UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Mecânica

LUÍS OTÁVIO GARAVASO

Enhancing LightGBM Fault Prediction in Rotating Machines using Knowledge-based Resampling and Bayesian Optimization

Aprimorando a Previsão de Falhas em Máquinas Rotativas com LightGBM usando Reamostragem Baseada em Conhecimento em Otimização Bayesiana

Campinas

2024

LUÍS OTÁVIO GARAVASO

Enhancing LightGBM Fault Prediction in Rotating Machines using Knowledge-based Resampling and Bayesian Optimization

Aprimorando a Previsão de Falhas em Máquinas Rotativas com LightGBM usando Reamostragem Baseada em Conhecimento em Otimização Bayesiana

Dissertation presented to the School of Mechanical Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Mechanical Engineering, in the area of Solids and Mechanical Projects.

Dissertação de Mestrado apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Mecânica, na Área de Mecânica dos Sólidos e Projeto Mecânico.

Orientador: Profa. Dra. Katia Lucchesi Cavalca Dedini

Coorientador: Prof. Dr. Gregory Bregion Daniel

ESTE TRABALHO CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DE MESTRADO DEFENDIDA PELO ALUNO LUÍS OTÁVIO GARAVASO, E ORIENTADA PELO PROFA. DRA. KATIA LUCCHESI CAVALCA DEDINI.

Campinas

2024

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

Garavaso, Luís Otávio, 1998-
G162e Enhancing LightGBM fault prediction in rotating machines using knowledge-based resampling and bayesian optimization / Luís Otávio Garavaso. – Campinas, SP : [s.n.], 2024.

Orientador: Katia Lucchesi Cavalca Dedini.
Coorientador: Gregory Bregion Daniel.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Rotores - Dinâmica. 2. Rolamento de esferas. 3. Otimização bayesiana. 4. Aprendizado de máquina. 5. Inteligência artificial. I. Dedini, Katia Luccesi Cavalca, 1963-. II. Daniel, Gregory Bregion, 1984-. III. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. IV. Título.

Informações Complementares

Título em outro idioma: Aprimorando a previsão de falhas em máquinas rotativas com LightGBM usando reamostragem baseada em conhecimento em otimização bayesiana
Palavras-chave em inglês:
Rotors - Dynamics
Ball-bearings
Bayesian optimization
Machine learning
Artificial intelligence
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Mestre em Engenharia Mecânica
Banca examinadora:
Katia Lucchesi Cavalca Dedini [Orientador]
João Paulo Dias
Mateus Giesbrecht
Data de defesa: 12-01-2024
Programa de Pós-Graduação: Engenharia Mecânica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-3624-8134
- Currículo Lattes do autor: https://lattes.cnpq.br/4386613644718863

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

DISSERTAÇÃO DE MESTRADO ACADÊMICO

Enhancing LightGBM Fault Prediction in Rotating Machines using Knowledge-based Resampling and Bayesian Optimization

Aprimorando a Previsão de Falhas em Máquinas Rotativas com LightGBM usando Reamostragem Baseada em Conhecimento em Otimização Bayesiana

Autor: Luís Otávio Garavaso

Orientador: Profa. Dra. Katia Lucchesi Cavalca Dedini
Coorientador: Prof. Dr. Gregory Bregion Daniel

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação de Mestrado:

Profa. Dra. Katia Lucchesi Cavalca Dedini
DSI/FEM/UNICAMP

Prof. Dr. Mateus Giesbrecht
DEEM/FEEC/UNICAMP

Prof. Dr. João Paulo Dias
Civil and Mechanical Engineering/Shippensburg University

A Ata de Defesa com as respectivas assinaturas dos membros encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 12 de Janeiro de 2024

RESUMO

Máquinas rotativas são necessárias para a geração de energia, pois desempenham papéis vitais que se estendem desde a extração de recursos, como combustíveis, até a conversão de energia cinética da água e do fluxo de ar em eletricidade para consumo. No entanto, essas máquinas enfrentam diferentes tipos de falhas mecânicas que alteram a resposta de vibração do sistema, criando padrões conhecidos como assinaturas de falhas. Portanto, interpretar essas marcas de vibração com algoritmos de aprendizado de máquina leva à identificação adequada de falhas, melhorando o agendamento de manutenção e reduzindo o tempo de reparo e a quebra de maquinário. Esta pesquisa utiliza o modelo Light Gradient Boosting Machine (LightGBM) para identificar defeitos em rolamentos de esferas por meio de sinais de vibração. Os dados são extraídos do Repositório da Universidade de Paderborn e contêm amostras de uma máquina rotativa sujeita a três condições operacionais: saudável, com uma falha na pista interna e uma falha na pista externa. Uma técnica de reamostragem de dados baseada em frequências específicas de rolamentos é proposta para aprimorar as capacidades preditivas do LightGBM. Além disso, o algoritmo de otimização Bayesiana é usado para ajustar os hiperparâmetros do modelo e maximizar a acurácia e o recall.

Palavras–chave: Dinâmica de Rotores, Otimização Bayesiana, Light Gradient Boosting Machine, Reamostragem de Dados, Identificação de Falhas

ABSTRACT

Rotating machines are necessary for power generation since they perform vital roles that extend from extracting resources, such as fuels, to converting kinetic energy from water and airflow into electricity for consumption. These machines, however, face different types of mechanical faults that alter the system vibration response, creating patterns known as fault signatures. Therefore, interpreting these vibration marks with machine learning algorithms leads to proper fault identification, improving maintenance scheduling, and reducing repair time and machinery breakage. This research uses the Light Gradient Boosting Machine (LightGBM) model to identify ball-bearing defects through vibration signals. The data is extracted from the Paderborn University Repository and contains samples of a rotating machine subject to three operational conditions: healthy, with an inner race, and an outer race fault. A data resampling technique based on specific rolling bearing frequencies is proposed to enhance LightGBM's predictive capabilities. Furthermore, the Bayesian optimization algorithm is used to fine-tune the model hyperparameters and maximize accuracy and recall.

Keywords: Rotordynamics, Bayesian optimization, Light Gradient Boosting Machine, Data Resampling, Fault Identification

LIST OF ABBREVIATIONS AND ACRONYMS

ANN	Artificial Neural Network
BPFI	Ball Pass Frequency Inner race
BPFO	Ball Pass Frequency Outer race
GBM	Gradient Boosting Machine
LGBM	Light Gradient Boosting Machine
LightGBM	Light Gradient Boosting Machine
RBF	Radial Basis Function
SVM	Support Vector Machine
UNICAMP	State University of Campinas

LIST OF SYMBOLS

D	Pitch circle diameter
n	Number of rolling elements
ϕ	Contact angle
d	Rolling element diameter
N	Number of training samples
M_i	Number of minimum features to assure the BPFI
M_o	Number of minimum features to assure the BPFO
M_r	Number of minimum features to assure a full bearing rotation
C_l	Number of target classes
C	SVM regularization factor
y	Actual class label
p	Probability of the predicted class
$\hat{y}$	Predicted class label
i	Samples index
c	Classes index
s	Sign vector or function
γ	Similarity normalization factor
K	Kernel function
M	Number of SVM machines trained
f	Black-box function
f^*	Black-box function best value during optimization

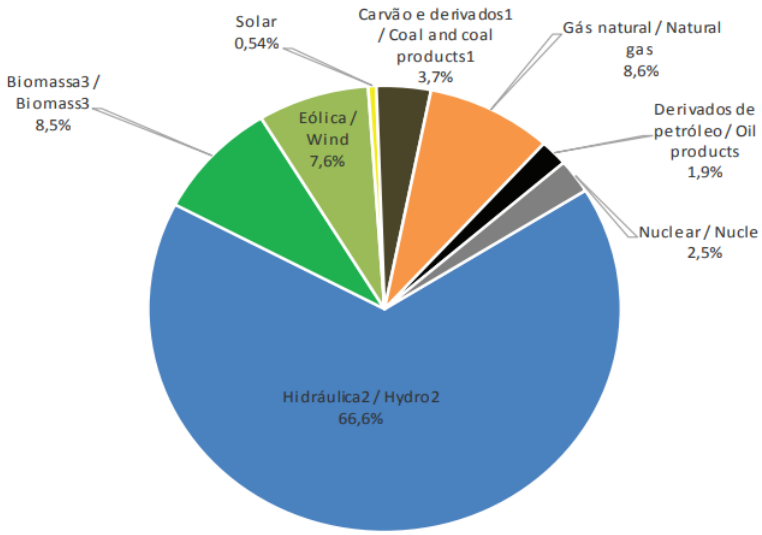
EI	Expected Improvement acquisition function
x	Black-box function evaluation point

CONTENTS

1 INTRODUCTION

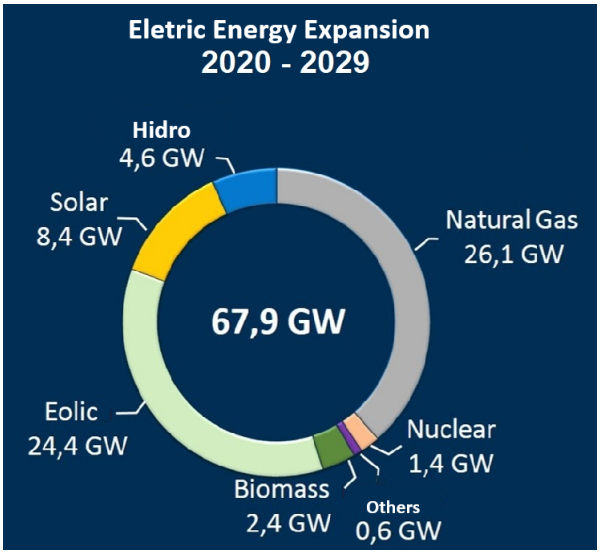
In the National Energy Balance of 2019 (Brasil, 2019a), it was observed that the Brazilian energy matrix has 153 GW of installed capacity. This value is distributed across various forms of generation, as shown in Figure 1.1.

Figure 1.1 – Distribution of the Brazilian Energy Matrix (Brasil, 2019a)



However, as illustrated in Figure 1.2, the Ten-Year Energy Expansion Plan (Brasil, 2019b) predicts an expansion of the system by 68 GW, representing an investment of R$ 2.3 trillion over 10 years, with R$ 1.9 trillion in oil, natural gas, and biofuels and R$ 456 billion in electricity generation and transmission.

Figure 1.2 – Projected Expansion of the Brazilian Energy Matrix by 2029 (Brasil, 2019b)



The extraction of these resources and their transformation into energy for the end consumer depend directly on rotating machinery, such as hydraulic turbines, compressors, and reducers. Therefore, system failures can result in significant economic losses and undesirable societal disruptions, and they must be identified and repaired promptly. One of the tools that can be used for this identification is machine learning.

Machine learning is the field of study that enables computers to perform tasks without being explicitly programmed. This definition was proposed by the American computer scientist Arthur Lee Samuel in 1959 after developing an algorithm capable of learning to play checkers and beating a human opponent (Samuel, 1959).

In classical programming, the programmer can describe rules to the computer through lines of code, where for each input, an output is obtained based on what the machine has been programmed to interpret. On the other hand, with machine learning, the programmer knows the inputs and outputs but not the rules that perform this translation. A practical example is an autonomous car, with specifications including the origin, route, and destination that the vehicle should follow. However, programming all the necessary rules to guide the car during its operation, considering adverse and unforeseen conditions, surpasses the capabilities of classical programming.

This limitation can be overcome by creating algorithms that analyze a significant amount of data with the help of mathematical and statistical methods. Thus, when the model becomes capable of providing reliable results, it can be said that the algorithm has learned, through mathematical methods, to transform input data into output, similar to intelligence, giving rise to the field known as Artificial Intelligence, which, according to Harvard Business School (Wellers *et al.*, 2017), will be worth $100 billion by 2025.

The justification for this value is that Machine Learning can be applied in various areas, such as fraud detection, facial recognition, and, related to energy generation and of greater importance for this project, the detection of mechanical failures.

Rotating components are subject to various defects that alter the system's vibration, causing characteristic patterns known as fault signatures. Consequently, the study of these marks allows extracting information that identifies the fault present in the machinery. Thus, Machine Learning becomes a promising tool for system condition diagnosis.

This research trains the Light Gradient Boosting Machine or LightGBM (Ke *et al.*, 2017b) machine learning algorithm on experimental data from an accelerated lifetime test

rig that created realistic faults on ball bearings. The data was extracted from the Paderborn University Repository (Lessmeier *et al.*, 2016a), and it contains samples of rotating machines with ball bearings mapped to three operational conditions: healthy, with an inner race, or with an outer race fault.

However, achieving perfect accuracy in fault detection with machine learning is highly unlikely since the data in this field is scarce and costly. As an alternative, this work proposes a data resampling technique based on knowledge from the field of bearing research to improve LightGBM's performance.

Another factor that affects accuracy in machine learning for fault detection is that the algorithms essentially work as mathematical approximations of complex physical phenomena. Because of that, handling misclassifications is fundamental, and specifically for this research, minimizing false negatives is crucial since they can lead to catastrophic failure. Consequently, this research uses Bayesian optimization to maximize LightGBM's accuracy and recall.

Rotating machines play a vital role in society, which makes identifying system failure fundamental. This research's main contribution to the area of mechanical fault detection in rotating machines is a methodology that combines LightGBM predictive capabilities, a knowledge-based resampling technique, and the Bayesian optimization to build a robust and reliable classifier capable of improving maintenance strategies and enhancing rotating machines' performance and longevity. It should be noted, however, that this project does not aim to implement the algorithms but rather to use open-source packages such as *Scikit-Learn* (Pedregosa *et al.*, 2011a), *LightGBM* (LightGBM, 2023), and *BayesianOptimization* (Nogueira, 2014), available in the Python programming language (Rossum; Drake, 2009).

The present study is divided into five chapters. Chapter 2 addresses the literature review, establishing the context for fault diagnosis in rolling bearings and the application of machine learning algorithms. Chapter 3 describes the theoretical modeling and machine learning algorithms employed, emphasizing LightGBM, the proposed resampling technique, and the Bayesian optimizer. The experimental results are presented in Chapter 4, where the effectiveness of the LightGBM model in identifying the operating conditions of rotating machinery is discussed. Finally, Chapter 5 synthesizes the conclusions and recommendations for future research, highlighting the practical and theoretical implications of the findings.

2 LITERATURE REVIEW

As mentioned in the Introduction Chapter of this work, machine learning can surpass the boundaries of classical programming with mathematical and statistical methods. According to Igual and Seguí (2017), most of these methods originated in the 18th and 19th centuries with statistical pioneers like Thomas Bayes, who discovered Bayes' Theorem in 1763, and Simon Laplace, who published the Analytical Theory of Probabilities in 1812.

Still in the 19th century, with the aid of the theories of Bayes and Laplace, Cramer (2002) states that the logistic function was created to describe the behavior of population growth and chemical reactions of autocatalysis. This model arose from the need to limit exponential growths since extrapolations with these models often result in impossible values.

Because of this, Pierre-François Verhulst, proposed adding a term to the exponential model responsible for denoting a saturation limit. Verhulst named the new model the logistic function, and the results of his research were published between 1838 and 1847 in three articles: *Notice sur la loi que la population poursuit dans son accroissement*, translated by Vogels *et al.* (1975); *Recherches mathématiques sur la loi d'accroissement de la population* (Verhulst, 1845); and *Deuxième mémoire sur la loi d'accroissement de la population* (Verhulst, 1847). However, Cramer (2002) argues that Verhulst's publications took almost a century to be used as a statistical regression model, which was possible thanks to the contributions of the American statistician Joseph Berkson, who popularized logistic regression in 1944 (Berkson, 1944).

Shortly after, Alan Turing, considered the father of theoretical computer science and artificial intelligence, published the article *Computing Machinery and Intelligence* (Turing, 1950) in the academic journal *Mind* at the University of Oxford, where he proposed the Turing Test, which aims to evaluate if a machine can replicate human behavior.

The test involves two players: a person and a machine designed to mimic human behaviors. Both communicate with a human judge responsible for asking questions. Based on the responses obtained, if the mediator cannot confidently distinguish which player is the machine, it is considered to have passed the Turing Test.

Although Alan Turing presented the theoretical foundation for a learning machine, the first practical demonstration occurred in 1952 with computer scientist Arthur Samuel. According to McCarthy and Feigenbaum (1990), Samuel joined the IBM Poughkeepsie Laboratory

in 1949 and began his research on self-learning machines.

Three years after his arrival, Samuel presented the first checkers game in which the opponent was a machine. The result attracted so much attention from the scientific and academic communities that IBM's stock price increased by $15 in just one day. Consequently, this event marks the beginning of machine learning as it is known today. However, the term "machine learning" was only popularized by Samuel in 1959 (Samuel, 1959).

Just before the popularization of the field, Rosenblatt (1958) created the Perceptron while working at the Cornell Aeronautical Laboratory. The algorithm, which currently serves as the basis for artificial neural networks, as described by Rosenblatt, is based on a connectionist brain theory. The researcher believed that information storage and pattern recognition happen through a network of activity centers in the central nervous system.

Mathematically, the Perceptron consists of a binary classifier built from the dot product between input values and their corresponding weights. Thus, if an input has a low weight, its contribution to the final output will be less significant, and the same logic applies in the opposite case. The final response is then evaluated based on a threshold value. If the obtained dot product exceeds the threshold, the inputs are classified as "1" and otherwise, as "0". In this algorithm, machine learning is understood as the process of iteratively updating one weight at a time, minimizing a loss function. Finally, once the loss function has been minimized, it is known that the inputs will be classified with the algorithm's maximum performance. However, this does not mean they will be correctly classified.

Although Rosenblatt's Perceptron was booming in the media, its development into modern Artificial Neural Networks (ANNs) was hindered due to the release of Minsky and Papert (1969)'s book, *Perceptrons: An Introduction to Computational Geometry*, in which various mathematical limitations of the model were presented. The most significant limitation was the inability of a single-layer Perceptron to reproduce the "exclusive or" or "XOR" function, a fundamental computing operation. As a result, Rosenblatt's model lost the attention of the technology community, and research in the field gradually diminished.

The impact of Minsky and Papert's book was so significant that seventeen years passed before Artificial Neural Networks regained media attention. This was made possible thanks to the article by Rumelhart *et al.* (1986) published in *Nature*, in which the backpropagation method was introduced.

The backpropagation algorithm consists of calculating the gradients of the loss func-

tion for a single-layer Perceptron based on the weights and updating them all at once, as opposed to the model initially proposed by Rosenblatt. This method is so efficient that it allows the construction of Perceptrons with more than one layer, known as Multi-layer Perceptrons, which can overcome all the limitations presented in Minsky and Papert's book. This marked the emergence of the first artificial neural networks (ANNs) as they are known today.

Due to the presentation of backpropagation, the evolution of ANNs, and the development of computers, the technology community at the end of the 20th century felt motivated to develop the field of machine learning further. Consequently, new algorithms began to emerge, such as Classification and Regression Trees (CARTs) (Breiman *et al.*, 1984), Recurrent Neural Networks (Hochreiter; Schmidhuber, 1997), and, of greater importance for this work, Support Vector Machines (SVMs) (Cortes; Vapnik, 1995) and Gradient Boosting Machines (Friedman, 2000; Mason *et al.*, 1999).

According to Cortes and Vapnik (1995), the Support Vector Machine (SVM) corresponds to a classifier that performs a nonlinear transformation of an input vector into a higher-dimensional space. The algorithm then creates a hyperplane constructed from support vectors. Next, an iterative process is initiated over the dataset in which, at each step, the components of the support vectors are updated until the best-separating hyperplane between the classes is obtained.

Boosting is understood as a technique for building robust regression or classification algorithms (referred to as "strong" algorithms) from sets of high-bias, high-variance algorithms (referred to as "weak" algorithms), such as decision trees and linear and logistic regressions. The method was introduced by Freund and Schapire (1997) through an algorithm called AdaBoost (Adaptive Boosting) and further refined by Mason *et al.* (1999) and Friedman (2000), who generalized the Boosting idea to optimize any differentiable loss function using the Gradient Descent method, giving rise to the family of algorithms known as Gradient Boosting Machines (GBMs), which contains popular algorithms such as the XGBoost (Chen; Guestrin, 2016), the CatBoost (Prokhorenkova *et al.*, 2019) and, most importantly for this research, the LightGBM (Ke *et al.*, 2017b).

LightGBM (Ke *et al.*, 2017b) was chosen since it has become one of the most used machine learning methods (Kaggle, 2022) for its efficient processing speed and high accuracy and has already been successfully used on bearing fault detection (Jia *et al.*, 2021; Xu *et al.*, 2021). It is a gradient-boosting framework that successively fits decision trees trained to cor-

rect the mistakes of the previous ones. However, LightGBM is unique due to two techniques that reduce training time and memory consumption without affecting accuracy: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

According to Ke *et al.* (2017b), while gradient-boosting decision tree algorithms usually scan all the data to estimate the information gain of all possible split points for each feature, the GOSS method in LightGBM maintains only the ones with more significant gradients and randomly drops the smaller ones. EFB, on the other hand, is a technique that attempts to merge highly similar features. The authors state that combining these methods speeds up training without affecting the algorithm's accuracy.

Regardless of the machine learning algorithm chosen to solve a specific task, hyperparameter tuning will be necessary. Hyperparameters are the algorithm settings defined before the training phase. For example, in a Support Vector Machine (SVM), the regularization parameter "C" controls the balance between maximizing the margin and minimizing classification errors (Géron, 2019). Similarly, in gradient boosting algorithms like LightGBM, hyperparameters like the learning rate and number of trees significantly impact the model's predictive abilities (LightGBM, 2023). One of the tools that stand out for hyperparameter tuning is Bayesian optimization.

Bayesian optimization (Frazier, 2018; Nogueira, 2014) is an efficient optimization technique widely used in machine learning problems since it searches for the best hyperparameters statistically rather than exhaustively, as done in a grid or random search. This method has already been used to enhance the identification of mechanical faults in rotating machines (Kolar *et al.*, 2021). It consists of optimizing "black-box" functions that cannot be maximized through standard techniques such as gradient descent, Newton's, or quasi-Newton methods due to their unknown properties, such as concavity, linearity, or first or second-order derivatives.

The Bayesian optimizer (Frazier, 2018; Nogueira, 2014) builds a probabilistic model of the original function based on initial observed points. This statistical approximation allows the construction of an acquisition function that balances exploration and exploitation of the observation space. The point that maximizes the acquisition function is chosen as the subsequent evaluation of the original function, and the new information updates the probabilistic model. This process is repeated until a global optimum is achieved.

The success of the techniques and algorithms created in the late 20th century can be seen in a wide range of applications in the 21st century, extending from medicine, with the

automation of diagnoses and drug development (JCU, 2019), to recommendation systems in streaming services, such as Netflix's movie recommendation algorithm (Chong, 2020), and, most notably for this work, fault diagnosis in rotating machines.

According to Liu *et al.* (2018), fault diagnosis is a pattern recognition problem regarding the rotating machinery condition, aiming to solve three tasks: determining whether the equipment is normal, finding the incipient failure and its reason, and predicting the trend of fault development. This work focuses on solving the first two and the fault detection process is made using statistical techniques and machine learning models capable of identifying trends, correlations, and characteristic behaviors of mechanical failures through measured signals, such as the wear (Gecgel *et al.*, 2020) and the ovalization of hydrodynamic bearings (Alves *et al.*, 2020), and, most notably for this work, the inner and outer race faults in ball-bearings.

According to Randall and Antoni (2011), when a rolling element within a bearing encounters a local fault on the inner or outer race, it produces an impact signal at one of the bearing fault frequencies, such as the Ball Pass Frequency Outer race (BPFO) or Ball Pass Frequency Inner race (BPFI). This behavior of generating specific fault-related frequencies in the impact signal allows fault identification through machine learning algorithms.

However, data in the field of fault detection is scarce and costly. As an alternative, various data augmentation techniques can be applied, extending from simply scaling the signal to using generative models, such as Generative Adversarial Networks (GANs) (Iwana; Uchida, 2021; Iglesias *et al.*, 2023). This work, on the other hand, proposes a data resampling technique based on knowledge from the field of bearing research to improve the performance of LightGBM. Therefore, instead of training LightGBM with fully extracted signals, the algorithm is trained and tested on random slices constructed based on the BPFO and the BPFI.

Finally, due to the growing use of machine learning techniques in various scientific and technological fields and the undeniable importance of rotating machinery in energy generation for society, this work's contribution to developing the fault detection field is evident.

3 METHODOLOGY

3.1 Problem Description

Rolling bearings are critical components in various industrial applications, and their reliable operation is essential to prevent catastrophic failures and costly downtime. This research addresses the fault detection problem in rolling bearings by employing machine learning techniques, primarily focusing on LightGBM as the main algorithm and benchmarking the Support Vector Machine (SVM).

The methodology analyzes time signal datasets and uses a resampling technique during training to enhance LightGBM's performance in building decision trees. While SVM is also evaluated as a benchmark, the methodology does not expect to improve its performance since SVM has a computational complexity that scales quadratically with the number of training samples, which is significantly increased with the proposed resampling method.

Using time-domain data offers several advantages. Firstly, time-domain data often requires less preprocessing, as it is the raw signal directly obtained from sensors like accelerometers, making it more straightforward to work with and less computationally intensive. Secondly, time-domain features such as peak-to-peak distance can be very effective at capturing the characteristics of bearing faults, as they directly reflect the physical impacts and vibrations caused by such faults. In addition, time-domain analysis can be more robust to variations in operating conditions since frequency-domain features may be more sensitive to changes in speed and load, which can alter the spectral distribution of the vibration signals. These advantages make time-domain data a practical and powerful choice for developing machine learning models for bearing fault detection.

In addition, the model selection focuses on building a reliable and robust classifier that achieves high accuracy and prevents false negatives in predicting healthy samples, which can potentially lead to catastrophic failure.

A scoring mechanism that combines the overall accuracy with the recall of the healthy samples is employed to achieve this goal. This scoring choice will be discussed later in this chapter. The model selection process involves using a Bayesian optimizer for hyperparameter tuning to maximize this composite score.

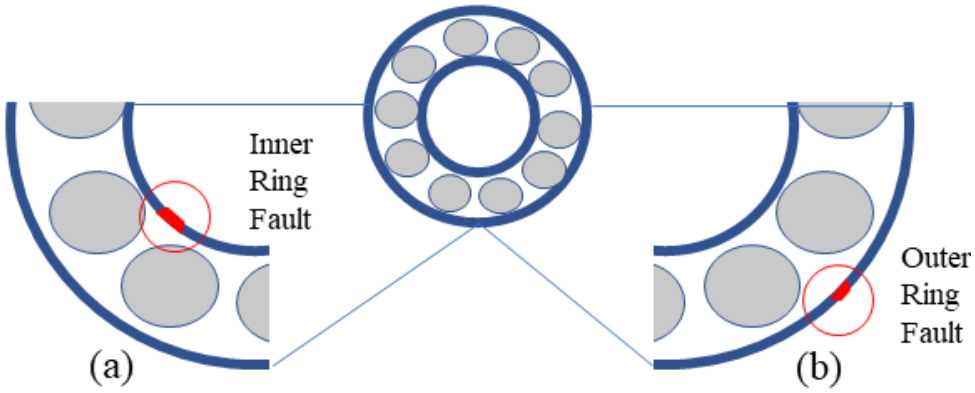
Therefore, the methodology combines a knowledge-based resampling technique

with the Bayesian optimizer for hyperparameter tuning to enhance LightGBM's predictive capabilities and training efficiency. Hence, this research aims to contribute to developing a fault detection system that offers both high accuracy and a precaution against false negatives, ultimately improving the reliability and safety of industrial machinery.

3.2 The Data

According to Lessmeier *et al.* (2016b), bearing damage is often monitored by acceleration sensors that permit vibration analysis. To support technological development in this area, researchers at Paderborn University made available a benchmark dataset for classifying common ball-bearing faults: the inner and outer ring faults, shown in Figure 3.1.

Figure 3.1 – Schematics of a ball bearing with an inner ring (a) and an outer ring fault (b)



The experiment used an accelerated lifetime test rig to create realistic faults for the bearing type 6203, whose geometry is shown in Table 3.1.

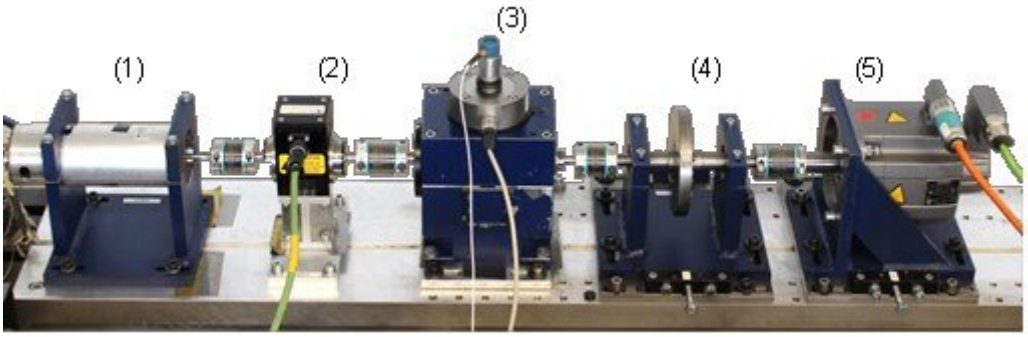
Table 3.1 – Geometry of the Bearing 6203 Used for Data Gathering (Lessmeier *et al.*, 2016b)

Property	Value	Symbol
Pitch circle diameter	28.55 mm	D
Number of rolling elements	8 pc.	n
Contact angle	0 rad	ϕ
Rolling element diameter	6.75 mm	d
Length of rolling element	6.75 mm	
Diameter of inner raceway	24.0 mm	
Diameter of outer raceway	33.1 mm	
Static load rating	4750 N	
Dynamic load rating	9500 N	
Speed limit	12000 rpm	
Manufacturer	FAG	

Note from Table 3.1 that only the pitch circle diameter, the number of rolling elements, the contact angle, and the rolling element diameter have mathematical symbols. That was done purposely for clarity since they will be used in Section 3.5 to calculate the ball-bearings fundamental frequencies.

Data generation was conducted on a modular test rig, collecting vibration data for health, outer, and inner race faults, all available in (Lessmeier *et al.*, 2016a).

Figure 3.2 – Modular Test Rig For Data Gathering (Lessmeier *et al.*, 2016b)



The test rig in Figure 3.2 contains five modules: an electric motor (1), a torque-measurement shaft (2), a rolling bearing test module (3), a flywheel (4), and a load motor (5).

The main characteristics of the electric motor (1) are shown in Table 3.2.

Table 3.2 – Motor and Inverter Specifications (Lessmeier *et al.*, 2016b)

Parameter	Specification
Motor Type	Permanent Magnet Synchronous Motor (PMSM)
Motor Power	425 W
Nominal Torque (T)	1.35 Nm
Nominal Speed	3,000 rpm
Nominal Current (I)	2.3 A
Pole Pair Number (p)	4
Motor Model	Type SD4CDu8S009
Motor Brand	Hanning Elektro-Werke GmbH
Frequency Inverter Model	KEB Combivert 07F5E 1D-2B0A
Inverter Switching Frequency	16 kHz

The speed and the torque are measured by the torque-measuring shaft (2) model TM305 Magtrol Inc., which has a nominal torque of 2 Nm and an accuracy of ±0.1 % of the nominal torque.

The rolling bearing test module (3) measures the acceleration of the bearing housing using a piezoelectric accelerometer (Model No. 336C04, PCB Piezotronics, Inc.) and a charge amplifier (Type 5015A, Kistler Group) with a low-pass filter at 30 kHz.

Finally, the authors state that the flywheel (4) and the load machine (5) are used to simulate the inertia and load of the driven equipment, respectively. The load motor is a PMSM Siemens-Motor 1FT7062-1AF70-1DG1 with a nominal torque of 6 Nm.

The modular setup developed by Lessmeier *et al.* (2016b) allows varying multiple operational parameters resulting in various experiments. This research merges datasets from fifteen experiments labeled by the researchers as shown in Table 3.3.

Table 3.3 – Categorization of the datasets used (Lessmeier *et al.*, 2016b)

Healthy	**Outer Ring Damage**	**Inner Ring Damage**
K001	KA04	KI04
K002	KA15	KI14
K003	KA16	KI16
K004	KA22	KI18
K005	KA30	KI21

Each dataset in Table 3.3 contains 80 measurements of approximately 4 seconds of vibration signal with a sampling frequency of 64 kHz. The 80 measurements were collected varying the rotor operational parameters according to Table 3.4 with 20 measurements each.

Table 3.4 – Rotor Operational Parameters

Rotational speed [rpm]	**Load Torque [Nm]**	**Radial force [N]**	**Name of Setting**
1500	0.7	1000	N15_M07_F10
900	0.7	1000	N09_M07_F10
1500	0.1	1000	N15_M01_F10
1500	0.7	400	N15_M07_F04

Target labeling was done visually by the authors and two examples are shown in Figure 3.3.

To build the complete dataset the samples are cropped to 250000 features. The rotational speeds of 900 rpm (15 Hz) and 1500 rpm (25 Hz) were also available as a feature. The individual samples were merged into one balanced dataset with 1200 rows and 250001 columns.

Figure 3.4 shows samples for the three operational conditions.

Figure 3.3 – Indentation at the raceway of the outer ring (a); small pitting at the raceway of the inner ring (b) (Lessmeier *et al.*, 2016b)

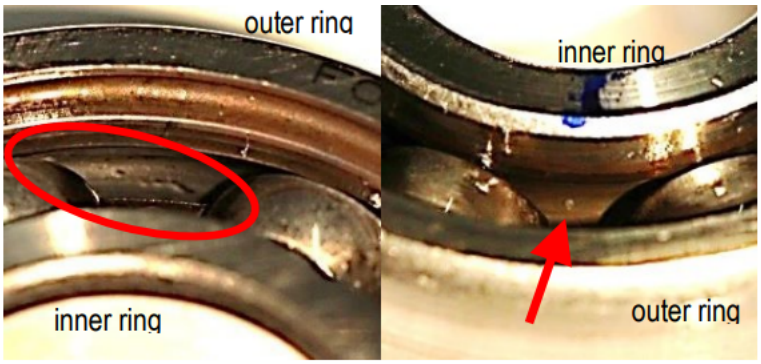
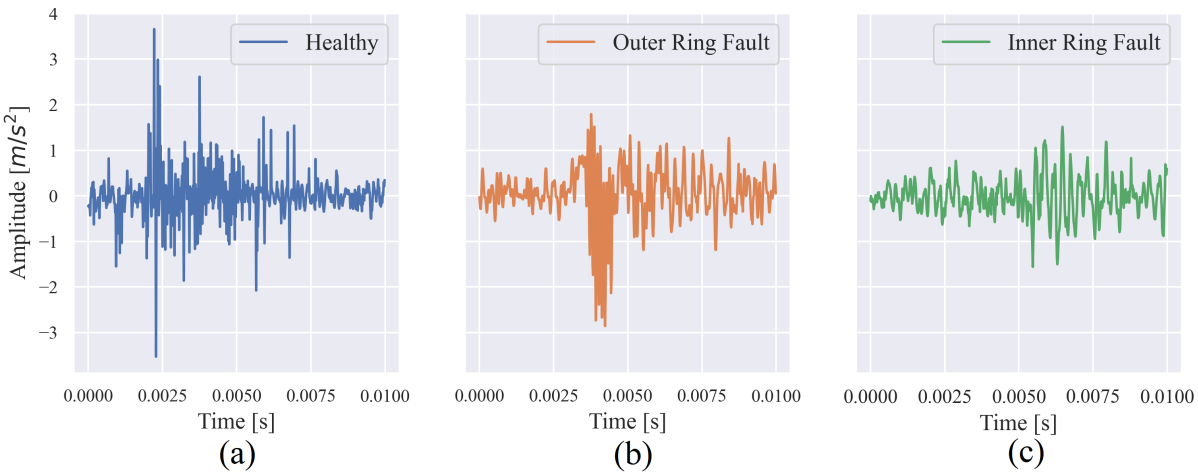


Figure 3.4 – Data samples of the healthy (a), the outer ring fault (b), and the inner ring fault (c) operational conditions



Smith and Randall (2015) define specific characteristics for benchmark datasets based on their extensive experience with various diagnostic techniques applied to a vibration-based condition monitoring dataset focused on bearings. According to the authors, a benchmark dataset must have organized and comprehensive documentation, sampling rates higher than 40 kHz, and data validation with verified methods before data publication. The dataset published by Lessmeier *et al.* (2016b) achieves all the criteria and is reliable for studying mechanical faults in rolling bearings.

3.3 LightGBM Algorithm

LightGBM is a gradient-boosting decision tree (GBDT) algorithm designed to be efficient and scalable. It can solve multiple machine learning tasks, such as multi-class clas-

sification, regression, and ranking problems. These properties make LightGBM one of the industry's most used machine learning frameworks, according to the "State of Data Science and Machine Learning 2022" survey made by (Kaggle, 2022).

LightGBM advantages, functionality, and hyperparameters are discussed in this section.

3.3.1 Advantages

LightGBM's primary advantages over its similars, such as XGBoost and AdaBoost, are training time optimization and lower memory usage. This is due to two techniques called Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), according to Ke *et al.* (2017a).

According to Ke *et al.* (2017a), while GBDT algorithms usually scan all the data instances to estimate the information gain of all possible split points for each feature, the Gradient-based One-Side Sampling (GOSS) method in LightGBM maintains only the ones with more significant gradients and randomly drops the smaller ones. The authors have proven in their research that this technique leads to more accurate predictions and lower training time since fewer data points need to be evaluated at each iteration.

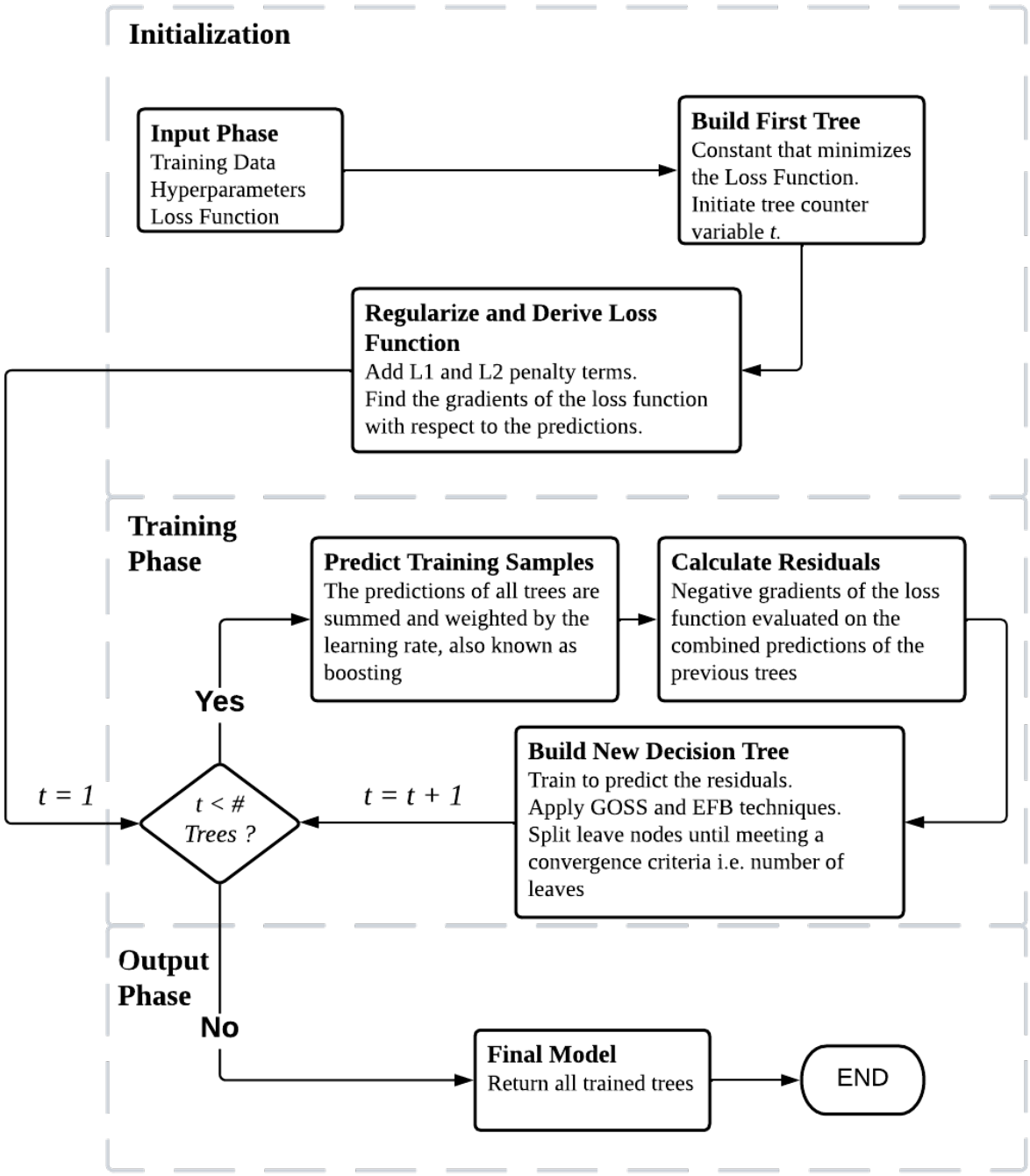
Exclusive Feature Bundling (EFB), on the other hand, is a technique that attempts to merge highly similar features. Ke *et al.* (2017a) state that high-dimensional data are usually sparse, and consequently, many features are mutually exclusive, i.e., they never take nonzero values simultaneously. In the context of this research, where the data samples are time signals, many features can be mostly related to higher or lower amplitudes. The authors propose that these features are exclusive and can be bundled into a single one, significantly speeding up training without affecting the algorithm's accuracy.

3.3.2 Functioning

LightGBM functioning occurs according to the flowchart in Figure 3.5 (Friedman, 2000; Ke *et al.*, 2017a) and can be summarized in three phases:

- **Initialization:** This step initializes some internal variables, such as the loss function and the hyperparameters. In addition, the first decision tree is fitted to the training data. There are three steps in this phase:

Figure 3.5 – LightGBM Functioning Flowchart



– **Regularization:** Regularization techniques prevent overfitting and improve the model generalization ability by adding penalty terms to the loss function. LightGBM uses the L1 and the L2 regularizations, which are penalties proportional to the absolute value and the square of the feature weights, respectively.

– **Finding the Gradients:** The regularized loss function is derived with respect to the predictions, which gives the gradient function that will be used to calculate the

pseudo-residuals during training.

  - **Building the First Tree:** The first classifier of the model is a constant that minimizes the loss function. Also, in this step, the tree counter parameter t is initialized and set to one.

- **Training Phase:** This step iteratively builds new decision trees until the tree counter parameter reaches a predefined value, shown in the diagram as $\#Trees$. Training trees that successively learn from the previous ones and have their predictions combined is known as boosting. There are also three steps in this phase:

  - **Predict Training Samples:** The training data is fed to all built trees, and the predictions are summed and weighted by the learning rate hyperparameter.
  - **Calculate Residuals:** The predictions in the previous step and the actual values of the target variable are input to the gradient function calculated in the initialization phase, outputting the pseudo-residuals.
  - **Build New Decision Tree:** A new decision tree is trained to predict the pseudo-residuals calculated in the previous step. A decision tree model is built by recursively splitting the leaf nodes until meeting a convergence criterion, i.e., achieving a predefined number of leaves. LightGBM does leaf splitting distinctively from other GBDT methods due to the GOSS and the EFB techniques, optimizing training time without affecting accuracy.

- **Output Phase:** Once the tree counter parameter reaches the predefined number of trees, the training phase stops, and all trained classifiers are returned.

As mentioned above, when LightGBM is initialized, the loss function is defined. According to Géron (2019), a standard loss function in multi-class classification problems is the cross-entropy, which can be modified by adding weights related to the predicted classes, as shown in Equation 3.1.

$$L(y, p) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C_l} w_j y_{ij} \log(p_{ij}) \tag{3.1}$$

Where N is the total number of samples, C_l is the number of classes, y_{ij} is the actual label (encoded as a one-hot vector) for the i-th instance and j-th class, p_{ij} is the predicted probability for the i-th instance and j-th class, and w_j is the weight for the j-th class.

Adding class weights allows the loss function to better deal with typical issues in machine learning projects, such as imbalanced datasets, and, most notably for this research, the minimization of false negatives, which will be discussed further in the Metrics section.

3.3.3 Hyperparameters

As mentioned previously, LightGBM is flexible toward various machine-learning problems. One of the key features contributing to its versatility is the variety of hyperparameters available to tune its performance.

In this research, five of the algorithm hyperparameters were explicitly chosen for their effects on training time, model accuracy, and overfitting (LightGBM, 2023):

1. *learning_rate*: controls the step size at each iteration during the gradient descent optimization process. A lower *learning_rate* leads to slower convergence but can help avoid overfitting.

2. *n_estimators*: specifies the number of boosting iterations or the number of decision trees in the model. Increasing *n_estimators* can improve the model accuracy and increase the training time and the risk of overfitting.

3. *num_leaves*: controls the maximum number of leaves in each decision tree. A higher *num_leaves* can increase the model complexity and accuracy, but it can also increase the risk of overfitting and slow down the training time.

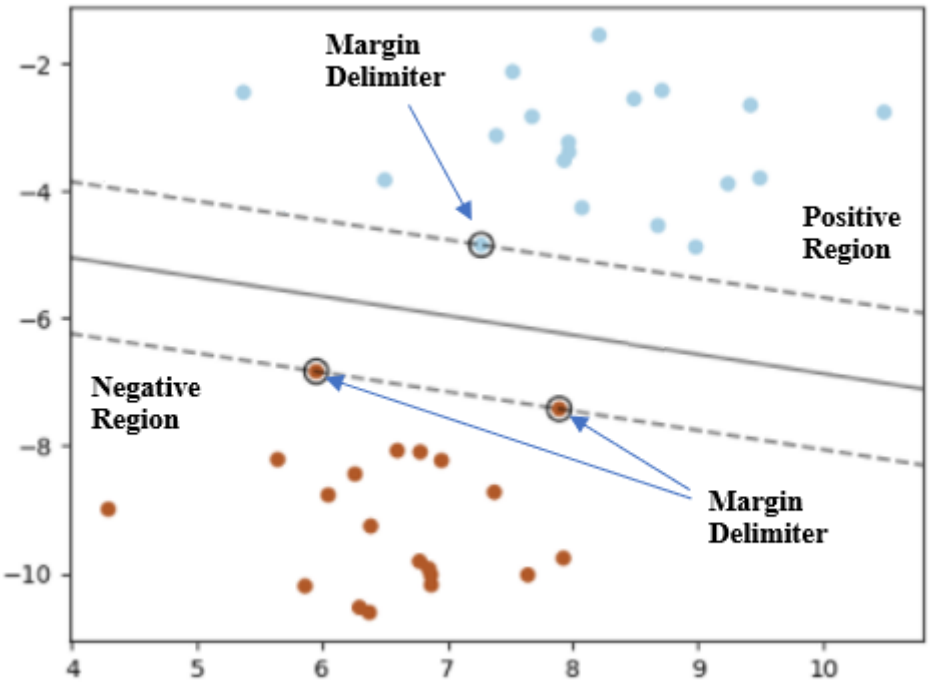
4. *reg_alpha* and *reg_lambda*: control the L1 and L2 regularization strength, respectively. Both hyperparameters can help prevent overfitting by adding a penalty term to the objective function. A higher value of *reg_alpha* or *reg_lambda* can lead to a simpler model and reduce accuracy.

3.4 SVM Algorithm

3.4.1 Binary Classification

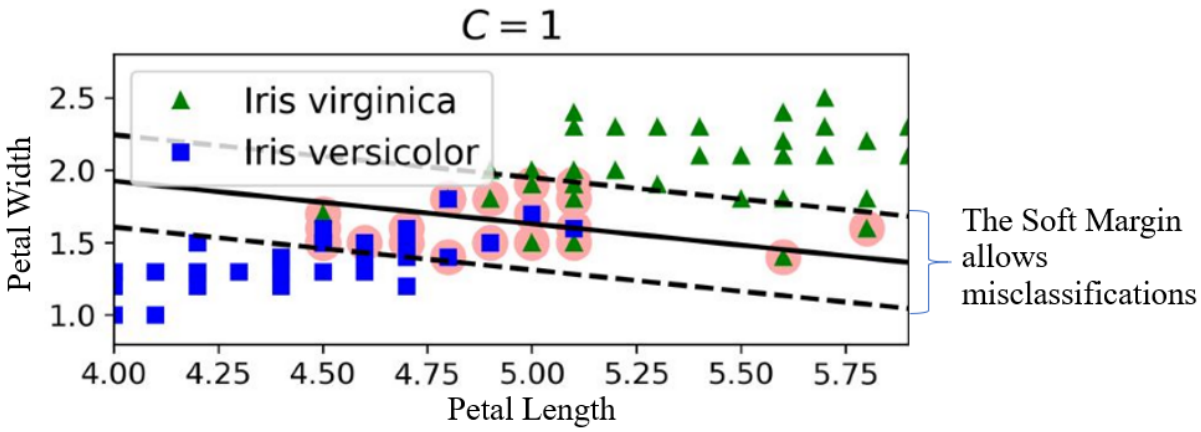
The Support Vector Machine (SVM) is an algorithm that, as shown in Figure 3.6, separates operational conditions into two different regions in the space through a kernel trick (Géron, 2019). In this operation, each vector pair in the dataset is mapped to a higher dimension using specific functions capable of quantifying similarities between observations.

Figure 3.6 – Separation Hyperplane formed by SVM (Pedregosa *et al.*, 2011c)



As shown in Figure 3.6, SVM would ideally be able to perfectly separate the dataset into failure (1) and non-failure (0) regions bounded by dashed lines, known as margins. However, perfect separation between regions is practically impossible in real data due to interference and noise in measurement equipment, among other factors. Therefore, optimizing the set of weights $\mathbf{w}$ of this algorithm requires that some observations may be incorrectly classified by a distance ζ from their appropriate decision margins, a condition known as "soft margin" (Géron, 2019), represented in Figure 3.7.

Figure 3.7 – Separation Hyperplane with Soft Margin (Géron, 2019)



Unlike LightGBM, SVM finds the best set of weights $\mathbf{w}$ through a constrained minimization problem. Additionally, incorrectly classified observations are penalized by a regularization factor C during the optimization process. Thus, the optimization problem can be summarized as Equation 3.2:

$$\begin{cases} \min_{\mathbf{w},b,\zeta} & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{k=1}^{N}\zeta_k \\ subject\ to & s_k(\mathbf{w}^T\phi(\mathbf{x}) + b) \geq 1 - \zeta_k \end{cases} \tag{3.2}$$

Where s is a component acting as a sign function, and ϕ is a mapping function designed to transform vector $\mathbf{x}$ into a higher dimension.

According to Géron (2019), SVM is part of a particular group of methods known as duality problems. The principle of Lagrangian duality applies to problems where the objective function and its constraints are convex and differentiable, stating that the problem can be solved equivalently from two perspectives, which in the case of SVM corresponds to Equation 3.3.

$$\begin{cases} \min_{\alpha} & \frac{1}{2}\alpha^T Q\alpha - d^T\alpha \\ subject\ to & s^T\alpha = 0 \\ with & 0 \leq \alpha_k \leq C,\ for\ k = 1, 2, 3, \ldots, N \\ & Q_{k,l} \equiv s_k s_l K(x_k, x_l) = s_k s_l \phi(x_k)^T\phi(x_l) \end{cases} \tag{3.3}$$

Where:

- α_k: dual coefficients
- Q: positive-definite Matrix
- K: kernel function
- d: unit vector
- l: row index in the data matrix

In Equation 3.3, it can be observed that the kernel function K corresponds to the product of the mapping functions ϕ mentioned earlier. According to Géron (2019), this relationship is guaranteed by Mercer's Theorem, which states that the kernel function can be used in place of the mapping functions, even without knowledge of their specific form, as long as ϕ exists.

In this project, the kernel function used is the Gaussian Radial Basis Function (RBF), defined as:

$$K(x_k, x_l) = \exp\left(-\gamma \|x_k - x_l\|^2\right) \tag{3.4}$$

Where γ represents the similarity normalization factor.

It can be observed that if x_k and x_l are distant points in space, $K(x_k, x_l) \approx 0$, and if they are close, $K(x_k, x_l) \approx 1$.

Thus, prediction is made by determining if a given point x is more similar to points indicating healthy operation (0) or failure (1), as mathematically represented by Equation 3.5.

$$\hat{y}_k(x) = \sum_{k=1}^{N} s_k \alpha_k K(x_k, x) + b \tag{3.5}$$

Similar to LightGBM, SVM is an algorithm that shows significant gains when trained on a larger dataset. However, according to Géron (2019), due to the kernel function, the method has a computational complexity that scales quadratically with the number of training samples. Therefore, SVM is suitable for medium-sized datasets, with approximately 30000 samples.

3.4.2 Multiclass Classification

The previous section clearly shows that Support Vector Machines are not naturally designed for handling multiclass classification. However, by breaking down the problem into several binary classification tasks, it is possible to turn SVM into a multiclass classifier using two distinct approaches: *One-vs-One* and *One-vs-Rest*.

In the *One-vs-One* approach, one SVM is trained for each pair of classes in the dataset. Therefore, two classes are selected at a time, and the remaining are ignored during the training process. The total number of SVMs trained in this approach is represented by Equation 3.6.

$$M_{1vs.1} = \frac{C_l(C_l - 1)}{2} \tag{3.6}$$

Where C_l and M are the number of classes and SVMs trained, respectively.

In the *One-vs-Rest* approach, on the other hand, one SVM is trained for each class in the dataset. Therefore, for each machine trained, one class becomes the positive (1) and the

remaining classes become the negative (0). The total number of SVMs trained in this approach is represented by Equation 3.7.

$$M_{1vs.R} = C_l \tag{3.7}$$

Note from Equations 3.6 and 3.7 that the *One-vs-One* method generally involves training more SVMs, leading to extended training times compared to the *One-vs-Rest* approach. Nevertheless, in this study where $C_l = 3$, both scenarios involve training an equal number of SVMs. Hence, both cases will be evaluated.

3.4.3 Hyperpameters

SVM is not as flexible as LightGBM and its hyperparameter tuning generally happens by adjusting the C parameter.

According to Pedregosa *et al.* (2011b), the C parameter behaves as a regularization factor trading off the correct classification of training examples against maximizing the decision function's margin. A smaller margin for larger values of C will be allowed if the decision function is better at correctly classifying all training points. A lower C, on the other hand, facilitates a more extensive margin and, therefore, a less complex decision function at the cost of training accuracy.

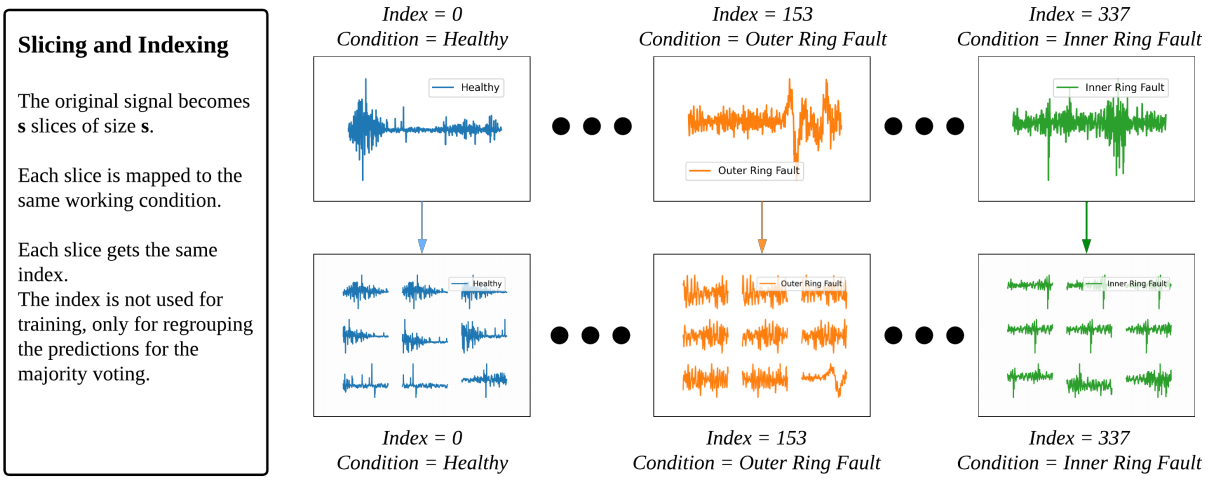
3.5 Knowledge-based Resampling

Despite the EFB technique, an excess of features in a dataset can significantly increase LightGBM's training time as the algorithm constructs decision trees from the gradients and distributions of feature values. As an alternative to improve LightGBM's performance, this work proposes a data resampling technique based on knowledge from the field of bearing research, which is discussed in this section. The methodology is done in two steps: firstly, slicing, indexing, and stacking, and lastly, testing and majority voting.

3.5.1 Slicing, Indexing, and Stacking

The technique takes s random slices with size s of an original time sample and stacks them into multiple time samples mapped to the same index and working condition, as shown in Figure 3.8.

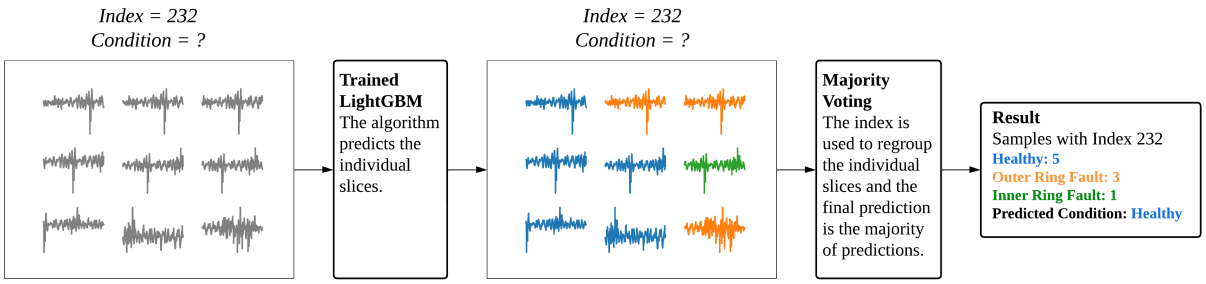
Figure 3.8 – Slicing, Indexing and Stacking Time Samples



3.5.2 Testing and Majority Voting

After the algorithm's training phase, the resampled testing dataset is evaluated, and each slice gets its own prediction. The index is then used to regroup the individual predictions, and the final value is achieved by majority voting, as shown in Figure 3.9.

Figure 3.9 – Testing and Majority Voting



3.5.3 Choosing the Slice Size

Recognizing amplitude patterns is more important than knowing when they happen in the original signal. That means specific signal regions are sufficient for classifying the working conditions of a rotating machine. Because of that, rolling bearings knowledge will be used to decide the slice size s.

First, reshaping the dataset so that the number of features matches one bearing rotation likely enhances the training process of LightGBM. This alignment facilitates the model's ability to capture patterns and variations in the data, preventing overfitting and improving its

generalization ability. Additionally, resampling the data in this manner reduces the computational complexity of the model, resulting in faster and more efficient training. The number of features to assure one bearing rotation is defined in Equation 3.8.

$$M_r = \frac{f_s}{f_r} \tag{3.8}$$

Where f_s and f_r are the sampling frequency and the rotor rotational speed respectively.

Another theoretical aspect that must be considered is that when a rolling element within a bearing encounters a local fault on the inner or outer race, it produces an impact signal at one of the bearing fault frequencies, such as the Ball Pass Frequency Outer race (BPFO) or Ball Pass Frequency Inner race (BPFI), which are defined in Equations 3.9 and 3.10 (Randall; Antoni, 2011):

$$BPFO = \frac{nf_r}{2}\left(1 - \frac{d}{D}\cos\phi\right) \tag{3.9}$$

$$BPFI = \frac{nf_r}{2}\left(1 + \frac{d}{D}\cos\phi\right) \tag{3.10}$$

Where D, d, ϕ, n are the pitch diameter, ball diameter, contact angle between the ball and the cage, and the number of rolling elements, respectively.

Given the bearing's geometry shown in Table 3.1 and the two rotor speeds of 15 and 25 Hz, the BPFI and the BPFO are shown in Table 3.5.

Table 3.5 – Ball Pass Frequencies in the Inner and Outer Race of the Bearing 6203

	$f_r = 15\ Hz$	$f_r = 25\ Hz$
BPFI	74.2	123.6
BPFO	45.8	76.4

The minimum number of features to assure at least one event of impact signal in the outer or the inner race is defined in Equations 3.11 and 3.12:

$$M_o = \frac{f_s}{BPFO} \tag{3.11}$$

$$M_i = \frac{f_s}{BPFI} \tag{3.12}$$

From Equations 3.8, 3.11, and 3.12, it is clear that the rotor speed affects the minimum amount of features required to represent a full rotation or an impact signal in case of a bearing fault. Because of that, every training step of the LightGBM will be done exclusively on one rotor speed. In addition, the seven distinct slice sizes for each rotor speed will be evaluated to measure the algorithm's sensitivity, as shown in Table 3.6.

Table 3.6 – Slice Sizes for Resampling the Dataset

		$f_r = 15\ Hz$	$f_r = 25\ Hz$
Half the BPFI	$\frac{M_i}{2}$	431	259
Number of points due to BPFI	M_i	863	518
Mid-point between BPFO and BPFI	$\frac{M_i+M_o}{2}$	1130	678
Number of points due to BPFO	M_o	1397	838
Mid-point between BPFO and Full Rotation	$\frac{M_r+M_o}{2}$	2832	1699
Full Rotation	M_r	4267	2560
Two Full Rotations	$2M_r$	8533	5120

The optimal results are likely achieved when employing a slice size between the BPFO and one bearing rotation. This hypothesis is grounded in the expectation that a slice size with fewer features than the BPFO might not provide sufficient information for detecting outer race faults. In addition, a slice size with more features than a full rotation could introduce confusion into the algorithm's training, rendering it less effective in fault detection.

3.6 Measuring the Algorithms' Performance

This work evaluates the accuracy and the recall for model selection. Once the algorithms are built based on these performance indicators, the confusion matrix is plotted for visualizing and quantifying the models' capacity to distinguish between fault classes.

3.6.1 Accuracy

Accuracy in multi-class classification is calculated as the sum of the true positives (TP) for each class divided by the total number of instances:

$$Accuracy = \frac{\sum_{c=1}^{C_l} TP_c}{n} \tag{3.13}$$

Where C_l is the number of classes and n is the total number of testing samples.

3.6.2 Recall

Achieving perfect accuracy in fault detection with machine learning is highly unlikely since the data are scarce and costly and the algorithms work as mathematical approximations of complex physical phenomena based on the provided data. While these models can capture specific patterns and features that may indicate faults, they cannot account for all factors contributing to fault occurrence. As a result, false positives and negatives are common in machine learning-based fault detection systems. Consequently, handling misclassifications is a crucial aspect of the fault detection process.

While false positives can disrupt operations and result in monetary losses, they are not as critical as false negatives in the context of fault detection in rotating machines since the inability to detect a fault can lead to machine breakdown, which could be potentially life-threatening to workers. Therefore, while maximizing accuracy in machine learning models for fault detection in rotating machines is preferable, the recall should also be evaluated and maximized to ensure the misclassifications will be mainly false positives.

The recall for a given class is calculated as the ratio of the true positives (TP) to the sum of true positives and false negatives (FN) for that class.

$$Recall_c = \frac{TP_c}{TP_c + FN_c} \tag{3.14}$$

Even though it is possible to calculate an overall recall for a machine learning model, only the recall for the healthy samples is considered in this work.

3.6.3 Scoring and Model Selection

That way, the LightGBM algorithm can be tuned to maximize accuracy while ensuring no faulty condition is classified as healthy.

To do so, the Bayesian optimization will maximize the score on Equation 3.15.

$$Score = 2 \cdot Accuracy + Recall_{healthy} \tag{3.15}$$

Note that the score on Equation 3.15 gives twice the importance to the accuracy than the recall. That way, the hyperparameters will be chosen to make fewer misclassifications, while the mistakes will be primarily false positives. This can be achieved using the Bayesian optimization algorithm for hyperparameter tuning.

3.6.4 Confusion Matrix

By definition, the confusion matrix A is such that $A_{i,j}$ equals the number of observations that belong to the i group and are predicted to belong to the j group (Pedregosa *et al.*, 2011a). For a multi-class classification problem with three classes (Healthy, Outer Ring Fault, and Inner Ring Fault), the confusion matrix will be a 3x3 matrix that illustrates how well a model distinguishes between different fault categories. It provides a comprehensive breakdown of true positives, true negatives, false positives, and false negatives, facilitating informed decision-making in fault detection applications. Lastly, it is essential to clarify that the confusion matrices in this work are exclusively built on the testing samples.

3.7 Bayesian Optimization

Bayesian optimization is a robust optimization algorithm widely used in machine learning problems since it can efficiently find the best set of hyperparameters for a specific situation. This is because this technique searches for the best hyperparameters statistically rather than exhaustively, as done in Grid Search.

This project uses a Python package made available by Nogueira (2014) called bayesian-optimization.

3.7.1 Properties

According to Frazier (2018), the Bayesian optimization focuses on maximizing a function f with the following properties:

- f is a "black-box", which means it lacks a particular known property like concavity or linearity that would make it easy to optimize using techniques that leverage such structure to improve efficiency.
- f may be expensive to evaluate, meaning each function evaluation may take significant time or resources.
- When evaluating f, only f is observed, and no first- or second-order derivatives. This prevents applying first- and second-order methods like gradient descent, Newton's, or quasi-Newton.

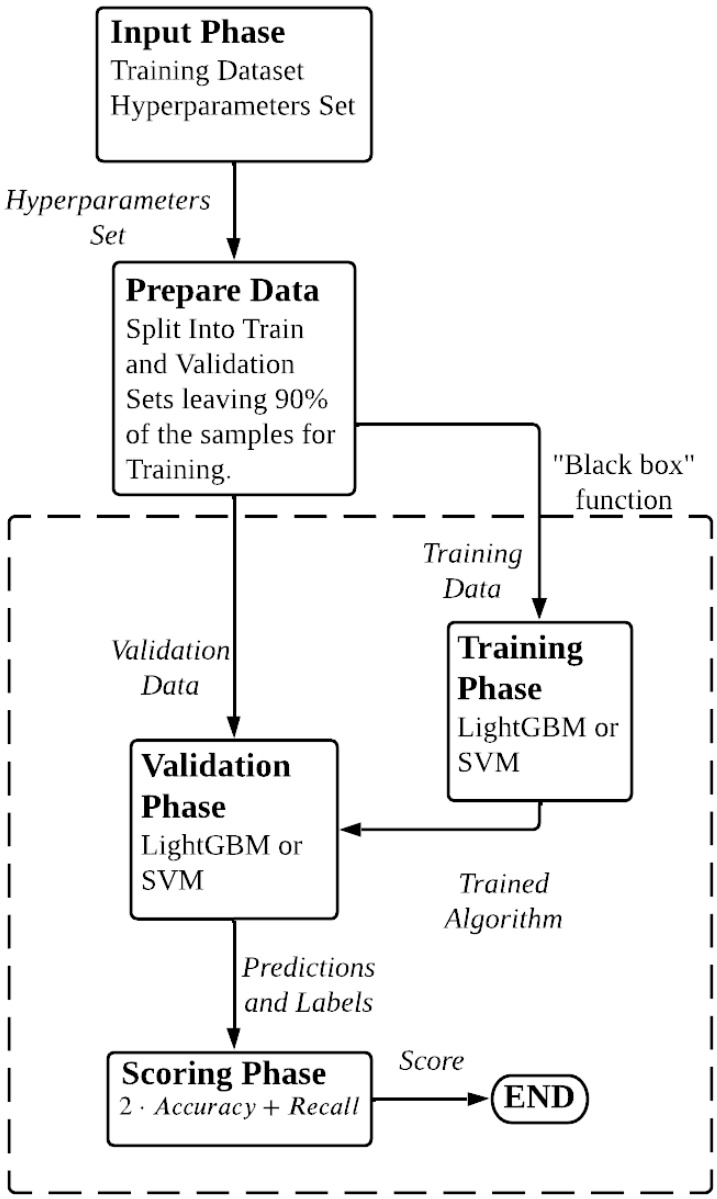
3.7.2 Functioning

The first step to applying Bayesian optimization to a machine learning problem is defining hyperparameters' search space and their ranges.

Next, an objective function is defined. In this case, the training and the validation of the LightGBM algorithm will occur inside the "black-box" function, and the predictions will be used to compute a score to be maximized. As stated in the previous section, the goal is to maximize the accuracy while forcing the misclassifications of the healthy sample to be mainly false positives, which is equivalent to maximizing the score from Equation 3.15.

One evaluation of the "black-box" function can be seen in Figure 3.10.

Figure 3.10 – "Black-box" Function For The Bayesian optimization



In Figure 3.10, only the training dataset is input in the "black-box" function, and the hyperparameters are optimized for a validation set extracted from it. This is done so the hyperparameters will not be chosen according to the testing set, preventing data leakage.

After the objective function is evaluated on the initial points, the Bayesian optimizer will construct a posterior distribution of functions through Gaussian processes (Nogueira, 2014). That way, a probabilistic model that best describes the objective function is achieved. Figure 3.11 shows the representation of a Gaussian Process applied to an objective function.

Figure 3.11 – Representation of a Gaussian Process of a Function (Pedregosa *et al.*, 2011a)

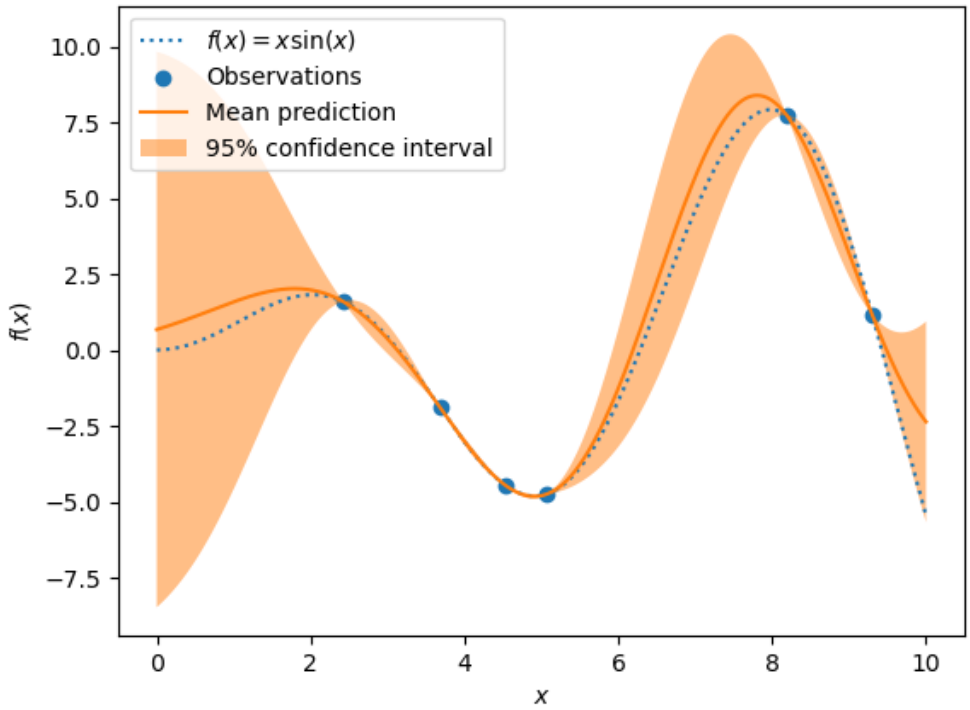


Figure 3.11 shows a "black-box" function evaluated on nine initial points represented by the blue dots. In addition, the orange line is the probabilistic model constructed by the Gaussian process based on the observed points of the original function. The light orange areas, on the other hand, are the confidence intervals, which are smaller when closer to the observed points and become more significant the further they get from them.

The next set of hyperparameters to evaluate will be chosen based on a trade-off between exploration and exploitation of the search space.

According to El-Omari (2021), exploration is the activity through which the search algorithm tries to explore as much broad search space as possible to evade falling into the local optima traps, which means the function should be evaluated in the least known areas. Whereas exploitation is the activity in which the searching algorithm tries to develop the finest

discovered solutions through some targeted approaches, which means the function should be evaluated closer to the best-known result so far.

This trade-off is achieved by the Expected Improvement acquisition function, which according to Brochu *et al.* (2010), is mathematically represented by Equation 3.16.

$$EI(x) = \begin{cases} (\mu(x) - f^*)\Phi(Z) + \sigma(x)\phi(Z) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases} \tag{3.16}$$

Where x is the input to the function based on the search space, $\mu(x)$ is the function being optimized at the proposed point, f^* is the best value of the function achieved so far, $\sigma(x)$ is the standard deviation of the function at x, $\Phi(Z)$ and $\phi(Z)$ are the cumulative distribution function and probability density function of the standard normal distribution respectively, and Z is the normalized distance between $\mu(x)$ and f^*, shown in Equation 3.17.

$$Z(x) = \begin{cases} \frac{\mu(x) - f^*}{\sigma(x)} & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases} \tag{3.17}$$

The x that maximizes the acquisition function from Equation 3.16 is the point with the best trade-off between exploration and exploitation. Therefore, the following evaluation of the "black-box" function will be at the x_{next} point, shown in Equation 3.18.

$$x_{next} = \underset{x}{\operatorname{argmax}}\, EI(x) \tag{3.18}$$

With a new evaluation of the function, the probabilistic model constructed by the Gaussian process is updated, allowing new regions of exploration and exploitation to be discovered. This process is repeated until the global optimum is achieved.

According to Brochu *et al.* (2010), it is often unclear how to handle the trade-off between exploration and exploitation in the acquisition function. In addition, unlike standard optimization algorithms such as gradient descent, Newton's, or quasi-Newton methods, the Bayesian optimization technique will not necessarily output the global optimum in its last iteration. Because of that, there is no general rule about how many initial points or the total number of iterations Bayesian optimization should evaluate. The optimization will evaluate five initial points and perform 45 iterations in this work.

4 RESULTS AND DISCUSSION

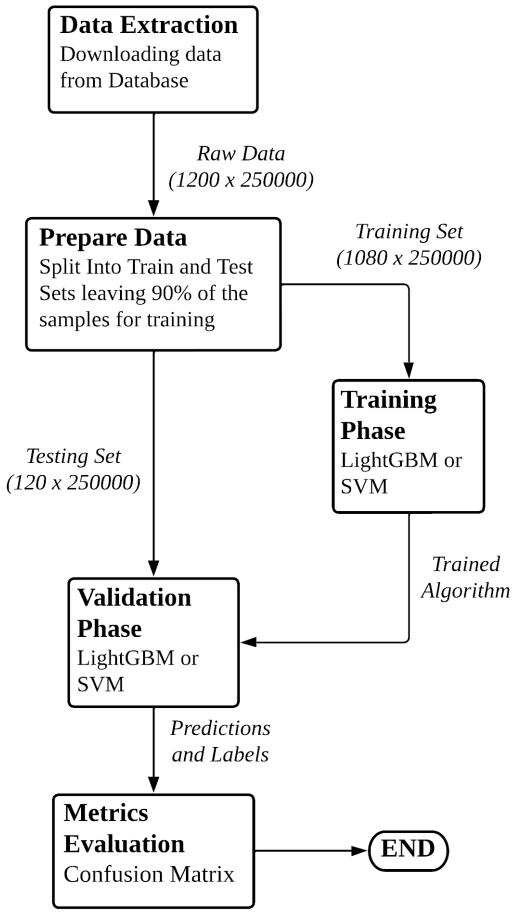
In this work, three sequential experiments with LightGBM and SVM are presented.

4.1 Unprocessed Data

The first step was a regular machine learning pipeline where LightGBM and SVM were trained on unprocessed data signals and their default hyperparameters (Ke *et al.*, 2017a; Pedregosa *et al.*, 2011c).

The proposed experiment is shown in Figure 4.1.

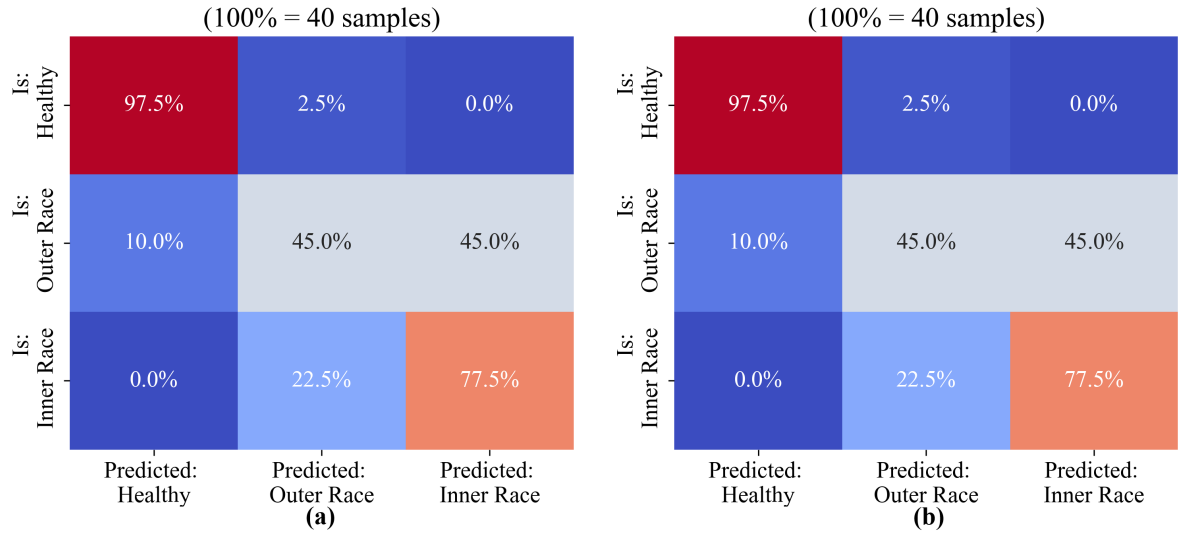
Figure 4.1 – Regular Machine Learning Pipeline



4.1.1 SVM

First, as shown in Figure 4.2, two SVMs were trained on the unprocessed data using the One-vs-One and the One-vs-Rest approaches.

Figure 4.2 – Confusion Matrix of SVM's Regular Machine Learning Pipeline using the One-vs-One (a) and the One-vs-Rest (b) approaches



The confusion matrices in Figure 4.2 show the SVM yielding the same results despite the different decision function. Because of that, the following experiments will use the One-vs-Rest approach, which is the model's default and recommended by Pedregosa *et al.* (2011c).

Figure 4.2 also demonstrates that the model accurately predicts instances of the healthy class, correctly identifying 97.5% of the samples. However, it misclassifies 2.5% of healthy instances as outer race fault. Notably, there are no instances of healthy being misclassified as inner race faults, indicating a clear distinction made by the model between these two classes.

In contrast, the model's performance on the outer race fault class is less satisfactory. It correctly classifies only 45.0% of these instances while misclassifying an equal percentage as inner race fault. A further 10.0% of outer race fault instances are incorrectly identified as healthy, suggesting a significant degree of confusion between the outer race fault and inner race fault classes.

The model's performance on the inner race fault class is moderately better than on the outer race fault class. It correctly identifies 77.5% of inner race fault instances but misclassifies 22.5% as outer race fault. There are no instances of inner race fault being misclassified as healthy, mirroring the pattern observed in the healthy class.

In addition, in this experiment, SVM took two minutes to train.

4.1.2 LightGBM

Next, the results of the LightGBM model are shown in Figure 4.3.

Figure 4.3 – Confusion Matrix of LGBM's Regular Machine Learning Pipeline

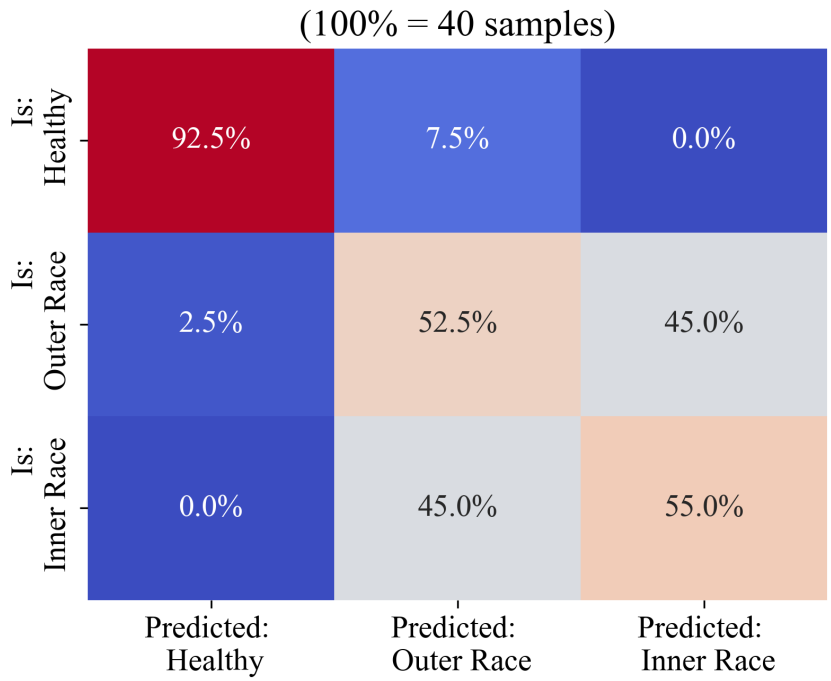


Figure 4.3 shows that while the model could accurately identify 92.5% of healthy operational conditions, 7.5% were wrongly classified as outer race faults. However, just like SVM, LightGBM predicted no healthy instances as having a defect in the inner race.

In contrast, the model shows unsatisfactory performance when identifying the inner and the outer race faults, misclassifying 45% of the faulty samples. The 52.5% and 55% accuracies show that the model performs slightly better than random chance when distinguishing the defects.

Regarding false negatives, similar to SVM, LightGBM predicted no inner race faults as being healthy. The model, however, misclassified 2.5% of the outer race samples as a normal rotor operation.

Furthermore, in this setup, LightGBM took 571 minutes to train.

4.1.3 Comparing the Algorithms

The results of the first experiment are shown in Table 4.1.

In this experiment, SVM and LGBM were trained on the same dataset, consisting of 1080 samples with 250,000 features each. However, the training time for the two algorithms

	SVM	LGBM
Training Time	2m	571m
Training Samples	1080	1080
Number of Features	250000	250000
Correct Healthy	97.5%	92.5%
Correct Outer Ring	45.0%	52.5%
Correct Inner Ring	77.5%	55.0%
Outer Ring Predicted Healthy	10.0%	2.5%
Inner Ring Predicted Healthy	0.0%	0.0%

Table 4.1 – Results of the First Experiment

was significantly different. The SVM model was trained in just 2 minutes, while the LGBM model required a much longer time, approximately 571 minutes. Support Vector Machine's superior computational efficiency in this setup is due to the vector-based training that deals better with more features, using a kernel function to compare all the features between two samples. LightGBM, on the other hand, uses a boosting approach that trains multiple decision trees, each calculating feature gradients to decide the best splitting point, as discussed previously.

Regarding performance, SVM correctly classified 97.5% of the healthy samples, which is higher than the 92.5% accuracy achieved by the LightGBM model. However, when classifying samples with outer ring defects, the LightGBM model outperformed the SVM model, correctly classifying 52.5% of the samples compared to SVM's 45.0%. The SVM model again outperformed the LightGBM model for inner ring defects, correctly classifying 77.5% of the samples compared to LightGBM's 55.0%.

Although neither model misclassified any inner ring defects as healthy, the SVM and the LightGBM misclassified 10.0% and 2.5% of outer ring defects, respectively. This false negative error was previously discussed as potentially life-threatening, and the following experiments aim to enhance the models' performance, eradicating these misclassifications.

4.2 Choosing the Slice Size

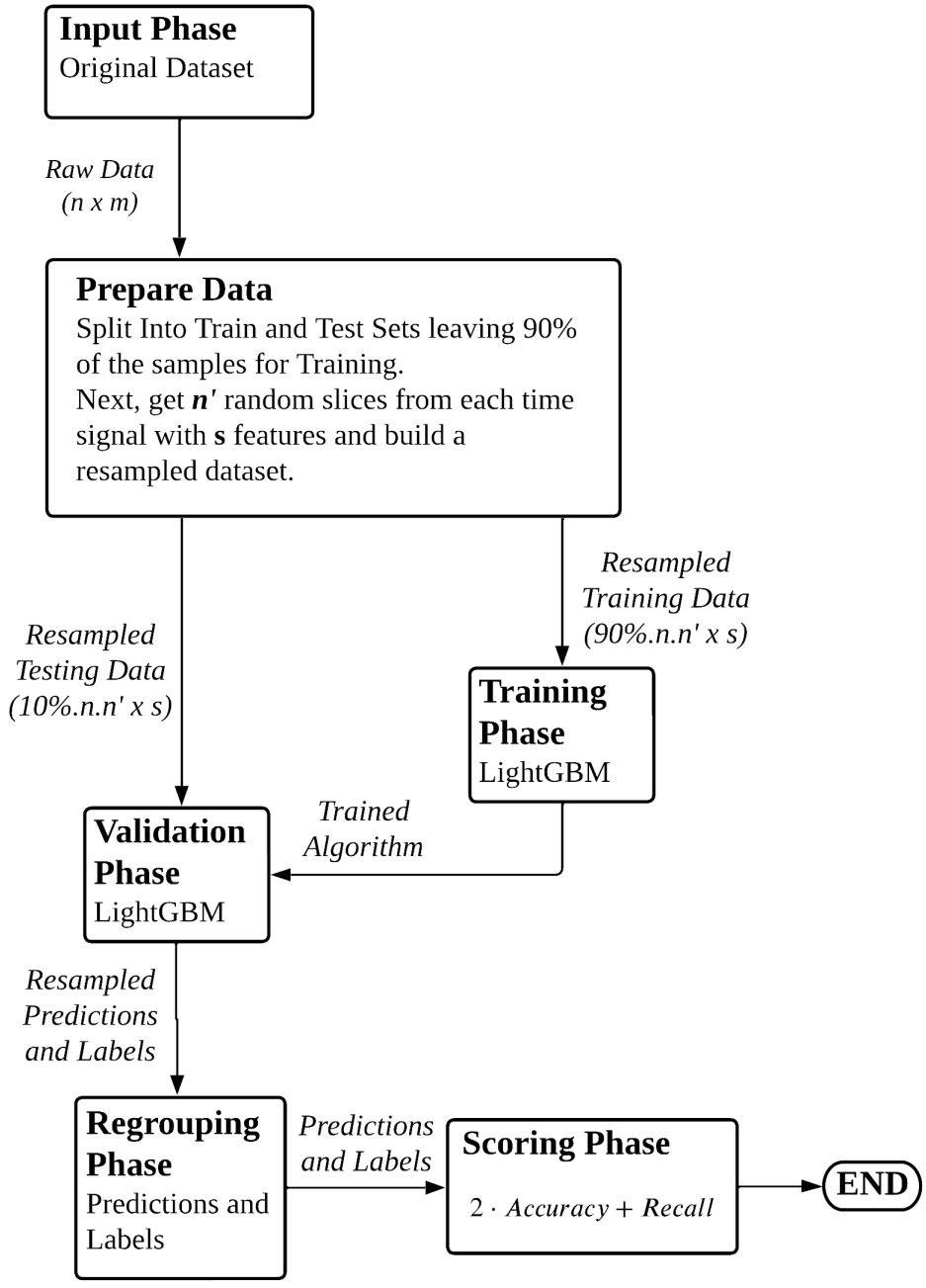
Next, the second experiment added the knowledge-based resampling technique to the machine learning pipeline.

It is essential to clarify that this experiment was proposed to enhance LightGBM's predictive capabilities and not SVM's. This is because resampling the dataset in this approach significantly augments the training samples and SVM's computational complexity scales

quadratically with the number of training samples. Therefore, only LightGBM is evaluated in this step.

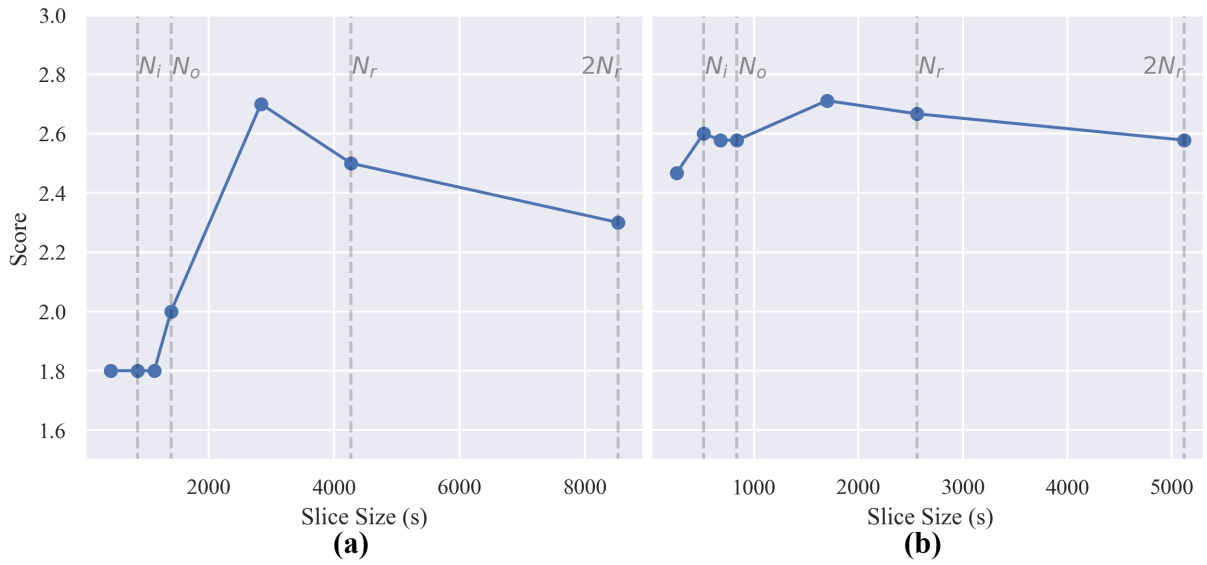
The proposed experiment is shown in Figure 4.4.

Figure 4.4 – Choosing the Slice Size Pipeline



As discussed previously and shown in Table 3.6, seven different slice sizes were evaluated for each rotor speed in this experiment, making a total of 14 results. Every evaluation was scored using Equation 3.15, allowing understanding of the model's sensitivity to slice size, as shown in Figure 4.5.

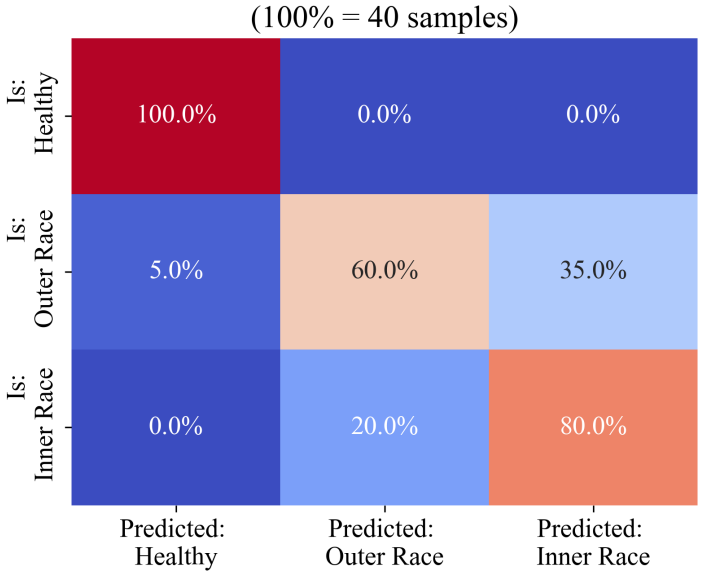
Figure 4.5 – Score Sensitivity to Slice Size for Rotor Speed of 15 Hz (a) and 25 Hz (b)



As expected, Figure 4.5 shows the best scores when employing a slice size between the BPFO and one bearing rotation, which consists of 2832 and 1699 samples for the 15 Hz and the 25 Hz speeds respectively. This behavior is because a slice size with fewer features than the BPFO does not provide sufficient information for detecting outer race faults, and more features than a full rotation introduces confusion into the algorithm's training, rendering it less effective in fault detection.

Figure 4.6 shows the resulting confusion matrix after training LightGBM with the knowledge-based resampled datasets.

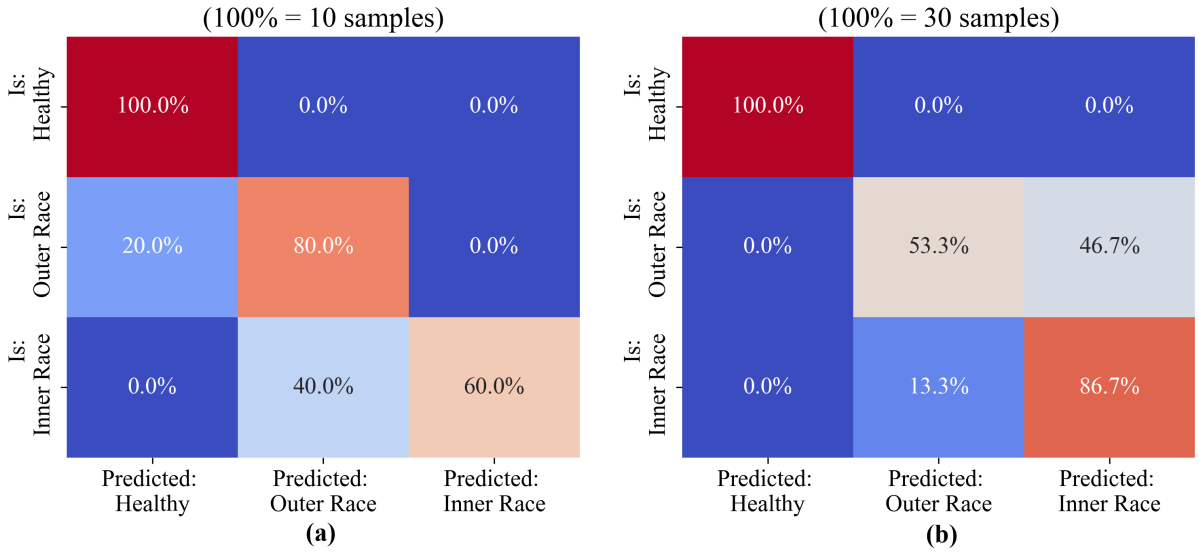
Figure 4.6 – Resulting Confusion Matrix of LightGBM with Knowledge-based Resampling



In comparison to the last experiment, Figure 4.6 shows that the knowledge-based resampling had a positive effect on correctly identifying healthy samples and distinguishing the faulty conditions. In this setup, the healthy samples are classified with 100% accuracy while the inner and the outer race fault predictions improved by 7.5% and 25% respectively.

Figure 4.7 separates the confusion matrix from Figure 4.6 by rotor speed to allow a better understanding of the model's limitations.

Figure 4.7 – Resulting Confusion Matrix of LightGBM under knowledge-based Resampling at Rotor Speed 15 Hz (a) and 25 Hz (b)



In Figure 4.7 it is clear that LightGBM in this setup only yields false negatives in the outer race samples at 15 Hz. However, overall outer race fault identification was better at 15 Hz than at 25 Hz, with 80% and 53.3%, respectively.

For the inner race fault, on the other hand, the superior speed made it easier for the algorithm to output accurate predictions, with 86.7% as opposed to the 60% achieved by the machine learning method trained with the 15 Hz dataset.

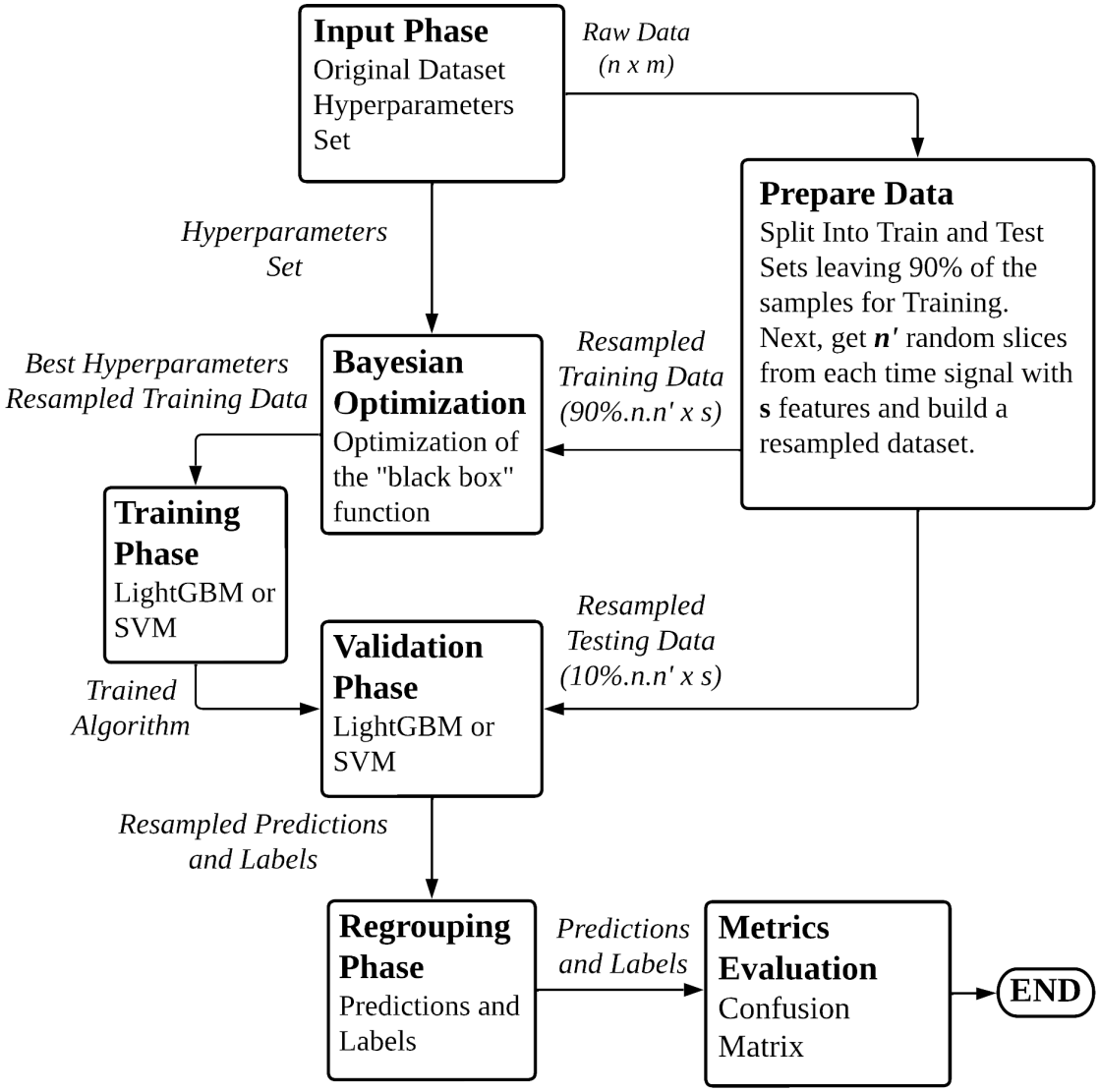
From this experiment, it is clear that the knowledge-based resampling was beneficial for LightGBM's training process. Additionally, it was shown that different speeds affect the algorithm's predictive capabilities, which justifies using Bayesian optimization for individual hyperparameter tuning to maximize the results.

4.3 Optimizing Hyperparameters

According to (Hutter *et al.*, 2019), it is widely acknowledged that hyperparameter optimization can significantly improve machine learning models' performance over the default settings provided by their libraries. Because of that, this work uses Bayesian optimization to find the best set of hyperparameters.

Figure 4.8 shows the whole proposed experiment.

Figure 4.8 – Optimizing Hyperparameters Machine Learning Pipeline



4.3.1 Choosing the Observation Range

There is no general rule for choosing the observation range for hyperparameters, but in this work, they were chosen as follows for SVM:

- **Regularization Factor C:** A wide range from one to one hundred was evaluated. A low C leads to a permissive soft margin, while a high C leads to overfitting.

For LightGBM, on the other hand:

- **Regularization Factors:** There can be no negative values, and the upper bound was set to two so that the regularizations would not have a higher effect on the loss function than the class weights. This hyperparameter can be continuous.
- **Learning Rate:** It must be higher than zero and generally smaller than one. This hyperparameter can be continuous.
- **Number of Estimators:** LightGBM default value is one hundred, which was set as the lower bound. The higher bound, on the other hand, should not be too high since the optimizer could choose a value that would lead to model overfitting. Because of that, a value of three hundred was chosen as the upper limit. This hyperparameter must be an integer.
- **Number of Leaves:** LightGBM default value is thirty-two, and according to the documentation (LightGBM, 2023), setting the number of leaves to eighty, which is set as the upper bound, generally gives a high accuracy without making the model too complex. This hyperparameter must be an integer.

The algorithms' class weights are also tuned, and they were chosen as:

- **Class Weights:** A wide range from one to ten was evaluated to allow the interpretation of an operation condition's importance compared to another, i.e., the inner ring fault being twice more important than the outer ring one. Even though the class weights can be continuous, they will be rounded to discrete values.

4.3.2 SVM

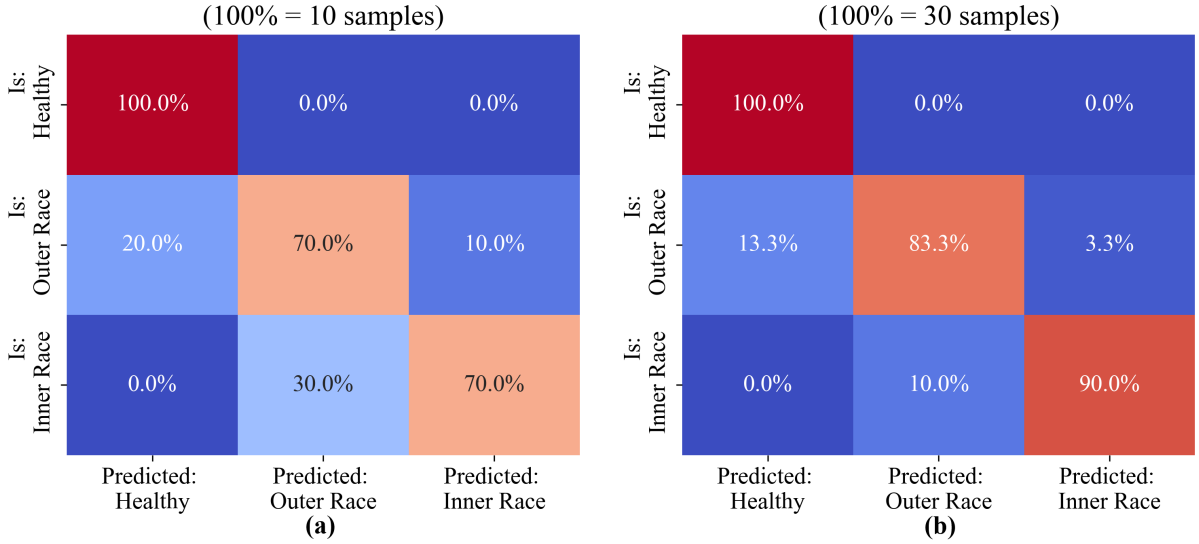
The observation ranges for SVM's hyperparameters and the optimization results can be seen in Table 4.2.

Table 4.2 – SVM's Optimized Hyperparameters

Parameter	Observation Range	$f_r = 15Hz$	$f_r = 25Hz$
Slice Size	-	2832	1699
Regularization Factor C	(1, 100)	15	35
Healthy Class Weight	(1, 10)	5	1
Outer Ring Class Weight	(1, 10)	5	2
Inner Ring Class Weight	(1, 10)	2	6
	Score	2.113	2.684

Table 4.2 shows SVM could be better tuned for 25 Hz with a score of 2.684 in contrast to the 2.113 achieved by the model trained with the 15 Hz data. The smaller score is due to the higher false negative rate in the outer race, as shown in Figure 4.9. Additionally, in the 25 Hz modeling, Bayesian optimization gave higher importance to the faults than to the healthy samples, which allowed the algorithm to achieve higher accuracies resulting in a better score.

Figure 4.9 – Resulting Confusion Matrix of SVM's Complete Pipeline at Rotor Speeds 15 Hz (a) and 25 Hz (b)



Compared to SVM's original setup, shown in Figure 4.2, Figure 4.9 shows a significant improvement of more than 20 pp in predicting the inner and the outer race faults. In addition, all healthy samples were correctly classified. However, SVM's limited hyperparameter flexibility worsened the method's false negative prediction, as 15% of the outer race faults were predicted as healthy, as opposed to the starting 10%.

It is important to remember that SVM's training samples were limited to 30000 due to the quadratic computational complexity. In this setting, the training time tripled, with each algorithm taking 6 minutes to train.

4.3.3 LightGBM

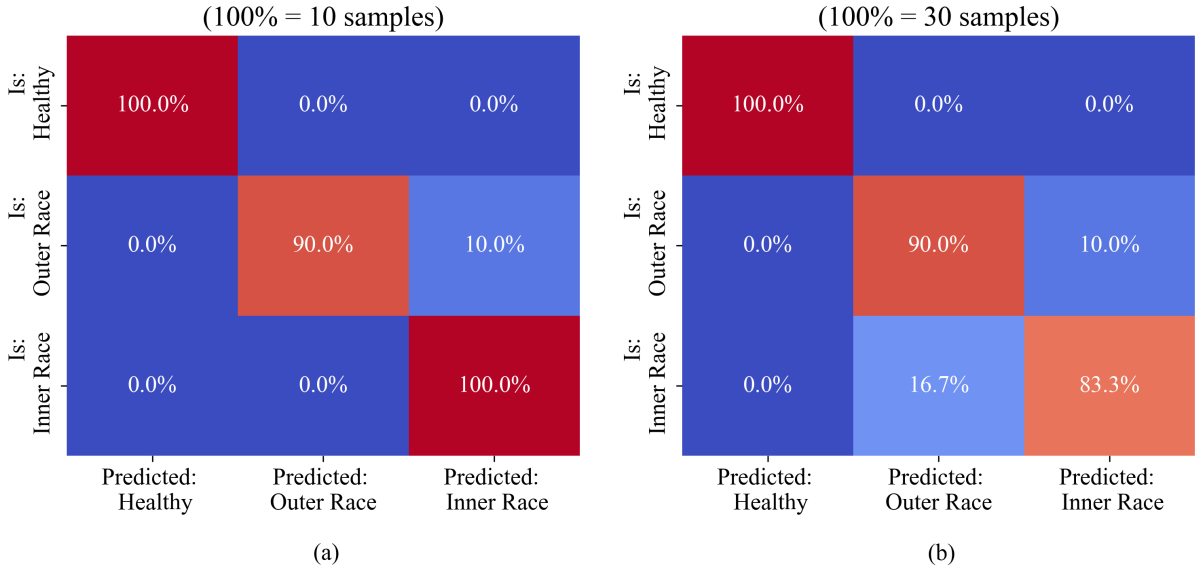
The observation ranges for LightGBM's hyperparameters and the optimization results can be seen in Table 4.3.

Table 4.3 – LightGBM's Optimized Hyperparameters

Parameter	Observation Range	$f_r = 15Hz$	$f_r = 25Hz$
Slice Size	-	2832	1699
L1 Regularization	(0, 2)	0.187	0.863
L2 Regularization	(0, 2)	1.676	0.233
Learning Rate	(0.01, 1)	0.45	0.70
Number of Estimators	(100, 300)	202	195
Number of Leaves	(20, 80)	20	57
Healthy Class Weight	(1, 10)	2	3
Outer Ring Class Weight	(1, 10)	1	6
Inner Ring Class Weight	(1, 10)	1	6
	Score	2.672	2.889

The results from the knowledge-based resampling along with hyperparameter optimization are shown in Figure 4.10.

Figure 4.10 – Resulting Confusion Matrix of LightGBM's Complete Pipeline at Rotor Speeds 15 Hz (a) and 25 Hz (b)



Compared to the last experiment, shown in Figure 4.6, Figure 4.10 shows a significant predictive power improvement of more than 30 pp in predicting the outer race faults. Additionally, although inner race accuracy only improved by 7.5 pp, it is notable that the hyperparameter tuning made the algorithm more capable of fault distinction.

In addition, all healthy samples were correctly classified and there were no false negatives, which was a fundamental goal of this work. Furthermore, in this setting, the training time was reduced to 7 minutes, which corresponds to a 99.8% reduction from the original experiment.

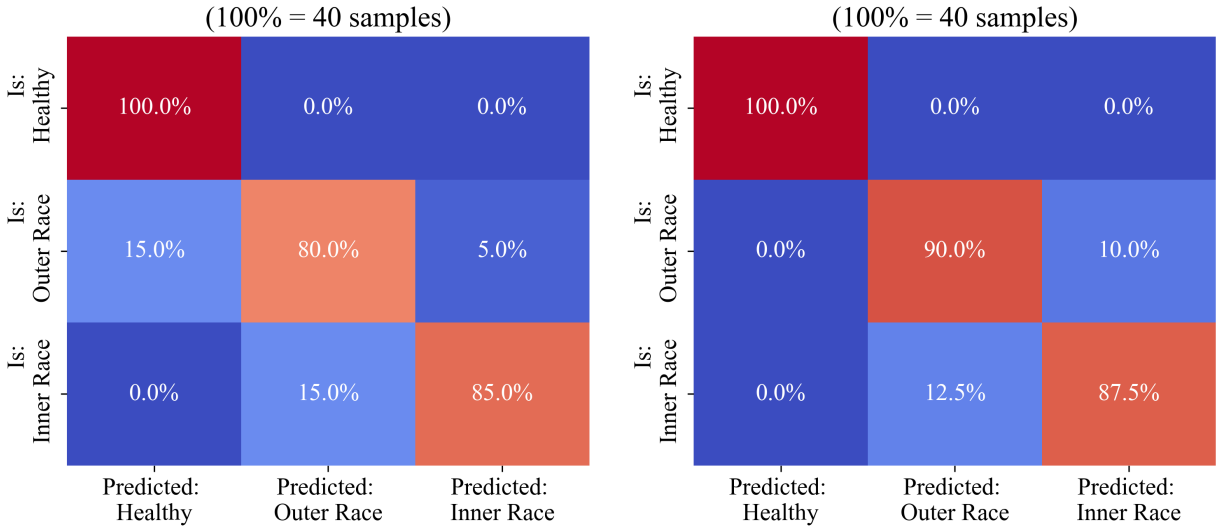
The major training time reduction is explained by the significant reduction in features since LightGBM's decision tree approach evaluates the gradient of each feature individually to find the best splitting point.

From this experiment, it is clear that combining the knowledge-based resampling with the Bayesian optimization can significantly improve LightGBM predictive power and training time, making a robust and reliable classifier that mitigates the prediction of any false negatives.

4.3.4 Comparing the Algorithms

The final confusion matrix of the algorithms can be seen in Figure 4.11.

Figure 4.11 – SVM's (a) and LightGBM's(b) Final Confusion Matrix



The results of the algorithms after the knowledge-based resampling and the optimization process are summarized in Table 4.4.

Despite being trained with 35 times more data, LightGBM took almost the same time to train as SVM, clearly showing LGBM's superiority when handling larger datasets. The optimization score, which measures the algorithm's tuning performance in relation to the goal

Table 4.4 – Results of the Complete Pipeline

	SVM	LGBM
Training Time	6m	7m
Training Samples (15 Hz)	30000	764640
Training Samples (25 Hz)	30000	1376190
Optimization Score (15 Hz)	2.113	2.672
Optimization Score (25 Hz)	2.684	2.889
Correct Healthy	100.0%	100.0%
Correct Outer Ring	80.0%	90.0%
Correct Inner Ring	85.0%	87.5%
Outer Ring Predicted Healthy	15.0%	0.0%
Inner Ring Predicted Healthy	0.0%	0.0%

of this work, was higher for LGBM at both speeds, scoring 2.672 and 2.889 compared to SVM's 2.113 and 2.684 at 15 Hz and 25 Hz, respectively.

The superior score is visible when the algorithm's performance regarding the individual classes is discussed. Both algorithms had a perfect score of 100% when identifying healthy samples. However, when identifying samples with outer and inner ring issues, LGBM outperformed SVM. LGBM correctly identified 90% of outer ring issues and 87.5% of inner ring issues, compared to SVM's 80% and 85%, respectively. Furthermore, although neither algorithm misclassified any inner ring defects as healthy, LGBM did not yield outer ring false negatives, while SVM misclassified 15% of these instances as healthy, which lowered the score significantly.

4.4 Consolidated Results

The results of all experiments can be seen in Table 4.5.

Table 4.5 – Results of All Experiments

	Original		Optimized	
	SVM	LGBM	SVM	LGBM
Training Samples	1080	1080	60000	2140830
Training Time	2m	571m	6m	7m
Correct Healthy	97.5%	92.5%	100.0%	100.0%
Correct Outer Ring	45.0%	52.5%	80.0%	90.0%
Correct Inner Ring	77.5%	55.0%	85.0%	87.5%
Outer Ring Predicted Healthy	10.0%	2.5%	15.0%	0.0%
Inner Ring Predicted Healthy	0.0%	0.0%	0.0%	0.0%

In the initial experiment, both LightGBM and SVM were tested on the original data. While LightGBM classified healthy samples with 92.5% accuracy, SVM outperformed it by correctly identifying 97.5% of the healthy samples. However, when it came to distinguishing defects, SVM's performance was not as robust as LightGBM's. Specifically, 10.0% of the Outer Ring faults were incorrectly classified as healthy by SVM, compared to only 2.5% by LightGBM.

The SVM algorithm also showed a varied performance in correctly identifying Outer and Inner Ring faults, with accuracies of 45.0% and 77.5% respectively. In comparison, LightGBM had a more balanced performance, but slightly higher than random chance, correctly classifying 52.5% of Outer Ring faults and 55.0% of Inner Ring faults.

Next, a knowledge-based resampling technique was proposed to improve LightGBM's predictive capabilities, and using a slice size between the BPFO and a full bearing rotating, not only reduced LightGBM's training time by 99.8% (from 571 to 7 minutes) but slightly improved the algorithm's overall accuracy.

The Bayesian optimization extended LightGBM's predictive capabilities even further. The combined techniques allowed LightGBM to predict healthy samples with 100% accuracy, and SVM to match this performance. Furthermore, even though the methodology was not developed for SVM, the accuracy of both algorithms in correctly classifying outer and inner ring faults improved significantly, with SVM reaching 80.0% and 85.0% respectively, and LightGBM achieving 90.0% and 87.5%.

In this optimized setting, both LightGBM and SVM predicted no false negatives for Inner Ring faults, which was a fundamental goal of this work. However, SVM still had a 15.0% rate of false negatives for outer ring faults, due to the algorithm's limited flexibility in hyperparameter tuning and the absence of the boosting process.

In comparison to Paderborn's benchmark research (Lessmeier *et al.*, 2016b), the last LightGBM setup achieved a total accuracy of 92.5%, which is higher than most machine learning models tested by authors except for a Classification and Regression Tree (CART), a Random Forest (RF), and ensemble model, which achieved 98.3% accuracy each. However, a time-domain approach was evaluated in this work, while in the Paderborn study, a feature engineering pipeline based on the frequency domain was proposed. Finally, the original research showed that tree-based models performed better than other machine learning algorithms, so it is likely that LightGBM can achieve similar or better results using the same methodology.

5 CONCLUSION

This research combines LightGBM with the slicing of time series and the Bayesian optimization to build a robust and reliable classifier of mechanical faults in rotating machines that achieve high accuracy while minimizing false negative rates.

The data was extracted from the Paderborn University Repository and contained experimental samples gathered on an accelerated lifetime test rig. The tested rotating machine was subject to three operational conditions: healthy, with an inner race and an outer race fault. The samples contained acceleration data that were measured for approximately 4 seconds at a sampling frequency of 64 kHz. In addition, various rotor parameters could be adjusted, and to build the complete dataset, the samples were cropped to 250000 features. The most critical parameter for this research was the rotational speeds of 900 rpm (15 Hz) and 1500 rpm (25 Hz), which were also available as a feature. The individual samples were merged into one balanced dataset with 1200 rows and 250001 columns.

The first experiment consisted of a regular machine learning pipeline, splitting the original data into training and testing sets. The training set was input to LightGBM and SVM default settings. The first was this research's primary goal, while the second was mainly used as a benchmark since it is a widely used algorithm in fault detection with machine learning. The initial results showed that SVM outperformed LightGBM in classifying healthy samples, with 97.5% and 92.5% accuracy, respectively. However, although SVM could predict inner race faults with 77.5% accuracy, the algorithm classified 10.0% of the outer race faults as healthy, compared to only 2.5% by LightGBM. Therefore, the first experiment showed neither algorithm was by this research's goals.

A knowledge-based resampling technique was proposed as an alternative to enhance LightGBM's predictive capabilities. It consisted of randomly slicing and stacking the time series, allowing overlapping, meaning a feature could appear multiple times on the dataset, but necessarily in different positions. It is essential to remember that this methodology was not developed for SVM due to the kernel function that causes the algorithm to quadratically scale its complexity to the number of training samples, which is significantly increased in this methodology.

Seven distinct slice sizes were evaluated based on the BPFI, the BPFO, and the

bearing rotation to find the best slice size. A sensitivity plot showed an optimal result for a slice size between the BPFO and one bearing rotation. That is because fewer features than one BPFO impact signal event do not provide sufficient information for detecting outer race faults, while more features than a full rotation introduce confusion into the algorithm's training, rendering it less effective in fault detection.

The knowledge-based resampling positively affected the correct identification of healthy samples and the distinction of faulty conditions. In this setup, the healthy samples were classified with 100% accuracy while the inner and the outer race fault predictions improved by 7.5% and 25%, respectively. Additionally, LightGBM in this experiment only yielded false negatives in the outer race samples at 15 Hz, which clarified that different speeds affect the algorithm's predictive capabilities. This justified using Bayesian optimization for individual hyperparameter tuning to maximize the results.

The Bayesian optimizer was used to enhance the algorithms' predictive capabilities. While SVM has only one main hyperparameter, its regularization factor, LightGBM has five, along with the boosting process, which trains multiple classifiers, showing superior flexibility toward various machine learning problems. Both algorithm's loss functions had their class weights optimized. The Bayesian optimization and knowledge-based resampling allowed SVM and LightGBM to predict healthy samples with 100% accuracy. Furthermore, even though the methodology was not developed for SVM, the accuracy of both algorithms in correctly classifying outer and inner ring faults improved significantly, with SVM reaching 80.0% and 85.0%, respectively, and LightGBM achieving 90.0% and 87.5%.

From the results obtained, it is clear that this research's main contribution is the development of a machine-learning pipeline that works on raw signal data. The resampling technique focuses on critical data sequences without increasing computation complexity. Another advantage of using time instead of frequency-domain approaches is that the latter can be less robust to variations in operating conditions since frequency-domain features may be more sensitive to changes in speed and load, which can alter the spectral distribution of the vibration signals.

In conclusion, the LightGBM algorithm trained with raw time-series data is promising for detecting and classifying mechanical faults in rotating machines. The proposed data resampling technique has improved the model's accuracy and recall. Lastly, the Bayesian optimization extended LightGBM's predictive capabilities further, eradicating false negatives and

building a robust and reliable fault classifier.

Future studies should investigate whether the methodology can also be applied to correctly classify other types of mechanical faults. It could also be evaluated whether the proposed time series slicing technique can enhance the predictive power of different machine learning methods, such as various architectures of artificial neural networks. Furthermore, whether other data augmentation techniques can achieve similar or even better results could be researched.

BIBLIOGRAPHY

Alves, D. S.; Daniel, G. B.; Castro, H. F. de; Machado, T. H.; Cavalca, K. L.; Gecgel, O.; Dias, J. P.; Ekwaro-Osire, S. Uncertainty quantification in deep convolutional neural network diagnostics of journal bearings with ovalization fault. **Mechanism and Machine Theory**, v. 149, p. 103835, jul 2020.

Berkson, J. Application of the logistic function to bio-assay. **Journal of the American Statistical Association**, v. 39, p. 357–365, 1944.

Brasil. **Balanço Energético Nacional**. Ministério de Minas e Energia. Secretaria de Planejamento e Desenvolvimento Energético. Brasília, DF, 2019. Disponível em: <https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/balanco-energetico-nacional-2019>.

Brasil. **Plano Decenal de Expansão de Energia 2029**. Ministério de Minas e Energia. Secretaria de Planejamento e Desenvolvimento Energético. Brasília, DF, 2019. Disponível em: <https://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/Documents/PDE2029.pdf>.

Breiman, L.; Friedman, J.; Stone, C. J.; Olshen, R. A. **Classification and Regression Trees**. Philadelphia, PA: Chapman & Hall/CRC, 1984.

Brochu, E.; Cora, V. M.; Freitas, N. de. **A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning**. 2010.

Chen, T.; Guestrin, C. XGBoost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2016. (KDD '16), p. 785–794. ISBN 978-1-4503-4232-2. Disponível em: <http://doi.acm.org/10.1145/2939672.2939785>.

Chong, D. **Deep Dive into Netflix's Recommender System**. 2020. Disponível em: <https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48>.

Cortes, C.; Vapnik, V. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, 1995.

Cramer, J. S. **The Origins of Logistic Regression**. Faculty of Economics and Econometrics. University of Amsterdam and Tinbergen Institute. Amsterdam, 2002.

El-Omari, N. Sea lion optimization algorithm for solving the maximum flow problem. **IJCSNS International Journal of Computer Science and Network Security**, v. 20, p. 30–68, 11 2021.

Frazier, P. I. **A Tutorial on Bayesian Optimization**. 2018.

Freund, Y.; Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of Computer and System Sciences**, v. 55, n. 1, p. 119–139, 1997.

Friedman, J. Greedy function approximation: A gradient boosting machine. **The Annals of Statistics**, v. 29, 11 2000.

Gecgel, O.; Dias, J. P.; Ekwaro-Osire, S.; Alves, D. S.; Machado, T. H.; Daniel, G. B.; Castro, H. F. de; Cavalca, K. L. Simulation-Driven Deep Learning Approach for Wear Diagnostics in Hydrodynamic Journal Bearings. **Journal of Tribology**, v. 143, n. 8, nov 2020.

Géron, A. **Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow**. Sebastopol, CA: O'Reilly Media, Inc, 2019. ISBN 978-1492032649.

Hochreiter, S.; Schmidhuber, J. Long short-term memory. **Neural computation**, v. 9, p. 1735–80, 12 1997.

Hutter, F.; Kotthoff, L.; Vanschoren, J. (Ed.). **Automated Machine Learning: Methods, Systems, Challenges**. 1. ed. Cham, Switzerland: Springer Nature, 2019. (The Springer Series on Challenges in Machine Learning).

Iglesias, G.; Talavera, E.; González-Prieto, Á.; Mozo, A.; Gómez-Canaval, S. Data augmentation techniques in time series domain: a survey and taxonomy. **Neural Computing and Applications**, mar 2023.

Igual, L.; Seguí, S. **Introduction to Data Science: a Python Approach to Concepts, Techniques and Applications**. Cham, Switzerland: Springer, 2017. ISBN 978-3-319-50016-4.

Iwana, B. K.; Uchida, S. An empirical survey of data augmentation for time series classification with neural networks. **PLOS ONE**, v. 16, p. 1–32, 07 2021.

JCU. **The Role Of Data Science In Advancing Medicine**. 2019. Disponível em: <https://online.jcu.edu.au/blog/data-science-medicine>.

Jia, X.; Xiao, B.; Zhao, Z.; Ma, L.; Wang, N. Bearing fault diagnosis method based on CNN-LightGBM. **IOP Conference Series: Materials Science and Engineering**, v. 1043, n. 2, p. 022066, jan. 2021.

Kaggle. **State of Data Science and Machine Learning 2022**. 2022. Disponível em: <https://www.kaggle.com/kaggle-survey-2022>.

Ke, G.; Meng, Q.; Finely, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In: **Advances in Neural Information Processing Systems 30 (NIP 2017)**. [s.n.], 2017. Disponível em: <https://www.microsoft.com/en-us/research/publication/lightgbm-a-highly-efficient-gradient-boosting-decision-tree/>.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In: Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; Garnett, R. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2017. v. 30. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf>.

Kolar, D.; Lisjak, D.; Pająk, M.; Gudlin, M. Intelligent fault diagnosis of rotary machinery by convolutional neural network with automatic hyper-parameters tuning using bayesian optimization. **Sensors**, v. 21, n. 7, 2021.

Lessmeier, C.; Kimotho, J.; Zimmer, D.; Sextro, W. Paderborn University, 2016. Accessed: 2023-04-18. KAt-DataCenter. Paderborn University. Disponível em: <mb.uni-paderborn.de/kat/datacenter>.

Lessmeier, C.; Kimotho, J. K.; Zimmer, D.; Sextro, W. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification. In: **European Conference of the Prognostics and Health Management Society**. [S.l.: s.n.], 2016. v. 3, n. 1.

LightGBM. **LightGBM's documentation: Parameter Tuning**. 2023. Disponível em: <https://lightgbm.readthedocs.io/en/v3.3.2/index.html>.

Liu, R.; Yang, B.; Zio, E.; Chen, X. Artificial intelligence for fault diagnosis of rotating machinery: A review. **Mechanical Systems and Signal Processing**, v. 108, p. 33–47, 2018.

Mason, L.; Baxter, J.; Bartlett, P.; Frean, M. Boosting algorithms as gradient descent. **Advances in neural information processing systems**, v. 12, p. 512–518, 01 1999.

McCarthy, J.; Feigenbaum, E. A. In Memoriam: Arthur Samuel: Pioneer in Machine Learning. **AI Magazine**, v. 11, n. 3 SE - Editorials, p. 10, sep 1990.

Minsky, M.; Papert, S. A. **Perceptrons : An Introduction to Computational Geometry**. Cambridge, Mass: MIT Press, 1969. ISBN 978-0262631112.

Nogueira, F. **Bayesian Optimization: Open source constrained global optimization tool for Python**. 2014. Disponível em: <https://github.com/fmfn/BayesianOptimization>.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Rbf svm parameters. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Support vector machines. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A. V.; Gulin, A. **CatBoost: unbiased boosting with categorical features**. 2019.

Randall, R. B.; Antoni, J. Rolling element bearing diagnostics—a tutorial. **Mechanical Systems and Signal Processing**, v. 25, n. 2, p. 485–520, 2011.

Rosenblatt, F. The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain. **Psychological review**, v. 65 6, p. 386–408, 1958.

Van Rossum, G.; Drake, F. L. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986.

Samuel, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, 1959.

Smith, W.; Randall, R. B. Rolling element bearing diagnostics using the case western reserve university data: A benchmark study. **Mechanical Systems and Signal Processing**, v. 64-65, 05 2015.

Turing, A. M. **COMPUTING MACHINERY AND INTELLIGENCE**. Mind. A Quartely Review of Psychology ad Philosofy, 1950. Disponível em: <https://academic.oup.com/mind/article/LIX/236/433/986238>.

Verhulst, P.-F. Recherches mathématiques sur la loi d'accroissement de la population. p. 1–42, 1845.

Verhulst, P.-F. **Deuxième mémoire sur la loi d'accroissement de la population**. [S.l.]: Mémoires de l'Académie Royale des Sciences et Belles Lettres de Bruxelles, 1847. 1–32 p.

Vogels, M.; Zoeckler, R.; Stasiw, D. M.; Cerny, L. C. P. F. Verhulst's "notice sur la loi que la populations suit dans son accroissement" from correspondence mathematique et physique. Ghent, vol. X, 1838. **Journal of Biological Physics**, v. 3, n. 4, p. 183–192, 1975.

Wellers, D.; Elliot, T.; Noga, M.; Uga, B.; Buga, U. **8 Ways Machine Learning Is Improving Companies' Work Processes**. 2017. Disponível em: <https://hbr.org/2017/05/8-ways-machine-learning-is-improving-companies-work-processes>.

Xu, Y.; Cai, W.; Wang, L.; Xie, T. Intelligent diagnosis of rolling bearing fault based on improved convolutional neural network and lightgbm. **Shock and Vibration**, v. 2021, p. 1205473, Dec 2021.