الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry for Higher Education and Scientific Research
جامعة غرداية
University of Ghardaïa

Registration N°

/…../…../…../…../…..

كلية العلوم والتكنولوجيا
Faculty of Science and Technology
قسم الآلية و الكهروميكانيك
Department of Automation and Electromechanical

Thesis of end study, with a view to obtaining the diploma

Master

Domaine: Science and Technology
Field: Automation and Electromechanical
Specialty : Industrial Maintenance

Theme

Fault Diagnosis in HVDC Systems Using Machine Learning

Presented by :
Bouras Taha
Blidi Djamal

Publicly defended on …../06/2025

Jury members :

Hamed BOUKARI	**MCB**	**Univ. Ghardaia**	**Chairman**
Hemza MEDOUKALI	**MCA**	**Univ. Ghardaia**	**Supervisor**
Fatma BOUCHELGA	**MCA**	**Univ. Ghardaia**	**Examiner**
Farid HADJRIOUA	**MRB**	**UREAR**	**Examiner**

University Year: 2024/2025

ملخص

تتناول هذه المذكرة تشخيص أعطال أنظمة التيار المستمر عالي الجهد (HVDC) باستخدام تعلم الآلة، مع التركيز على التعميم لظروف أعطال غير مرئية. تم نمذجة نظام HVDC بمحول (LCC) في MATLAB/Simulink لمحاكاة أعطال متنوعة، وتوليد بيانات لاستخلاص ميزات إحصائية. تم تقييم سبعة نماذج تعلم آلي، بما في ذلك الانحدار اللوجستي، آلة المتجهات الداعمة، أقرب الجيران، شجرة القرار، الغابة العشوائية، تعزيز التدرج، والشبكة العصبونية (MLP). شمل التقييم تقسيما عشوائيا قياسيا واختبار تعميم حاسم على مقاومات أعطال لم تُستخدم في التدريب. أظهرت الغابة العشوائية والشبكة العصبية وتعزيز التدرج متانة فائقة، حيث حققت الغابة العشوائية أعلى درجة دقة في اختبار التعميم. وهذا يؤكد أهمية اختبار التعميم لتشخيص موثوق.

كلمات مفتاحية: تشخيص الأعطال، التعلم الآلي، أنظمة التيار المستمر عالي الجهد، التعميم، الغابة العشوائية، المحاكاة.

Abstract

This thesis investigates machine learning (ML) for fault diagnosis in high voltage direct current (HVDC) systems, emphasizing generalization to unseen fault conditions. A line commutated converter (LCC) HVDC system was modeled in MATLAB/Simulink to simulate diverse faults, generating data from which statistical features were extracted. Seven ML models (logistic regression, SVM, KNN, decision tree, random forest, gradient boosting, MLP) were evaluated. Experiments included a standard random split and a crucial generalization test on unseen fault resistances. Random forest, neural network, and gradient boosting demonstrated superior robustness, with random forest achieving the highest accuracy in generalization. The study highlights the importance of generalization testing for reliable fault diagnosis.

Key words: *Fault diagnosis, Machine learning, HVDC systems, Generalization, Random forest, Simulation.*

Résumé

Cette thèse étudie l'utilisation de l'apprentissage automatique (AA) pour le diagnostic des défauts dans les systèmes à courant continu haute tension (CCHT), en mettant l'accent sur la généralisation à des conditions de défaut non observées. Un système CCHT à convertisseur commuté par le réseau (LCC) a été modélisé dans MATLAB/Simulink pour simuler divers défauts, générant des données à partir desquelles des caractéristiques statistiques ont été extraites. Sept modèles d'AA (régression logistique, SVM, KNN, arbre de décision, forêt aléatoire, gradient boosting, MLP) ont été évalués. Les expériences comprenaient une division aléatoire standard et un test de généralisation crucial sur des résistances de défaut non utilisées lors de l'entraînement. La forêt aléatoire, le réseau neuronal et le gradient boosting ont démontré une robustesse supérieure, la forêt aléatoire atteignant la meilleure précision lors du test de généralisation. L'étude souligne l'importance des tests de généralisation pour un diagnostic de défaut fiable.

Mots clés : *Diagnostic des défauts, Apprentissage automatique, Systèmes CCHT, Généralisation, Forêt aléatoire, Simulation.*

Acknowledgements

Praise be to the Almighty God who has given us faith, courage, and patience to carry out this work.

We would like to extend our sincere appreciation to those who supported us during this research.

We want to express our gratitude to our supervisor Dr. Medoukali Hemza from Djelfa university, for the confidence he has placed in us, through his presence always with us, by his direction, his modesty, his advice, and constructive remarks for the good progress of this work.

We would like to thank everyone who helped us to improve our work. and who gave us any remark that helped us to perfect this manuscript.

We express our deep gratitude to our parents, our brothers, our sisters, and our whole families for their encouragement and prayers that allowed us to achieve this modest job. We am very grateful for the confidence they have placed in us.

Finally, we express our gratitude to all those who have contributed in one way or another to the development of this work.

O Allah, send your blessings on your noble messenger, his family, and companions, and bless us in our life.

Contents

List of Figures

List of Tables

List of Abbreviations

AC:	Alternating Current
ACO:	Ant Colony Optimization
AEs:	Autoencoders
ANFIS:	Adaptive Neuro-Fuzzy Inference System
ANNs:	Artificial Neural Networks
CNN:	Convolutional Neural Network
CNN-LSTM:	Convolutional Neural Network - Long Short-Term Memory
CSV:	Comma-Separated Values
DC:	Direct Current
DFRs:	Digital Fault Recorders
DL:	Deep Learning
DTs:	Decision Trees
EMD:	Empirical Mode Decomposition
EMI:	Electromagnetic Interference
EMT:	Electromagnetic Transient
FDD:	Fault Detection and Diagnosis
FFT:	Fast Fourier Transform
FN:	False Negative
FP:	False Positive
GBM:	Gradient Boosting Machines
GRU:	Gated Recurrent Unit
GWO:	Grey Wolf Optimization
HHT:	Hilbert-Huang Transform
HVDC:	High Voltage Direct Current
IGBT(s):	Insulated Gate Bipolar Transistor(s)
KNN:	K-Nearest Neighbors
L-G:	Line-to-Ground (fault type)
L-L:	Line-to-Line (fault type)
LCC:	Line Commutated Converter
LSTM:	Long Short-Term Memory

MATLAB:	Matrix Laboratory
ML:	Machine Learning
MLP(s):	Multi-Layer Perceptron(s)
MMC(s):	Modular Multilevel Converter(s)
MTDC:	Multi-Terminal Direct Current
NaN:	Not-a-Number
OvR:	One-vs-Rest
P-G:	Pole-to-Ground (fault type)
PCA:	Principal Component Analysis
PMUs:	Phasor Measurement Units
PSCAD/EMTDC:	Power Systems Computer Aided Design / Electromagnetic Transients including DC
RBF:	Radial Basis Function
ReLU:	Rectified Linear Unit
RF:	Random Forest
RMS:	Root Mean Square
RNNs:	Recurrent Neural Networks
STFT:	Short-Time Fourier Transform
SVC:	Support Vector Classifier (a type of SVM)
SVM(s):	Support Vector Machine(s)
TN:	True Negative
TP:	True Positive
VSC:	Voltage Source Converter
WT:	Wavelet Transform

General Introduction

High voltage direct current (HVDC) systems have emerged as a cornerstone of modern power transmission, offering significant advantages over traditional alternating current (AC) systems for long-distance and inter-grid connectivity. HVDC systems enable efficient power transfer with reduced transmission losses, facilitate the integration of renewable energy sources, and allow asynchronous grid interconnections [1]. These benefits have driven the global adoption of HVDC technology, particularly in applications such as offshore wind farms, cross-border power exchanges, and large-scale renewable energy integration [2]. However, the complexity and high power ratings of HVDC systems make them susceptible to faults, which can lead to severe consequences, including system outages, equipment damage, and economic losses [3].

Faults in HVDC systems, such as DC line-to-ground, phase-to-phase, and phase-to-ground faults, arise from various causes, including insulation failures, lightning strikes, and equipment malfunctions [4]. Detecting and diagnosing these faults rapidly and accurately is critical to maintaining system reliability and preventing cascading failures. Traditional fault detection methods, such as overcurrent relays and differential protection, often struggle with the dynamic and nonlinear behavior of HVDC systems, leading to delayed responses or false positives [5]. The limitations of these conventional approaches have spurred interest in advanced techniques, particularly machine learning (ML), which offers the potential to model complex patterns in fault data and achieve high diagnostic accuracy [6].

Machine learning has shown promise in power system applications, including fault detection, due to its ability to learn from large datasets and generalize to unseen conditions [7]. In HVDC systems, ML can leverage simulated or real-world data to identify fault signatures, classify fault types, and even predict system behavior under normal and faulty conditions. The motivation for this study stems from the need to enhance fault diagnosis in HVDC systems using ML, addressing the shortcomings of traditional methods and exploring innovative approaches to improve model generalization. Unlike many existing studies that randomly split fault data for training and testing, this research recognizes that the probability of a fault recurring with the same resistance is low in real-world scenarios. Therefore, evaluating ML models on unseen fault resistances is essential to ensure robust performance in practical applications.

The significance of this study lies in its comprehensive approach to fault diagnosis, combining simulation techniques with a diverse set of ML models to benchmark performance. By open-sourcing the MATLAB/Simulink simulation scripts and models on GitHub, this work aims to foster collaboration and enable other researchers to build upon or improve the proposed methods. Additionally, the inclusion of a non-fault case in the dataset allows for the characterization

of normal system behavior, laying the groundwork for future real-time monitoring capabilities. These contributions position this study as a valuable resource for advancing HVDC system reliability and supporting the global transition to sustainable energy systems.

Fault detection in HVDC systems is a challenging task due to the diversity of fault types, the variability of fault parameters (e.g., resistance, location, and duration), and the nonlinear dynamics of the system [3]. Conventional fault detection methods often rely on predefined thresholds or manual feature engineering, which may not capture the full complexity of fault signatures [8]. Moreover, the random splitting of fault data for training and testing, as commonly practiced in prior studies, does not adequately address the real-world scenario where faults with specific resistances are unlikely to repeat [9]. This raises questions about the generalization ability of ML models when applied to unseen fault conditions.

The problem addressed in this study is the development of an effective fault diagnosis framework for HVDC systems using ML, with a focus on evaluating model performance under realistic conditions. Specifically, the study aims to simulate a variety of fault scenarios using MATLAB/Simulink, generate comprehensive fault and non-fault datasets, and train multiple ML models to diagnose faults accurately. A key challenge is to assess whether these models can generalize to fault resistances not seen during training, mimicking the unpredictability of real-world faults.

The primary objectives of this study are as follows:

- Develop a detailed MATLAB/Simulink model of an HVDC system and simulate various fault scenarios, including DC line-to-ground, phase-to-phase, and phase-to-ground faults, as well as non-fault conditions.
- Generate a comprehensive dataset using custom MATLAB scripts and the `parsim` function for parallel simulations, capturing voltage and current signals for ML training.
- Train and benchmark multiple ML models—logistic regression, support vector machine (SVM), k-nearest neighbors (KNN), decision tree, random forest, gradient boosting, and neural network—using the scikit-learn library and grid search for hyperparameter optimization.
- Evaluate the generalization performance of these models by testing on unseen fault resistances, in addition to standard train-test splits, to reflect real-world fault variability.
- Open-source the simulation scripts and Simulink models, enabling contributions and further development.

- Analyze the results and recommend the best-performing model for HVDC fault diagnosis, with insights into future applications like real-time monitoring.

Industrial maintenance focuses on ensuring the reliability, safety, and efficiency of complex industrial systems. In the context of HVDC transmission systems, early fault detection plays a crucial role in preventive and predictive maintenance strategies. This research directly contributes to industrial maintenance by providing a machine learning-based framework for accurate and timely fault diagnosis. By enabling maintenance teams to identify potential issues before they escalate into major failures, the proposed approach helps to reduce downtime, optimize repair schedules, and extend the lifespan of critical components. Thus, the outcomes of this study align closely with the core objectives of industrial maintenance, supporting data-driven and proactive decision-making for improved system reliability.

The scope of this study encompasses the simulation, data generation, preprocessing, and ML-based fault diagnosis for a two-terminal HVDC system modeled in MATLAB/Simulink. The fault types considered include DC line-to-ground, phase-to-phase (in the AC section), and phase-to-ground (in the AC section) faults, with a non-fault case included to characterize normal behavior. The ML models are implemented using the scikit-learn library, and their performance is evaluated using standard metrics (accuracy, precision, recall, F1-score) and generalization tests on unseen resistances.

The main limitations of this study are:

- The study does not involve real-world HVDC system data; findings are based on simulation results.
- The study relies on simulated data rather than real-world measurements, which may not fully capture environmental factors like temperature or aging effects.
- The inclusion of non-fault data is a step toward future real-time monitoring, but this application is not implemented within the current study.

This thesis is organized as follows:

- Chapter 1: reviews HVDC systems, fault types, and fault detection methods.
- Chapter 2: describes HVDC modeling in MATLAB/Simulink and fault simulation.
- Chapter 3: explains scenario generation, dataset creation, and data preprocessing.

- Chapter 4: presents machine learning models, experiments, and results analysis.

We conclude this thesis with Conclusion and future work, summarizing the key findings of our study and providing suggestions for future research directions to enhance HVDC fault diagnosis and system reliability.

Chapter 1

Bibliography and Literature Review

1.1 Introduction

This chapter reviews the existing literature pertinent to fault diagnosis in High Voltage Direct Current (HVDC) systems, emphasizing Machine Learning (ML) applications. It aims to survey the current state of the art, identify key advancements, and analyze limitations and gaps in existing research to contextualize this thesis.

The review covers HVDC technology fundamentals, common fault types and their characteristics, and various diagnostic methods, from traditional approaches to contemporary ML techniques. A significant focus is placed on ML in HVDC fault detection, including algorithm surveys, feature engineering, and dataset utilization in prior studies. Finally, the chapter synthesizes these findings to highlight research gaps, particularly concerning model generalization and open-source contributions.

1.2 HVDC Technology and Applications

High Voltage Direct Current (HVDC) transmission systems have become integral to modern power grids, offering efficient and reliable solutions for long-distance and high-capacity power transfer. Among the various HVDC technologies, thyristor-based systems, also known as Line Commutated Converter (LCC) HVDC, have been widely adopted due to their robustness and suitability for bulk power transmission [5].

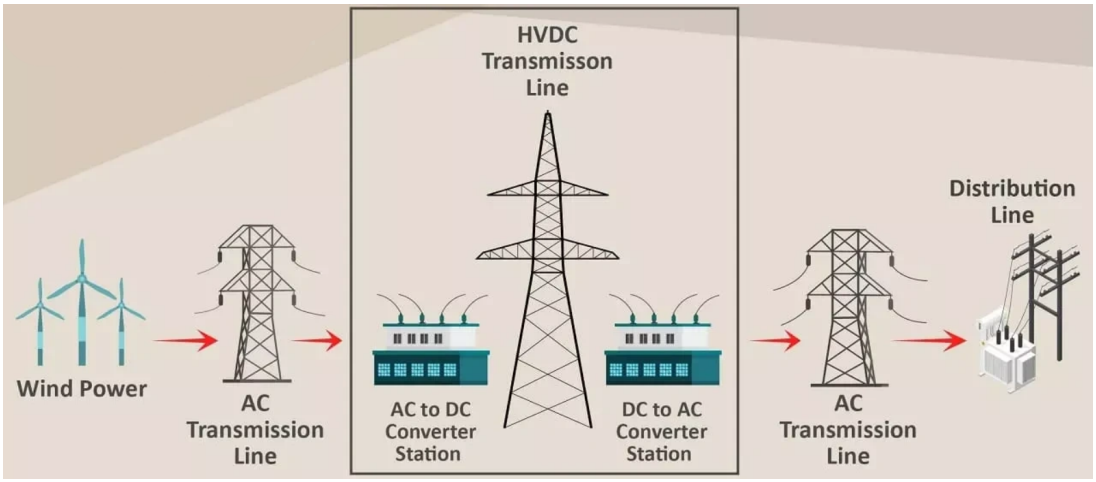

Figure 1.1: HVDC system schematic representation [10].

1.2.1 Evolution and Key Milestones

The concept of transmitting electricity using direct current dates back to the late 19th century. However, practical HVDC systems emerged in the mid-20th century, with the first commercial HVDC link established between Sweden and the island of Gotland in 1954. This pioneering project utilized mercury-arc valves for conversion between AC and DC, marking a significant milestone in power transmission technology [8].

Figure 1.2: The first commercial HVDC link: Gotland, Sweden (1954) [11].

Subsequent advancements led to the development of thyristor-based Line Commutated Converters (LCC) in the 1970s, enhancing the efficiency and capacity of HVDC systems. The introduction of Voltage Source Converters (VSC) in the 1990s further revolutionized HVDC

technology, enabling more flexible and compact converter stations suitable for a wider range of applications [12].

1.2.2 Fundamental Principles of HVDC Systems

HVDC transmission involves converting alternating current (AC) to direct current (DC) at the sending end, transmitting the DC power over long distances, and then converting it back to AC at the receiving end. This process is facilitated by converter stations equipped with rectifiers and inverters. The primary components of an HVDC system include:

- Converter stations: utilize power electronic devices (e.g., thyristors or IGBTs) to perform AC/DC and DC/AC conversions.
- Transmission medium: comprises overhead lines or submarine cables designed for high-voltage DC transmission.
- Control systems: manage the operation of converters, ensuring stability and efficient power flow.

The choice between LCC and VSC technologies depends on factors such as system requirements, cost, and the nature of the connected networks. LCC-HVDC systems are typically employed for bulk power transmission over long distances, while VSC-HVDC systems are favored for their ability to connect weak or isolated grids and for applications requiring rapid control of power flow [13].

HVDC systems can also be configured in various ways, including monopolar, bipolar, homopolar, back-to-back, and multi-terminal systems (MTDC), each suited to specific application needs and offering different levels of reliability and cost [3].

1.2.3 Advantages and Disadvantages of HVDC Transmission

High Voltage Direct Current (HVDC) transmission systems offer several advantages over traditional Alternating Current (AC) systems, particularly for specific applications. However, they also present certain challenges and limitations.

A - Advantages of HVDC:

- HVDC is more efficient for long-distance transmission due to lower resistive (I^2R) losses and the absence of reactive power, which significantly reduces transmission losses over hundreds of kilometers. This makes HVDC especially beneficial for remote generation sources.
- Reduced transmission losses: HVDC transmission lines experience lower electrical losses than AC lines, especially over extended distances. this reduction in losses contributes to improved overall system efficiency [14].
- Asynchronous interconnection of grids: HVDC allows for the interconnection of power systems operating at different frequencies or without synchronized phases. this capability facilitates energy exchange between regions with incompatible AC systems [2].
- Enhanced control and stability: HVDC systems provide precise control over power flow, which enhances the stability of interconnected power networks. this controllability is particularly beneficial in mitigating the effects of disturbances and preventing cascading failures [8].
- Integration of renewable energy sources: HVDC systems are ideally suited for integrating remote renewable sources like offshore wind and desert solar farms. they support bulk power transfer with minimal loss and help overcome the geographical mismatch between generation and load centers [1, 6].
- Lower right-of-way requirements: compared to AC transmission, HVDC lines require narrower right-of-way and smaller tower sizes, which is advantageous in urban or environmentally protected areas [8].
- Reduced electromagnetic interference (EMI): HVDC lines produce minimal EMI due to constant voltage and current, which is beneficial for minimizing disturbances to nearby communication infrastructure.
- Simplified cable design for submarine and underground applications: unlike AC systems, which suffer from capacitive charging currents in long cables, HVDC cables can be laid over much longer distances with reduced losses, making them ideal for submarine or underground installations (e.g., inter-island or cross-border links) [3].

B - Disadvantages of HVDC:

- High initial investment and converter costs: The major drawback of HVDC is the high capital cost of converter stations, which require complex power electronic devices like thyristor or IGBT-based valves, transformers, and filters [3].

- Converter station complexity: Converter stations (especially Line-Commutated Converters or LCCs, used in thyristor-based systems) require intricate control systems, harmonic filters, and reactive power support equipment, making their design and operation complex [12].
- Limited short-term overload capability: HVDC converters often have limited overload capacity compared to AC transformers or lines. This constraint can affect emergency operation scenarios [15].
- DC circuit breaker challenges: DC current lacks natural zero crossings, which makes interrupting faults more difficult. Specialized and often expensive DC circuit breakers are required, particularly in meshed or multi-terminal HVDC grids [12].
- Harmonics and filtering requirements: LCC-based HVDC systems generate harmonics that must be filtered out to protect equipment and ensure power quality. This adds to both the cost and footprint of converter stations.
- Lower operational flexibility in voltage transformation: Unlike AC systems, where voltage levels can be easily modified with transformers, changing DC voltage levels requires complex DC-DC converters, which are not yet widely implemented in high-power applications.
- Skilled maintenance and operational requirements: HVDC systems demand a specialized workforce for design, commissioning, and maintenance, especially due to the use of advanced digital control and protection systems.
- Integration with existing AC infrastructure: Integrating HVDC into an AC-dominated grid must be carefully engineered to avoid operational issues like sub-synchronous resonance, harmonic interactions, or inadequate reactive power support [14].

1.3 Types of Faults in HVDC Systems

High voltage direct current (HVDC) transmission systems are pivotal for long-distance and high capacity power transfer. However, their reliable operation is susceptible to various faults that can compromise system stability, damage equipment, and interrupt power supply. Understanding these faults is essential for developing effective detection and protection strategies, especially in LCC-HVDC systems, which are widely used due to their efficiency and cost-effectiveness.

1.3.1 Classification of HVDC faults

HVDC transmission systems can experience various types of faults, impacting performance and reliability. The following figure categorizes these fault types:

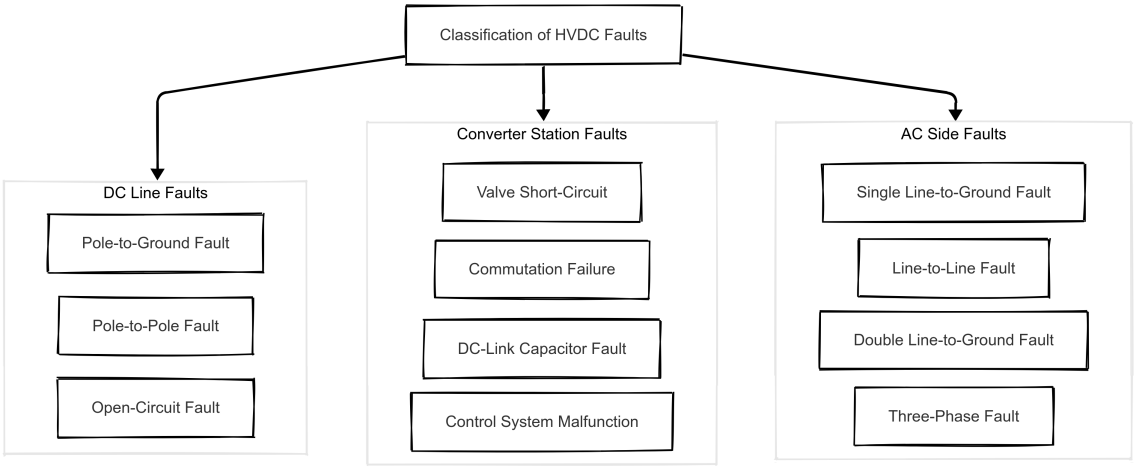

Figure 1.3: Classification of faults in HVDC transmission systems.

Faults in HVDC systems can be broadly categorized based on their location and nature:

- DC line faults: these occur along the HVDC transmission lines and are primarily due to external factors. Common types include:
  - Pole-to-ground faults: caused by insulation failure or external interference, leading to a direct connection between one pole and the ground.
  - Pole-to-pole faults: result from simultaneous faults on both positive and negative poles, often due to severe insulation breakdown.
  - Open-circuit faults: occur when a conductor is physically broken or disconnected, interrupting the current flow.
- Converter station faults: these internal faults within converter stations can arise from:
  - Valve short circuits: due to semiconductor device failures (e.g., thyristors in LCC or IGBTs in VSC), leading to uncontrolled current paths.
  - Commutation failures: specific to LCC systems, inadequate commutation margin can cause overlap between incoming and outgoing valves, disrupting current transfer [15].
  - DC-link capacitor faults: in VSC systems, failures in the DC-link capacitors can lead to significant operational issues.

  - Submodule faults: in Modular Multilevel Converters (MMCs), a common VSC topology, faults within individual submodules (e.g., IGBT open-circuit or short-circuit) are critical [12, 16].
  - Control system malfunctions: errors in control logic or hardware can lead to improper firing of valves or incorrect converter operation.

- AC side faults: while primarily affecting the AC network, these faults can influence HVDC operation, especially in LCC systems where commutation depends on the AC voltage stability. these can also impact VSC systems by affecting power exchange capabilities.

In LCC-HVDC systems, DC line faults are particularly critical due to the rapid rise in fault current and the absence of natural current zero crossings, which complicates fault interruption [5].

1.3.2 Characteristics of HVDC faults

HVDC faults exhibit distinct characteristics that differentiate them from AC system faults:

- Rapid fault current rise: the low inductance in HVDC lines allows fault currents to escalate rapidly, often reaching several kiloamperes within milliseconds [17].
- Absence of natural current zero: unlike AC systems, HVDC lacks natural current zero crossings, making it challenging to interrupt fault currents using conventional circuit breakers [18, 19].
- Voltage collapse: faults can lead to a sudden drop in DC voltage, potentially causing system instability and equipment damage [16].
- Commutation failures: in LCC systems, faults (both DC and AC side) can disrupt the commutation process, leading to misfiring of valves and further exacerbating system instability [15].
- Overvoltages: certain fault types or clearing actions can lead to significant overvoltages in the HVDC system, stressing insulation and components [15].

1.4 Fault Diagnosis Methods in HVDC Systems

High voltage direct current (HVDC) systems are pivotal in modern power transmission, offering efficient long-distance electricity transfer. However, their unique characteristics, such as low system inertia and rapid fault propagation, necessitate advanced fault diagnosis methods. These methods integrate principles from control theory, signal processing, artificial intelligence, and statistical analysis to ensure system reliability and safety.

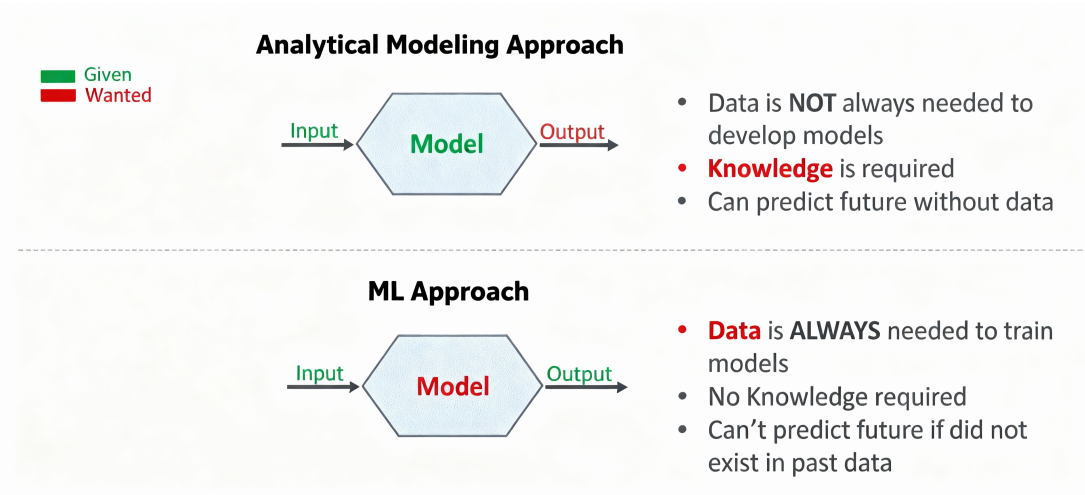

Figure 1.4: Model based Analytics vs. data based ML.

1.4.1 Analytical Model-Based Methods

Analytical model-based approaches involve creating mathematical representations of HVDC systems to detect deviations indicative of faults. These methods are categorized into:

- State estimation: this technique estimates the system's internal states using observers or filters. Discrepancies between estimated and actual measurements generate residuals, which, when analyzed statistically, can indicate faults. The accuracy of this method hinges on the observability of the system.
- Parameter estimation: faults often alter system parameters (e.g., line resistance, inductance). By estimating these parameters in real-time and monitoring their variations from nominal values, faults can be detected and localized. This method is particularly effective when direct measurement of certain parameters is challenging [3].
- Parity space method: this approach compares actual system outputs with those predicted by the model. Significant deviations, or residuals, suggest the presence of faults. The

method is advantageous due to its ability to handle incomplete or uncertain information.

1.4.2 Signal Processing-Based Methods

Signal processing techniques analyze electrical signals (voltages and currents) to identify anomalies indicative of faults. Key methods include:

- Wavelet transform (WT): WT decomposes signals into time-frequency components, making it adept at detecting transient events in non-stationary signals. It is particularly useful for identifying fault-induced transients (e.g., high-frequency components) in HVDC systems and is often used for feature extraction [5, 17].
- Hilbert–Huang transform (HHT): HHT combines empirical mode decomposition (EMD) with Hilbert spectral analysis to extract instantaneous frequency data from non-linear and non-stationary signals. This method is effective in capturing the dynamic behavior of HVDC systems during faults and can adaptively analyze fault signatures.
- Natural frequency-based methods: these techniques analyze the natural frequencies generated during faults, which are influenced by system parameters (e.g., line length, grounding resistance) and fault locations. By examining these frequencies, faults can be detected and localized without necessarily tracking traveling wavefronts [20].
- S-Transform: the S-Transform provides a time-frequency representation with frequency-dependent resolution, combining features of WT and Short-Time Fourier Transform (STFT). It has been applied to extract features from non-stationary fault signals in HVDC systems for subsequent classification [21].

1.4.3 Pattern Recognition and Machine Learning Methods

Pattern recognition and machine learning (ML) methods learn to classify system states (normal or faulty) or identify fault types based on features extracted from operational data. These methods encompass:

- Artificial neural networks (ANNs): ANNs learn complex patterns from data, making them suitable for fault detection and classification in HVDC systems. They can handle non-linear relationships and adapt to varying system conditions [3, 22].

- Support vector machines (SVMs): SVMs classify data by finding the optimal hyperplane that separates different fault conditions. They are effective in scenarios with limited data and high-dimensional feature spaces [6].
- K-nearest neighbors (KNN): KNN classifies faults based on the majority class among the 'k' closest data points in the feature space. It is a simple yet effective non-parametric method, especially when dealing with small and well-characterized datasets [23, 24].
- Hybrid models: combining different ML algorithms or integrating ML with signal processing techniques (e.g., WT-ANN) can enhance fault diagnosis accuracy by capturing diverse data characteristics, such as both spatial and temporal features from CNN-LSTM models [23].

1.4.4 Hybrid and Advanced Methods

Advanced fault diagnosis approaches integrate multiple methodologies to leverage their respective strengths and overcome individual limitations:

- Hybrid protection algorithms: these algorithms combine various detection techniques, sometimes dynamically selecting the most appropriate one based on system conditions or confidence levels. For instance, a hybrid primary protection algorithm for multi-terminal HVDC systems might utilize context clustering to choose the optimal fault detector from a pool, enhancing robustness against diverse fault scenarios [3, 24].
- Bayesian regression models: bayesian approaches incorporate prior knowledge (e.g., about fault probabilities) and observed data to predict fault locations or classify fault types. A Bayesian Ridge Regression model has been proposed to estimate fault locations in multi-terminal HVDC networks using single-ended measurements, demonstrating resilience to measurement noise and varying fault conditions [25].
- Transient-based protection: these methods analyze high-frequency components of voltage and current signals generated during transients to detect faults rapidly. By examining the differences in transient energy, frequency content, or arrival times at various points in the system, faults can be accurately identified and localized [3, 5]. These often serve as inputs to ML classifiers.
- Fuzzy logic systems: fuzzy logic can handle uncertainties and imprecise information inherent in fault diagnosis. It uses linguistic variables and fuzzy rules derived from expert knowledge or data to make decisions [26]. Fuzzy logic is often combined with other

techniques, like ANNs (Adaptive Neuro-Fuzzy Inference Systems - ANFIS) or WT, to improve diagnostic performance [27].

1.5 Machine Learning in Power System Fault Detection (Focus on HVDC)

Machine Learning (ML) has emerged as a powerful paradigm for enhancing fault detection and diagnosis (FDD) in power systems, particularly in complex High Voltage Direct Current (HVDC) environments [3,28]. The ability of ML algorithms to learn intricate patterns from vast amounts of operational data, without necessarily requiring an exact mathematical model of the system, makes them well-suited for the non-linear and dynamic nature of HVDC systems and their associated faults [8]. This section provides a survey of ML algorithms applied to HVDC fault detection, discusses input feature considerations, contrasts data-driven with model-driven approaches, critically examines dataset handling in previous studies, and highlights research addressing key operational challenges.

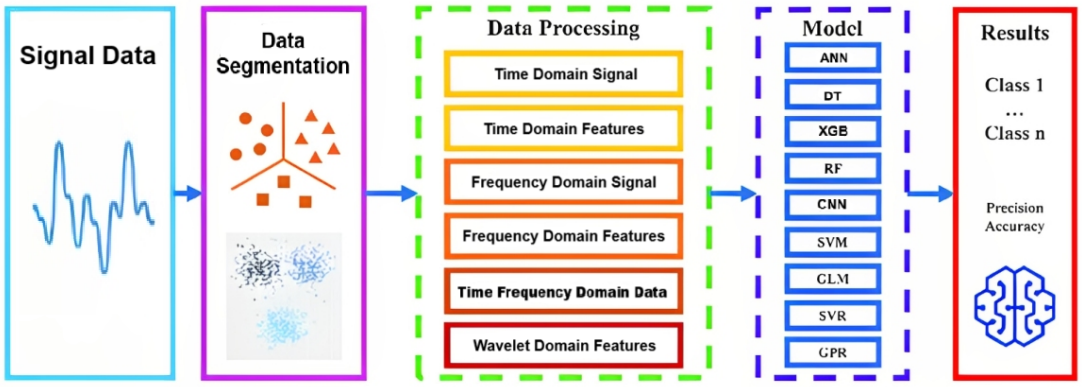

Figure 1.5: General workflow for ML model building.

1.5.1 Survey of Machine Learning Algorithms

A variety of ML algorithms have been investigated for HVDC fault detection, each with its own strengths and weaknesses.

- **Artificial neural networks (ANNs) and deep learning (DL):** ANNs, inspired by biological neural systems, are widely used due to their capability to model complex non-linear relationships. Multi-Layer Perceptrons (MLPs) are common foundational architectures [3, 22]. Deep Learning, a subfield of ML involving ANNs with multiple hidden layers (deep architectures), has shown significant promise.

  - *Convolutional neural networks (CNNs)* are adept at automatically extracting hierarchical features from grid-like data, such as time-series signals (1D CNNs) or time-frequency representations (2D CNNs), making them effective for analyzing raw current and voltage waveforms [23, 29].
  - *Recurrent neural networks (RNNs)*, particularly Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) variants, are designed to handle sequential data and capture temporal dependencies, crucial for understanding fault evolution over time [23, 30].
  - *Autoencoders (AEs)*, a type of unsupervised neural network, can be used for anomaly detection by learning to reconstruct normal operational data; high reconstruction errors may indicate faults. Deep AEs are also used for feature learning [31].

  The combination of metaheuristic optimization algorithms like Ant Colony Optimization (ACO) or Grey Wolf Optimization (GWO) with ANNs has also been explored to enhance feature selection and network training for HVDC fault detection [32, 33].

- **Support vector machines (SVMs):** SVMs are powerful supervised learners effective for classification tasks, especially with high-dimensional feature spaces and potentially small datasets [6]. They work by finding an optimal hyperplane that maximizes the margin between different classes (e.g., fault types). The choice of kernel function (e.g., linear, polynomial, Radial Basis Function - RBF) is critical for SVM performance. SVMs have been used for identifying, classifying, and locating faults in MT-HVDC systems.

- **Decision trees (DTs) and ensemble methods:**

  - *Decision trees* create a tree-like model of decisions. They are relatively easy to interpret but can be prone to overfitting.
  - *Ensemble methods* combine multiple base learners to improve prediction accuracy and robustness. *Random forests (RF)* build multiple decision trees on different subsets of data and features, averaging their predictions. RFs are robust to overfitting and can handle high-dimensional data. *Gradient boosting machines (GBM)*, such as XGBoost and LightGBM, build trees sequentially, where each new tree corrects the errors of the previous ones, often achieving state-of-the-art results [34]. Ensemble methods like Bagged Trees have shown high accuracy in HVDC fault diagnosis.

- **K-nearest neighbors (KNN):** KNN is a non-parametric, instance-based learning algorithm. It classifies a new data point based on the majority class of its 'k' nearest neighbors in the feature space [23, 24]. While simple and often effective for well-defined datasets, its performance can degrade with high-dimensional data and large datasets due to computational costs and the "curse of dimensionality."

- **Other ensemble and hybrid approaches:** researchers frequently combine different ML models (heterogeneous ensembles) or integrate ML with other techniques like signal processing (e.g., Wavelet-ANN, S-Transform-SVM) or fuzzy logic to leverage their complementary strengths and achieve superior diagnostic performance [35].

1.5.2 Input Features for Machine Learning Models

The performance of ML models heavily relies on the quality and relevance of input features. Common feature types include:

- **Raw signals:** direct use of sampled voltage and current waveforms, often suitable for deep learning models like CNNs and LSTMs that can learn features automatically [23].
- **Statistical features:** calculated from segments of raw signals, such as mean, variance, Root Mean Square (RMS), skewness, kurtosis, peak values, energy, and standard deviation [6, 35].
- **Time-domain features:** features extracted directly from the time-domain signals, including peak-to-peak values, crest factor, and shape factor.
- **Frequency-domain features:** derived using techniques like Fast Fourier Transform (FFT), providing information about the spectral content of signals, such as harmonic magnitudes and total harmonic distortion.
- **Time-frequency domain features:** extracted using methods like Wavelet Transform (WT), Hilbert-Huang Transform (HHT), or S-Transform, which provide simultaneous time and frequency information. Coefficients from WT (e.g., detail and approximation coefficients) are commonly used as inputs to ML classifiers [5, 21].
- **Derived quantities:** features engineered based on domain knowledge, such as rates of change of voltage/current (di/dt, dv/dt), differential currents/voltages, sequence components (in AC-side faults affecting HVDC), or correlation coefficients between signals.

Feature selection and dimensionality reduction techniques (e.g., Principal Component Analysis (PCA) [6, 36]) are often employed to select the most informative features, reduce model complexity, and improve generalization.

1.5.3 Data-Driven vs. Model-Driven Approaches

Fault diagnosis methodologies can be broadly categorized as model-driven or data-driven, with a growing trend towards hybrid approaches.

- **Model-driven approaches:** as discussed in Section 1.4 (Analytical Model-Based Methods), these rely on an accurate mathematical model of the HVDC system. They offer interpretability and can perform well if the model is precise. However, developing such models can be complex, and their accuracy may degrade with system uncertainties or unmodeled dynamics.
- **Data-driven approaches:** these primarily use historical or simulated data to train ML models that learn the mapping between input measurements and fault status/type [3]. Their main advantage is the ability to handle complex systems without requiring an explicit physical model. However, they depend heavily on the availability of large, diverse, and high-quality datasets, and some ML models (especially deep learning) can act as "black boxes," lacking transparency.
- **Hybrid approaches:** these aim to combine the strengths of both. For instance, a physical model might be used to generate synthetic data for training ML models, or ML techniques could be used to estimate parameters within a physical model or to process residuals from model-based observers. Knowledge graphs, which structure system knowledge, are also being explored with ML algorithms like KNN for HVDC fault diagnosis.

1.5.4 Critical Examination of Datasets in Previous Studies

The reliability and generalizability of ML-based FDD heavily depend on the datasets used for training and testing.

- **Data sources:**
  - *Simulation data:* the majority of studies utilize data generated from simulation software like PSCAD/EMTDC or MATLAB/Simulink [6, 8, 30, 34]. This allows for extensive fault scenarios with varying types, locations, and parameters. However, the fidelity of simulated data to real-world conditions is always a concern.
  - *Real-world data:* data from actual HVDC installations (e.g., from Phasor Measurement Units (PMUs) or Digital Fault Recorders (DFRs)) is the most desirable but

is often scarce, proprietary, imbalanced (few fault instances compared to normal operation), and may lack detailed labels for fault characteristics [23].

- **Dataset size and diversity:** many studies rely on datasets of limited size or diversity, which may not adequately represent the wide spectrum of possible fault types, fault resistances, locations, varying operational conditions (e.g., power flow levels, system topology changes), and noise levels. This can lead to models that perform well on specific test sets but generalize poorly to unseen conditions.
- **Data splitting methodologies:** as highlighted in Section 1.6, a common practice is random splitting of data into training, validation, and testing sets. While straightforward, this approach may not adequately assess a model's ability to generalize to truly novel fault scenarios (e.g., fault resistances or locations not encountered during training). More rigorous generalization testing, such as splitting based on specific fault parameter ranges or different operational scenarios, is crucial but less frequently reported.

1.6 Gaps and Research Opportunities

The application of machine learning (ML) to fault detection in High Voltage Direct Current (HVDC) systems, particularly Thyristor-Based Line-Commutated Converter (LCC-HVDC) systems, has demonstrated considerable potential to improve system reliability, reduce downtime, and enhance operational efficiency. However, a comprehensive review of the existing literature reveals several critical shortcomings that impede the practical deployment and scalability of ML-based fault diagnosis techniques. These gaps include the lack of comprehensive datasets covering diverse fault types and parameters, insufficient investigation into model generalization beyond simple random data splits, limited benchmarking comparing a wide range of ML models on the same dataset, and the need for open-source tools and datasets to facilitate reproducibility and further research. This section elaborates on these shortcomings, clearly articulates how this study addresses them in alignment with the objectives outlined in general introduction, and identifies opportunities for contribution to advance the field.

1.6.1 Shortcomings in Existing Literature

A - Insufficient Investigation into Model Generalization Beyond Simple Random Splits

A critical methodological gap in the literature is the reliance on random dataset splitting for training and testing ML models, which fails to adequately assess model generalization to

unseen fault conditions. Nearly all studies split their datasets randomly across fault resistances, locations, and other parameters, assuming that this approach sufficiently evaluates model performance. However, in real-world HVDC systems, faults are stochastic, and the likelihood of a fault recurring with the same resistance (e.g., 2Ω) or at the same location (e.g., 50% of the line length) is extremely low. For example, a model trained on faults with resistances between 0.1Ω and 5Ω may perform well on test data within this range but fail when encountering a fault with a resistance of 8Ω or at a different line position (e.g., 10% of the line). This raises significant concerns about the true generalization capability of existing models, as they are rarely tested on truly novel fault scenarios that differ substantially from the training data. The lack of rigorous generalization testing limits the reliability of ML models in operational settings, where adaptability to new fault conditions is essential for ensuring system stability and preventing outages. This gap is particularly pronounced in LCC-HVDC systems, where fault dynamics are influenced by complex interactions between DC and AC components, necessitating models that can handle variability in fault parameters [23, 37, 38].

B - Need for Open-Source Tools and Datasets to Facilitate Reproducibility and Further Research

Reproducibility and collaboration are cornerstones of scientific progress, yet the field of HVDC fault detection using ML suffers from a significant lack of open-source simulation model, scripts, and datasets. Most studies utilize proprietary or custom-built simulation environments, such as MATLAB/Simulink models or PSCAD/EMTDC configurations, which are not shared with the broader research community. Similarly, fault datasets are rarely made publicly available, forcing researchers to independently generate their own data, often duplicating efforts and introducing inconsistencies in simulation parameters or fault scenarios. This lack of accessibility restricts the ability of other researchers to validate findings, replicate experiments, or build upon existing work, slowing the pace of innovation. The absence of open-source resources also limits collaborative opportunities, as researchers cannot easily contribute improvements or adapt existing tools to new fault detection challenges. Open-source tools and datasets are critical for fostering a collaborative research ecosystem, enabling standardized comparisons, and accelerating the development of reliable ML-based fault detection systems for HVDC applications.

1.6.2 Addressing the Gaps Through This Study

This study directly addresses the identified shortcomings by implementing a series of targeted contributions that align with the objectives outlined in general introduction, which include developing robust ML models for HVDC fault detection, ensuring generalization to unseen fault conditions, laying the groundwork for system monitoring, and promoting reproducibility through open-source resources. The following sections detail how each gap is addressed, linking back to the study's objectives.

A - Comprehensive Dataset Development

To address the lack of diverse datasets, MATLAB/Simulink is used to simulate various faults in a two-terminal LCC-HVDC system. The dataset includes DC line-to-ground, AC phase-to-phase/ground faults, with variations in resistance (0.01Ω, 0.1Ω, 1Ω, 10Ω, 50Ω, 100Ω, 500Ω), location (10%, 50%, 90%), duration (0.01s, 0.05s, 0.1s, 0.2s, 0.3s, 0.5s). No-fault scenarios are also included to support system monitoring.

B - Generalization Testing

To improve robustness, the dataset is split by excluding certain fault resistances and locations from training. Models trained on a subset (0.01Ω, 0.1Ω, 1Ω, 50Ω, 500Ω) are tested on unseen ranges (10Ω, 100Ω), simulating real-world unpredictability and ensuring models generalize to new conditions.

C - Model Benchmarking

Multiple ML models (Logistic Regression, SVM, KNN, Decision Trees, Random Forests, Gradient Boosting, and Neural Networks) are benchmarked using Scikit-learn with hyperparameter tuning. Metrics like accuracy, precision, and F1-score are used to evaluate model performance, addressing the lack of standardized comparisons in HVDC fault detection.

D - Open-Source Tools and Dataset

To promote reproducibility all MATLAB scripts and Simulink models are open-sourced on GitHub. These include fault scenario generation, automation tools, and documentation, allowing researchers to replicate or extend the work.

1.7 Conclusion

This literature review has surveyed HVDC technology, associated fault types, and the evolution of diagnostic methods, culminating in modern machine learning approaches. A key finding is the persistent need for more rigorous ML model generalization testing beyond standard random data splits. Additionally, there is a clear call for greater availability of open-source tools and datasets to enhance reproducibility and collaboration in the field.

These identified shortcomings have directly informed the objectives of this thesis. The subsequent chapters will focus on developing a comprehensive simulation framework, generating an open dataset, and evaluating ML models with a strong emphasis on their generalization capabilities, aiming to contribute to more reliable HVDC fault diagnosis solutions.

Chapter 2

HVDC System Modeling, Fault Characterization, and Simulation

2.1 Introduction

The accurate detection and classification of faults in high voltage direct current (HVDC) systems using machine learning techniques necessitates a thorough understanding of the system's operational principles, component characteristics, and the distinct signatures of various fault types. Furthermore, the generation of high-quality, representative data through simulation is paramount for training robust machine learning models. This chapter details the modeling of a benchmark 12-pulse HVDC transmission system, describes its principal components, outlines the simulation environment and parameters, and characterizes the various AC and DC side faults investigated in this thesis. The simulation model, developed in MATLAB/Simulink, serves as the foundation for generating the dataset used for training and evaluating the machine learning algorithms discussed in subsequent chapters.

2.2 HVDC System Components and Configuration

The HVDC system under consideration is a point-to-point, 12-pulse, two-level, thyristor-based line commutated converter (LCC) system, a widely adopted configuration for bulk power transmission. The key components of such a system, as represented in the simulation model (refer to figure 2.4), are described below.

2.2.1 Converters (Rectifier and Inverter)

Converters are the heart of high voltage direct current (HVDC) systems, playing a crucial role in converting alternating current (AC) to direct current (DC) at the sending end (rectification) and reversing the process at the receiving end (inversion). This conversion is essential for efficient long-distance power transmission, as DC reduces energy losses compared to AC. HVDC systems utilize two main types of converters: line-commutated converters (LCC), which are robust and suited for high-power applications, and voltage-source converters (VSC), which offer greater flexibility and are ideal for integrating renewable energy. Beyond enabling efficient power transmission, converters enhance grid stability by allowing precise control of power flow, making them indispensable in modern power networks [14, 39].

Figure 2.1: LCC valve hall with thyristor valves [40].

- **Operation principle:** the simulated system employs 12-pulse converters. A 12-pulse arrangement is achieved by using two three-phase, 6-pulse thyristor bridges connected in series on the DC side. The AC supply to these two bridges is phase-shifted by 30 degrees, typically achieved using two converter transformers with Yg-Y (wye-grounded secondary to wye) and Yg-D (wye-grounded secondary to delta) winding configurations [41].
- **Advantages of 12-pulse configuration:** this configuration significantly reduces harmonic distortion on both the AC and DC sides compared to a 6-pulse system. Specifically, it eliminates characteristic harmonics of orders $6k \pm 1$ (where k is an integer), leaving $12k \pm 1$ (i.e., 11th, 13th, 23rd, 25th, etc.) on the AC side and $12k$ (i.e., 12th, 24th, etc.) on the DC side [42].
- **Thyristor valves:** each 6-pulse bridge consists of six thyristor valves. These are high-

power semiconductor devices that can be controlled (turned on) by a gate pulse when forward biased. They turn off when the current through them falls below a holding current (line commutation).

- **In the simulation (figure 2.4):** the blocks labeled "Rectifier" and "Inverter" represent these 12-pulse thyristor bridge configurations. They are controlled by firing angle controllers (alpha for rectifier, gamma for inverter) to regulate DC voltage, current, and power flow. The diagram shows the standard bridge connections.

2.2.2 AC Filters and Reactive Power Source

LCC-HVDC converters inherently generate current harmonics on the AC side and consume reactive power.

Figure 2.2: AC and DC filter reactors in an HVDC system [43].

- **Harmonic filters:** these are passive L-C circuits tuned to specific harmonic frequencies (e.g., 11th, 13th, high-pass for higher orders) to shunt these harmonic currents, preventing them from flowing into the AC network and ensuring power quality [14].
- **Reactive power source:** converters typically consume reactive power equivalent to 40-60% of the active power transmitted. This reactive power demand is usually supplied by the AC filters themselves (which are capacitive at fundamental frequency), shunt capacitor banks, or synchronous condensers [14].
- **In the simulation (figure 2.4):** blocks labeled "AC filters" are present on the AC side of both the rectifier (60 Hz, 600 Mvar) and the inverter (50 Hz, 600 Mvar). These typically represent a combination of tuned harmonic filters and shunt capacitor banks.

2.2.3 DC Smoothing Reactors

Large inductors, known as smoothing reactors, are connected in series on the DC side, typically at each end of the DC line.

Figure 2.3: HVDC smoothing reactors [44].

- **Functions:** smoothing reactors serve several critical functions in HVDC systems. They smooth the DC current ripple generated by the converters, thereby reducing harmonic currents on the DC line. Additionally, they limit the rate of rise of DC fault current during DC line faults, providing protection systems with sufficient time to operate effectively. By mitigating current dips during transients, smoothing reactors also help prevent commutation failures in the inverter. Furthermore, they play a vital role in addressing resonance issues within the DC circuit, ensuring stable and reliable system operation [8].
- **In the simulation (figure 2.4):** smoothing reactors are represented by inductors (labeled 0.5 H in the diagram) on the DC side of both the rectifier and inverter, connected between the converter bridges and the DC line.

2.2.4 DC Transmission Line

The DC transmission line carries the bulk power between the converter stations.

- **Configuration (poles):** HVDC lines can be monopolar (one conductor, typically with ground or metallic return) or bipolar (two conductors, one positive and one negative polarity). The provided diagram suggests a bipolar setup as it shows positive and negative terminals (+ and -) and fault locations between these and ground.
- **Modeling:** for transient studies, DC lines are often modeled using distributed parameter models or cascaded pi-sections to accurately represent wave propagation phenomena. The diagram shows the DC line segmented into multiple sections (30 km, 45 km, 75 km, 75 km, 45 km, 30 km, totaling 300 km), allowing for faults to be applied at various points along its length.
- **In the simulation (figure 2.4):** the DC line is clearly depicted with segments. The "DC Fault", "DC Fault1", ..., "DC Fault4" blocks indicate locations where DC line-to-ground faults can be initiated.

2.2.5 AC Circuit Breakers

While not explicitly detailed as a separate HVDC component block in the core DC system, AC circuit breakers are essential on the AC side of the converter transformers.

- **Function:** they protect the HVDC system from AC side faults and are used to connect/disconnect the HVDC converter stations from the AC grid. Their operation is critical during fault conditions to isolate the faulted section [20].
- **In the simulation context:** the AC sources (5000 MVA and 10,000 MVA equivalents) implicitly include the upstream AC network with its protective devices. AC faults are simulated on the AC bus near the converters.

2.3 Simulation Model and Parameters

The study uses a detailed simulation of a 1000 MW high-voltage direct current (HVDC) system, designed to mirror real-world setups. Operating at ±500 kV with a 2 kA current, this 12-pulse LCC-HVDC model is built in MATLAB/Simulink using the Simscape Power Systems toolbox. It simplifies complex electrical behavior into a clear, accurate representation. This benchmark system helps researchers study how HVDC systems work without needing the full technical details. Its main purpose is to provide data for training machine learning tools to

detect and classify faults. This makes it a valuable tool for improving the reliability of power transmission in practical applications.

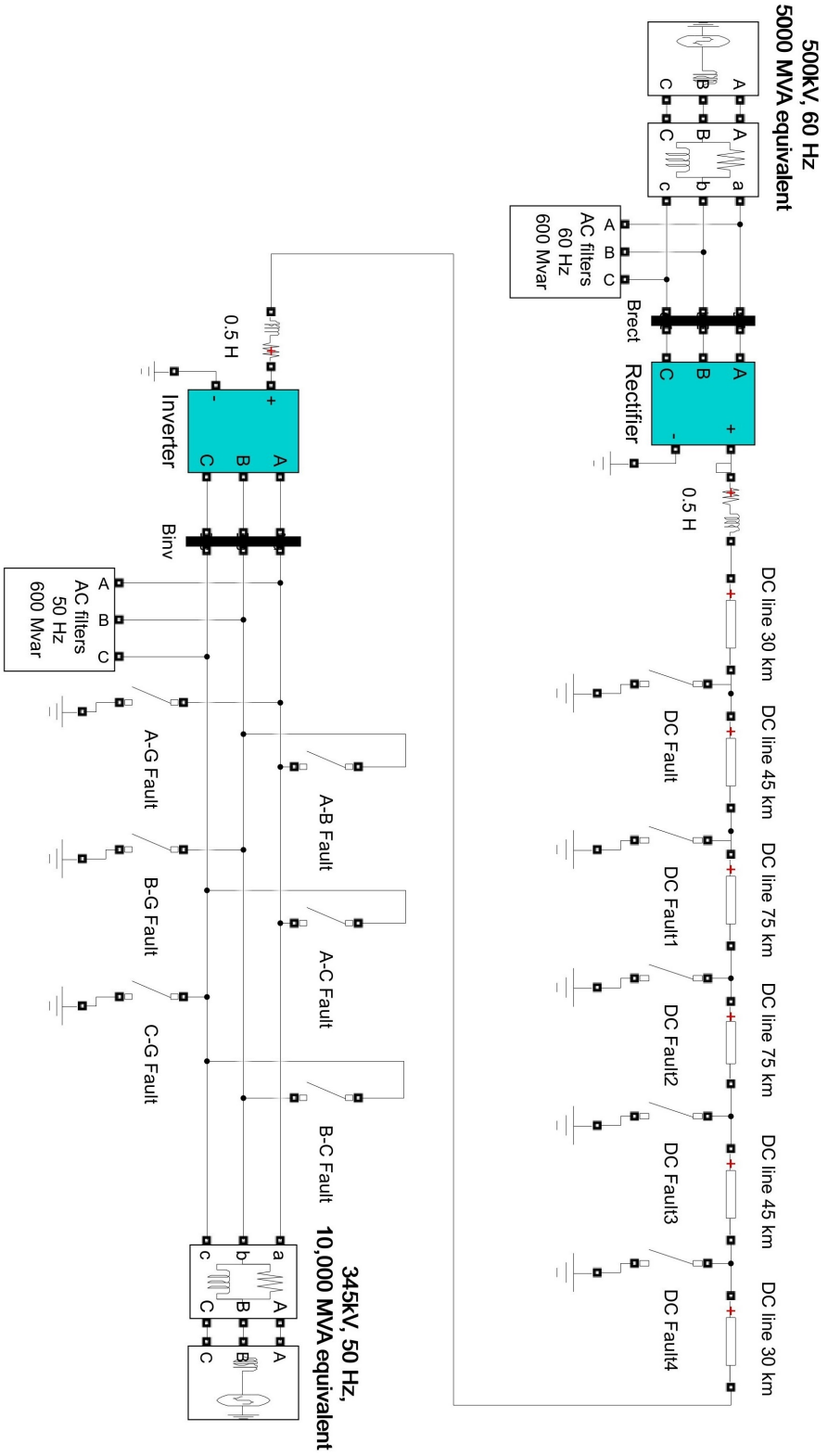

Figure 2.4: Overview of the MATLAB/Simulink model of the 12-pulse HVDC transmission system.

2.3.1 System Specifications

Table 2.1: Main specifications of the simulated HVDC system.

Parameter	**Value / Description**
Rated power	1000 MW
DC voltage	$\pm$500 kV (bipolar configuration assumed based on fault locations)
Rated DC current	2 kA
DC transmission line length	300 km
Rectifier side AC system	500 kV, 60 Hz. The AC source is modeled as a 5000 MVA equivalent three-phase source.
Inverter side AC system	345 kV, 50 Hz. The AC source is modeled as a 10,000 MVA equivalent three-phase source.
Converter configuration	Two 6-pulse thyristor bridges per pole, forming a 12-pulse converter.
Smoothing reactors	0.5 H at each end (rectifier and inverter side).
AC filters	600 Mvar (capacitive at fundamental frequency) on both AC sides.

The choice of different frequencies (60 Hz sending end, 50 Hz receiving end) and voltage levels for the connected AC systems is typical for asynchronous interconnections facilitated by HVDC.

2.3.2 Simulation Parameters

To capture both steady-state operation and transient fault behavior accurately, the following simulation parameters were adopted:

- **Simulation software:** MATLAB/Simulink with Simscape Power Systems.
- **Total simulation time:** 3.5 seconds. This duration allows the system to reach a steady state before fault inception and provides sufficient post-fault data.

- **Solver time step:** 500 μs (microseconds). This small time step is necessary to accurately simulate the fast electromagnetic transients associated with faults and converter switching.
- **Fault initiation time:** faults are triggered at 2.5 seconds into the simulation. A random start-time variation (e.g., $\pm$ a few milliseconds around 2.5s) is introduced for each fault scenario. This ensures that faults are not always initiated at the exact same point on the AC waveform, adding realism and diversity to the dataset. The pre-fault duration allows the system to operate in a stable steady state.

2.3.3 Note on Per-Unit (pu) System Representation

Throughout this thesis, particularly in the simulation results and graphical representations, electrical quantities such as voltage and current are often expressed in **per-unit (pu)** values. This is a standard and widely adopted practice in power system analysis.

A per-unit value is the ratio of an actual physical quantity to a pre-defined base value of the same dimension:

$$\text{Per-Unit Value} = \frac{\text{Actual Value}}{\text{Base Value}} \tag{2.1}$$

The use of the per-unit system offers several significant advantages, making it particularly useful for simulations and graphical analysis:

- **Normalization and Comparison:** it normalizes quantities, bringing values from different parts of the system (which may have vastly different absolute magnitudes) onto a common dimensionless scale. This greatly simplifies the comparison of equipment performance and system behavior, especially in graphs where multiple signals are plotted.
- **Simplified Analysis:** system parameters, such as transformer impedances, tend to fall within a relatively narrow range when expressed in per-unit.
- **Clarity of Deviation:** per-unit values provide an immediate and intuitive understanding of how much a quantity deviates from its nominal or rated value. For instance, a voltage of 0.9 pu clearly indicates it is 10% below its nominal level.

In the context of this work, the specific base values for the per-unit system are derived from the HVDC system's rated specifications and the conventions used in the simulation model, as detailed in the provided simulation parameters.

For DC Quantities: The base values for DC voltage and current are explicitly defined by the simulation:

- The base DC voltage is 500 kV. Therefore, in graphs depicting DC voltage, 1 pu corresponds to 500 kV.
- The base DC current is 2 kA. Therefore, in graphs depicting DC current, 1 pu corresponds to 2 kA.

For AC Quantities: The representation of AC quantities in per-unit is as follows:

- **AC Voltages:** voltages on the AC side are presented in per-unit based on the nominal AC system line-to-line voltage at the point of measurement. This is 500 kV for the rectifier side AC system and 345 kV for the inverter side AC system.
- **AC Currents:** currents on the AC side, often indicated by axis labels such as 'pu/100 MVA', are expressed in per-unit relative to a base current. This base current is derived using a common system apparent power base 'S_base' of 100 MVA and the respective nominal AC system line-to-line voltage at the point of measurement (500 kV for the rectifier side or 345 kV for the inverter side). The formula for this base current 'I_base' is $I_{\text{base}} = S_{\text{base}}/(\sqrt{3} \cdot V_{\text{base,LL}})$.

This consistent use of the per-unit system with these defined base values allows for a standardized representation and straightforward interpretation of the system's dynamic behavior under various fault conditions across different parts of the HVDC link. This is particularly evident in the graphical analysis presented in subsequent figures.

2.4 Fault Characterization and Modeling in Simulation

This study explores various fault types in high voltage direct current (HVDC) systems, dividing them into DC line faults and AC side faults, while also considering normal (no-fault) operation. Each fault type produces distinct electrical signatures, marked by specific voltage and current deviations. Using simulations, the research models these faults to understand their effects on system performance. This approach is vital for improving fault detection and protection strategies without relying on real-world incidents. The comprehensive analysis of diverse fault scenarios enhances the study's relevance to practical HVDC applications. Ultimately,

identifying and classifying these unique signatures advances the reliability and safety of HVDC power networks.

2.4.1 Normal Operation ('None')

Under normal, steady-state operation, the HVDC system transmits power smoothly from the rectifier to the inverter. DC voltages and currents are relatively constant (with some ripple due to converter operation), and AC voltages and currents at the converter buses are balanced and sinusoidal (after filtering).

- **Characteristics:** stable DC power flow, nominal DC voltage and current levels, filtered AC waveforms.
- **Simulation:** this scenario involves simulating the system without any fault triggers for the entire 3.5-second duration.

2.4.2 DC Line Faults

Faults on the DC transmission line are common and can be severe. This study focuses on DC line-to-ground faults at various locations.

- **Simulated locations:** 10%, 25%, 50%, 75%, and 90% of the 300 km transmission line length, measured from the rectifier end. These are denoted as 'DC10', 'DC25', 'DC50', 'DC75', and 'DC90'.
- **Modeling in simulation:** a DC line-to-ground fault is typically simulated by closing a switch that connects one of the DC poles to ground through a specified fault resistance. The Simulink diagram (figure 2.4) shows fault blocks ("DC Fault" to "DC Fault4") along the DC line for this purpose.
- **General characteristics:**
  - Rapid collapse of DC voltage on the faulted pole.
  - Sharp increase in DC current flowing from the converters towards the fault point, limited initially by smoothing reactors and line impedance.
  - Potential for commutation failures at the inverter due to DC voltage depression.

– The severity and characteristics (e.g., rate of rise of current, voltage drop magnitude) depend on the fault location, fault resistance, and system parameters.

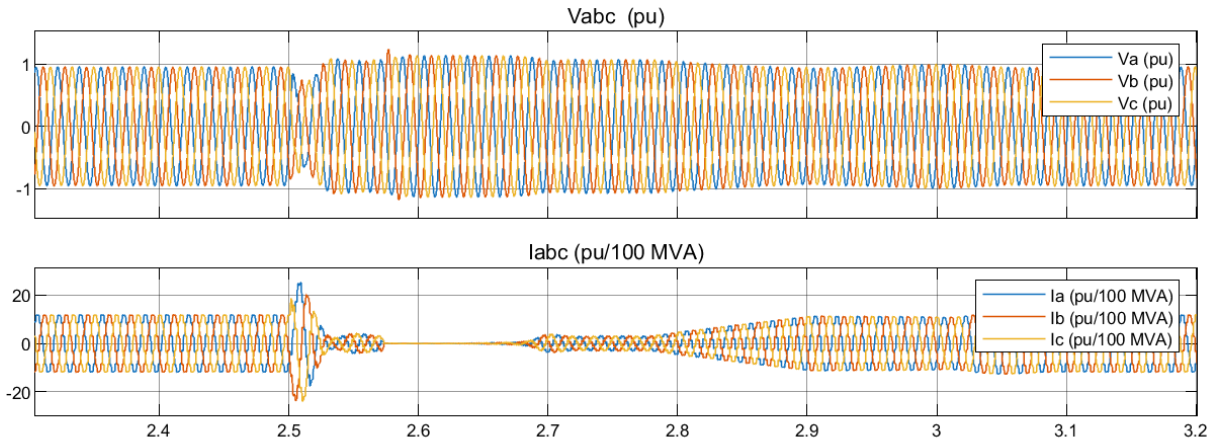

Figure 2.5: Waveforms of rectifier AC voltage and current during a DC line-to-ground fault at 50% of the transmission line.

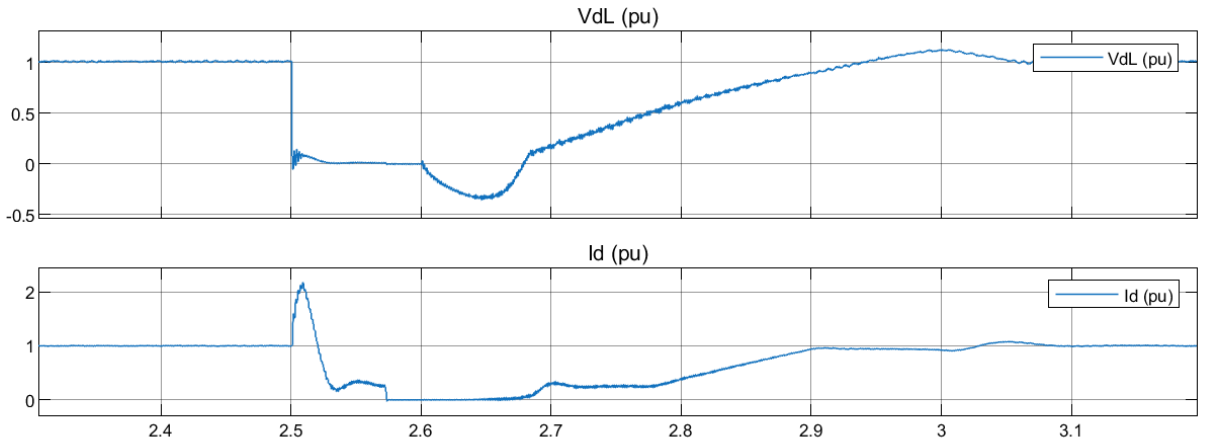

Figure 2.6: Waveforms of rectifier DC voltage and current during a DC line-to-ground fault at 50% of the transmission line.

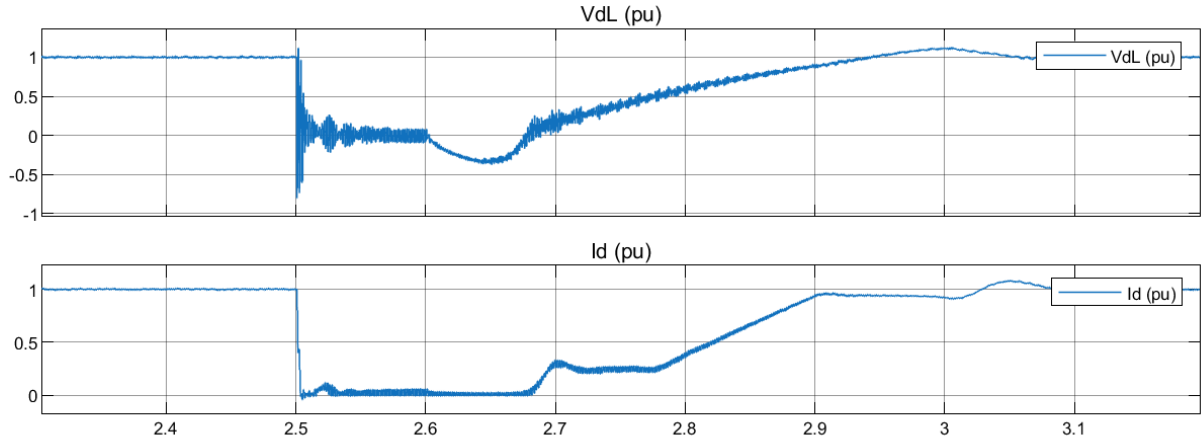

Figure 2.7: Waveforms of inverter DC voltage and current during a DC line-to-ground fault at 50% of the transmission line.

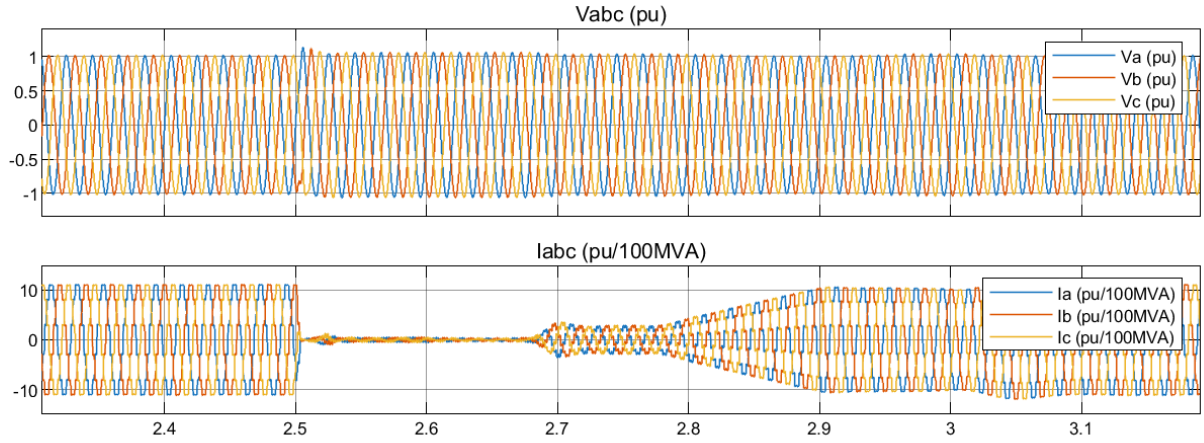

Figure 2.8: Waveforms of inverter AC voltage and current during a DC line-to-ground fault at 50% of the transmission line.

2.4.3 AC Side Faults

Faults on the AC systems connected to the rectifier or inverter can significantly impact the operation of the HVDC link. The simulated AC faults occur on the AC bus close to the converter transformers (as shown near the inverter in figure 2.4, but applicable to either side).

A - Single-Phase-to-Ground (L-G) AC Faults

These faults involve one phase conductor shorting to ground.

- **Simulated types:** 'AG' (phase A to ground), 'BG' (phase B to ground), 'CG' (phase C to ground).
- **Modeling in simulation:** achieved by closing a switch connecting the faulted phase to ground through a fault resistance at the AC bus.
- **General characteristics:**
  - Voltage drop in the faulted phase and potential voltage swells in healthy phases (depending on grounding).
  - Increase in current in the faulted phase.
  - Unbalanced AC voltages supplied to the converters, leading to increased non-characteristic harmonics.
  - High probability of commutation failures, especially if the fault is electrically close to the converter and severe. This results in a temporary inability of the inverter to commutate current from one valve to the next, often leading to a DC-like short circuit through the converter valves and a rapid increase in DC current.

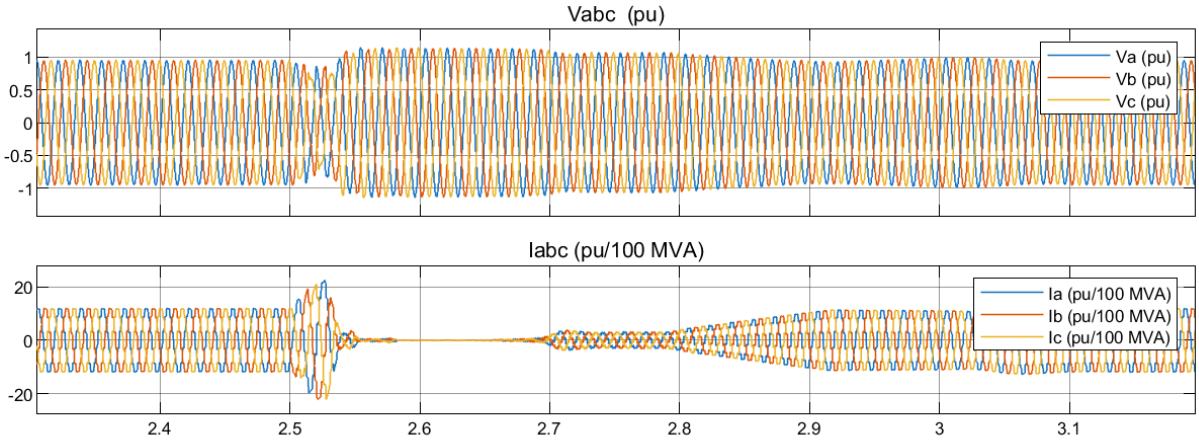

Figure 2.9: Waveforms of rectifier AC voltage and current during an AC single-phase-to-ground fault (AG).

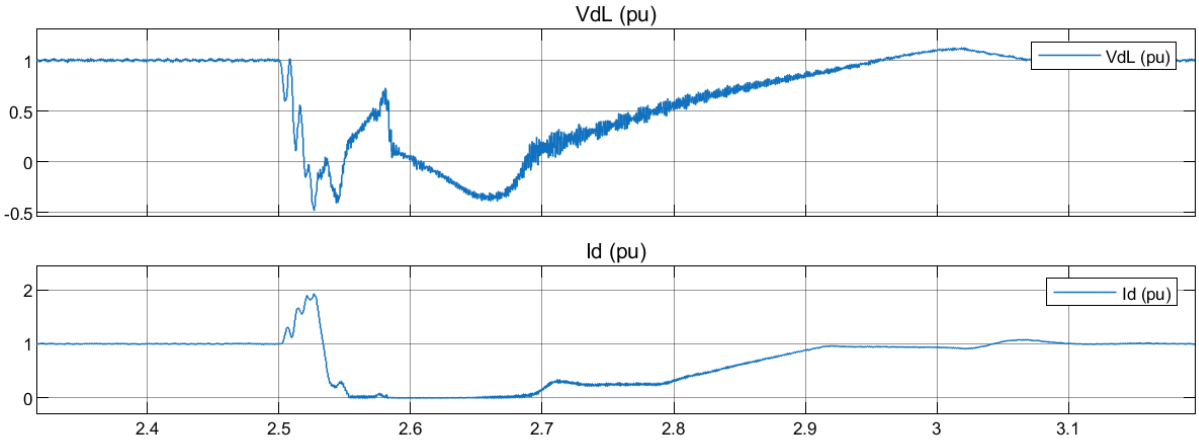

Figure 2.10: Waveforms of rectifier DC voltage and current during an AC single-phase-to-ground fault (AG).

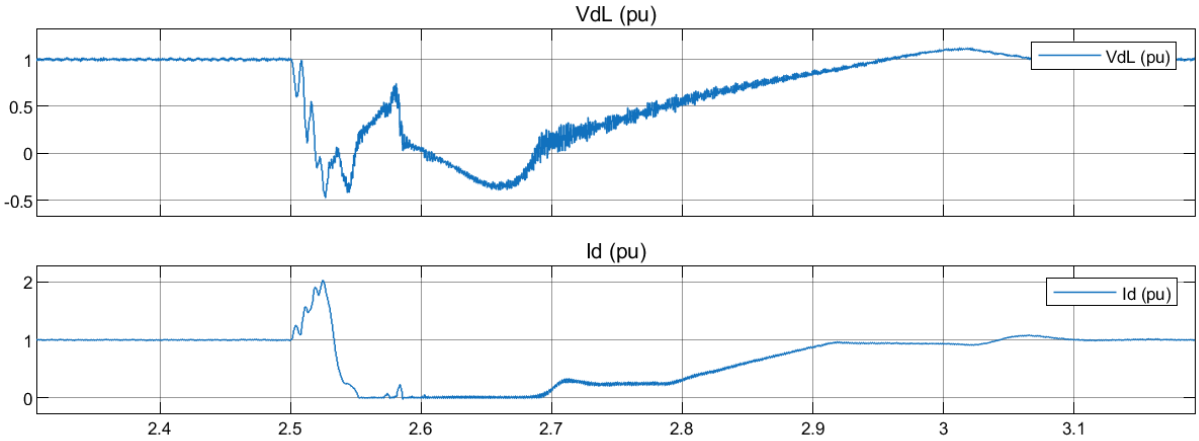

Figure 2.11: Waveforms of inverter DC voltage and current during an AC single-phase-to-ground fault (AG).

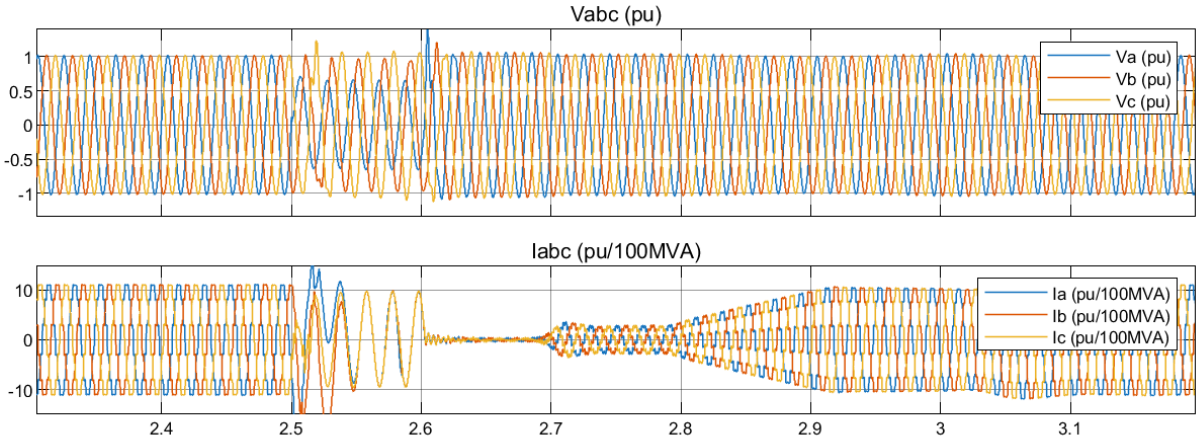

Figure 2.12: Waveforms of inverter AC voltage and current during an AC single-phase-to-ground fault (AG).

B - Phase-to-Phase (L-L) AC Faults

These faults involve two phase conductors shorting together.

- **Simulated types:** 'AB' (phase A to phase B), 'BC' (phase B to phase C), 'AC' (phase A to phase C).
- **Modeling in simulation:** achieved by closing a switch connecting the two faulted phases through a fault resistance at the AC bus.
- **General characteristics:**
  - Severe voltage dips in the faulted phases and potential overvoltages or distortions in the unfaulted phase.
  - Large fault currents flowing between the faulted phases.
  - Significant unbalance in the AC voltages supplied to the converters.
  - Very high likelihood of repeated commutation failures in the converters, often leading to a complete, albeit temporary, interruption of power transfer. The DC current rises significantly due to the effective short-circuiting of the DC link through the commutating valves.

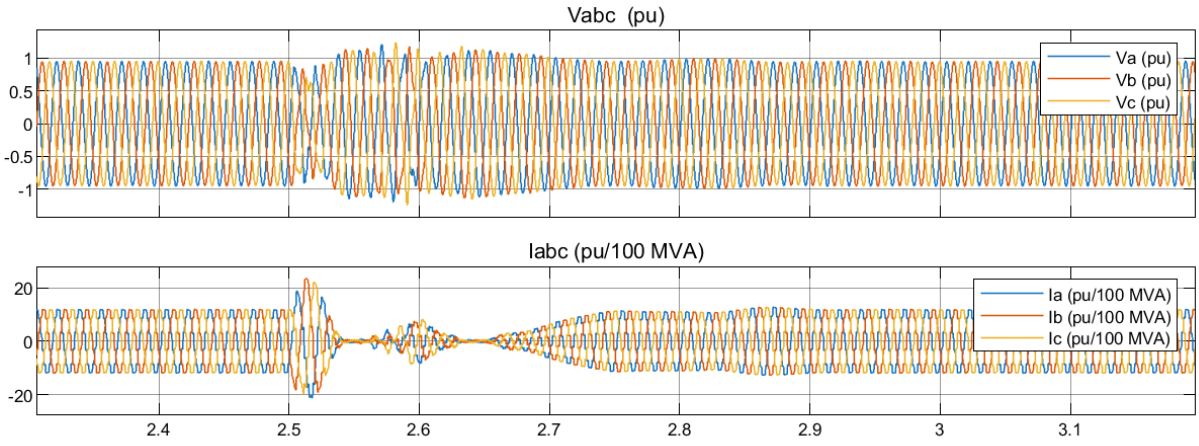

Figure 2.13: Waveforms of rectifier AC voltage and current during an AC phase-to-phase fault (AB).

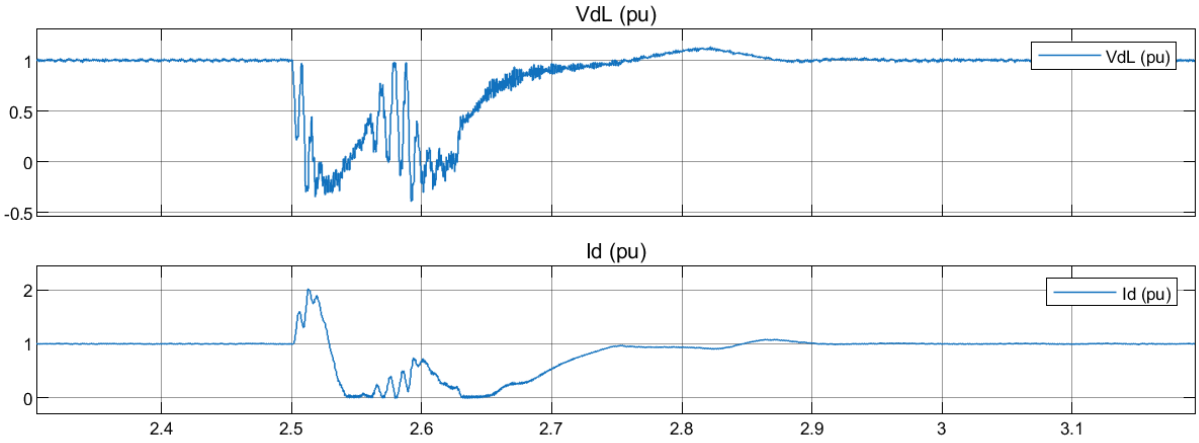

Figure 2.14: Waveforms of rectifier DC voltage and current during an AC phase-to-phase fault (AB).

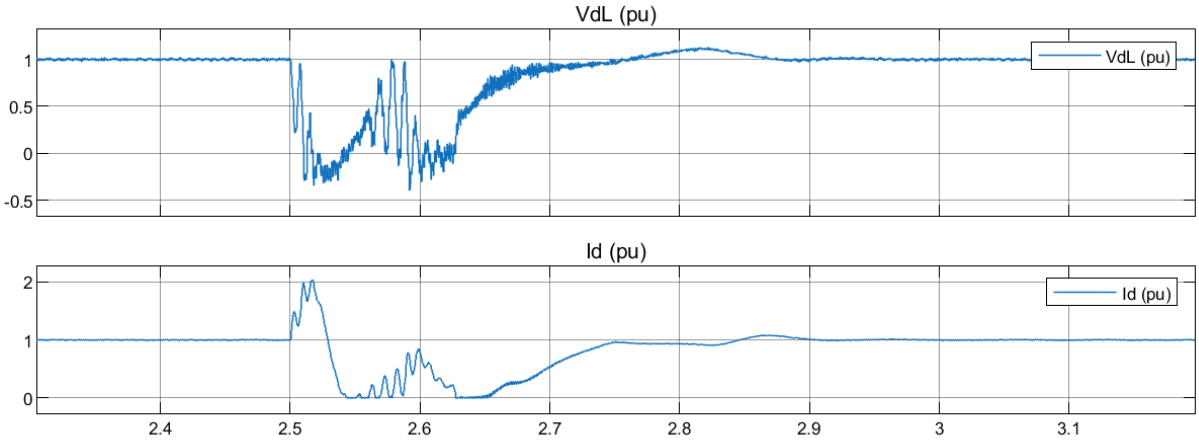

Figure 2.15: Waveforms of inverter DC voltage and current during an AC phase-to-phase fault (AB).

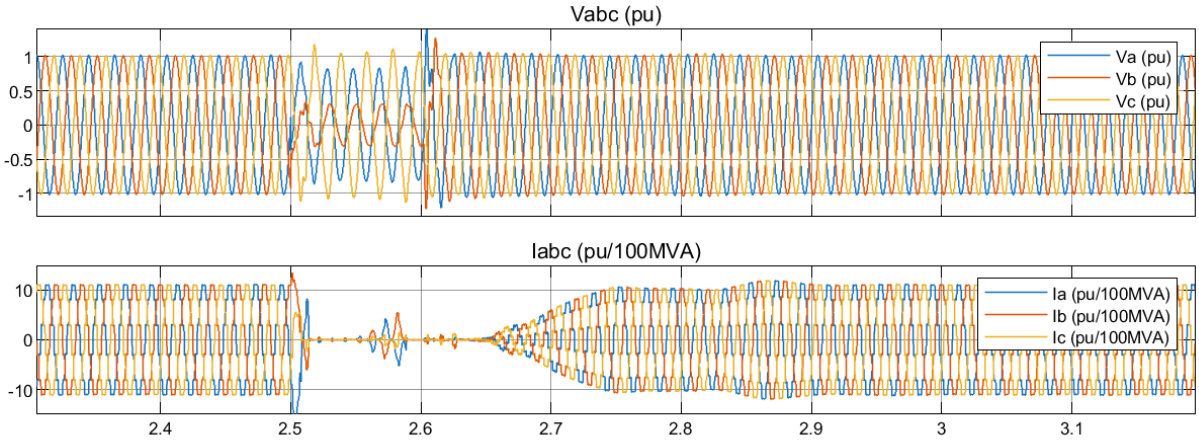

Figure 2.16: Waveforms of inverter AC voltage and current during an AC phase-to-phase fault (AB).

For all fault types, variations in fault resistance and fault duration are also introduced during data generation (as detailed in Chapter 3) to create a diverse dataset for machine learning.

2.5 Conclusion

This chapter explores the 12-pulse line-commutated converter high-voltage direct current (LCC-HVDC) transmission system, a cornerstone of this research. It details critical components: `converters`, `transformers`, `AC filters`, `DC smoothing reactors`, and the `DC line` and their functions, all modeled in MATLAB/Simulink. The system, a robust 1000 MW, ±500 kV, 300 km setup, showcases large-scale power transmission. Simulation specifics include a 3.5-second duration and a 500 µs time step, ensuring accuracy. These elements are meticulously represented to mirror real-world behavior. The model enables safe analysis of system responses, like faults, in a controlled environment. MATLAB/Simulink's platform enhances accessibility and precision. This approach deepens insights into HVDC dynamics and supports advanced research outcomes. The different fault types analyzed in high voltage direct current (HVDC) systems, including normal operation, DC line-to-ground faults occurring at five distinct locations, AC single-phase-to-ground faults, and AC phase-to-phase faults. For each type, the discussion covers their specific electrical characteristics and their effects on the HVDC system, offering valuable insights into how the system responds under various fault conditions. Additionally, the chapter explains how these faults are carefully modeled within simulations to reflect real-world behavior accurately. This thorough fault analysis and modeling effort is vital for producing a high-fidelity dataset, which is essential for building and evaluating machine learning-based fault detection and classification methods central to this thesis. The following chapter will provide a deeper look into the systematic creation of fault scenarios and the preparation of this data for machine learning applications.

Chapter 3

Scenario Generation and Data Preparation

3.1 Introduction

The foundation of any successful machine learning endeavor, particularly in complex domains like high voltage direct current (HVDC) fault diagnosis, lies in the quality, diversity, and representativeness of the data used for training and evaluation. This chapter meticulously details the systematic methodology employed to generate and prepare a comprehensive dataset tailored for this study.

The process begins with the careful design of various fault scenarios pertinent to line-commutated converter (LCC) HVDC systems, encompassing different fault types, locations, resistances, and durations, alongside normal operating conditions. Subsequently, it describes the automated simulation framework developed, leveraging MATLAB/Simulink for system modeling and Python for orchestrating parallel simulations and efficient data aggregation. The chapter then outlines the comprehensive data acquisition strategy, the selection of key electrical signals, and the subsequent feature engineering process, where meaningful statistical features are extracted from the raw time-series data. Finally, it details the crucial data preprocessing pipeline, including signal segmentation, data cleaning, and normalization, all essential steps to create a robust and well-structured dataset ready for the machine learning tasks discussed in the subsequent chapter.

3.2 Fault Scenario Design for LCC-HVDC Systems

To develop ML models capable of accurately identifying and classifying various disturbances in an LCC-HVDC system, a diverse range of fault scenarios was meticulously designed and simulated. The selection of fault types and their parameter variations was guided by their prevalence in real-world systems and their impact on system operation, as documented in existing literature [3, 5].

3.2.1 Fault Types and Characteristics

The simulated dataset encompasses the following primary fault categories:

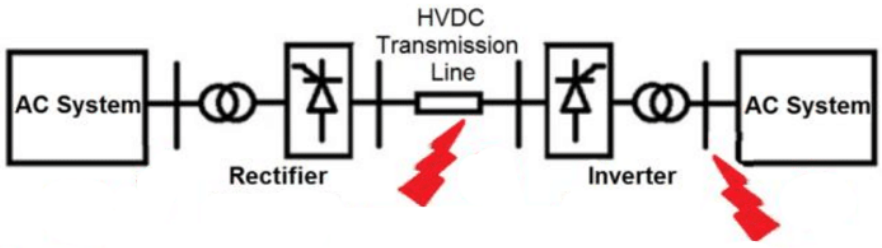

Figure 3.1: Simulated HVDC fault types.

A - DC Line Faults: these are among the most common and critical faults in HVDC transmission lines [17].

- *Pole-to-Ground (P-G) Faults:* involving a short circuit between one of the DC poles and the ground.

For DC line faults, the following parameters were systematically varied to ensure a rich dataset:

- *Fault Resistance (R_f):* values such as 0.01Ω, 0.1Ω, 1Ω, 10Ω, 50Ω, 100Ω, and 500Ω were used. This range covers low-impedance (severe) to high-impedance (incipient or arcing) faults, significantly influencing the fault current magnitude and voltage depression [16].
- *Fault Location (L_f):* faults were simulated at different points along the DC transmission line, specifically at 10%, 25%, 50%, 75%, and 90% of the line length from the rectifier end. Fault location critically affects the traveling wave phenomena and the measured impedances from either terminal [20].

- *Fault Duration (T_d):* varied durations such as 0.01s, 0.05s, 0.1s, 0.2s, 0.3s, and 0.5s were considered to capture both transient and more sustained fault events.

B - AC System Faults at Converter Terminals: faults on the AC side connected to the converter stations can lead to commutation failures and significantly impact HVDC operation [15].

- *Single Phase-to-Ground (AG, BG, CG) Faults:* on the AC bus at both rectifier and inverter sides.
- *Phase-to-Phase (AB, BC, CA) Faults:* on the AC bus at both rectifier and inverter sides.

For AC system faults, the same range of *Fault Resistance (R_f)* and *Fault Duration (T_d)* values as used for DC line faults were applied to ensure consistency and comprehensive coverage of fault severities and persistence. The impact of these AC faults on the HVDC system is highly dependent on these parameters, influencing the extent of voltage distortion and potential for commutation failure.

C - No-Fault (Normal Operation) Scenarios: a significant number of simulations under normal operating conditions, including minor load variations or system parameter drifts, were included. This is crucial for training ML models to distinguish between fault and non-fault states, thereby minimizing false alarms.

The systematic variation of these parameters ensures that the generated dataset captures a wide operational envelope, crucial for training generalizable ML models as discussed in Section 1.6.

3.3 Automated Simulation Framework

Generating a large and diverse dataset, as described above, necessitates an automated simulation framework. This study leveraged MATLAB/Simulink for modeling the LCC-HVDC system and developed custom Python scripts for orchestrating the numerous individual simulation runs and managing the resultant data.

3.3.1 MATLAB/Simulink Model and Scripted Single Simulations

A detailed electromagnetic transient (EMT) model of a two-terminal LCC-HVDC system was developed in MATLAB/Simulink. This model includes representations of the converter

bridges, transformers, AC and DC filters, DC transmission line, and control systems. Each fault scenario was simulated individually by a MATLAB script. This script was responsible for:

1. Programmatically setting the specific fault parameters (type, location, resistance, duration, etc.) within the Simulink model workspace.
2. Running a single simulation for the configured scenario.
3. Extracting the specified output signals and saving them to a temporary CSV file upon simulation completion.

This approach of scripting individual simulations provides fine-grained control over each scenario. The simulation automation scripts, along with the base Simulink model, are open-sourced and available for reproducibility and further research (see Appendix .2).

3.3.2 Python-Orchestrated Parallel Simulation and Data Aggregation

To expedite the generation of the extensive dataset required (tens of thousands of scenarios) and to manage the workflow efficiently, a distributed and parallelized simulation strategy was implemented using Python.

- **Distributed Simulation Chunks:** the total list of defined fault scenarios was logically divided into manageable chunks.
- **Python-based Parallel Orchestration:** a Python script utilizing the multiprocessing library was developed to manage the execution of these individual MATLAB simulations in parallel. This script was responsible for:
  - Iterating through the list of all defined fault scenarios (or chunks thereof).
  - For each scenario, preparing the necessary input configuration to be passed to the MATLAB simulation script.
  - Launching multiple instances of MATLAB (each running one simulation scenario at a time) in parallel across available CPU cores or even distributed across multiple servers. In this study, each server was configured to run 10 such MATLAB simulations concurrently.
  - Monitoring the completion of each MATLAB process and handling the output CSV file.

- **Data Storage and Organization on Hugging Face Datasets:** upon successful completion and data extraction, each individual fault scenario simulation resulted in a CSV file. These CSV files were then systematically uploaded to a dedicated Hugging Face Dataset repository. This centralized cloud-based storage offers robust accessibility, versioning, and management capabilities. The dataset within the Hugging Face repository is organized into folders, where each folder name indicates a specific fault type and, for DC faults, its location. The folder structure includes names such as: `DC10`, `DC25`, `DC50`, `DC75`, `DC90` (for DC line faults at 10%, 25%, 50%, 75%, and 90% of the line length, respectively), and `AG`, `BG`, `CG`, `AB`, `AC`, `BC` (for various AC fault types). Each CSV file within these folders is named according to the convention:
  `runNumber_faultType_tInception_dDuration_rResistance.csv`
  For example, `123_AG_t0.50_d0.10_r50.csv` would represent run number 123, a A phase-to-ground fault, occurring at 0.50s, with a duration of 0.10s, and a fault resistance of 50Ω.

This parallelized approach, combining MATLAB's simulation capabilities with Python's orchestration and Hugging Face's data infrastructure, enabled the efficient generation of a substantial dataset comprising **22612 individual CSV files**, each representing a unique fault or operational scenario.

3.4 Comprehensive Data Acquisition

During each simulation run, a comprehensive set of electrical signals from various points within the LCC-HVDC model was captured. The rationale for collecting such a wide array of signals is to provide a rich dataset that can support not only the current study but also future research, exploration of alternative features, or the development of more detailed diagnostic algorithms. Table 17 in Appendix .1 provides a detailed overview of the key signals logged.

These signals represent voltages, currents at various key locations, control system parameters, and internal operational states from both the rectifier and inverter stations, as well as currents directly associated with the fault path. Each CSV file generated, as mentioned in Section 3.3, contains a time-series record of these signals for the duration of that specific simulation scenario.

3.5 Feature Engineering and Selection for Machine Learning

While a vast amount of raw signal data was captured, specific signals were selected, and statistical features were derived from them for the initial phase of ML model training detailed in this thesis. This selection and engineering process is crucial for transforming raw data into a format that ML algorithms can effectively learn from.

3.5.1 Signal Selection for Current Study

For the present study, the following subset of signals was chosen as the primary input for feature extraction, representing key electrical quantities at both ends of the HVDC link and on the AC sides. These signals are known to exhibit distinct transient and steady-state changes during fault conditions, providing valuable information for fault detection and classification [8]:

- DC Voltage at Rectifier.
- DC Current at Rectifier.
- DC Voltage at Inverter.
- DC Current at Inverter.
- AC Phase Voltages at Rectifier.
- AC Phase Currents at Rectifier.
- AC Phase Voltages at Inverter.
- AC Phase Currents at Inverter.

3.5.2 Feature Calculation

From each selected time-series signal within a defined post-fault window (see Section 3.6.1), a set of statistical features was calculated. These features aim to summarize the temporal behavior of the signals into a concise numerical representation that is more readily digestible by ML algorithms than raw time-series data. The calculated features for each signal segment include:

- **Maximum (max):** this feature captures the peak positive amplitude of the signal within the window. During faults, voltages might experience sags (lower max) or swells (higher max), while currents often exhibit significant overshoots. The maximum value is thus crucial for identifying the severity and type of transient disturbance.

- **Minimum (min):** this represents the peak negative amplitude or the lowest point of a sag in the signal window. Similar to the maximum, it provides critical information about the extent of voltage depressions or negative peaks in current waveforms, characteristic of many fault types.

- **Mean (μ):** the average value of the signal over the window. Post-fault, the mean value of a signal may shift significantly from its pre-fault steady-state level. For example, DC voltage typically drops, and its mean value reflects this new quasi-steady state during the fault.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.1}$$

- **Standard Deviation (σ):** this measures the dispersion or variability of the signal around its mean within the window. A higher standard deviation indicates greater fluctuation, which can be indicative of oscillatory transients, harmonics, or instability introduced by the fault. It quantifies the signal's dynamic activity.

$$\text{std} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{3.2}$$

- **Energy (E):** calculated as the sum of the squared instantaneous values of the signal within the window. The energy of a signal is related to its overall power content. Faults typically cause a substantial change in the energy of voltage and current signals—currents often see increased energy due to fault current contribution, while voltage energy might decrease due to sags. This feature provides a robust indicator of the disturbance's intensity.

$$\text{energy} = \sum_{i=1}^{N} x_i^2 \tag{3.3}$$

The combination of these five statistical features for each of the selected signals creates a feature vector. This vector serves as a "fingerprint" for the system's behavior during the analyzed window, aiming to provide sufficient discriminatory information for the ML models to distinguish between different fault types, characteristics, and normal operation [45,46]. Extracting such features helps transform complex temporal patterns into a more structured and lower-dimensional representation, which can improve both the performance and computational efficiency of the subsequent ML classification task.

3.6 Data Preprocessing Pipeline

Before feeding the extracted features into ML models, several preprocessing steps were essential to ensure data quality and optimize model performance.

3.6.1 Signal Segmentation and Windowing

The raw time-series data generated from simulations includes the pre-fault steady-state, the fault onset, and the subsequent transient response. In this study, feature extraction was performed using a fixed time window centered around the fault initiation point. Specifically, a few milliseconds of signal were captured both before and after the fault occurred, allowing the window to capture the transition from normal operation to fault conditions. This approach ensures that both pre-fault behavior and the immediate fault transient are represented, which is critical for distinguishing subtle signal changes.

3.6.2 Data Cleaning

The generated dataset was meticulously inspected for any inconsistencies or missing data. Although simulations generally produce clean data, it is good practice to check for:

- **NaN (Not-a-Number) Values:** in rare cases, simulation convergence issues or data logging errors might introduce NaN values. Any scenario (CSV file) or signal containing NaN values within the analysis window was either carefully inspected and corrected if the issue was minor and identifiable, or excluded from the training/testing set to prevent errors in ML model training.
- **Constant or Corrupted Signals:** signals that were unexpectedly constant (indicating a possible issue with the sensor modeling or data logging in the simulation) or showed obviously erroneous patterns were flagged. If a scenario contained such data that could not be rectified, it was typically removed from the dataset.

This cleaning process ensures that the ML algorithms are trained on reliable and meaningful data.

3.6.3 Normalization/Scaling

Features extracted from different signals or calculated using different methods can have widely varying scales and ranges (e.g., voltage in per-unit vs. energy which can be much larger). Most ML algorithms, particularly those relying on distance calculations (like KNN, SVM) or gradient descent optimization (like Neural Networks, Logistic Regression), perform better or converge faster when input features are on a relatively similar scale [47]. Normalization or scaling prevents features with larger numerical values from dominating those with smaller values during model training.

Two common techniques considered are:

- **Min-Max Scaling (Normalization):** rescales features to a fixed range, typically [0, 1] or [-1, 1]. It is calculated as: $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$
- **Z-score Standardization:** rescales features to have zero mean and unit variance. It is calculated as: $X_{std} = \frac{X - \mu}{\sigma}$ where μ is the mean and σ is the standard deviation of the feature across the training samples.

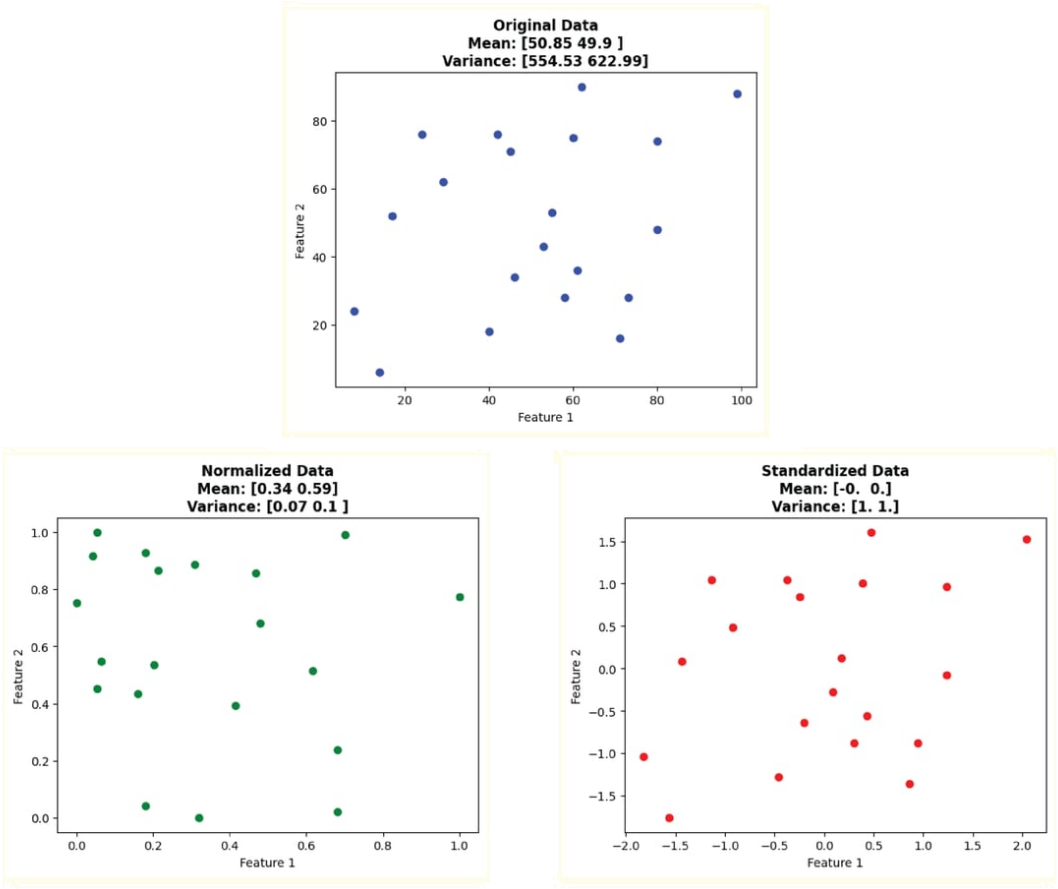

Figure 3.2: Example of normalization and standardization.

In this study, Z-score standardization using scikit-learn's `StandardScaler` was applied to the feature set. The scaling parameters (μ and σ for standardization, or X_{min} and X_{max} for normalization) were computed from the training data and then applied consistently to the validation and test datasets. This critical step prevents data leakage from the test set into the training process, ensuring an unbiased evaluation of model performance and promoting robust generalization.

3.7 Conclusion

This chapter has comprehensively outlined the systematic approach undertaken to generate and prepare the dataset crucial for the machine learning-based fault diagnosis in HVDC systems. The process commenced with a detailed fault scenario design for a two-terminal LCC-HVDC system, incorporating a wide range of fault types (DC line faults, AC system faults) and parameters (resistance, location, duration), alongside no-fault conditions. An automated simulation framework, combining MATLAB/Simulink for modeling and Python for parallel execution and data management, was developed to efficiently generate a substantial dataset of 22612 individual simulation runs.

Key electrical signals were captured, from which statistical features such as maximum, minimum, mean, standard deviation, and energy were calculated for selected signals within a defined post-fault window. This feature set was then subjected to data preprocessing pipeline, including data cleaning to handle any anomalies, and Z-score standardization to ensure feature consistency. The result of these well-structured dataset, ready to support training, validating, and testing the machine learning models in the next chapter. The open availability of these simulation scripts and the dataset itself is intended to foster further research and reproducibility in this domain.

Chapter 4

Machine Learning Framework and Results Analysis

4.1 Introduction

This chapter presents the core experimental work of the thesis, focusing on two main evaluation scenarios for HVDC fault detection and classification using machine learning models:

- **Standard evaluation (random 80/20 split):** models are trained and tested using a conventional random 80/20 train-test split, serving as a baseline and reflecting common practice.
- **Generalization testing (unseen fault resistances):** models are trained without samples containing fault resistances of 10 Ω and 100 Ω, and then evaluated specifically on these unseen conditions to assess real-world robustness.

This approach enables a critical assessment of both standard performance and generalization capability. The chapter concludes with a comparative analysis of all models across both scenarios, highlighting their strengths, weaknesses, and robustness. All datasets and scripts used, as described in Chapter 3, are openly available on Hugging Face and GitHub to support reproducibility and further research, as discussed in Section 1.6.

4.2 Machine Learning Models and Framework

This section details the machine learning algorithms chosen for the HVDC fault detection task. A diverse set of models was selected from the `scikit-learn` library to benchmark their performance and generalization capabilities. For each model, a brief theoretical overview is provided, along with key hyperparameters that were tuned. Hyperparameter optimization was performed using `GridSearchCV` with 5-fold cross-validation on the training set to find the best combination of parameters for each model.

4.2.1 Logistic Regression

Logistic regression is a linear model used for binary or multi-class classification problems. Despite its name, it is a classification algorithm rather than a regression one. For binary classification, it models the probability that an input $\mathbf{x}$ belongs to a particular class using the logistic (or sigmoid) function:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}} \tag{4.1}$$

where $\mathbf{w}$ are the weights and b is the bias term, learned during training by minimizing a loss function (typically log-loss). For multi-class problems, as in this study (multiple fault types), a "one-vs-rest" (OvR) or multinomial approach can be used. Scikit-learn's 'LogisticRegression' handles this automatically [48].

- **Key hyperparameters tuned:**
  - `C`: inverse of regularization strength. Smaller values specify stronger regularization.
  - `solver`: algorithm to use in the optimization problem (e.g., 'liblinear', 'lbfgs', 'saga').
  - `penalty`: specifies the norm used in the penalization (e.g., 'l1', 'l2').
- **Strengths:** simple, interpretable, computationally efficient, performs well on linearly separable data.
- **Weaknesses:** may not perform well on complex, non-linear data; assumes linear relationship between features and log-odds.

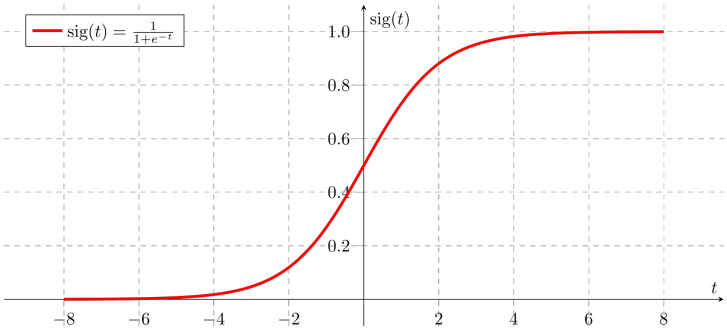

Figure 4.1: The Sigmoid function used in Logistic Regression [48].

4.2.2 Support Vector Machine (SVM)

Support vector machines (SVMs) are powerful supervised learning models used for classification and regression. For classification, SVMs aim to find an optimal hyperplane in an N-dimensional space (where N is the number of features) that maximally separates data points of different classes. The "margin" is the distance between the hyperplane and the closest data points (support vectors) from each class. For non-linearly separable data, SVMs can use the "kernel trick" to map the data into a higher-dimensional space where a linear separation might be possible.

$$\text{Decision Function: } \operatorname{sign}\left(\sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{4.2}$$

where $K(\mathbf{x}_i, \mathbf{x})$ is the kernel function, α_i are Lagrange multipliers, y_i are class labels, and b is the bias term [49, 50].

- **Key hyperparameters tuned:**
  - `C`: regularization parameter. A smaller C creates a wider margin but may misclassify more points, while a larger C aims to classify all training examples correctly but may lead to a narrower margin and overfitting.
  - `kernel`: specifies the kernel type to be used in the algorithm (e.g., 'linear', 'poly', 'rbf', 'sigmoid'). The radial basis function (RBF) kernel is commonly used.
  - `gamma`: kernel coefficient for 'rbf', 'poly', and 'sigmoid'. Higher gamma means more influence of single training examples, potentially leading to overfitting.
- **Strengths:** effective in high-dimensional spaces, robust to overfitting if C is chosen carefully, versatile due to different kernel functions.

- **Weaknesses:** can be computationally intensive for large datasets. Performance depends heavily on hyperparameter tuning (especially C, kernel, and gamma). Less interpretable than decision trees.

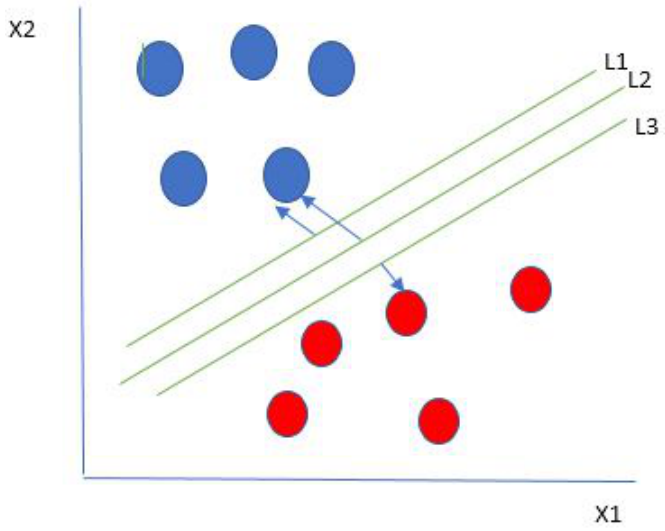

Figure 4.2: Conceptual Illustration of a Support Vector Machine (SVM) Classifier [50].

4.2.3 K-Nearest Neighbors (KNN)

K-nearest neighbors (KNN) is a non-parametric, instance-based learning algorithm. It classifies a new data point based on the majority class of its 'k' nearest neighbors in the feature space. The "nearness" is typically measured using a distance metric, such as Euclidean distance:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{4.3}$$

where $\mathbf{p}$ and $\mathbf{q}$ are two points in n-dimensional space [51, 52].

- **Key hyperparameters tuned:**
  - `n_neighbors` (k): the number of neighbors to consider.
  - `weights`: weight function used in prediction. 'uniform' assigns equal weights to all neighbors, while 'distance' assigns weights proportional to the inverse of the distance from the query point.
  - `metric`: the distance metric to use (e.g., 'euclidean', 'manhattan', 'minkowski').

- **Strengths:** simple to understand and implement, no explicit training phase (lazy learner), adapts well to complex decision boundaries.
- **Weaknesses:** computationally expensive during prediction for large datasets (needs to compute distances to all training points), sensitive to irrelevant features and the scale of data (necessitating normalization), performance degrades in high-dimensional spaces (curse of dimensionality). Choosing an optimal 'k' is crucial.

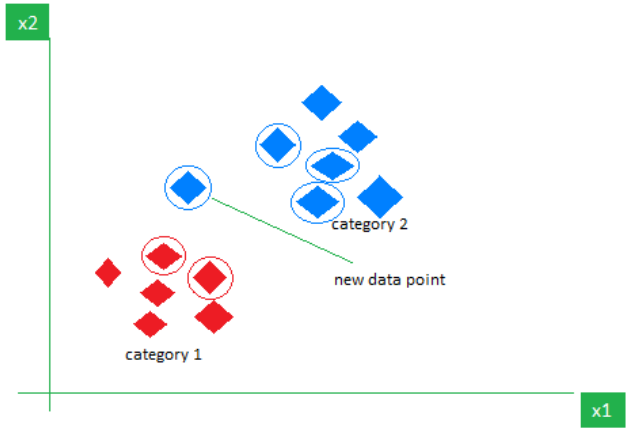

Figure 4.3: Conceptual Illustration of K-Nearest Neighbors (KNN) Classification [52].

4.2.4 Decision Tree

Decision trees (DTs) are non-parametric supervised learning methods used for classification and regression. They create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The tree is built by recursively splitting the data based on feature values that best separate the classes, typically measured by impurity metrics like Gini impurity or entropy.

$$\text{Gini impurity} = 1 - \sum_{k=1}^{K} p_k^2 \tag{4.4}$$

where p_k is the proportion of samples belonging to class k at a given node [53, 54].

- **Key hyperparameters tuned:**
  - `criterion`: the function to measure the quality of a split (e.g., 'gini', 'entropy').

  - `max_depth`: the maximum depth of the tree. Controls complexity and potential for overfitting.
  - `min_samples_split`: the minimum number of samples required to split an internal node.
  - `min_samples_leaf`: the minimum number of samples required to be at a leaf node.

- **Strengths:** easy to understand and interpret, can handle both numerical and categorical data, requires little data pre-processing (e.g., normalization is often not needed), captures non-linear relationships.
- **Weaknesses:** prone to overfitting, especially with deep trees. Can be unstable (small changes in data can lead to different trees). Biased towards features with more levels.

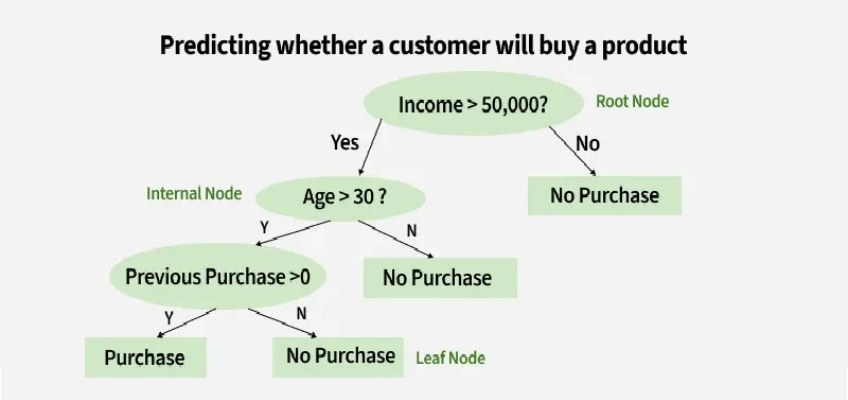

Figure 4.4: Example of a Simple Decision Tree Structure [54].

4.2.5 Random Forest

Random forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It introduces randomness by:

1. Building each tree on a bootstrapped sample of the training data (sampling with replacement).
2. Considering only a random subset of features for splitting at each node in each tree.

This randomness helps to de-correlate the trees, reducing variance and making the model more robust to overfitting compared to a single decision tree [55, 56].

- **Key hyperparameters tuned:**
  - `n_estimators` : the number of trees in the forest.
  - `max_features` : the number of features to consider when looking for the best split.
  - `max_depth` : the maximum depth of each tree.
  - `min_samples_split` , `min_samples_leaf` : same as for decision trees.
  - `criterion` : same as for decision trees.
- **Strengths:** high accuracy, robust to overfitting, handles high-dimensional data well, can estimate feature importance.
- **Weaknesses:** less interpretable than a single decision tree, can be computationally intensive for a large number of trees.

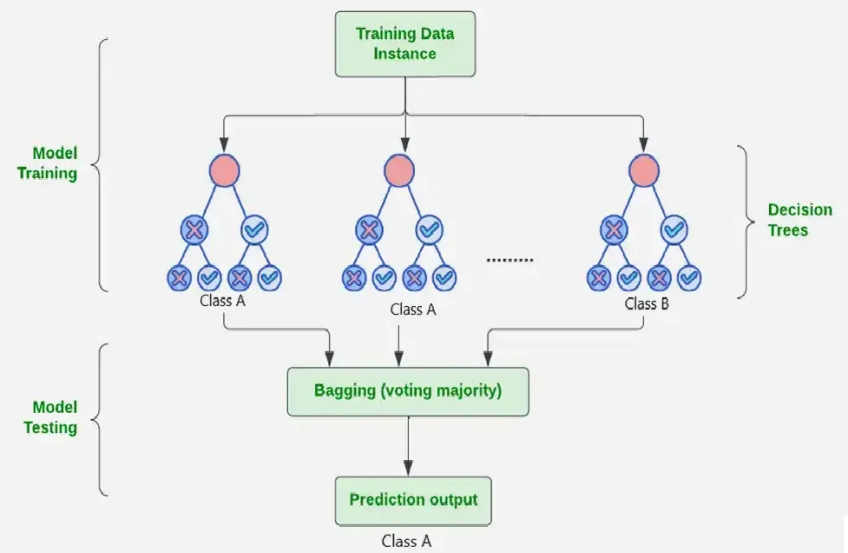

Figure 4.5: Illustration of a Random Forest Ensemble [56].

4.2.6 Gradient Boosting

Gradient boosting (specifically, gradient boosting machines or GBMs) is another powerful ensemble technique that builds models (typically decision trees) in a sequential, stage-wise fashion. Each new tree attempts to correct the errors made by the previous ensemble of trees. It fits

new models to the residual errors of the previous models, effectively "boosting" the performance by focusing on difficult-to-classify instances. The scikit-learn ‘GradientBoostingClassifier‘ is an implementation of this [55, 57].

- **Key hyperparameters tuned:**
  - `n_estimators`: the number of boosting stages (trees) to perform.
  - `learning_rate`: shrinks the contribution of each tree. Lower values require more trees but often lead to better generalization.
  - `max_depth`: maximum depth of the individual regression estimators.
  - `subsample`: the fraction of samples to be used for fitting the individual base learners. If smaller than 1.0, this results in stochastic gradient boosting.
- **Strengths:** often achieves state-of-the-art performance on many tasks, can handle various types of data, provides feature importance.
- **Weaknesses:** prone to overfitting if not tuned carefully (especially `n_estimators` and `learning_rate`), can be computationally expensive and slower to train than random forests.

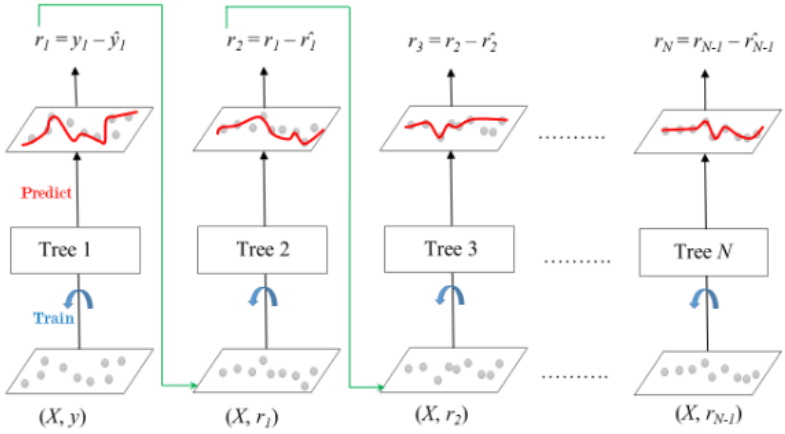

Figure 4.6: Illustration of Gradient Boosted Trees [57].

4.2.7 Neural Network (Multi-Layer Perceptron)

A multi-layer perceptron (MLP) is a class of feedforward artificial neural network (ANN). An MLP consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. Each node (neuron) in one layer connects with a certain weight to every

node in the following layer. Neurons in the hidden layers typically use a non-linear activation function (e.g., ReLU, sigmoid, tanh). The network learns by adjusting the weights using an algorithm like backpropagation based on a loss function. The output y_k of a neuron k in a layer is typically:

$$y_k = \phi\left(\sum_j w_{jk}x_j + b_k\right) \tag{4.5}$$

where ϕ is the activation function, w_{jk} are the weights from neuron j in the previous layer, x_j are the outputs of those neurons (or input features), and b_k is the bias [58].

- **Key hyperparameters tuned (using 'MLPClassifier'):**
  - `hidden_layer_sizes`: a tuple specifying the number of neurons in each hidden layer (e.g., (100,), (50, 25)).
  - `activation`: activation function for the hidden layers (e.g., 'relu', 'tanh', 'logistic').
  - `solver`: the solver for weight optimization (e.g., 'adam', 'sgd', 'lbfgs').
  - `alpha`: L2 penalty (regularization term) parameter.
  - `learning_rate_init`: the initial learning rate used.
- **Strengths:** can model highly complex, non-linear relationships; capable of learning features automatically in deeper architectures (though this MLP is relatively shallow).
- **Weaknesses:** prone to overfitting, computationally intensive to train, requires careful hyperparameter tuning, can be a "black box" (difficult to interpret). Sensitive to feature scaling.

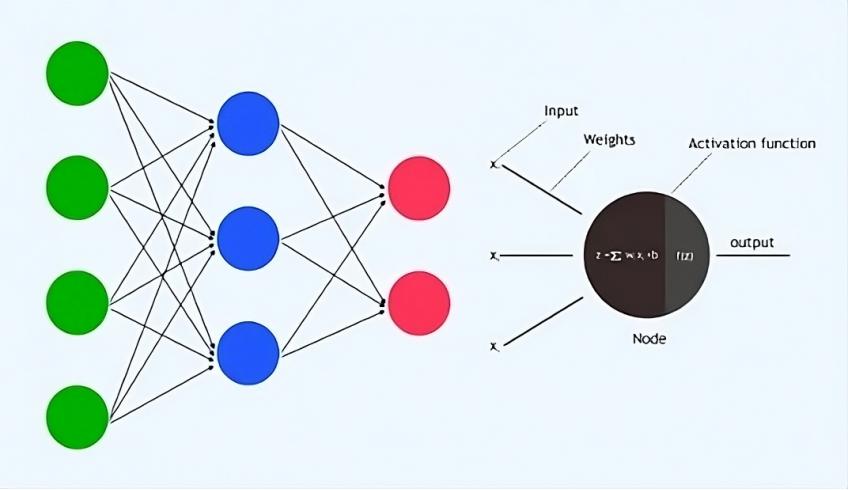

Figure 4.7: Structure of a Multi-layer Perceptron (MLP) with one hidden layer.

4.3 Experimental Setup and Evaluation Metrics

This section outlines the methodology used to evaluate the performance of the selected machine learning models. Two main experimental scenarios are considered to assess both standard performance and generalization capabilities.

4.3.1 Performance Metrics

To quantitatively evaluate the performance of the classification models, the following standard metrics derived from the confusion matrix are used:

- **Accuracy:** the proportion of correctly classified samples out of the total number of samples.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.6}$$

  (For multi-class, TP, TN, FP, FN are considered per class in a one-vs-rest manner or via macro/micro averaging of per-class metrics).

- **Precision:** the ability of the classifier not to label as positive a sample that is negative. For a given class, it is the ratio of true positives to the sum of true positives and false positives.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.7}$$

- **Recall (sensitivity or true positive rate):** the ability of the classifier to find all the positive samples. For a given class, it is the ratio of true positives to the sum of true positives and false negatives.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.8}$$

- **F1-score:** the harmonic mean of Precision and Recall. It provides a balance between these two metrics.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.9}$$

For multi-class classification problems like this one, these metrics are typically reported as weighted averages (weighted by the number of true instances for each label) to account for class imbalance, or as macro averages (unweighted mean per class). In this study, weighted averages will be primarily reported.

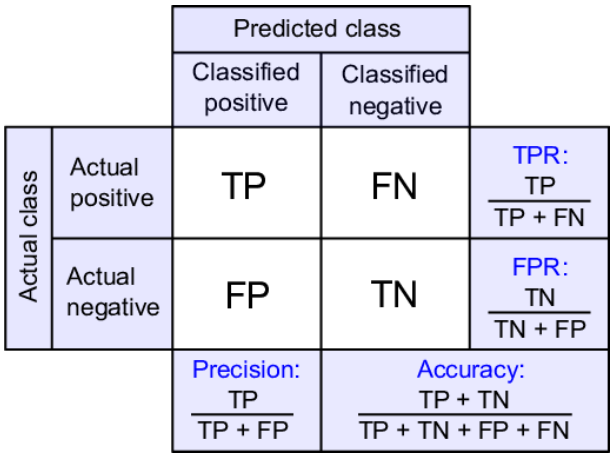

Figure 4.8: Confusion matrix illustrating binary classification metrics.

4.3.2 Experiment 1: Standard Evaluation (Random 80/20 Split)

This experiment follows a common methodology in machine learning literature. The entire dataset, containing all fault types and variations in fault resistance, duration, and start time, is randomly shuffled and split into:

- **Training set (80%):** used to train the machine learning models and perform hyperparameter tuning via 5-fold cross-validation with GridSearchCV.
- **Testing set (20%):** used to evaluate the performance of the trained models on unseen data.

The random split ensures that the training and testing sets have similar distributions of fault types and parameters. This scenario assesses how well the models learn from a representative sample of the overall data distribution.

4.3.3 Experiment 2: Generalization Testing (Unseen Fault Resistances)

This experiment is designed to critically evaluate the generalization capability of the models, particularly their ability to correctly classify faults with characteristics (specifically, fault resistances) that were not encountered during training. This scenario more closely mimics real-world situations where novel fault conditions can occur. The dataset is split as follows:

- **Training set:** comprises all normal operation samples and all fault samples *except* those

with fault resistances of 10 Ω and 100 Ω. This modified training set is then further split internally (e.g., 80/20) for training and validation during hyperparameter tuning.

- **Testing set (unseen resistances):** consists exclusively of fault samples with resistances of 10 Ω and 100 Ω. These specific resistance values are completely held out from the training process.

This setup tests whether the models can interpolate or extrapolate learned patterns to correctly identify faults even when a key parameter like fault resistance falls into a range not explicitly seen during training.

4.4 Results and Analysis

This section presents the performance results of the seven machine learning models for both experimental scenarios. The discussion will focus on comparing model accuracies, F1-scores, and their generalization abilities.

4.4.1 Results of Experiment 1: Standard Evaluation (Random 80/20 Split)

This section presents the detailed performance results for each of the seven models when trained and tested using the standard random 80/20 split. This serves as a baseline representing performance under conditions where the test data distribution closely mirrors the training data.

A - Logistic Regression Results (Experiment 1)

The Logistic Regression model achieved the performance shown in Table 4.1.

Table 4.1: Performance of Logistic Regression (Random 80/20 Split - Exp 1).

Metric	Score
Accuracy	0.7554
Precision (Weighted)	0.7745
Recall (Weighted)	0.7554
F1-Score (Weighted)	0.7422

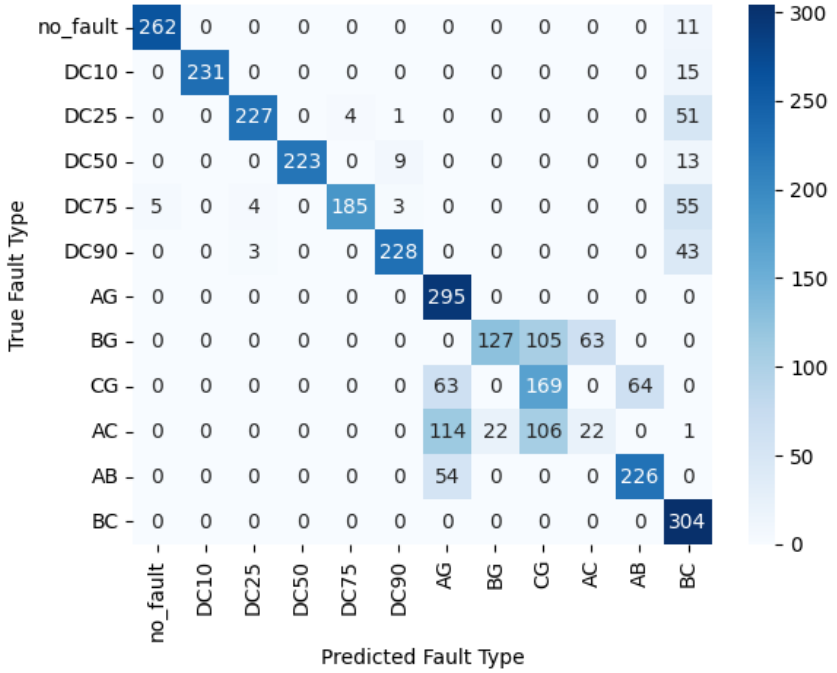

Figure 4.9: Confusion Matrix for Logistic Regression (Random 80/20 Split - Exp 1).

Discussion: Logistic Regression provided a solid baseline performance with an F1-score (weighted) of 0.7422 and an accuracy of 0.7554. However, as seen in Figure 4.9, the model exhibited significant confusion among several fault types. A prominent pattern was the misclassification of various DC fault types (‘DC10‘, ‘DC25‘, ‘DC50‘, ‘DC75‘, ‘DC90‘) as the ‘BC‘ fault type. For instance, ‘DC75‘ was misclassified as ‘BC‘ 55 times, ‘DC25‘ as ‘BC‘ 51 times, and ‘DC90‘ as ‘BC‘ 43 times. Furthermore, there was substantial inter-class confusion among the AC-ground/phase faults, particularly ‘AC‘, ‘AG‘, ‘BG‘, and ‘CG‘. Key examples include ‘AC‘ being misclassified as ‘AG‘ (114 times) and ‘CG‘ (106 times), and ‘BG‘ being misclassified as ‘CG‘ (105 times). The ‘AB‘ fault was also notably confused with ‘AG‘ (54 times), and ‘CG‘ with ‘AB‘ (64 times). Minor confusion was also observed for the ‘no_fault‘ class (e.g., misclassified as ‘BC‘ 11 times) and between different DC fault types (e.g., ‘DC50‘ as ‘DC90‘ 9 times). The performance of this linear model suggests that accurately distinguishing these highly confused fault types likely requires models capable of capturing more complex, non-linear relationships in the feature space.

B - Support Vector Machine (SVC) Results (Experiment 1)

The Support Vector Classifier, using the best hyperparameters found (e.g., likely an RBF kernel), yielded the results in Table 4.2.

Table 4.2: Performance of Support Vector Machine (Random 80/20 Split - Exp 1).

Metric	Score
Accuracy	0.5807
Precision (Weighted)	0.5981
Recall (Weighted)	0.5807
F1-Score (Weighted)	0.5606

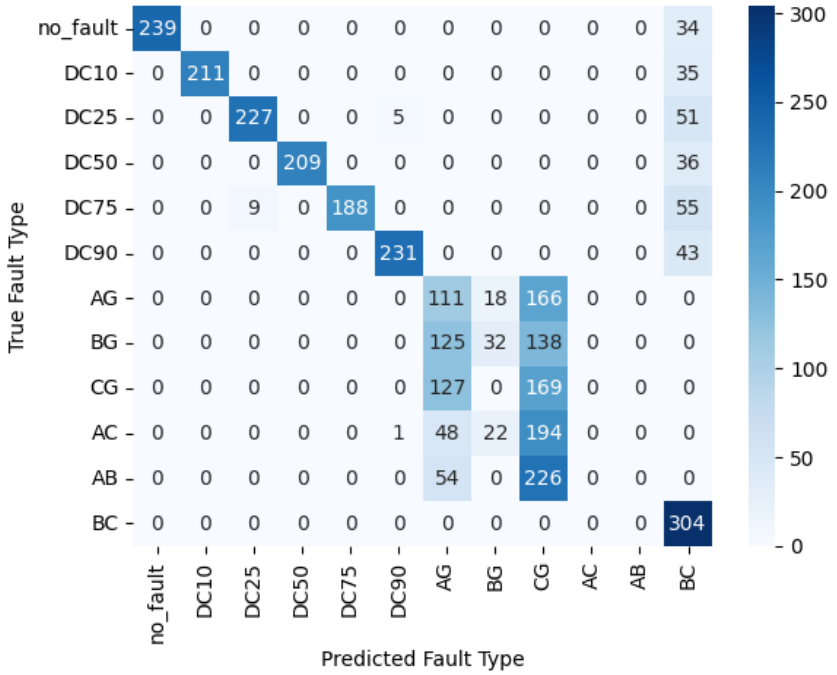

Figure 4.10: Confusion Matrix for Support Vector Machine (Random 80/20 Split - Exp 1).

Discussion: The Support Vector Classifier achieved moderate performance in this standard evaluation, with a weighted F1-score of 0.5606 and an accuracy of 0.5807. An analysis of its misclassifications (visualized in Figure 4.10) reveals several key challenges. A very significant issue is the misclassification of the 'AB' fault type, predominantly into 'CG' (226 instances). Strong confusion also exists with the 'AC' fault type, which is frequently misclassified as 'CG' (194 instances) and 'AG' (48 instances). Similarly, the 'AG' fault is often mistaken for 'CG' (166 instances).

Furthermore, there is considerable inter-confusion among the 'AG', 'BG', and 'CG' classes; for example, 'BG' is often predicted as 'CG' (138 instances) or 'AG' (125 instances), and 'CG' is frequently misclassified as 'AG' (127 instances). Another widespread pattern is the misclassification of 'no_fault' and various DC fault types ('DC10', 'DC25', 'DC50', 'DC75',

'DC90') into the 'BC' class. These systematic misclassifications highlight the SVM's difficulty in effectively separating these specific fault categories under the current experimental setup.

C - K-Nearest Neighbors (KNN) Results (Experiment 1)

The KNN classifier performance, based on the optimal 'k' neighbors found, is presented in Table 4.3.

Table 4.3: Performance of K-Nearest Neighbors (Random 80/20 Split - Exp 1).

Metric	Score
Accuracy	0.9958
Precision (Weighted)	0.9958
Recall (Weighted)	0.9958
F1-Score (Weighted)	0.9958

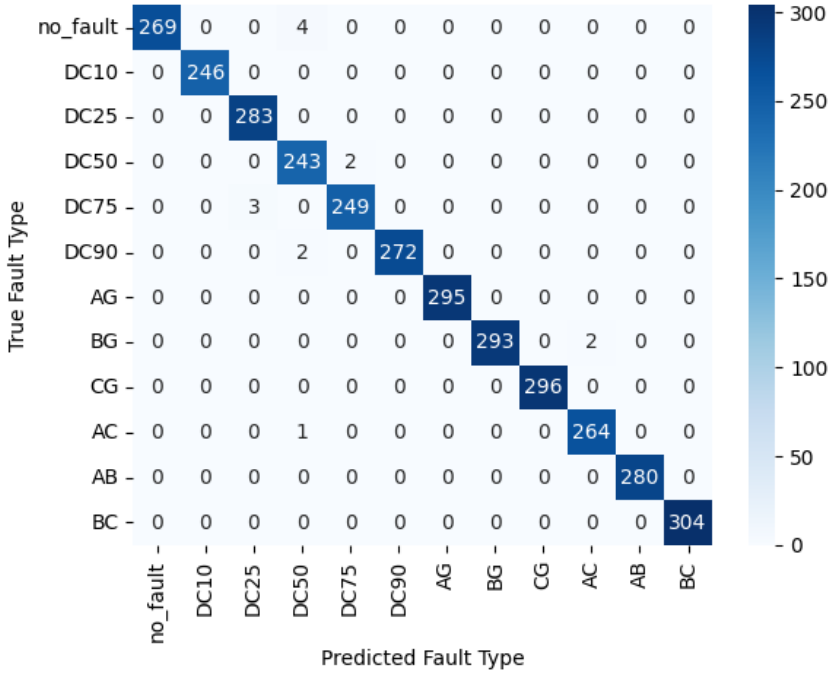

Figure 4.11: Confusion Matrix for K-Nearest Neighbors (Random 80/20 Split - Exp 1).

Discussion: The K-Nearest Neighbors (KNN) classifier achieved exceptionally high performance in the standard random 80/20 split scenario, demonstrating near-perfect classification with a weighted F1-score, accuracy, precision, and recall all at 0.9958. The confusion matrix

(Figure 4.11) visually confirms this, showing that the vast majority of predictions fall along the diagonal, indicating correct classifications. There were only a minimal number of misclassifications. Specifically, 'no_fault' was misclassified as 'DC50' in 4 instances. Other minor errors included 'DC50' being misclassified as 'DC75' (2 times), 'DC75' as 'DC25' (3 times), 'DC90' as 'DC50' (2 times), 'BG' as 'AC' (2 times), and 'AC' as 'DC50' (1 time). The high performance suggests that in this standard split, the different fault types and the 'no_fault' condition are highly separable in the feature space based on proximity. This outcome highlights the effectiveness of KNN's instance-based approach and its ability to model potentially complex decision boundaries when sufficient, similar training data is available close to the test points.

D - Decision Tree Results (Experiment 1)

The Decision Tree classifier, when evaluated on the standard random 80/20 train-test split, produced the performance metrics listed in Table 4.4.

Table 4.4: Performance of Decision Tree (Random 80/20 Split - Exp 1).

Metric	Score
Accuracy	0.5299
Precision (Weighted)	0.5130
Recall (Weighted)	0.5299
F1-Score (Weighted)	0.4684

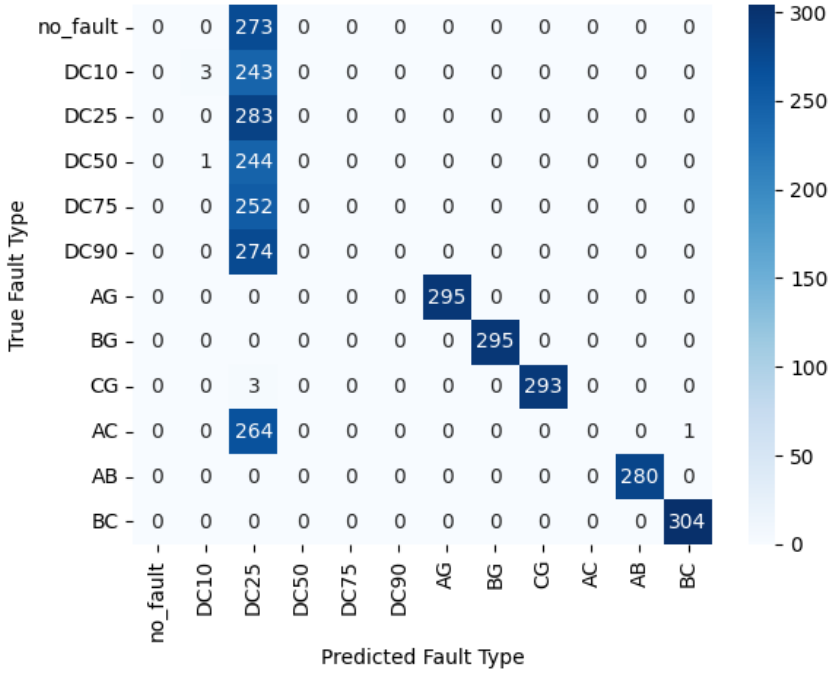

Figure 4.12: Confusion Matrix for Decision Tree (Random 80/20 Split - Exp 1).

Discussion: The Decision Tree classifier exhibited relatively low performance in this standard evaluation, achieving an F1-score (weighted) of 0.4684 and an accuracy of 0.5299. A detailed look at its misclassifications (visualized in Figure 4.12) reveals a significant and dominant pattern: a large number of instances from various classes are overwhelmingly misclassified as ‘DC25‘. Specifically, ‘no_fault‘ (273 times), ‘DC10‘ (243 times), ‘DC50‘ (244 times), ‘DC75‘ (252 times), ‘DC90‘ (274 times), and ‘AC‘ (264 times) were all predominantly misclassified as ‘DC25‘. Other misclassifications, such as ‘CG‘ to ‘DC25‘ (3 times), ‘DC50‘ to ‘DC10‘ (1 time), and ‘AC‘ to ‘BC‘ (1 time), were minor in comparison. This extreme tendency to predict ‘DC25‘ suggests that the learned decision rules are overly simplistic or biased towards features indicative of the ‘DC25‘ fault, leading to poor discrimination between ‘DC25‘ and many other fault types, as well as the ‘no_fault‘ condition. The model struggles significantly to establish effective boundaries for most classes in this experimental setup.

E - Random Forest Results (Experiment 1)

The ensemble Random Forest classifier yielded the performance metrics listed in Table 4.5 when evaluated on a standard random 80/20 train-test split.

Table 4.5: Performance of Random Forest (Random 80/20 Split - Exp 1).

Metric	Score
Accuracy	0.9631
Precision (Weighted)	0.9735
Recall (Weighted)	0.9631
F1-Score (Weighted)	0.9642

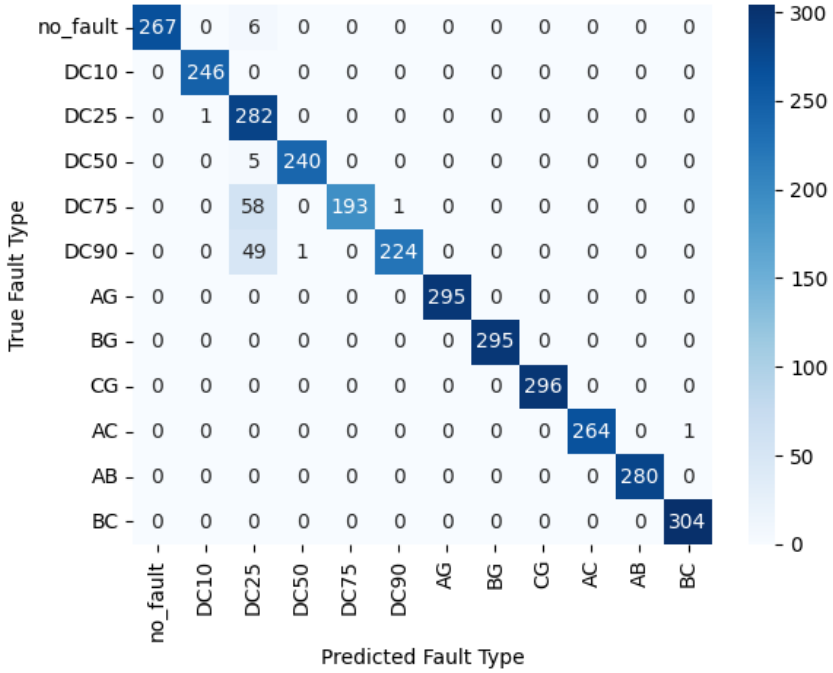

Figure 4.13: Confusion Matrix for Random Forest (Random 80/20 Split - Exp 1).

Discussion: Random Forest demonstrated strong performance in this standard evaluation, achieving a high F1-score (weighted) of 0.9642 and an accuracy of 0.9631. This indicates good overall predictive capability. However, an analysis of its misclassifications (visualized in Figure 4.13) reveals specific challenges. The model notably struggled with distinguishing certain DC fault types, with ‘DC75‘ and ‘DC90‘ faults being frequently misclassified as ‘DC25‘. While most other classes were well-identified, this confusion involving ‘DC25‘ represents the primary area where the model’s discriminative power could be improved in this experimental setup. The ensemble nature of Random Forest generally contributes to robust performance, yet these specific inter-class similarities present a hurdle.

F - Gradient Boosting Results (Experiment 1)

The Gradient Boosting classifier performance on the standard random 80/20 train-test split is summarized in Table 4.6.

Table 4.6: Performance of Gradient Boosting (Random 80/20 Split - Exp 1).

Metric	Score
Accuracy	0.8975
Precision (Weighted)	0.9188
Recall (Weighted)	0.8975
F1-Score (Weighted)	0.8987

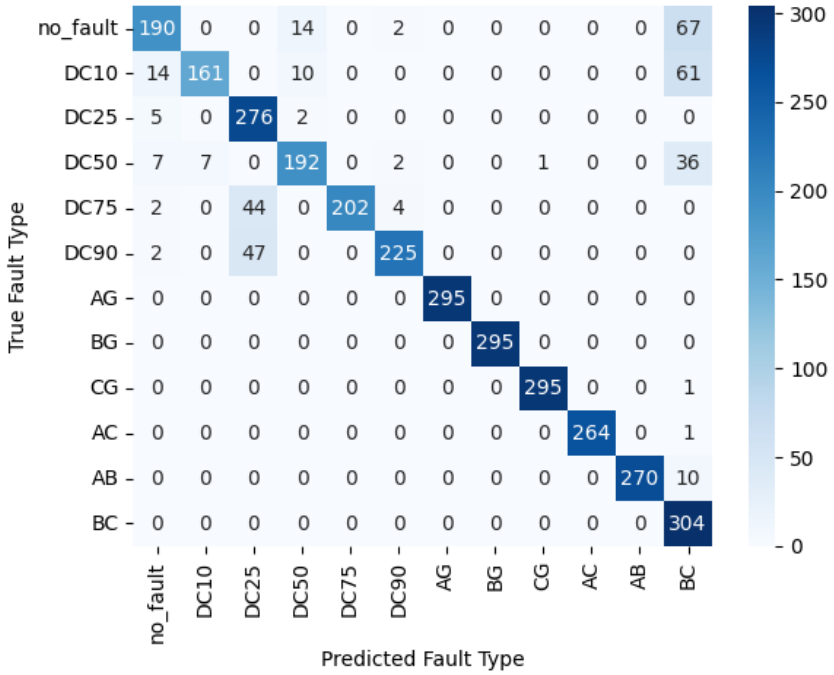

Figure 4.14: Confusion Matrix for Gradient Boosting (Random 80/20 Split - Exp 1).

Discussion: Gradient Boosting delivered a good performance in this standard evaluation, achieving an F1-score (weighted) of 0.8987 and an accuracy of 0.8975. However, the misclassification analysis (visualized in Figure 4.14) highlights specific areas of difficulty. A prominent issue is the misclassification of several fault types into the ‘BC‘ class; notably, ‘no_fault‘ was misclassified as ‘BC‘ 67 times, ‘DC10‘ as ‘BC‘ 61 times, and ‘DC50‘ as ‘BC‘ 36 times. Additionally, there was notable confusion among certain DC fault categories: ‘DC75‘ was frequently misclassified as ‘DC25‘ (44 times), and ‘DC90‘ was also commonly mistaken for ‘DC25‘ (47

times). Other misclassifications, such as ‘no_fault‘ into ‘DC50‘ (14 times) and ‘DC10‘ into ‘no_fault‘ (14 times), were also observed. While Gradient Boosting’s sequential ensemble approach often leads to strong predictive power, these specific inter-class confusions, particularly with the ‘BC‘ class and between certain DC faults, impacted its overall effectiveness in this experiment.

G - Neural Network (MLP) Results (Experiment 1)

The Multi-layer Perceptron (MLP) classifier, when evaluated on the standard random 80/20 train-test split, achieved the performance metrics shown in Table 4.7.

Table 4.7: Performance of Neural Network (MLP) (Random 80/20 Split - Exp 1).

Metric	Score
Accuracy	0.9571
Precision (Weighted)	0.9654
Recall (Weighted)	0.9571
F1-Score (Weighted)	0.9579

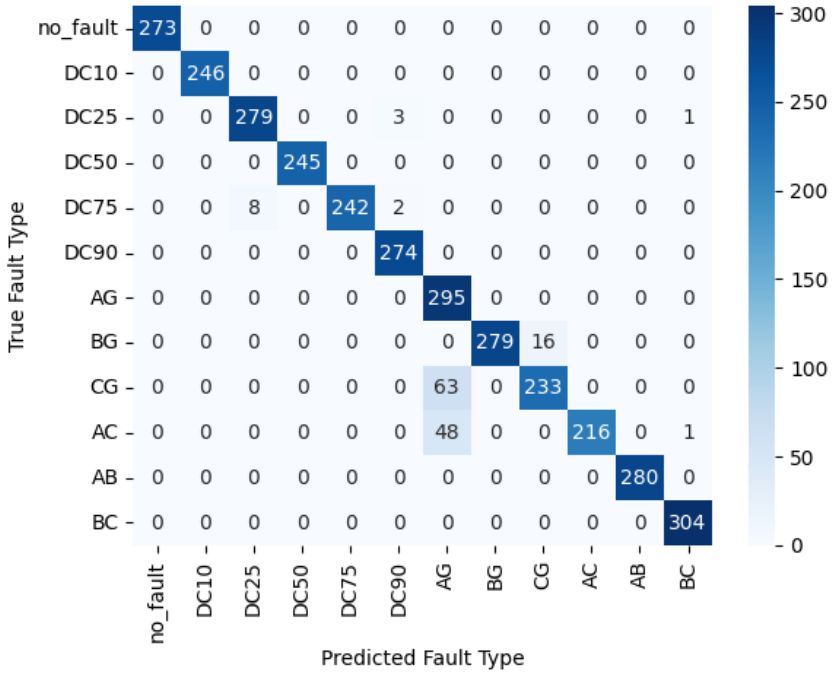

Figure 4.15: Confusion Matrix for Neural Network (MLP) (Random 80/20 Split - Exp 1).

Discussion: The Neural Network (MLP) demonstrated strong performance in this standard evaluation, with a high F1-score (weighted) of 0.9579 and an accuracy of 0.9571. This suggests its capability to learn complex, non-linear relationships within the HVDC fault data. However, an analysis of misclassifications (visualized in Figure 4.15) indicates specific areas of confusion. The most notable issue was the misclassification of 'CG' faults, which were frequently predicted as 'AG' (63 instances). Similarly, 'AC' faults were often misclassified as 'AG' (48 instances), and 'BG' faults were mistaken for 'CG' (16 instances). Confusion among DC faults, such as 'DC75' being misclassified as 'DC25' (8 times), was also observed but to a lesser extent. While the MLP achieved good overall results, these particular inter-class confusions, especially involving the 'AG', 'CG', and 'AC' fault types, represent the primary challenges for the model in this experimental setup.

4.4.2 Summary of Experiment 1 Results

Experiment 1 evaluated the performance of various machine learning models using a standard random 80/20 train-test split. This serves as a baseline to understand how well each model can learn to classify HVC faults when the test data is drawn from the same distribution as the training data. The key performance metrics for all evaluated models are summarized in Table 4.8 and visually compared in Figure 4.16.

Table 4.8: Performance Summary of All Models (Random 80/20 Split - Exp 1). Best values per column are bolded.

Model	Accuracy	Precision (W)	Recall (W)	F1-Score (W)
Logistic Regression	0.7554	0.7745	0.7554	0.7422
Support Vector Machine	0.5807	0.5981	0.5807	0.5606
K-Nearest Neighbors	**0.9958**	**0.9958**	**0.9958**	**0.9958**
Decision Tree	0.5299	0.5130	0.5299	0.4684
Random Forest	0.9631	0.9735	0.9631	0.9642
Gradient Boosting	0.8975	0.9188	0.8975	0.8987
Neural Network (MLP)	0.9571	0.9654	0.9571	0.9579

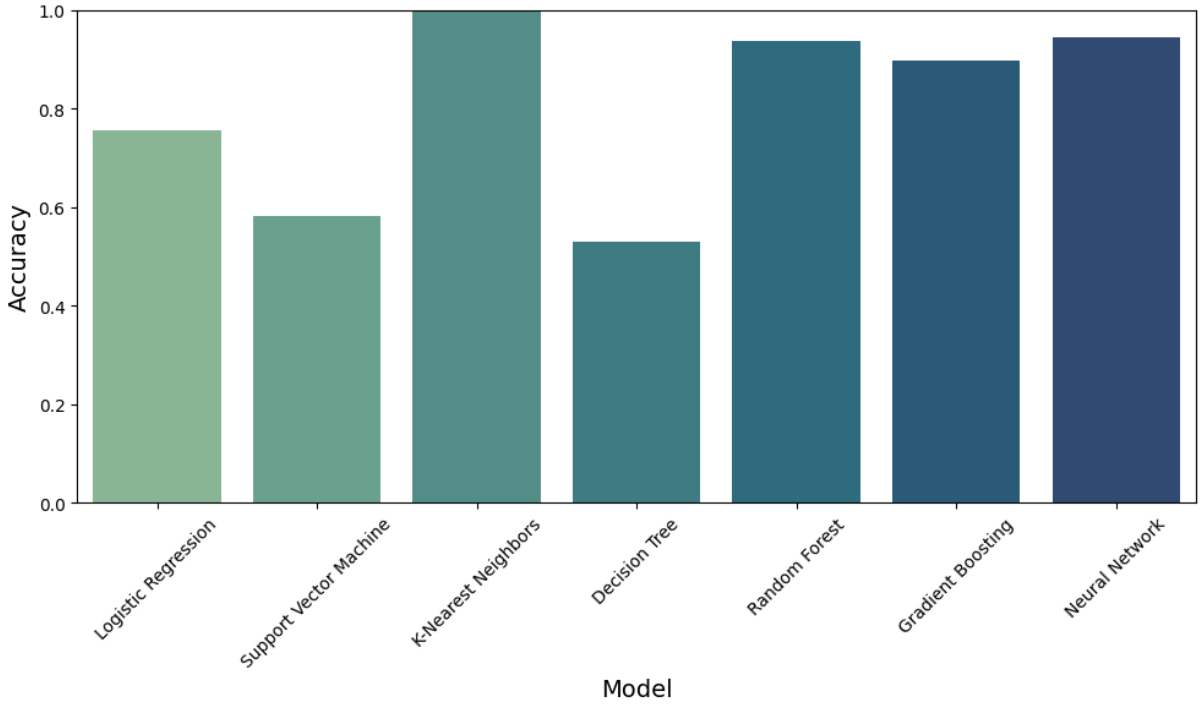

Figure 4.16: Comparison of Accuracy for All Models in Experiment 1 (Random Split).

Summary Discussion: As presented in Table 4.8 and illustrated in Figure 4.16, the models exhibited a wide range of performances in the standard 80/20 random split evaluation. K-Nearest Neighbors (KNN) delivered an outstanding, near-perfect performance, achieving the highest scores across all metrics (F1-score of 0.9958). This suggests that, under these conditions, fault classes are highly separable based on feature proximity. The ensemble methods, Random Forest (F1-score 0.9642) and Neural Network (MLP) (F1-score 0.9579), also demonstrated excellent classification capabilities, achieving high F1-scores indicative of their ability to model complex data patterns effectively. Gradient Boosting (F1-score 0.8987) followed, showing strong performance. Logistic Regression (F1-score 0.7422) offered a respectable baseline performance, though it encountered more significant challenges with inter-class confusions compared to the top-tier models, highlighting the limitations of a linear model for this dataset.

In contrast, the Support Vector Machine (F1-score 0.5606) and particularly the Decision Tree (F1-score 0.4684) showed considerably weaker performance. The Decision Tree's low score was largely due to its tendency to misclassify many instances into a single dominant class ('DC25'), while the SVM struggled with significant confusion between several specific fault categories.

These results establish a crucial baseline, highlighting that several sophisticated models can achieve high accuracy and F1-scores when the training and testing data share a similar

distribution. However, this scenario does not test the models' robustness to variations or novel conditions not seen during training. The true measure of practical applicability will come from assessing generalization performance, particularly in Experiment 2, which focuses on unseen fault resistances.

4.4.3 Results of Experiment 2: Generalization Testing (Unseen Fault Resistances)

This section presents the crucial results from the generalization test. Models were trained on data excluding 10 and 100 fault resistances and then evaluated on samples containing these specific, unseen resistances. This tests the models' ability to extrapolate or interpolate effectively.

A - Logistic Regression Results (Experiment 2)

When tested on unseen fault resistances (10 Ω and 100 Ω), the Logistic Regression model achieved the performance metrics detailed in Table 4.9.

Table 4.9: Performance of Logistic Regression (Unseen Resistances - Exp 2).

Metric	Score
Accuracy	0.8520
Precision (Weighted)	0.8066
Recall (Weighted)	0.8520
F1-Score (Weighted)	0.8130

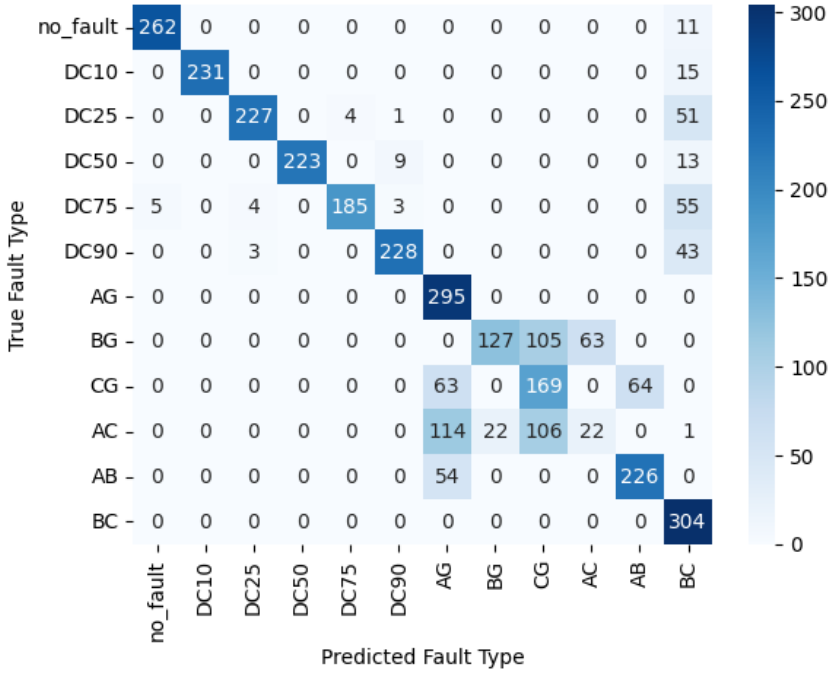

Figure 4.17: Confusion Matrix for Logistic Regression (Unseen Resistances - Exp 2).

Discussion: In the generalization test on unseen fault resistances, Logistic Regression achieved an F1-score (weighted) of 0.8130 and an accuracy of 0.8520. Despite this overall score, the confusion matrix (Figure 4.17) reveals significant challenges in generalizing to these new conditions. Most notably, there was extensive misclassification involving the ‘CG‘ fault type, which was frequently confused with ‘BG‘ (152 times) and ‘AB‘ (143 times). Additionally, the ‘AC‘ fault type was commonly misclassified as ‘AG‘ (140 times). While some DC fault confusions were present (e.g., ‘DC25‘ as ‘DC90‘ 21 times), they were less prominent than the AC/ground/phase fault misclassifications. This indicates that the linear boundaries learned by Logistic Regression did not adequately generalize to the feature variations introduced by the unseen resistances, particularly for these complex fault types.

B - Support Vector Machine (SVC) Results (Experiment 2)

The SVC’s performance when tested on the unseen fault resistances (10 Ω and 100 Ω) is detailed in Table 4.10.

Table 4.10: Performance of Support Vector Machine (Unseen Resistances - Exp 2).

Metric	Score
Accuracy	0.6037
Precision (Weighted)	0.5976
Recall (Weighted)	0.6037
F1-Score (Weighted)	0.5728

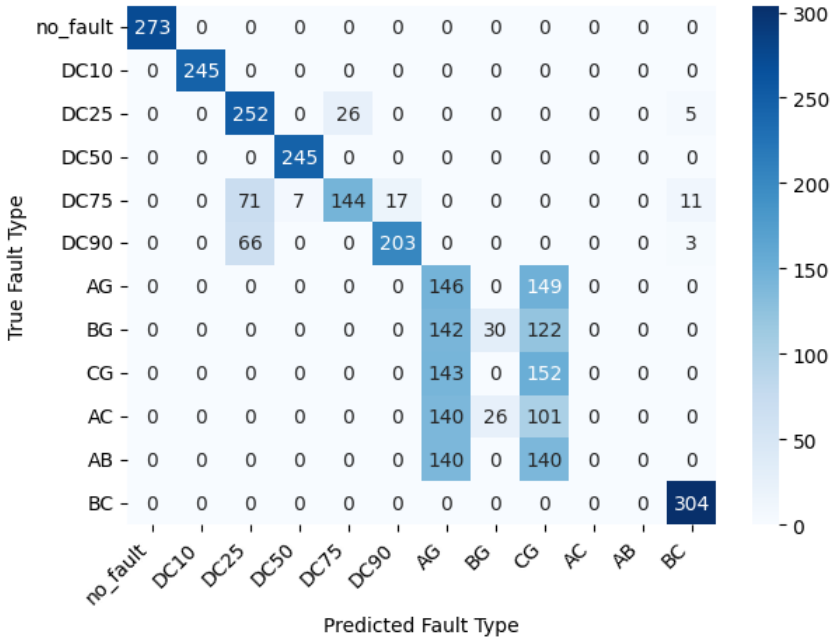

Figure 4.18: Confusion Matrix for Support Vector Machine (Unseen Resistances - Exp 2).

Discussion: In the generalization test on unseen fault resistances, the Support Vector Machine achieved an F1-score (weighted) of 0.5728 and an accuracy of 0.6037. The confusion matrix (Figure 4.18) indicates significant difficulties in generalizing to these new conditions. There was widespread misclassification among the AC-ground/phase faults, with very high instances of confusion such as ‘AG‘ being misclassified as ‘CG‘ (149 times), ‘BG‘ as ‘AG‘ (142 times) or ‘CG‘ (122 times), ‘CG‘ as ‘AG‘ (143 times), ‘AC‘ as ‘AG‘ (140 times) or ‘CG‘ (101 times), and ‘AB‘ as ‘AG‘ or ‘CG‘ (140 times each). Additionally, significant confusion persisted within DC fault types, for example, ‘DC75‘ was frequently misclassified as ‘DC25‘ (71 times) and ‘DC90‘ as ‘DC25‘ (66 times). These results suggest that the decision boundaries learned by the SVM did not adapt well to the feature variations introduced by the unseen fault resistances, leading to poor discrimination across many classes.

C - K-Nearest Neighbors (KNN) Results (Experiment 2)

KNN's performance when classifying faults with unseen resistances (10 Ω and 100 Ω) is given in Table 4.11.

Table 4.11: Performance of K-Nearest Neighbors (Unseen Resistances - Exp 2).

Metric	Score
Accuracy	0.7078
Precision (Weighted)	0.7466
Recall (Weighted)	0.7078
F1-Score (Weighted)	0.6816

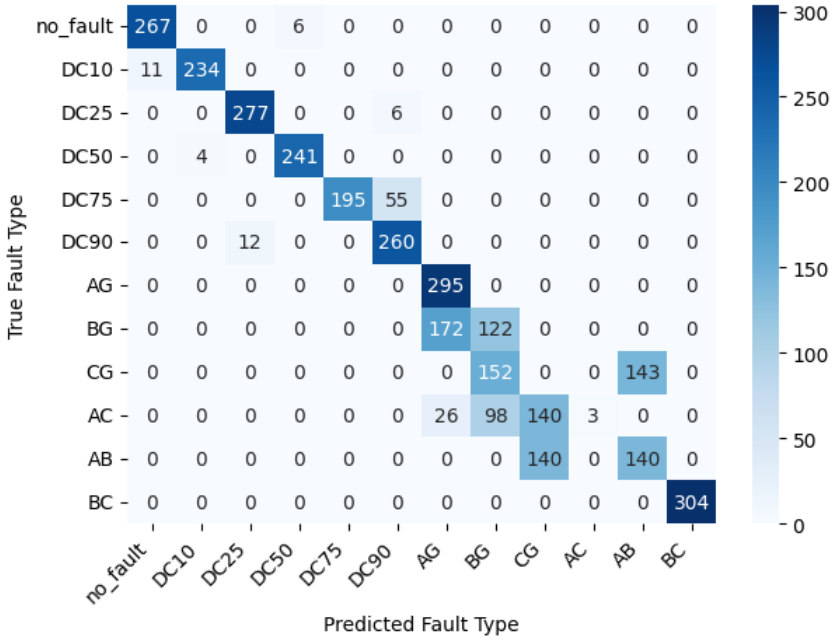

Figure 4.19: Confusion Matrix for K-Nearest Neighbors (Unseen Resistances - Exp 2).

Discussion: When faced with unseen fault resistances, the K-Nearest Neighbors classifier achieved an F1-score (weighted) of 0.6816 and an accuracy of 0.7078. This represents a notable decrease from its near-perfect performance in Experiment 1. The confusion matrix (Figure 4.19) reveals significant generalization challenges. Prominent misclassifications included 'BG' faults being mistaken for 'AG' (172 times), 'CG' for 'BG' (152 times) or 'AB' (143 times), 'AC' for 'CG' (140 times) or 'BG' (98 times), and 'AB' for 'CG' (140 times). Additionally, 'DC75' was frequently misclassified as 'DC90' (55 times). KNN's reliance on local proximity makes it sensitive to shifts in the feature space caused by unseen conditions. The feature values for

the unseen resistances likely placed these instances in neighborhoods dominated by incorrect classes from the training data, leading to these widespread misclassifications.

D - Decision Tree Results (Experiment 2)

The single Decision Tree's generalization performance when tested on unseen fault resistances (10 Ω and 100 Ω) is shown in Table 4.12.

Table 4.12: Performance of Decision Tree (Unseen Resistances - Exp 2).

Metric	Score
Accuracy	0.5301
Precision (Weighted)	0.4577
Recall (Weighted)	0.5301
F1-Score (Weighted)	0.4673

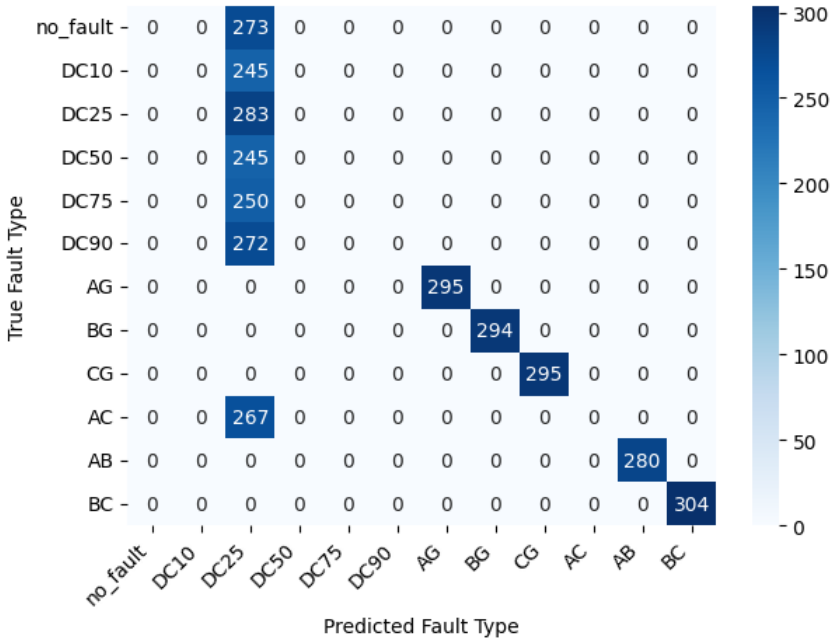

Figure 4.20: Confusion Matrix for Decision Tree (Unseen Resistances - Exp 2).

Discussion: The Decision Tree demonstrated poor generalization to unseen fault resistances, achieving a low F1-score (weighted) of 0.4673 and an accuracy of 0.5301. The confusion matrix (Figure 4.20) reveals a severe and dominant misclassification pattern: a vast majority of instances from various classes, including 'no_fault' (273 times), 'DC10' (245 times), 'DC50'

(245 times), ‘DC75‘ (250 times), ‘DC90‘ (272 times), and ‘AC‘ (267 times), were overwhelmingly misclassified as ‘DC25‘. This indicates that the specific decision rules and feature thresholds learned by the tree during training were not robust to the feature variations introduced by the unseen resistances. The model failed to discriminate effectively, collapsing many distinct fault types into a single category (‘DC25‘) when faced with these new conditions.

E - Random Forest Results (Experiment 2)

Random Forest’s performance on the unseen fault resistances (10 Ω and 100 Ω) test set is presented in Table 4.13.

Table 4.13: Performance of Random Forest (Unseen Resistances - Exp 2).

Metric	Score
Accuracy	0.9682
Precision (Weighted)	0.9721
Recall (Weighted)	0.9682
F1-Score (Weighted)	0.9672

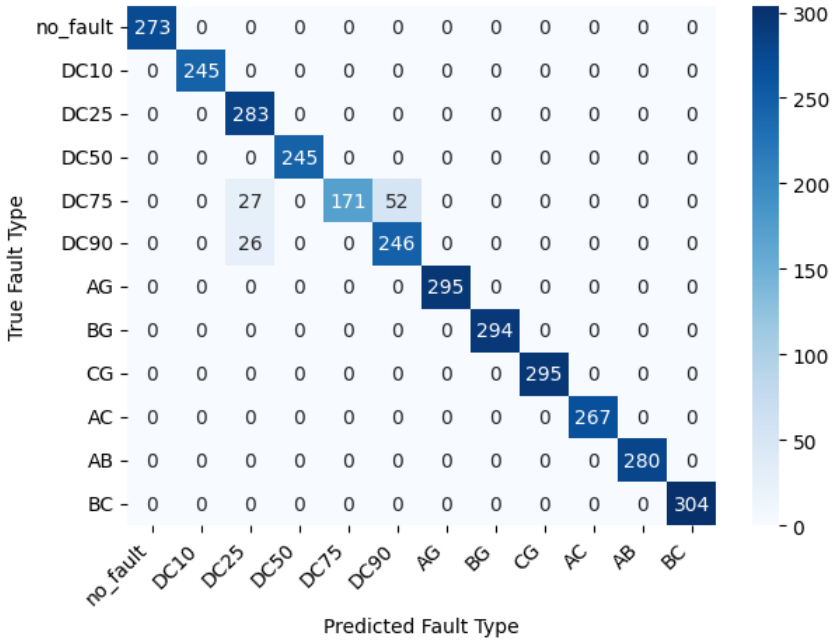

Figure 4.21: Confusion Matrix for Random Forest (Unseen Resistances - Exp 2).

Discussion: Random Forest demonstrated excellent generalization capabilities when tested on unseen fault resistances, achieving a high F1-score (weighted) of 0.9672 and an accuracy of

0.9682. Remarkably, these scores are comparable to, and even slightly better than, its performance in Experiment 1, indicating strong robustness. The ensemble nature of Random Forest likely contributed significantly to this robust generalization. An analysis of misclassifications (Figure 4.21) shows that the errors were minimal and primarily confined to confusion between specific DC fault types: ‘DC75‘ was sometimes misclassified as ‘DC25‘ (27 times) or ‘DC90‘ (52 times), and ‘DC90‘ as ‘DC25‘ (26 times). Overall, Random Forest proved highly effective at classifying fault types even under novel resistance conditions not encountered during training.

F - Gradient Boosting Results (Experiment 2)

Gradient Boosting’s generalization performance when tested on unseen fault resistances (10 Ω and 100 Ω) is summarized in Table 4.14.

Table 4.14: Performance of Gradient Boosting (Unseen Resistances - Exp 2).

Metric	Score
Accuracy	0.9173
Precision (Weighted)	0.9523
Recall (Weighted)	0.9173
F1-Score (Weighted)	0.9205

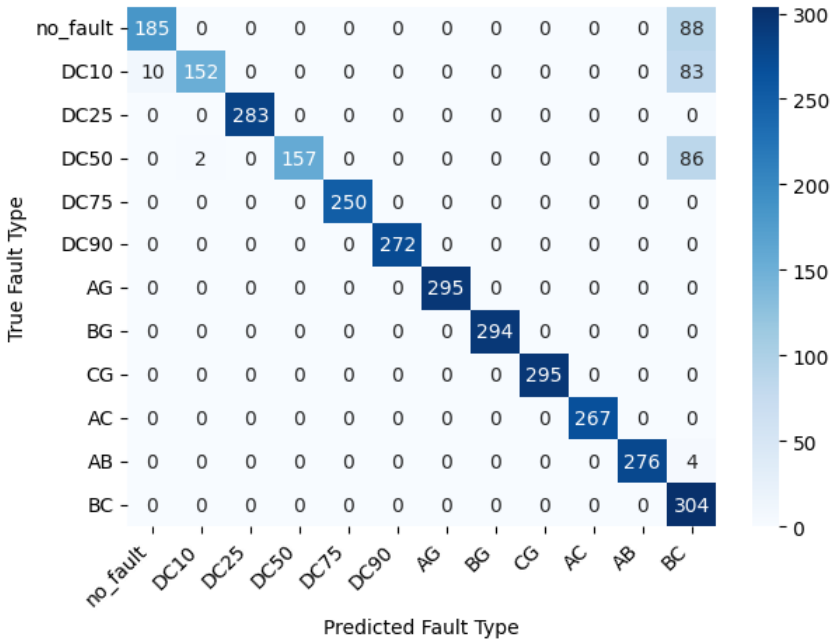

Figure 4.22: Confusion Matrix for Gradient Boosting (Unseen Resistances - Exp 2).

Discussion: Gradient Boosting demonstrated good generalization capabilities, achieving an F1-score (weighted) of 0.9205 and an accuracy of 0.9173 when evaluated on unseen fault resistances. While the overall performance was strong, an analysis of misclassifications (Figure 4.22) indicates that the primary challenge involved the ‘BC‘ fault type. Specifically, ‘no_fault‘ was misclassified as ‘BC‘ 88 times, ‘DC10‘ as ‘BC‘ 83 times, and ‘DC50‘ as ‘BC‘ 86 times. Other misclassifications, such as ‘DC10‘ to ‘no_fault‘ (10 times) and ‘AB‘ to ‘BC‘ (4 times), were less frequent. The sequential nature of Gradient Boosting, which aims to correct errors of previous learners, likely contributed to its solid generalization, though it still struggled to perfectly distinguish certain classes from ‘BC‘ under these novel conditions.

G - Neural Network (MLP) Results (Experiment 2)

The MLP’s performance when tested on unseen fault resistances (10 Ω and 100 Ω) is shown in Table 4.15.

Table 4.15: Performance of Neural Network (MLP) (Unseen Resistances - Exp 2).

Metric	Score
Accuracy	0.9316
Precision (Weighted)	0.9343
Recall (Weighted)	0.9316
F1-Score (Weighted)	0.9306

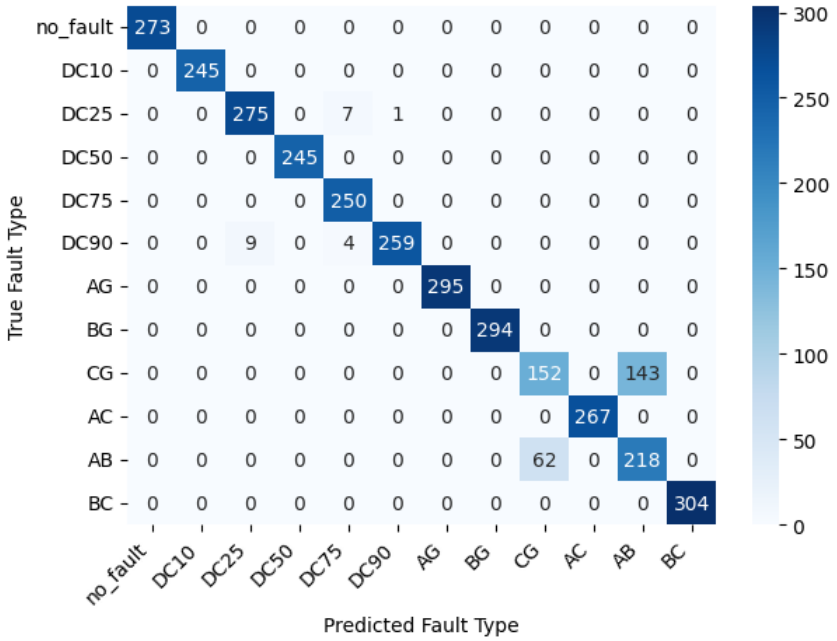

Figure 4.23: Confusion Matrix for Neural Network (MLP) (Unseen Resistances - Exp 2).

Discussion: The Neural Network (MLP) demonstrated strong generalization to unseen fault resistances, achieving a high F1-score (weighted) of 0.9306 and an accuracy of 0.9316. While the overall performance was robust, an analysis of misclassifications (Figure 4.23) highlighted a specific area of difficulty: the ‘CG‘ fault type was frequently misclassified as ‘AB‘ (143 times), and conversely, ‘AB‘ was often misclassified as ‘CG‘ (62 times). Misclassifications among DC faults (e.g., ‘DC90‘ as ‘DC25‘ 9 times) were comparatively minor. The MLP’s ability to learn complex feature representations likely enabled it to adapt well to most variations from the unseen resistances, though the similarity between ‘CG‘ and ‘AB‘ faults under these new conditions posed a notable challenge.

4.4.4 Summary of Experiment 2 Results

Experiment 2 was designed to rigorously test the models’ generalization capabilities by evaluating them on fault data with resistances (10 Ω and 100 Ω) that were entirely excluded from the training set. This scenario mimics a more realistic deployment where models encounter conditions not perfectly represented in their initial training. The performance of each model under these challenging conditions is summarized in Table 4.16 and visually compared in Figure 4.24.

Table 4.16: Performance Summary of All Models (Unseen Resistances - Exp 2). Best values per column are bolded.

Model	Accuracy	Precision (W)	Recall (W)	F1-Score (W)
Logistic Regression	0.8520	0.8066	0.8520	0.8130
Support Vector Machine	0.6037	0.5976	0.6037	0.5728
K-Nearest Neighbors	0.7078	0.7466	0.7078	0.6816
Decision Tree	0.5301	0.4577	0.5301	0.4673
Random Forest	**0.9682**	**0.9721**	**0.9682**	**0.9672**
Gradient Boosting	0.9173	0.9523	0.9173	0.9205
Neural Network (MLP)	0.9316	0.9343	0.9316	0.9306

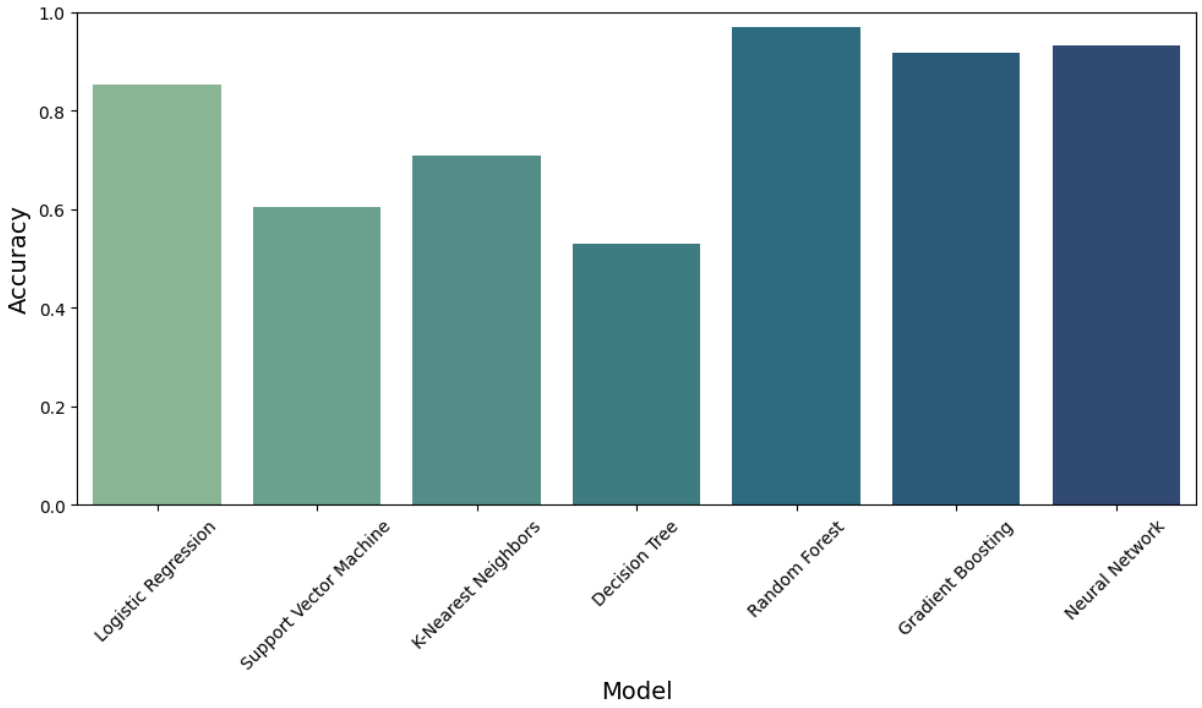

Figure 4.24: Comparison of Accuracy for All Models in Experiment 2 (Unseen Resistances).

Summary Discussion: Experiment 2 provided critical insights into the models' ability to generalize to novel fault conditions, as shown in Table 4.16 and Figure 4.24. The results clearly differentiated the models based on their robustness. Random Forest emerged as the top-performing model, achieving an outstanding F1-score of 0.9672, remarkably maintaining its high performance from Experiment 1. This highlights its excellent generalization capabilities. The Neural Network (MLP) (F1-score 0.9306) and Gradient Boosting (F1-score 0.9205) also demonstrated strong robustness, yielding high F1-scores that indicated a good ability to adapt to the feature variations introduced by the unseen resistances.

Logistic Regression (F1-score 0.8130) showed a notable improvement compared to its Experiment 1 performance, suggesting that while it struggled with the overall complexity of the full dataset in Experiment 1, it found more generalizable (albeit still imperfect) linear boundaries that were coincidentally effective for the specific subset of unseen resistances, albeit with some significant misclassifications remaining.

In contrast, K-Nearest Neighbors (F1-score 0.6816) experienced a substantial performance degradation compared to its near-perfect score in Experiment 1. Its instance-based nature proved sensitive to the feature shifts. The Support Vector Machine (F1-score 0.5728) and the single Decision Tree (F1-score 0.4673) performed poorly, struggling significantly with the unseen conditions and exhibiting widespread misclassifications. The Decision Tree, in particular, continued to collapse many classes into one, underscoring its lack of generalization.

This experiment underscores the importance of robust model architectures, such as ensembles (Random Forest, Gradient Boosting) and potentially well-tuned Neural Networks, when dealing with real-world scenarios where operational conditions might deviate from the training data. Simple models or those overly sensitive to local data structure may not generalize effectively.

4.5 Comparative Analysis and Discussion

This section synthesizes the findings from both Experiment 1 (Standard Evaluation) and Experiment 2 (Generalization Testing) to provide a holistic comparison of the models' performance and, critically, their generalization capabilities when faced with unseen fault resistances. The aim is to identify models that are not only accurate on data similar to training but also robust to novel conditions.

Figure 4.25 directly compares the Accuracy for each model across both experiments. This visualization is crucial for understanding how each model's performance changes when transitioning from a standard evaluation scenario to a more challenging generalization test.

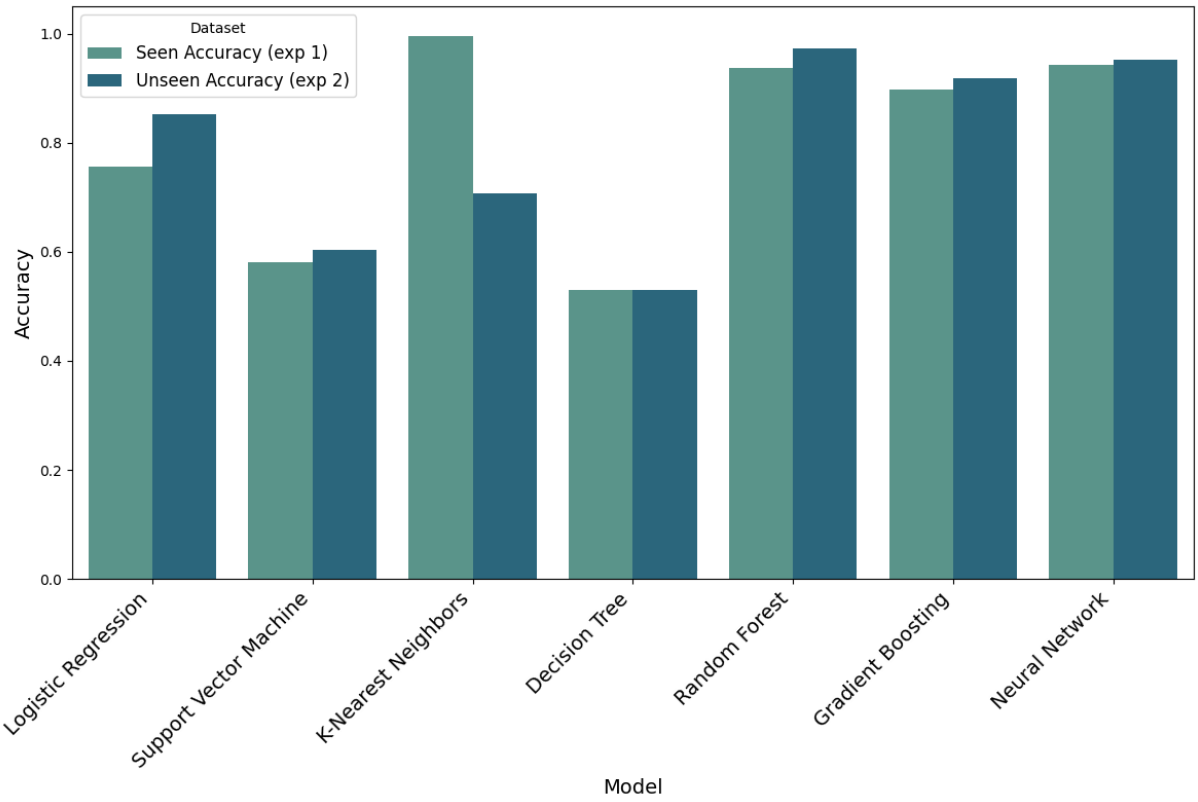

Figure 4.25: Comparison of Models Accuracy: Experiment 1 (Random 80/20 Split) vs. Experiment 2 (Unseen Fault Resistances of 10 Ω and 100 Ω).

Overall Discussion: The comparative analysis starkly illustrates the difference between standard evaluation and generalization testing, which is a central theme of this thesis. While several models, notably K-Nearest Neighbors (F1-score 0.9958), Random Forest (0.9642), and Neural Network (MLP) (0.9579), achieved high to near-perfect scores on the random 80/20 split (Experiment 1), their reliability and performance under the novel conditions of unseen fault resistances (Experiment 2) varied considerably.

The results from Experiment 2 are particularly revealing. Random Forest emerged as the most robust model, not only performing exceptionally well in Experiment 1 (F1-score 0.9642) but also maintaining, and even slightly improving, this high level of performance in Experiment 2 (F1-score 0.9672) when faced with unseen resistances. This demonstrates its excellent generalization capability. The Neural Network (MLP) also showed strong generalization, with a high F1-score of 0.9579 in Experiment 1 and a resilient 0.9306 in Experiment 2. Gradient Boosting similarly proved robust, with F1-scores of 0.8987 and 0.9205 in Experiment 1 and 2 respectively. The ability of these ensemble methods (Random Forest, Gradient Boosting) to average out biases and reduce variance, and the capacity of Neural Networks to learn complex, robust feature representations, likely contribute to their superior generalization.

Conversely, K-Nearest Neighbors, despite its stellar performance in Experiment 1, experienced a dramatic drop in Experiment 2 (F1-score from 0.9958 to 0.6816). This highlights KNN's sensitivity to shifts in the feature space; its reliance on local proximity means that when test instances with unseen characteristics fall into "neighborhoods" dominated by incorrect classes from the training data (which did not include those specific resistances), its performance degrades significantly. Logistic Regression showed an interesting trend: its F1-score improved from 0.7422 in Experiment 1 to 0.8130 in Experiment 2. This suggests that while it struggled with the overall complexity of the full dataset in Experiment 1, it might have learned more generalizable (albeit still imperfect) linear boundaries that were coincidentally effective for the specific subset of unseen 10 Ω and 100 Ω fault resistances. However, its confusion matrices still indicated notable misclassifications. The Support Vector Machine also showed a slight increase (F1-score from 0.5606 to 0.5728) but remained a relatively poor performer in both scenarios, struggling with significant inter-class confusion. The single Decision Tree consistently performed poorly (F1-scores of 0.4684 and 0.4673), often collapsing many classes into one, underscoring its tendency to overfit to specific paths in the training data and its lack of robustness.

These findings directly support the research objective of evaluating generalization and addressing the gap in literature that often relies solely on random splits. This study clearly shows that random splits can overestimate real-world performance. The necessity of targeted generalization testing, such as evaluating on unseen fault parameters, is paramount for developing reliable ML-based fault diagnosis systems.

In terms of trade-offs, the best generalizing models (Random Forest, Gradient Boosting, MLP) are generally more complex and computationally intensive to train than simpler models like Logistic Regression or a single Decision Tree. However, their superior robustness often justifies this additional upfront cost, especially for critical infrastructure like HVDC systems.

It is important to acknowledge limitations: this study focused on generalization to unseen fault resistances. Further research could explore generalization to other unseen parameters like fault location, duration, or different system operating conditions. Moreover, the results are based on simulation data, and validation on real-world HVDC system data would be an essential next step.

The comparative analysis strongly suggests that ensemble methods like Random Forest and Gradient Boosting, along with well-tuned Neural Networks, offer a promising balance of high accuracy and superior robustness for HVDC fault detection when faced with variations in fault parameters not explicitly seen during training.

4.6 Computational Considerations

While accuracy and generalization were the primary evaluation criteria in this study, computational load is a relevant factor for the practical implementation and deployment of machine learning models in HVDC fault diagnosis systems.

Training times varied significantly across the evaluated models, particularly when considering the hyperparameter optimization phase using `GridSearchCV` with 5-fold cross-validation. Simpler models like Logistic Regression and a single Decision Tree were the fastest to train, often completing in minutes. K-Nearest Neighbors has a negligible explicit training phase (as it is an instance-based learner), but its prediction time can be high for large datasets as it needs to compute distances to all training points (though optimized implementations can mitigate this). Support Vector Machine training time, especially with non-linear kernels like RBF, can scale non-linearly with the size of the dataset, making it more time-consuming for larger datasets.

The ensemble methods (Random Forest and Gradient Boosting) and the Neural Network (MLP) were the most computationally intensive to train. Building multiple trees (for RF and GB) or training a network through multiple epochs with backpropagation, combined with an exhaustive `GridSearchCV` over a range of hyperparameters, required substantial computational resources and time, sometimes spanning several hours depending on the search space and dataset size.

However, once the models were trained and optimized, their prediction times for classifying new, unseen fault instances were generally very low, typically in the range of milliseconds or even sub-milliseconds per instance on modern hardware. This rapid prediction capability is crucial for HVDC fault detection systems, which require swift responses to mitigate potential damage and ensure system stability.

The initial data generation phase, which involved simulating numerous fault scenarios in MATLAB/Simulink, was also computationally intensive. The use of MATLAB's Parallel Computing Toolbox ('parsim' command) significantly accelerated this phase by distributing the simulations across multiple processor cores, making it feasible to generate the comprehensive dataset required for this study in a reasonable timeframe.

In summary, while some of the best-performing and most robust models require a significant upfront investment in terms of training time and computational resources for hyperparameter tuning, their fast prediction speeds make them viable for real-time or near real-time fault diag-

nosis applications.

4.7 Conclusion

This chapter presented the machine learning framework and detailed results analysis for HVDC fault detection and classification. The core of the experimental work focused on two evaluation scenarios: Experiment 1 utilized a standard random 80/20 train-test split for baseline performance assessment, while Experiment 2 critically evaluated model generalization by testing on fault resistances (10 Ω and 100 Ω) entirely excluded from the training data. Data was derived from simulations (as detailed in Chapter 3), involving statistical feature extraction and scaling, with all resources made openly available.

Seven classifiers (Logistic Regression, Support Vector Machine, K-Nearest Neighbors, Decision Tree, Random Forest, Gradient Boosting, and a Neural Network (MLP)) were tuned using `GridSearchCV` and evaluated using metrics like the weighted F1-score.

Experiment 1 showed high baseline F1-scores, particularly for KNN (0.9958), Random Forest (0.9642), and MLP (0.9579). However, Experiment 2, the generalization test, revealed significant performance differences. Random Forest demonstrated excellent robustness, maintaining a high F1-score (0.9672). The MLP (0.9306) and Gradient Boosting (0.9205) also generalized well. In contrast, KNN's performance substantially decreased (F1-score 0.6816), while Decision Tree and SVM struggled with the unseen conditions.

These findings underscore that standard random splits can overestimate practical model performance, highlighting the necessity of rigorous generalization testing. Ensemble methods (Random Forest, Gradient Boosting) and Neural Networks showed the most promise for developing robust and accurate HVDC fault diagnosis systems capable of handling novel conditions. Computational aspects were also briefly considered. The insights gained here pave the way for the thesis's concluding discussions.

Conclusion and Future Work

This thesis embarked on an investigation into the application of machine learning techniques for fault diagnosis in high voltage direct current (HVDC) systems. The primary motivation stemmed from the critical need for reliable and rapid fault detection to ensure the stability and security of modern power grids, which increasingly rely on HVDC technology. A significant emphasis was placed on evaluating not just the standard performance of machine learning models but, more importantly, their generalization capabilities when faced with fault conditions not explicitly encountered during training—a crucial aspect for real-world deployment.

The work began with an extensive review of existing literature on HVDC systems, fault types, conventional diagnostic methods, and prior applications of machine learning. This review identified key gaps, particularly the common reliance on simple random data splits for model evaluation, which may not adequately reflect a model's ability to generalize to truly novel fault scenarios. The problem was thus defined as developing and rigorously evaluating machine learning models with a specific focus on their generalization to unseen fault parameters.

A detailed 12-pulse line commutated converter (LCC) HVDC transmission system was modeled in MATLAB/Simulink. This model served as the foundation for simulating a wide array of fault types, including DC line-to-ground faults at various locations, and AC side faults (phase-to-ground and phase-to-phase). The characteristics of these faults and their impact on system signals were analyzed. To generate a comprehensive dataset, numerous fault scenarios were scripted with variations in fault type, location, resistance, and duration. An automated framework, combining MATLAB for simulation and Python for orchestration of parallel simulations, was implemented. Key electrical signals were captured. The generated dataset and simulation scripts were made publicly available to promote reproducibility. Data preprocessing steps, including cleaning and standardization, were applied.

Seven machine learning classifiers (logistic regression, support vector machine, k-nearest neighbors, decision tree, random forest, gradient boosting, and a multi-layer perceptron neural network) were selected from the scikit-learn library. Hyperparameters for each model were tuned using grid search with cross-validation. Two distinct experimental evaluation scenarios were designed: a standard evaluation using a random 80/20 split, and a generalization test where models were trained on data excluding specific fault resistances and then evaluated exclusively on these unseen resistance values. Performance was assessed using accuracy, precision, recall, and F1-score, with detailed analysis of confusion matrices and comparative performance.

The results demonstrated that performance on standard random train-test splits can be overly

optimistic and does not reliably predict performance on novel fault conditions. For instance, k-nearest neighbors achieved near-perfect scores in the standard evaluation but suffered a significant performance drop in the generalization test. Random forest emerged as the most robust model, exhibiting excellent performance in both scenarios, while the neural network and gradient boosting also demonstrated strong generalization capabilities. This suggests that ensemble techniques and appropriately tuned neural networks are well-suited for handling the variability inherent in real-world fault scenarios. The study provides a comprehensive benchmark of seven different machine learning algorithms on a consistently generated and processed dataset for HVDC fault diagnosis, offering valuable insights into the relative strengths and weaknesses of these models for this specific application.

Several challenges and limitations were encountered during the research. The entire study was based on data generated from MATLAB/Simulink simulations, which may not fully capture all the complexities, noise, uncertainties, and unmodeled dynamics present in real-world HVDC operational data. The generalization testing focused specifically on unseen fault resistances, while real-world novelty can also arise from unseen fault locations, variations in fault inception angle, or evolving system operating conditions not explicitly covered in the generalization test. The study utilized a set of common statistical features; while effective, other advanced feature engineering techniques or features derived from deeper domain expertise might provide additional discriminatory power. The hyperparameter optimization process for the more complex models was computationally intensive and time-consuming. The study focused on a specific type of HVDC system (12-pulse LCC), so the findings might not directly translate to other HVDC technologies without further investigation.

Based on the findings and limitations of this study, several promising directions for future research can be identified. The most critical next step is to validate the developed models and the generalization testing methodology using data from actual HVDC installations. Future work should broaden the scope of generalization testing to include other types of unseen variations, such as faults at novel locations, variations in system operating conditions, and robustness to measurement noise. More sophisticated signal processing techniques for feature extraction and the application of deep learning models directly on raw time-series data could be explored to enable automatic feature learning. Applying explainable AI techniques to the best-performing complex models could provide insights into their decision-making processes and enhance model trustworthiness. Developing frameworks for online learning where models can adapt to evolving system conditions or new fault signatures, and addressing data imbalance and rare faults, are also important avenues. Extending the methodologies to multi-terminal HVDC systems and exploring hardware implementation for real-time performance would further enhance the practical applicability of the research. Additionally, simulating and testing on different HVDC configurations such as bipolar and homopolar links would help evaluate and improve the adaptability

and robustness of the developed models across various HVDC topologies.

In summary, this thesis successfully developed and evaluated a range of machine learning models for HVDC fault diagnosis, with a critical emphasis on their generalization capabilities. The findings clearly demonstrate that while many models can achieve high accuracy on data similar to their training set, their performance can vary significantly when faced with novel fault conditions. Ensemble methods, particularly random forest, and neural networks showed superior robustness and adaptability, making them promising candidates for practical deployment. The research underscored the importance of moving beyond simplistic random-split evaluations towards more rigorous generalization testing methodologies to build truly reliable ML-based diagnostic systems. The open-sourcing of the simulation tools and datasets aims to contribute to a more collaborative and reproducible research environment in this domain. While limitations exist, the insights gained and the proposed avenues for future work pave the way for continued advancements in developing intelligent, robust, and reliable fault diagnosis solutions for critical HVDC infrastructure, ultimately contributing to the resilience and efficiency of future power systems.

Bibliography

[1] D. Van Hertem and M. Ghandhari, "Multi-terminal VSC HVDC for the European supergrid: Obstacles," vol. 14, no. 9, pp. 3156–3163.

[2] N. Flourentzou, V. Agelidis, and G. Demetriades, "VSC-Based HVDC Power Transmission Systems: An Overview," vol. 24, no. 3, pp. 592–602.

[3] A. Pragati, M. Mishra, P. K. Rout, D. A. Gadanayak, S. Hasan, and B. R. Prusty, "A Comprehensive Survey of HVDC Protection System: Fault Analysis, Methodology, Issues, Challenges, and Future Perspective," vol. 16, no. 11, p. 4413.

[4] J. Yang, J. E. Fletcher, and J. O'Reilly, "Short-Circuit and Ground Fault Analyses and Location in VSC-Based DC Network Cables," vol. 59, no. 10, pp. 3827–3837.

[5] M. Muniappan, "A comprehensive review of DC fault protection methods in HVDC transmission systems," vol. 6, no. 1, p. 1.

[6] R. Muzzammel and A. Raza, "A Support Vector Machine Learning-Based Protection Technique for MT-HVDC Systems," vol. 13, no. 24, p. 6668.

[7] G. Wang, J. Xie, and S. Wang, "Application of Artificial Intelligence in Power System Monitoring and Fault Diagnosis," vol. 16, no. 14, p. 5477.

[8] Y. Chen, "Fault Diagnosis of HVDC Systems Using Machine Learning Based Methods."

[9] M. Z. Yousaf, A. R. Singh, S. Khalid, M. Bajaj, B. H. Kumar, and I. Zaitsev, "Enhancing HVDC transmission line fault detection using disjoint bagging and bayesian optimization with artificial neural networks and scientometric insights," vol. 14, no. 1, p. 23610.

[10] T. Hlalele, "Research trends in high voltage power transmission," International Journal of Engineering Trends and Technology, vol. 72, pp. 139–148, 08 2024.

[11] The gotland hvdc link | hitachi energy. [Online]. Available: https://www.hitachienergy.com/pl/pl/news-and-events/customer-success-stories/the-gotland-hvdc-link

[12] A. Alassi, S. Bañales, O. Ellabban, G. Adam, and C. MacIver, "HVDC Transmission: Technology Review, Market Trends and Future Outlook," vol. 112, pp. 530–554.

[13] D. Jovcic, High Voltage Direct Current Transmission: Converters, Systems and DC Grids, 1st ed. Wiley.

[14] J. Dragan, "VSC HVDC Applications and Topologies, Performance and Cost Comparison with LCC HVDC," pp. 137–163.

[15] D. Xu, X. Zhao, Y. Lu, K. Qin, and L. Guo, "Study on overvoltage of hybrid LCC-VSC-HVDC transmission," vol. 2019, no. 16, pp. 1906–1910.

[16] B. Chang, O. Cwikowski, M. Barnes, R. Shuttleworth, A. Beddard, and P. Coventry, "Review of different fault detection methods and their impact on pre-emptive VSC-HVDC dc protection performance," vol. 2, no. 4, pp. 211–219.

[17] R. Li and L. Xu, "Review of DC fault protection for HVDC grids," vol. 7, no. 2.

[18] O. Ikotun and S. Shehu, "An Examination of Modelling, Simulation and Detection of Fault Behaviours in HVDC Monopolar System," vol. 4, no. 1.

[19] A. Saber, H. Zeineldin, T. El-Fouly, and A. Al-Durra, "Current differential relay characteristic for bipolar HVDC transmission line fault detection," vol. 14, no. 23, pp. 5505–5513.

[20] M. Zain Yousaf, H. Liu, A. Raza, and M. Baber Baig, "Primary and backup fault detection techniques for multi-terminal HVdc systems: A review," vol. 14, no. 22, pp. 5261–5276.

[21] S. Z. Lv, J. Wen, S. Y. Zhou, and W. J. Cao, "S-Transform-Based Classification of Converter Faults in HVDC System by Support Vector Machines," vol. 347–350, pp. 1308–1312.

[22] K. Nagar and M. Shah, "A Review on Different ANN Based Fault Detection Techniques for HVDC Systems," vol. 3, no. 6, pp. 477–481.

[23] Q. Chen, Q. Li, J. Wu, J. He, C. Mao, Z. Li, and B. Yang, "State Monitoring and Fault Diagnosis of HVDC System via KNN Algorithm with Knowledge Graph: A Practical China Power Grid Case," vol. 15, no. 4, p. 3717.

[24] J. Sun, S. Debnath, M. Bloch, and M. Saeedifard. A Hybrid DC Fault Primary Protection Algorithm for Multi-Terminal HVdc Systems.

[25] T. Flavin, T. Steiner, B. Mitra, and V. nagaraju. Bayesian Ridge Regression Based Model to Predict Fault Location in HVdc Network.

[26] Z. S. Almajali, M. N. Bashabsheh, S. A. Alleimon, and L. R. Btoosh, "Fault Classification in HVDC Systems: A Fuzzy Logic Classifier Approach," in 2024 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET), pp. 1–6.

[27] P. M. Frank and B. Köppen-Seliger, "Fuzzy logic and neural network applications to fault diagnosis," vol. 16, no. 1, pp. 67–88.

[28] G. Porawagamage, K. Dharmapala, J. S. Chaves, D. Villegas, and A. Rajapakse, "A review of machine learning applications in power system protection and emergency control: Opportunities, challenges, and future directions," vol. 3.

[29] Q. Wang, Y. Yu, H. O. A. Ahmed, M. Darwish, and A. K. Nandi, "Fault Detection and Classification in MMC-HVDC Systems Using Learning Methods," vol. 20, no. 16, p. 4438.

[30] Wang, Qinghua and Yu, Yuexiao and Ahmed, Hosameldin O. A. and Darwish, Mohamed and Nandi, Asoke K., "Open-Circuit Fault Detection and Classification of Modular Multilevel Converters in High Voltage Direct Current Systems (MMC-HVDC) with Long Short-Term Memory (LSTM) Method," vol. 21, no. 12, p. 4159.

[31] G. Luo, M. Cheng, J. Hei, X. Wang, W. Huang, and J. He, "Stacked denoising autoencoder based fault location in voltage source converters-high voltage direct current," vol. 15, no. 9, pp. 1474–1485.

[32] R. S. Jawad and H. Abid, "Fault Detection in HVDC System with Gray Wolf Optimization Algorithm Based on Artificial Neural Network," vol. 15, no. 20, p. 7775.

[33] R. Rohani and A. Koochaki, "A Hybrid Method Based on Optimized Neuro-Fuzzy System and Effective Features for Fault Location in VSC-HVDC Systems," vol. 8, pp. 70 861–70 869.

[34] T. Flavin, B. Mitra, V. Nagaraju, and R. Meyur. Fault location in High Voltage Multiterminal dc Networks Using Ensemble Learning.

[35] J. C. Bravo-Rodríguez, F. J. Torres, and M. D. Borrás, "Hybrid Machine Learning Models for Classifying Power Quality Disturbances: A Comparative Study," vol. 13, no. 11, p. 2761.

[36] F. Nwokoma, J. Foreman, and C. M. Akujuobi, "Effective Data Reduction Using Discriminative Feature Selection Based on Principal Component Analysis," vol. 6, no. 2, pp. 789–799.

[37] Y. Wang, D. Zheng, and R. Jia, "Fault Diagnosis Method for MMC-HVDC Based on Bi-GRU Neural Network," vol. 15, no. 3, p. 994.

[38] Q. Chen, J. Wu, Q. Li, X. Gao, R. Yu, J. Guo, G. Peng, and B. Yang, "Long Short-Term Memory Network-Based HVDC Systems Fault Diagnosis under Knowledge Graph," vol. 12, no. 10, p. 2242.

[39] H. Li and K. Qin, "Dynamic Phasor Modelling of LCC-HVDC System Based on a Practical Project," vol. 256, p. 01034.

[40] HVDC Classic (LCC) | Hitachi Energy. [Online]. Available: https://www.hitachienergy.com/products-and-solutions/hvdc/hvdc-classic

[41] Y. F. Teng, D. X. Li, G. L. Su, and Y. H. Wang, "Simulation on Dual 12-Pulse Converter Groups in Respectively Controlled UHVDC System," vol. 805–806, pp. 700–705.

[42] H. Surya, "The Effect of 12 Pulse Converter Input Transformer Configuration on Harmonics of Input Current," vol. 1158, no. 1, p. 012007.

[43] AC/DC Filters Reactor – Quality Power. [Online]. Available: https://qualitypower.com/products/facts-hvdc-reactors/ac-dc-filters-reactor/

[44] E.-E. E. Portal and E. Csanyi. Major components of the HVDC converter station (single line diagram explained) | EEP. EEP - Electrical Engineering Portal. [Online]. Available: https://electrical-engineering-portal.com/hvdc-converter-station-single-line-diagram

[45] A. Ardakani, A. Ardakani, B. Meyer, J. J. Clark, and W. J. Gross. Standard Deviation-Based Quantization for Deep Neural Networks.

[46] S. Uddin and H. Lu, "Dataset meta-level and statistical features affect machine learning performance," vol. 14, no. 1, pp. 1–11.

[47] M. M. Ahsan, M. A. P. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique, "Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance," vol. 9, no. 3, p. 52.

[48] Logistic Regression in Machine Learning. GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/understanding-logistic-regression/

[49] scikit-learn developers. Support vector machines. scikit-learn. [Online]. Available: https://scikit-learn/stable/modules/svm.html

[50] Support Vector Machine (SVM) Algorithm. [Online]. Available: https://www.geeksforgeeks.org/support-vector-machine-algorithm/

[51] scikit-learn developers. Nearest neighbors. scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html

[52] K-Nearest Neighbor(KNN) Algorithm. [Online]. Available: https://www.geeksforgeeks.org/k-nearest-neighbours/

[53] scikit-learn developers. Decision trees. scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html

[54] Decision Tree in Machine Learning. [Online]. Available: https://www.geeksforgeeks.org/decision-tree-introduction-example/

[55] scikit-learn developers. Gradient boosting, random forests, bagging, voting, stacking. scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html

[56] Random Forest Algorithm in Machine Learning. [Online]. Available: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

[57] Gradient Boosting in ML. GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/ml-gradient-boosting/

[58] scikit-learn developers. Neural network. scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

Appendix

.1 Overview of Key Signals Captured During Simulation

This section provides a detailed overview of the key electrical signals captured during the simulation of fault scenarios in the LCC-HVDC system. These signals form the basis of the dataset used for training and evaluating the machine learning models, as discussed in Chapter 3.

Table 17: Overview of Key Signals Captured During Simulation

Signal Name(s) in CSV	**Description**
`DCFaultCurrent`	Current flowing directly through the fault impedance path.
`FaultCurrent_AG_PhA, _BG_PhB, _CG_PhC`	Fault current in the specified phase during AC phase-to-ground faults
`FaultCurrent_AB_PhA, _AC_PhA, _BC_PhB`	Fault current in phase A (for AB, AC) or B (for BC) during AC phase-to-phase faults
Rectifier Station Signals:	
`Rectifier_V[a,b,c]_pu`	AC side three-phase voltages
`Rectifier_I[a,b,c]_pu`	AC side three-phase currents
`Rectifier_VdL_pu`	DC line voltage at the rectifier output
`Rectifier_Id_pu`	DC current flowing from the rectifier
`Rectifier_IdrefLim_pu`	Limited DC current reference
`Rectifier_AlphaOrd_deg`	Ordered firing angle for rectifier valves
`Rectifier_LowACVolt`	Flag indicating low AC voltage condition
`Rectifier_ForcedAlpha`	Flag indicating forced alpha operation
`Rectifier_Valve1_Voltage`	Voltage across a specific rectifier valve
`Rectifier_Valve[1,3]_Current`	Current through specific rectifier valves
`Rectifier_Valve_AlphaOrd`	Actual firing angle of valves
Inverter Station Signals:	
`Inverter_V[a,b,c]_pu`	AC side three-phase voltages
`Inverter_I[a,b,c]_pu`	AC side three-phase currents
`Inverter_VdL_pu`	DC line voltage at the inverter input
`Inverter_VdRef_pu`	DC voltage reference for the inverter
`Inverter_Id_pu`	DC current flowing into the inverter
`Inverter_IdrefLim_pu`	Limited DC current reference for inverter
`Inverter_AlphaOrd_deg`	Ordered extinction angle for inverter valves
`Inverter_GammaMean_deg`	Mean measured extinction angle

Continued on next page

Table – continued from previous page

Signal Name(s) in CSV	**Description**
Inverter_GammaRef_deg	Reference extinction angle
Inverter_LowACVolt	Flag indicating low AC voltage condition
Inverter_AMin	Minimum alpha limit (related to inverter control)
Inverter_Valve1_Voltage	Voltage across a specific inverter valve
Inverter_Valve1_Current	Current through a specific inverter valve
Inverter_Ucom1_Current	Commutation overlap current (related to a specific valve)
Inverter_Valve_G_GammaMean_deg	Mean extinction angle of inverter valves

.2 Repository Link

The simulation automation scripts, the base Simulink model, and links to the Hugging Face dataset are available on GitHub:
https://github.com/taha2002/hvdc-fault-diagnosis-ml.git.

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

Université de Ghardaïa
Faculté des Sciences et de la Technologie

جامعة غرداية
كلية العلوم والتكنولوجيا

قسم :الآلية والكهروميكانيك

غرداية في :..30..أكتوبر..2025

شعبة :... الكهرو..ميكانيك..
تخصص:..صيانة..صناعية..

شهادة ترخيص بالتصحيح والايداع:

انا الاستاذ(ة)....بسو..جامي...حامد.....

بصفتي المشرف المسؤول عن تصحيح مذكرة تخرج (ليسانس/ماستر/دكتورا) المعنونة بـ:

Fault Diagnosis in HVDC Systems Using machine lerning

من انجاز الطالب (الطلبة):

.....جون..ابن..عبد.....

.....بلبدي..جمال.....

التي نوقشت بتاريخ :..12..جوان..2025....

اشهد ان الطالب/الطلبة قد قام /قاموا بالتعديلات والتصحيحات المطلوبة من طرف لجنة المناقشة
وقد تم التحقق من ذلك من طرفنا
وقد استوفت جميع الشروط المطلوبة.

مصادقة
رئيس قسم الآلية والكهروميكانيك
عزاوي محمد

جامعة غرداية
قسم الآلية والكهروميكانيك
كلية العلوم و التكنولوجيا

امضاء المسؤول عن التصحيح
رئيس القسم