy, Faulty	2	A	0.31	0.67	0.20
12	Dense Autoencoder (0.27)	Healthy, Faulty	2	A	0.32	0.82	0.20
13	Conv. Autoencoder (32)	Healthy, Faulty	2	A	0.53	0.79	0.39
14	Conv. Autoencoder (256)	Healthy, Faulty	2	A	0.55	0.72	0.44

In models 11 and 12 presented in Table 13, both implemented with Dense Autoencoder architecture, we applied varying threshold values to fine-tune the model's performance. For model 11, with a threshold of 0.29, the F1-score reached 0.31 (31%), indicating a balance between precision and recall. The recall, specifically at 0.67 (67%), suggests a decent ability to capture positive instances, while precision was 0.20, signifying that when the model predicted the positive class, it was correct (20%) of the

time.

In contrast, model 12 utilized a slightly lower threshold of 0.27. This adjustment resulted in an improved F1-score of 0.32 (32%), implying a slightly better overall balance between precision and recall. Notably, the recall surged to 0.82 (82%), indicating a higher capability to capture positive instances. However, precision remained at 0.20 (20%), suggesting consistency in correctly identifying the positive class but with a moderate overall precision.

In models 13 and 14 also presented in Table 13, both implemented with the Convolutional Autoencoder architecture, we explored the impact of varying window sizes on the model's performance. Model 13, with a window size of 32, exhibited an F1-score of 0.53 (53%), indicating a balanced trade-off between precision and recall. The recall, at 0.79 (79%), suggests a strong ability to capture positive instances, while precision was 0.39 (39%), reflecting the correctness of predictions when the model identified the positive class.

In contrast, model 14 utilized a larger window size of 256. This adjustment led to an improved F1-score of 0.55 (55%), signifying an enhanced balance between precision and recall. Although the recall slightly decreased to 0.72 (72%), the model demonstrated a substantial boost in precision, reaching 0.54 (54%). This suggests that, with a larger batch size, the model exhibited a more accurate identification of the positive class, resulting in increased precision.

Classifier. A classifier is a machine learning model that assigns labels to input data based on patterns learned during training. It predicts unseen data by transforming features into class predictions through interconnected neurons arranged in layers. Based on that, it was created the Architecture I (Arc I) presented in Table 9 that considers the batch size 128 and sequence length of 64, and another variation with the batch size of 64 and sequence length of 32. Another architecture was created (Arc II) as presented in Table 10 to analyze some defined aspects, considering batch size of 64 and sequence length of 32.

Table 14 – Classifier Results I

Id.	Algorithm	Split [Train/Test]	DS.	Sets	F1-Score	Recall	Precision
15	Classifier - Arc I (128/64)	80%, 20%	1-4	D	0.49	0.40	0.61
16	Classifier - Arc I (128/64)	50%, 50%	1-4	D	0.20	0.79	0.11
17	Classifier - Arc I (128/64)	80%, 20%	2-4	D	0.63	0.52	0.79
18	Classifier - Arc I (128/64)	50%, 50%	2-4	D	0.21	0.79	0.12

First, the splitting process was analyzed as shown in Table 14. Instances 15 and 17, with an 80/20 split, generally show better f1-scores than those with a 50/50 split (instances 16 and 18). There is a notable performance difference between datasets 1-4 and 2-4, indicating that the choice of dataset has a more significant impact when splitting the dataset into 80/20 than 50/50. Thus, the connection between splitting the

data of the datasets directly affects the model's performance, as demonstrated in the results from instances 15 to 18. Instances with higher recall (instances 16 and 18) tend to have lower precision, suggesting a trade-off between correctly identifying positive instances and minimizing false positives.

Table 15 – Classifier Results II

Id.	Algorithm	Split [Train/Test]	DS.	Sets	F1-Score	Recall	Precision
19	Classifier - Arc I (128/64)	80%, 20%	1-6	D	0.81	0.83	0.80
20	Classifier - Arc I (128/64)	50%, 50%	1-6	D	0.26	0.79	0.16
21	Classifier - Arc I (128/64)	80%, 20%	1-6	E	0.12	0.13	0.12
22	Classifier - Arc I (128/64)	50%, 50%	1-6	E	0.05	0.04	0.09
23	Classifier - Arc I (64/32)	80%, 20%	1-6	D	0.45	0.46	0.44
24	Classifier - Arc I (64/32)	80%, 20%	1-6	E	0.00	0.00	0.00
25	Classifier - Arc II (64/32)	80%, 20%	1-6	D	0.12	0.13	0.12

Now focusing on all datasets 1-6, as shown in Table 15, it is still notable that the influence of the splitting process in the model. Thus, the model has fewer aspects to capture standards and be generalized. This can be seen in the instances 19 and 20, 21 and 22. This process directly affects the precision, dropping from 0.80 (80%) to 0.16 (16%) in instances 19 and 20. Instance 19 achieved the highest result of 81% f1-score considering the Arc I, splitting 80/20, and set D. Another experiment was conducted using the set of variables E, representing the lowest f1-score so far.

In instances 23 and 24, it was made a change in the batch size from 128 to 64, and the sequence length from 64 to 32. The change was implemented to investigate the impact of using a smaller batch size in the context of knock detection. When comparing instances 19 and 23, it is clear that it significantly impacts recall and precision, leading to a smaller f1-score. The new architecture (Arc II), previously created, was used to check its performance. Comparing instances 23 and 25, it showcases a decrease in its performance.

Table 16 – Classifier Results III

Id.	Algorithm	Split [Train/Test]	DS.	Sets	F1-Score	Recall	Precision
26	Classifier - Arc I (128/64)	80%, 20%	6	D	0.28	0.50	0.19
27	Classifier - Arc I (128/64)	50%, 50%	6	D	0.14	0.26	0.10
28	Classifier - Arc I (128/64)	80%, 20%	4	D	0.50	0.36	0.81
29	Classifier - Arc I (128/64)	50%, 50%	4	D	0.01	0.01	0.03
30	Classifier - Arc I (128/64)	80%, 20%	1-6	B	0.72	0.78	0.66
31	Classifier - Arc I (128/64)	80%, 20%	1-6	C	0.78	0.80	0.77
32	Classifier - Arc I (128/64)	80%, 20%	1-6	F	0.12	0.12	0.11

The temporal aspect was given up since the datasets were aggregated in the previous experiments. In the following experiments, it was tested only with single dataset 6 on instances 26 and 27, and dataset 4 on instances 28 and 29, to understand the effects on performance. These instances were using the same set (D) of variables. The 50/50 splitting process still harms the performance, as it is possible to see in Table 16,

the f1-score dropped by half of the score from the instances 26 to 27. It got the worst performance considering using dataset 4. Thus, it is clear that the temporal aspect does not influence this model to achieve high performance for detecting engine knock.

In the last experiments, it was tested different sets of data B, C, and F, which represent instances 30, 31, and 32. The set of variables plays a crucial role in performance, with Set C (combining signals, including knock signals) outperforming both Sets B (only knock signals) and F (equal to C but without knock signals). Instances with more signals, especially those including knock signals (instances 30 and 31), tend to perform better than those without (instance 32). This assumption was previously proven as presented in the experiments that used set E (set D without knock) in instances 21 and 22. Thus, instances 30 and 31 achieved high f1-score 0.72 (72%) and 0.78 (78%).

5.2 COMPARISON BETWEEN ALGORITHMS

All the results are summarized and presented in a single Table 17, where the classifier achieved the best result with the knowledge-driven variables (Set D). It obtained an f1-score of 81%, indicating a significant improvement in engine knock detection compared to using only the signals, which achieved 72% f1-score. This classifier is also highlighted in the instances 19, 30, and 31.

Figure 11 illustrates the performance of various models. The convolutional autoencoder achieved an f1-score of 55%, followed by the SVM with 48%, the dense autoencoder with 32%, and the isolated forest with 31%. Furthermore, the recall results varied between 50% and 83%, indicating the models' ability to identify positive instances correctly presented at Figure 12. The precision results ranged from 20% to 80%, representing the models' ability to provide accurate positive results presented at Figure 13.

Based on the presented experimental results, several assumptions can be made:

- The Classifier - Architecture I model with a batch size of 128 and sequence length of 64 performs better on Set D when the dataset includes samples from 1 to 6 (instance 19). This configuration achieves a high f1-score of 0.81 (81%), indicating a good balance between precision and recall. However, when the dataset is limited to samples from 1 to 4, the model's performance drops significantly (instance 15).
- The performance of the Classifier - Architecture I model is lower on Set E than on Set D. The f1-scores are very low across different dataset configurations and splits (results 21, 22, and 24). It suggests the model needs the knock signals in Set E to generalize and detect accurately.
- Both results of instances 19 and 20, which have an 80/20 split, have a higher f1-score than those of instances 15 and 16, which have a 50%/50% split. This

suggests that the models trained on the larger training portion (80%) perform better overall accuracy, as indicated by the f1-score.

- The results with a batch size of 128 (result of instance in 15 and 19) generally outperform those with a batch size of 64 (result in 23) regarding f1-score, recall, and precision. Using a larger batch size contributes to better model performance in this analysis.
- Based on these results, the Convolutional Autoencoder performs better on Set A than the Dense Autoencoder. It has higher f1-score and recall values, suggesting better performance in identifying faulty instances. However, the precision values for both autoencoder models are relatively low, indicating a higher rate of false positives.

Based on the presented related works and experimental results, some additional assumptions can be made:

- Data-driven models rely on the availability of a quality dataset, and achieving exceptional performance is contingent upon a meticulous acquisition process, thorough data preparation, and effective training.
- Classifiers can detect Engine Knock in low-frequency using ECU's signal with f1-score of 81%.

Overall, the classifier with the knowledge-driven variables (Set D) demonstrated the best performance, achieving high f1-score, recall, and precision values. The convolutional autoencoder, SVM, dense autoencoder, and isolated forest also showed varying performance levels, with the convolutional autoencoder performing the best among them. These results highlight the effectiveness of the selected classifier and its potential for engine knock detection.

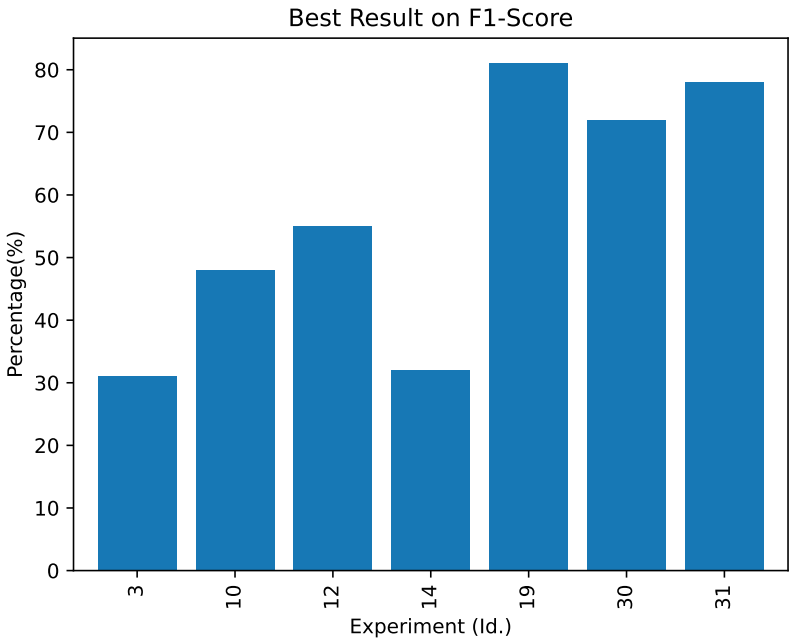

Figure 11 – Results of the Best F1-Score

Figure 12 – Results of the Best Recall

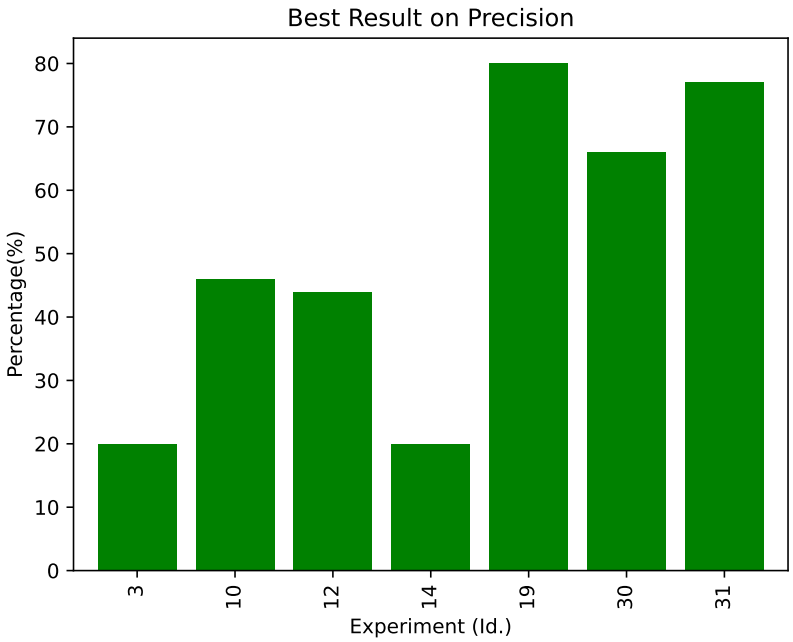

Figure 13 – Results of the Best Precision

Table 17 – Results of the All Experiments

Id.	Algorithm	Split [Train/Test]	DS.	Sets	F1-Score	Recall	Precision
1	Isolated Forest	75%, 25%	2	A	0.22	0.37	0.15
2	Isolated Forest	75%, 25%	2	A	0.26	0.42	0.19
3	Isolated Forest	75%, 25%	2	B	0.31	0.67	0.20
4	SVM (Linear/Sigmoid)	75%, 25%	2	B	0.00	0.00	0.00
5	SVM (RBF)	75%, 25%	2	B	0.07	0.04	0.50
6	SVM (Poly nº3)	75%, 25%	2	B	0.21	0.15	0.38
7	SVM (Poly nº4)	75%, 25%	2	B	0.28	0.19	0.56
8	SVM (Poly nº5)	75%, 25%	2	B	0.36	0.27	0.56
9	SVM (Poly nº6)	75%, 25%	2	B	0.39	0.47	0.33
10	SVM (Poly nº7)	75%, 25%	2	B	0.48	0.50	0.46
11	Conv. Autoencoder (32)	Healthy, Faulty	2	A	0.53	0.79	0.39
12	Conv. Autoencoder (256)	Healthy, Faulty	2	A	0.55	0.72	0.44
13	Dense Autoencoder (0.29)	Healthy, Faulty	2	A	0.31	0.67	0.20
14	Dense Autoencoder (0.27)	Healthy, Faulty	2	A	0.32	0.82	0.20
15	Classifier - Arc I (128/64)	80%, 20%	1-4	D	0.49	0.40	0.61
16	Classifier - Arc I (128/64)	50%, 50%	1-4	D	0.20	0.79	0.11
17	Classifier - Arc I (128/64)	80%, 20%	2-4	D	0.63	0.52	0.79
18	Classifier - Arc I (128/64)	50%, 50%	2-4	D	0.21	0.79	0.12
19	Classifier - Arc I (128/64)	80%, 20%	1-6	D	0.81	0.83	0.80
20	Classifier - Arc I (128/64)	50%, 50%	1-6	D	0.26	0.79	0.16
21	Classifier - Arc I (128/64)	80%, 20%	1-6	E	0.12	0.13	0.12
22	Classifier - Arc I (128/64)	50%, 50%	1-6	E	0.05	0.04	0.09
23	Classifier - Arc I (64/32)	80%, 20%	1-6	D	0.45	0.46	0.44
24	Classifier - Arc I (64/32)	80%, 20%	1-6	E	0.00	0.00	0.00
25	Classifier - Arc II (64/32)	80%, 20%	1-6	D	0.12	0.13	0.12
26	Classifier - Arc I (128/64)	80%, 20%	6	D	0.28	0.50	0.19
27	Classifier - Arc I (128/64)	50%, 50%	6	D	0.14	0.26	0.10
28	Classifier - Arc I (128/64)	80%, 20%	4	D	0.50	0.36	0.81
29	Classifier - Arc I (128/64)	50%, 50%	4	D	0.01	0.01	0.03
30	Classifier - Arc I (128/64)	80%, 20%	1-6	B	0.72	0.78	0.66
31	Classifier - Arc I (128/64)	80%, 20%	1-6	C	0.78	0.80	0.77
32	Classifier - Arc I (128/64)	80%, 20%	1-6	F	0.12	0.12	0.11

6 CONCLUSION

This study focused on detecting engine knock, a common fault in ICE, using a data-driven approach based on machine learning techniques. The automotive industry is vested in minimizing engine knock effects to reduce maintenance costs and prolong engine life. By leveraging the vast amount of data collected from a Renault Sandero car's ECU, we aimed to develop an effective method for detecting engine knock.

Through extensive experimentation and analysis, we explored various machine learning algorithms, including Classifier, Dense and Convolutional Autoencoders, SVM, and Isolated Forest. Training and evaluation were conducted on a comprehensive dataset comprising 19 variables extracted from the vehicle across 32 experiments. Our primary goal was to attain robust fault detection capabilities while minimizing false positives at a low-frequency rate utilizing the IASE Hardware.

Based on our exploration of related works, no existing research parallels the scope and methodology employed in this study. This distinctive approach contributes novel insights to the field of fault detection in vehicular systems. This comprehensive analysis and novel approach position our work as a valuable contribution to the existing body of knowledge in the domain, offering fresh perspectives and potential avenues for future research.

Our results demonstrate an impressive engine knock detection rate of 81%, surpassing limitations imposed by XCP on the sampling rate. We achieved a performance comparable to the literature, even at high frequencies.

We propose future improvements, expanding applications that use ML inside IASE hardware on vehicles. Additionally, include other sensing data from the vehicle. By leveraging additional engine variables, we aim to develop a comprehensive detection system that does not solely rely on the piezoelectric sensor. This approach will enable us to capture multiple dimensions of engine behavior and cross-validate the knock detection results. Considering a broader range of data inputs, we anticipate improved accuracy in identifying engine knock events and preventing false positives. Incorporating these multi-modal sensing techniques into the detection system will pave the way for more robust and reliable fault detection processes in automotive vehicles.

In conclusion, this study demonstrates the potential of machine learning techniques for engine knock detection in automotive vehicles. Although there is room for further improvement, our results highlight the importance of leveraging data-driven approaches to enhance the safety and performance of internal combustion engines. We hope our findings contribute to the ongoing research efforts in the automotive industry and inspire future advancements in fault detection and prevention systems.

REFERENCES

(IEA), International Energy Agency. **Oil 2021 Analysis and forecast to 2026**. [S.l.]: IEA, 2021. Accessed: March 2, 2023. Available from: https://www.iea.org/reports/oil-2021.

ABID, Anam; KHAN, Muhammad; IQBAL, Javaid. A review on fault detection and diagnosis techniques: basics and beyond. **Artificial Intelligence Review**, v. 54, p. 1–26, June 2021.

AHMED, Ryan; EL SAYED, Mohammed; GADSDEN, S. Andrew; TJONG, Jimi; HABIBI, Saeid. Automotive Internal-Combustion-Engine Fault Detection and Classification Using Artificial Neural Network Techniques. **IEEE Transactions on Vehicular Technology**, v. 64, n. 1, p. 21–33, 2015.

ALPAYDIN, Ethem. **Introduction to machine learning**. [S.l.]: MIT press, 2010.

BEDRETCHUK, João Paulo; ARRIBAS GARCÍA, Sergio; NOGIRI IGARASHI, Thiago; CANAL, Rafael; WEDDERHOFF SPENGLER, Anderson; GRACIOLI, Giovani. Low-Cost Data Acquisition System for Automotive Electronic Control Units. **Sensors**, v. 23, n. 4, 2023. ISSN 1424-8220.

BOUBAI, O. Knock detection in automobile engines. **IEEE Instrumentation Measurement Magazine**, v. 3, n. 3, p. 24–28, 2000.

BREIMAN, Leo; FRIEDMAN, JH; STONE, CJ; OLSHEN, RA. **Classification and regression trees**. [S.l.]: CRC press, 1984.

CHO, Kyunghyun; MERRIENBOER, Bart van; GULCEHRE, Caglar; BOUGARES, Fethi; SCHWENK, Holger; BENGIO, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014.

DOORNBOS, Gerben; HEMDAL, Stina; DENBRATT, Ingemar; DAHL, Daniel. Knock Phenomena under Very Lean Conditions in Gasoline Powered SI-Engines. **SAE International Journal of Engines**, SAE International, v. 11, n. 1, p. 39–54, 2018. ISSN 19463936, 19463944.

EU PARLIAMENT. **Directive (EU) 2018/2001 of the European Parliament and of the Council of 11 December 2018 on the promotion of the use of energy from renewable sources (recast)**. v. L 328. [S.l.]: Official Journal of the European Union, Dec. 2018. P. 82–209. Accessed: March 2, 2023. Available from: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32018L2001.

FALCÃO, Eduardo; BARROS, Péricles R.; MELO, Vinicius Mafra. A Model Based Design Approach For Knock Control in Internal Combustion Engines Using Machine Learning. Anais do 14º Simpósio Brasileiro de Automação Inteligente; Ouro Preto.Minas Gerais.Brasil. Campinas : Galoá; 2019, v. 1, 2019.

FIOLKA, J. A Fast Method For Knock Detection Using Wavelet Transform. **Proceedings of the International Conference Mixed Design of Integrated Circuits and System, 2006. MIXDES 2006.**, p. 621–626, 2006.

FIRMINO, João L.; NETO, João M.; OLIVEIRA, Andersson G.; SILVA, José C.; MISHINA, Koje V.; RODRIGUES, Marcelo C. Misfire detection of an internal combustion engine based on vibration and acoustic analysis. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, v. 43, n. 336, 2021. ISSN 1806-3691.

FRÖHLICH, Antônio Augusto. SmartData: an IoT-ready API for sensor networks. **International Journal of Sensor Networks**, Inderscience Publishers (IEL), v. 28, n. 3, p. 202–210, 2018.

GAO, Zhiwei; CECATI, Carlo; DING, Steven X. A survey of fault diagnosis and fault-tolerant techniques—Part I: Fault diagnosis with model-based and signal-based approaches. **IEEE Transactions on Industrial Electronics**, IEEE, v. 62, n. 6, p. 3757–3767, 2015.

GAO, Zhiwei; CECATI, Carlo; DING, Steven X. A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part II: Fault Diagnosis With Knowledge-Based and Hybrid/Active Approaches. **IEEE Transactions on Industrial Electronics**, v. 62, n. 6, p. 3768–3774, 2015.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. [S.l.]: Springer Science & Business Media, 2009.

HEINZ, Decker; HANS-ULRICH, Gruber. Knock Control of Gasoline Engines—A Comparison of Solutions and Tendencies, with Special Reference to Future European Emission Legislation. **SAE Transactions**, SAE International, v. 94, p. 806–813, 1985. ISSN 0096736X.

HEYWOOD, John B. **Internal Combustion Engine Fundamentals**. [S.l.]: McGraw-Hill, 1988. P. 323–325.

HORNER, Thomas G. **Engine Knock Detection Using Spectral Analysis Tecniques With a TMS320 DSP**. [S.l.: s.n.], 1995. P. 1–4. (SPRA039).

INCE, Turker; KIRANYAZ, Serkan; EREN, Levent; ASKAR, Murat; GABBOUJ, Moncef. Real-time motor fault detection by 1-d convolutional neural networks. **IEEE Transactions on Industrial Informatics**, IEEE, v. 12, n. 5, p. 1865–1874, 2016.

JAFARIAN, Kamal; DARJANI, Morteza; HONARKAR, Zahra. Vibration analysis for fault detection of automobile engine using PCA technique, p. 372–376, 2016.

JANAKIRAMAN, Vijay Manikandan; NGUYEN, XuanLong; STERNIAK, Jeff; ASSANIS, Dennis. Identification of the Dynamic Operating Envelope of HCCI Engines Using Class Imbalance Learning. **IEEE Transactions on Neural Networks and Learning Systems**, v. 26, n. 1, p. 98–112, 2015.

JIA, Feng; LEI, Yaguo; LIN, Jing; ZHOU, Xin; LU, Na. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. **Mechanical Systems and Signal Processing**, v. 72-73, p. 303–315, 2016. ISSN 0888-3270.

JIANG, Guoqian; HE, Haibo; YAN, Jun; XIE, Ping. Multiscale Convolutional Neural Networks for Fault Diagnosis of Wind Turbine Gearbox. **IEEE Transactions on Industrial Electronics**, v. 66, n. 4, p. 3196–3207, 2019.

JIANG, Hongkai; LI, Xingqiu; SHAO, Haidong; ZHAO, Ke. Intelligent fault diagnosis of rolling bearings using an improved deep recurrent neural network. **Measurement Science and Technology**, IOP Publishing, v. 29, n. 6, p. 065107, May 2018.

KALGHATGI, Gautam. Is it really the end of internal combustion engines and petroleum in transport? **Applied Energy**, v. 225, p. 965–974, 2018. ISSN 0306-2619.

KEFALAS, Achilles; OFNER, Andreas B.; PIRKER, Gerhard; POSCH, Stefan; GEIGER, Bernhard C.; WIMMER, Andreas. Detection of Knocking Combustion Using the Continuous Wavelet Transformation and a Convolutional Neural Network. **Energies**, v. 14, n. 2, 2021. ISSN 1996-1073.

KIRSANOVS, Vladimirs; BARISA, Aiga; SAFRONOVA, Alina. Cost-benefit assessment of electric vehicle vs internal combustion engine in Latvia, p. 1–4, 2020.

KOSENOK, B.B.; BALYAKIN, V.B. Optimization of Dynamic Characteristics of an Internal Combustion Engine with Opposing Pistons, p. 1–7, 2020.

KRUMMENACHER, Gabriel; ONG, Cheng Soon; KOLLER, Stefan; KOBAYASHI, Seijin; BUHMANN, Joachim M. Wheel Defect Detection With Machine Learning. **IEEE Transactions on Intelligent Transportation Systems**, v. 19, n. 4, p. 1176–1187, 2018.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LEVINE, Sergey; FINN, Chelsea; DARRELL, Trevor; ABBEEL, Pieter. End-to-end training of deep visuomotor policies. **The Journal of Machine Learning Research**, JMLR. org, v. 17, n. 1, p. 1334–1373, 2016.

LIU, Chengcai; GAO, Qing; JIN, Ying-ai; YANG, Wenhong. Application of wavelet packet transform in the knock detection of gasoline engines, p. 686–690, 2010.

LU, Peng; VAN KAMPEN, Erik-Jan; DE VISSER, Cornelis; CHU, Qiping. Aircraft fault-tolerant trajectory control using Incremental Nonlinear Dynamic Inversion. **Control Engineering Practice**, v. 57, p. 126–141, 2016. ISSN 0967-0661.

LUJÁN, José Manuel; CLIMENT, Héctor; GARCÍA-CUEVAS, Luis Miguel; MORATAL, Ausias. Volumetric efficiency modelling of internal combustion engines based on a novel adaptive learning algorithm of artificial neural networks. **Applied Thermal Engineering**, v. 123, p. 625–634, 2017. ISSN 1359-4311.

MAHENDAR, Senthil Krishnan. **Mitigating Knock in Heavy Duty Spark Ignition Engines**. 2021. PhD dissertation – KTH Royal Institute of Technology.

MILLO, F; FERRARO, CV. Knock in SI engines: a comparison between different techniques for detection and control. **SAE transactions**, JSTOR, p. 1091–1112, 1998.

MITCHELL, T. **Machine learning**. [S.l.]: McGraw Hill, 1997.

MOTAHARI, S.; CHITSAZ, Iman. Effects of Altitude and Temperature on the Performance and Efficiency of Turbocharged Direct Injection Gasoline Engine. **Journal of Applied Fluid Mechanics**, v. 12, p. 1825–1836, Nov. 2019.

NAIR, Vishal Vijayan; KOUSTUBH, Boppudi Pranava. Data analysis techniques for fault detection in hybrid/electric vehicles. **2017 IEEE Transportation Electrification Conference (ITEC-India)**, p. 1–5, 2017.

PAN, Honghu; HE, Xingxi; TANG, Sai; MENG, Fanming. An Improved Bearing Fault Diagnosis Method using One-Dimensional CNN and LSTM. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, v. 64, n. 7, p. 443–452, 2018.

PANZANI, Giulio; ÖSTMAN, Fredrik; ONDER, Christopher H. Engine Knock Margin Estimation Using In-Cylinder Pressure Measurements. **IEEE/ASME Transactions on Mechatronics**, v. 22, n. 1, p. 301–311, 2017.

PARK, You-Jin; FAN, Shu-Kai S.; HSU, Chia-Yu. A Review on Fault Detection and Process Diagnostics in Industrial Processes. **Processes**, v. 8, n. 9, 2020. ISSN 2227-9717.

PETRUCCI, Luca; RICCI, Federico; MARIANI, Francesco; CRUCCOLINI, Valentino; VIOLI, Mattia. Engine Knock Evaluation Using a Machine Learning Approach. SAE International, Sept. 2020. ISSN 0148-7191.

QIN, S. Joe. Data-driven Fault Detection and Diagnosis for Complex Industrial Processes. **IFAC Proceedings Volumes**, v. 42, n. 8, p. 1115–1125, 2009. 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes. ISSN 1474-6670.

QUINLAN, J. Ross. Induction of decision trees. **Machine learning**, Springer, v. 1, n. 1, p. 81–106, 1986.

SAIBANNAVAR, Deepa; MATH, Mallikarjun M.; KULKARNI, Umakant. A Survey on On-Board Diagnostic in Vehicles. In: RAJ, Jennifer S. (Ed.). **International Conference**

on Mobile Computing and Sustainable Informatics. Cham: Springer International Publishing, 2021. P. 49–60.

SAMIMY, B.; RIZZONI, G. Mechanical signature analysis using time-frequency signal processing: application to internal combustion engine knock detection. **Proceedings of the IEEE**, v. 84, n. 9, p. 1330–1343, 1996.

SANGHA, M.S.; GOMM, J.B.; YU, D.L.; PAGE, G.F. FAULT DETECTION AND IDENTIFICATION OF AUTOMOTIVE ENGINES USING NEURAL NETWORKS. **IFAC Proceedings Volumes**, v. 38, n. 1, p. 272–277, 2005. 16th IFAC World Congress. ISSN 1474-6670.

SHAHID, Syed Maaz Shahid; KO, Sunghoon; KWON, Sungoh. Real-time abnormality detection and classification in diesel engine operations with convolutional neural network. **Expert Systems with Applications**, v. 192, p. 116233, 2022. ISSN 0957-4174.

VIJAYAN, Arunkumar; TAHOORI, Mehdi B.; KINTZLI, Ewald; LOHMANN, Timm; HANS HANDL, Juergen. A Data-driven Approach for Fault Detection in the Alternator Unit of Automotive Systems, p. 1–4, 2022.

YAN, Weizhong; YU, Lijie. On Accurate and Reliable Anomaly Detection for Gas Turbine Combustors: A Deep Learning Approach. arXiv, 2019.

YIN, Shen; DING, Steven X.; XIE, Xiaochen; LUO, Hao. A Review on Basic Data-Driven Approaches for Industrial Process Monitoring. **IEEE Transactions on Industrial Electronics**, v. 61, n. 11, p. 6418–6428, 2014.

ZHENG, Zhihao; LIN, Xiaodong; YANG, Ming; HE, Zeming; BAO, Ergude; ZHANG, Hang; TIAN, Zhenyu. Progress in the Application of Machine Learning in Combustion Studies. **ES Energy & Environment**, v. 9, p. 1–14, 2020. ISSN 2576-9898.

ZILONG, Zhuang; WEI, Qin. Intelligent fault diagnosis of rolling bearing using one-dimensional multi-scale deep convolutional neural network based health state classification, p. 1–6, 2018.