FACULDADE
DE ECONOMIA
UNIVERSIDADE
DO PORTO

Fault Detection in Hydraulic Components of Wind Turbines Using SCADA Data and LSTM Autoencoders

Paulo Henrique de Lima

Dissertation

Master in Modelling, Data Analysis and Decision Support Systems

Supervised by:

PhD Bruno Miguel Delindro Veloso
PhD Maria Isabel Pinto Preto

Co-supervised by:

PhD João Manuel Portela da Gama

2024

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Bruno Veloso, for his invaluable guidance and continuous support throughout the development of this dissertation. His expertise, patience, and dedication were crucial to the completion of this work. I am also profoundly grateful to my co-supervisor, Professor João Gama, whose insights and inspiration during my master's studies greatly influenced my approach to this research.

A special thanks to Isabel Preto from Enlitia, whose expertise as a Data Scientist and weekly guidance were essential in helping me navigate the data and overcome technical challenges. Without her support, this thesis would not have been possible.

I am deeply thankful to my family, whose unwavering support and encouragement carried me through the most difficult moments of this journey. Their belief in me has been a constant source of strength.

I would also like to acknowledge my friends, especially Bruno Teixeira and António Campos, whose companionship during this master's degree was vital. They were by my side through the toughest times, whether it was exams, assignments, or the challenges of balancing work and academic life.

Lastly, I would like to express my heartfelt gratitude to ISMART (Instituto Social para Motivar, Apoiar e Reconhecer Talentos), which has supported me since 2011. ISMART gave me the opportunity to study at one of the best private schools in Brazil, and their belief in my potential has made all the difference. I am here today because they believed in me, and for that, I am eternally grateful.

Abstract

This work addresses the challenge of fault detection in wind turbines by leveraging advanced machine learning techniques, specifically focusing on the detection of anomalies in hydraulic systems. The study aims to improve the reliability of anomaly detection models by utilizing sensor data in conjunction with SCADA data preprocessing techniques. The primary objective is to detect faults in hydraulic components using an LSTM Autoencoder model, and to compare its performance with and without the application of a low-pass filter. The CRISP-DM (Cross-Industry Standard Process for Data Mining) framework was employed to guide the development and evaluation of the model. Enlitia provided the data used in this study, enabling the development and validation of the proposed methods. Results show that the LSTM Autoencoder, when combined with a low-pass filter, enhances the model's generalization and fault detection capabilities. Key sensor variables, such as rotor Pitch Angle and Hydraulic Oil Pressure, were identified as critical to the model's performance. Additionally, SHAP analysis was used to provide interpretability, highlighting the most important features influencing the model's predictions. This work offers a more robust and interpretable approach to detecting faults in wind turbine hydraulic systems, improving predictive maintenance and operational reliability in industrial applications.

Keywords: Wind Turbines, SCADA Data, LSTM Autoencoder, Low-Pass Filter and Anomaly Detection

Resumo

Este trabalho aborda o desafio da deteção de falhas em turbinas eólicas, aproveitando técnicas avançadas de machine learning, com um foco específico na deteção de anomalias nos sistemas hidráulicos. O estudo visa melhorar a fiabilidade dos modelos de deteção de anomalias, utilizando dados de sensores em conjunto com técnicas de pré-processamento de dados SCADA. O principal objetivo é detetar falhas em componentes hidráulicos através de um modelo LSTM Autoencoder, comparando o seu desempenho com e sem a aplicação de um filtro passa-baixo. O framework CRISP-DM (Cross-Industry Standard Process for Data Mining) foi utilizado para orientar o desenvolvimento e a avaliação do modelo. A Enlitia forneceu os dados utilizados neste estudo, permitindo o desenvolvimento e a validação dos métodos propostos. Os resultados mostram que o LSTM Autoencoder, quando combinado com um filtro passa-baixo, melhora a generalização do modelo e a sua capacidade de deteção de falhas. Variáveis-chave dos sensores, como o ângulo de inclinação do rotor e a pressão do óleo hidráulico, foram identificadas como críticas para o desempenho do modelo. Adicionalmente, a análise com SHAP foi usada para fornecer interpretabilidade, destacando as características mais importantes que influenciam as previsões do modelo. Este trabalho oferece uma abordagem mais robusta e interpretável para a deteção de falhas em sistemas hidráulicos de turbinas eólicas, melhorando a manutenção preditiva e a fiabilidade operacional em ambientes industriais.

Palavras-chave: Turbinas Eólicas, Dados SCADA, LSTM Autoencoder, Filtro Passa-Baixo, Deteção de Anomalias

Contents

Acknowledgements . . . iii
Abstract . . . v
Resumo . . . vii

1 Introduction **1**
1.1 Motivation and the Context of the Problem . . . 1
1.2 They Way that the Dissertation will be structured . . . 2

2 Literature Review **3**
2.1 Explanation of Supervised and Unsupervised Learning Algorithms . . . 3
2.2 The Main Supervised Learning Techniques For Wind Turbine Fault Detection 3
2.2.1 Neural Networks . . . 3
2.2.2 Support Vector Machines . . . 6
2.2.3 Ensemble Learning . . . 7
2.2.4 Decision Tree . . . 9
2.2.5 Bayes Algorithms . . . 10
2.3 The Main Unsupervised Learning Techniques for Wind Turbine Fault Detection . . . 11
2.3.1 Principal Component Analysis . . . 11
2.3.2 Autoencoders . . . 11
2.3.3 Fast Fourier Transform . . . 13
2.4 Techniques Comparison: Advantages and Disadvantages of Each Approach 14

3 The Problem Studied **17**
3.1 Definition of the Problem . . . 17
3.1.1 A brief explanation about the wind turbine components . . . 17
3.1.2 The main technologies of Wind Turbine . . . 18
3.1.3 The Goal . . . 19
3.2 Methodology Used . . . 19
3.3 The data Provided . . . 20
3.3.1 Description of the Data . . . 21
3.3.2 Exploratory Data Analysis . . . 23

4 Modeling **31**
4.1 The Theory of the Algorithm used - LSTM Autoencoder . . . 31
4.1.1 LSTM . . . 31
4.1.2 Autoencoder . . . 33
4.1.3 LSTM-Autoencoder . . . 34

4.1.4 Other Techniques Used to Outperform LSTM Autoencoder 36
4.2 Data Preparation and Treatment 37
4.2.1 Handling Missing Values 37
4.2.2 Choosing the Faults and Variables 38
4.2.3 Connecting the Fault Dataset and the Variables Dataset 39
4.2.4 Low Pass Filter Implementation 40
4.2.5 Transforming the Data 40
4.3 Implementation of LSTM Autoencoder 41
4.3.1 The LSTM Autoencoder 41
4.3.2 Gaussian Noise Implementation 42
4.3.3 Tunning of the Model 43
4.4 Evaluation and Results Obtained 44
4.4.1 Explicability of the Model 44
4.4.2 The Process of Training the Model Applied 46
4.4.3 Results Mestrics Before and After Low Pass Filter 47
4.4.4 Model Behavior in Predicting Failures in Advance 49

5 Conclusion **53**
5.1 Final Remarks 53
5.2 Limitations and Future Work 53

Bibliography **55**

A Appendix **i**
A.1 Other Information i

Glossary

HAWT - Horizontal Axis Wind Turbines
VAWT - Vertical Axis Wind Turbines
SCADA - Supervisory Control and Data Acquisition
CNN - Convolutional Neural Networks
SVM - Support Vector Machines
MSCNN - Multiscale Convolutional Neural Networks
MC-CNN - Multichannel Convolutional Neural Network
LSTM - Long and Short Term Memory Network
RNN - Recurrent Neural Network
AM - Attention Mechanism
AOC - Attention Octave Convolution
SGD - Stochastic Gradient Descent
BO - Bayesian optimization
DNN - Deep Neural Network
PCA - Principal Component Analysis
Artificial Neural Networks (ANNs)
SSA - Sparrow Search Algorithm
GA - Genetic Algorithm
PSO - Particle Swarm Optimization
TWSVM - Twin Support Vector Machine
RF - Random Forests
XGBoost - Extreme Gradient Boosting
RBF - Radial Basis Function
AdaBoost - Adaptive Boosting
LightGBM - Light Gradient-Boosting Machine
GBDT - Gradient Boosting (GB) with Decision Tree
BN - Bayes Net
DMNB - Discriminative Multinomial Naïve Bayes
NB - Naïve Bayes
SNB - Simple Naïve Bayes
UNB - Updateable Naïve Bayes
SVD - Value Decomposition
PC - Principal Components
DAE - Denoising Autoencoder
SW - Sliding Window
ROC curve - Receiver Operating Characteristic Curve
AUC - Area Under The Curve

SMLDAE - Stacked Multi-Level-Denotacked Denoising Autoencoder
SAE - Stacked Autoencoders
SDAE - Stacked Denoising Autoencoders
MLD-AE - Multi-Level-Denoising Autoencoder
FDR - Fault Detection Rate
AE - Autoencoder
FFT - Fast Fourier Transform
DPCA - Demixed Principal Component Analysis
MLP - Multilayer Perceptrons
DBI - Davies-Bouldin index
SI - Silhouette index
DBSCAN - Density-based spatial clustering of applications with noise
EDA - Exploratory Data Analysis
DE - Drive End
NDE - Non Drive End
SHAP - Shapley Additive Explanations

List of Figures

3.1 Generic HAWT Wind Turbine Components Example 17
3.2 The HAWT (left) and VAWT (right) Technologies of Wind Turbines 18
3.3 Steps of CRISP-DM Methodology 20
3.4 Energy Production Over Time 24
3.5 Boxplot of Energy Production 25
3.6 Spearman's Correlation Matrix of the Chosen Variables 29

4.1 How LSTM Works. Source: Wei et al. (2023) 33
4.2 How Autoencoder Works. Source: Wei et al. (2023) 34
4.3 LSTM-Autoencoder Schema. Source: Wei et al. (2023) 36
4.4 Model Summary of LSTM Autoencoder from Python Code 42
4.5 SHAP Summary Plot for Model Explanation 45
4.6 Model Training vs Validation Loss Curves Before Low Pass Filter 46
4.7 Model Training vs Validation Loss Curves After Low Pass Filter 47
4.8 Confusion Matrix Before Low Pass Filter for wt 15 48
4.9 Loss-MAE with Fault Stamp for wt 15 - Full Period of Test Dataset After Low Pass Filter 49
4.10 Loss-MAE with Fault Stamp for wt 15 - Zoomed Period of Test Dataset Before Low Pass Filter 50
4.11 Loss-MAE with Fault Stamp for wt 15 - Zoomed Period of Test Dataset After Low Pass Filter 50

A.1 Loss-MAE with Fault Stamp for wt 8 - Full Period of Test Dataset Before Low Pass Filter ii
A.2 Loss-MAE with Fault Stamp for wt 8 - Zoomed Period of Test Dataset Before Low Pass Filter ii
A.3 Confusion Matrix Before Low Pass Filter for wt 8 iii
A.4 Loss-MAE with Fault Stamp for wt 8 - Full Period of Test Dataset After Low Pass Filter iii
A.5 Loss-MAE with Fault Stamp for wt 8 - Zoomed Period of Test Dataset After Low Pass Filter iv
A.6 Confusion Matrix After Low Pass Filter for wt 8 iv
A.7 Loss-MAE with Fault Stamp for wt 12 - Full Period of Test Dataset Before Low Pass Filter v
A.8 Loss-MAE with Fault Stamp for wt 12 - Zoomed Period of Test Dataset Before Low Pass Filter v
A.9 Confusion Matrix Before Low Pass Filter for wt 12 vi

A.10 Loss-MAE with Fault Stamp for wt 12 - Full Period of Test Dataset After Low Pass Filter vi
A.11 Loss-MAE with Fault Stamp for wt 12 - Zoomed Period of Test Dataset After Low Pass Filter vii
A.12 Confusion Matrix After Low Pass Filter for wt 12 vii
A.13 Loss-MAE with Fault Stamp for wt 15 - Full Period of Test Dataset Before Low Pass Filter viii
A.14 Confusion Matrix Before Low Pass Filter for wt 15 viii
A.15 Confusion Matrix After Low Pass Filter for wt 15 ix
A.16 Loss-MAE with Fault Stamp for wt 21 - Full Period of Test Dataset Before Low Pass Filter ix
A.17 Loss-MAE with Fault Stamp for wt 21 - Zoomed Period of Test Dataset Before Low Pass Filter x
A.18 Confusion Matrix Before Low Pass Filter for wt 21 x
A.19 Loss-MAE with Fault Stamp for wt 21 - Full Period of Test Dataset After Low Pass Filter xi
A.20 Loss-MAE with Fault Stamp for wt 21 - Zoomed Period of Test Dataset After Low Pass Filter xi
A.21 Confusion Matrix After Low Pass Filter for wt 21 xii
A.22 A Sample of the Dataset xii
A.23 Spearman's Correlation of the All Variables xiii
A.24 Kendall's Correlation of the All Variables xiii
A.25 Kendall's Correlation Matrix of the Chosen Variables xiv

Chapter 1

Introduction

Early fault detection in wind turbines involves the proactive use of advanced monitoring techniques to identify potential issues before they escalate into critical failures. This approach integrates sensor technology to capture real-time data on factors such as vibration, temperature, and wind speed. Leveraging data analytics, machine learning algorithms, and condition monitoring systems, it enables the timely detection of anomalies in critical components like gearboxes, generators, and blades. Vibration analysis plays a key role in identifying irregularities in mechanical parts, while prognostics and health management strategies contribute to predicting the remaining useful life of components. The implementation of remote monitoring allows for real-time assessment, empowering operators and maintenance teams to take corrective measures and optimize maintenance schedules, ultimately enhancing the reliability and performance of wind turbines while minimizing downtime and reducing overall maintenance costs.

1.1 Motivation and the Context of the Problem

Throughout my master's program, one of the subjects that captivated my interest the most was machine learning. The capability to predict future outcomes based on historical data is nothing short of remarkable. This field's potential to transform vast amounts of data into actionable insights fascinated me from the beginning. As I progressed through my studies, the rapid advancements in artificial intelligence (AI) became increasingly apparent. When I first entered the master's program, tools like ChatGPT were not widely used or even well-known. However, during the course of my studies, AI technologies like ChatGPT have become ubiquitous, underscoring their immense potential and inevitability as future cornerstones of various industries.

As given by Maheshwari (2024) in your Forbes article, the global AI market is projected to grow at an impressive compound annual growth rate (CAGR) of 37.3% from 2023 to 2030, reaching a staggering $1.3 trillion by 2030. This growth is fueled by the widespread adoption of AI across diverse sectors such as healthcare, finance, and manufacturing. Witnessing this exponential growth and its implications on the future solidified my decision to focus on machine learning for my thesis.

In parallel, I have always been an ardent advocate for renewable energy. Growing up in Brazil, I was acutely aware of the significance of renewable energy in our country's energy strategy. Brazil is the largest electricity market in Latin America, the world's sixth-largest

consumer of electricity, and has the seventh-largest electricity generation capacity globally. According to Rego (2023), renewable energy sources account for 90% of Brazil's electricity matrix, significantly higher than the global average of around 25% Administration (2023). According to Brasil (2023) wind energy, in particular, plays a crucial role in Brazil's renewable energy landscape, representing about 20% of the energy matrix. Brazil stands as the largest wind energy producer in Latin America, with an installed capacity of approximately 25,04 gigawatts (GW).

Given my enthusiasm for both machine learning and renewable energy, I decided to integrate these two interests into my thesis. The rapid development and integration of AI in various fields, combined with the critical importance of renewable energy for sustainable development, provided a compelling foundation for my research. By leveraging machine learning techniques, we can develop more accurate predictive models for energy production, demand forecasting, and resource management, ultimately contributing to more efficient and sustainable energy systems.

1.2 They Way that the Dissertation will be structured

This dissertation will be structured based on the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology. CRISP-DM is a comprehensive framework that provides a structured and systematic approach to data mining projects, consisting of six phases, as seen in Wirth and Hipp (2000): business understanding, data understanding, data preparation, modeling, evaluation, and deployment. Each phase includes specific tasks and deliverables that ensure a thorough and repeatable process, promoting best practices and facilitating effective communication among stakeholders. This methodology enhances project success and consistency across various industries through its iterative nature and flexibility.

The iterative nature of CRISP-DM allows for revisiting and refining earlier phases based on insights gained in later stages, thereby improving model outcomes. Its industry-agnostic design has made it a cornerstone in data mining and analytics, contributing significantly to the field's advancement. By standardizing project workflows, CRISP-DM helps document and replicate successful strategies, ensuring robust data-driven decision-making processes.

In this dissertation, the CRISP-DM methodology is reflected in the following subchapters: business understanding is covered in subchapters 1.1, 1.2, and 3.1; data understanding is addressed in subchapter 3.2; the modeling phase is detailed in subchapters 4.2 and 4.3; and the evaluation of the model is discussed in subchapter 4.4. The deployment phase will be implemented in the next steps by Enlitia, ensuring a practical application of the research findings.

Chapter 2

Literature Review

2.1 Explanation of Supervised and Unsupervised Learning Algorithms

According to Berry, Mohamed, and Yap (2019), supervised learning is characterized by the algorithm's capacity to generalize knowledge from labeled or target cases within the available data. This enables the algorithm to make predictions for new, unlabeled cases. The algorithm learns to map input features to the correct output by generalizing from the labeled examples. Examples include Convolutional Neural Networks (CNN), Support Vector Machines (SVM), Artificial Neural Networks (ANN), Random Forest, among others.

Also for Berry et al. (2019), unsupervised learning are the techniques of clustering data through automated methods or algorithms without pre-existing classification or categorization. In this context, algorithms are tasked with 'learning' inherent relationships or features from the provided data and grouping cases based on shared features or characteristics. These algorithms explore the inherent structure of the data without predefined class labels. Examples include Fast Fourier Transform (FFT), Autoencoder Algorithms, Principal Component Analysis, among others.

2.2 The Main Supervised Learning Techniques For Wind Turbine Fault Detection

2.2.1 Neural Networks

Neural networks are computational models inspired by the structure and functioning of the human brain, composed of interconnected nodes organized into layers. Each connection between nodes has a weight, and through a process of learning from data, these weights are adjusted to enable the network to make predictions or classifications. The network is typically divided into input, hidden, and output layers, with the hidden layers extracting hierarchical features from the input data. Training involves feeding the network labeled data, adjusting weights through backpropagation, and optimizing performance. In their comprehensive book Goodfellow, Bengio, and Courville (2016a) provide an in-depth exploration of neural networks and the broader field of deep learning.

CNNs are a specialized class of deep neural networks designed for grid-like data, partic-

ularly effective in image processing tasks. CNNs consist of convolutional layers that extract local patterns, pooling layers for spatial dimension reduction, and fully connected layers for integrating features. One influential figure in the CNN domain is Yann LeCun, a pioneer in deep learning. His work, such as the introduction of convolutional network architectures for handwritten digit recognition in his paper Goodfellow et al. (2016a), has been pivotal in establishing CNNs as a powerful tool in computer vision applications. LeCun's continued contributions make him a leading authority on CNNs and their applications in various fields. CNN will be the most talked neural network in the following text.

In his article, Jiang, He, Yan, and Xie (2018) investigates the utilization of CNNs in the realm of fault detection for wind turbine gearboxes. The Multiscale Convolutional Neural Networks (MSCNN) system is designed to enhance the detection of gearbox issues by learning features at multiple scales, thereby improving fault classification in comparison to traditional methods. The paper systematically outlines the three main components of the MSCNN model: data preparation achieved through coarse-graining, learning from data at multiple scales using neural network layers, and the subsequent identification of faults. The experimental results, conducted on a wind turbine test rig, showcase the superior performance of MSCNN in identifying and classifying gearbox conditions when compared to traditional single-scale CNNs.

In his study, Zare and Ayati (2021) underscores the significance of multiscale analysis in the field of fault diagnosis for wind turbine applications. The results demonstrate that the MSCNN model outperforms traditional methods, emphasizing the importance of considering vibration signals at different scales for effective fault detection. The authors also acknowledge that MSCNN, while taking longer to train due to its complexity, proves to be fast enough for real-time applications once trained. The findings suggest that, particularly for complex tasks, the adoption of a deeper MSCNN might be necessary to further improve results in identifying and classifying problems associated with wind turbine gearboxes.

A similar approach was done by Zare and Ayati (2021) introduces an innovative approach to detecting faults in wind turbines utilizing SCADA signals. The authors employ a specialized neural network known as Multichannel Convolutional Neural Network (MC-CNN), designed to process distinct types of data separately. The MC-CNN comprises two primary components: one for initial feature extraction and another for amalgamating these features to reach a final decision. With seven input channels handling various data types such as power and speed, the MC-CNN efficiently utilizes a single channel for similar vibration signals to optimize computational resources.

The evaluation metrics of Zare and Ayati (2021), including accuracy, loss function, and a confusion matrix, demonstrate the MC-CNN's remarkable average accuracy of 99.85% in detecting wind turbine faults, with the confusion matrix revealing a high level of correct fault identification and minimal errors. The MC-CNN method, showcased in this research, eliminates the need for human input or additional sensors, relying solely on existing SCADA data. Designed for real-world conditions, the model can be implemented on industrial systems with a GPU, emphasizing its practical applicability. Despite the resource-intensive nature of CNNs in learning from raw data, the MC-CNN method demonstrates efficiency, speed, and reduced complexity compared to previous approaches.

Another way of applying CNN to the context of wind turbine fault detection is the method applied by Xiang, Wang, Yang, Hu, and Su (2021), in which the CNN is applied together with the Long and Short Term Memory Network (LSTM) - a type of recurrent neural network

(RNN) designed to process and learn sequences of data, and Attention Mechanism (AM). The method uses CNN to identify features of a wind turbine's state and LSTM to merge time-related features. AM assigns importance to these features to improve the model's predictions.

The article of Xiang et al. (2021) unveils the CNN-LSTM-AM model's proficiency in early wind turbine fault detection. This model, adept at predicting faults, displays stable RMSE values during normal operation and reveals discernible rising trends well in advance of actual faults. Notably, the study highlights instances where the CNN-LSTM-AM model detected potential failures significantly before their occurrence within the analyzed timeframe. For instance, in Case 2, the RMSE value surpassed the set threshold on the 109th day out of a total of 160 days, indicating an impressive capability for early fault detection.

In the article by Xiao, Liu, Zhang, and Zhang (2021), the authors use CNN with ResNet, which is a type of network that helps avoid these problems through a special design called residual learning. Residual learning facilitates the learning of networks by focusing on small changes in an identity function. There are many types of ResNet models, ResNet50 was chosen for its balance of speed and accuracy in this research. The authors also used the AM, proposed in the previous article, to enhance the ResNet50 network, creating the AOC-ResNet50 model. The attention mechanism helps to address issues with the original Octave Convolution (OctConv) by preserving the original features in the high and low frequency domains.

The authors Xiao et al. (2021) utilize radar charts derived from data 24 hours, 3 days, 7 days, and 15 days before failures to train and test the models. Impressively, the AOC–ResNet50 network exhibits superior fault detection performance in comparison to other models tested. Notably, the study underscores that the model's effectiveness improves when trained with data closer to the time of failure. However, the performance difference between the 3-day dataset and the 15-day dataset is approximately 6 percentage points (98.04% vs. 93.51%). This suggests that the AOC–ResNet50 model is exceptionally efficient in detecting early failures, showcasing its potential for timely and accurate fault identification in wind turbine converter systems.

As there will be a focus on failure detection of hydraulic components of wind turbines, articles on failure detection in strictly hydraulic components, such as the piston pump, were also reviewed. In the pursuit of enhancing fault diagnosis for hydraulic piston pumps, S. Tang, Zhu, and Yuan (2021) present a study introducing an advanced CNN model with an adaptable learning rate. The proposed model employs continuous wavelet transform to convert three raw signals—vibration, pressure, and sound—into two-dimensional time-frequency images, facilitating comprehensive fault diagnosis (similar to the approach of articles of wind turbines talked before). To optimize the model's performance, researchers rigorously tested five optimizers—SGD, Adam, RMSprop, Adagrad, and Adadelta. Among these, the Adam was the best one, achieving high accuracy (>97%) swiftly with minimal loss.

S. Tang, Zhu, and Yuan (2022) proposed another approach for the same problem that integrates CNNs with Bayesian optimization (BO) for enhanced diagnostic efficacy. The method involves transforming vibration signals into 2D images using the ComplexMorlet wavelet function (similar to the others showed in this review). Notably, the study employs Bayesian optimization to fine-tune hyperparameters (HPs), such as learning rate and batch size, optimizing the CNN models for improved performance. Three CNN models were assessed, and the proposed CNN-BO emerged with the highest accuracy, surpassing 97%.

In their article, L. Chen, Xu, Zhang, and Zhang (2019) tackles the issue of imbalanced

SCADA data in wind turbine fault diagnosis. The conventional oversight of data imbalances between normal and abnormal instances in SCADA data often results in biased fault diagnosis models. To address this, the paper introduces a deep neural network (DNN)-based method that leverages triplet pairs from imbalanced data, enabling the model to learn and distinguish between anchor points, positive, and negative counterparts. By mapping data into a new space and applying a k-nearest neighbor method, the proposed approach enhances fault diagnosis, addressing the challenges posed by imbalanced SCADA data.

Finally, in their model L. Chen et al. (2019) demonstrates high precision and F-measure, with detailed parameters provided for reproducibility. While achieving effective data dimensionality reduction through Principal Component Analysis (PCA), the TL-Model proves resilient in handling the variability of wind turbine conditions, showcasing its efficacy in feature extraction for fault diagnosis. Comparative analyses underscore the importance of the added bypass component, significantly improving precision in the fault diagnosis process.

2.2.2 Support Vector Machines

Support Vector Machines (SVMs) are a robust class of supervised learning algorithms widely recognized for their efficacy in handling complex classification tasks. Cortes and Vapnik (1995) introduced SVMs as a method for finding hyperplanes that maximize the margin between different classes in high-dimensional spaces, providing a solid foundation for binary and multi-class classification. The key idea lies in identifying the optimal hyperplane that maximizes the separation between data points of different classes, with the regularization parameter (C) allowing for a fine balance between achieving a wider margin and minimizing misclassifications. SVMs, often equipped with kernel functions, demonstrate adaptability to non-linear decision boundaries, making them indispensable in various domains, including image recognition, text classification, and bioinformatics.

In his article, Santos, Villa, Reñones, Bustillo, and Maudes (2015) focuses on detecting faults in the mechanical chain of wind turbines by utilizing accelerometers to capture vibrations indicating potential issues. Given the challenges posed by the dynamic nature of wind turbine operations, the study employs various sensors and compares the effectiveness of SVM and Artificial Neural Networks (ANNs) for data analysis. In the ANN investigation, parameters such as momentum and learning rate are optimized, revealing that a 17-neuron architecture achieves an accuracy of 97.47%.

Santos et al. (2015) then delves into SVM optimization, where the linear SVM emerges as the fastest and most accurate method, boasting an accuracy of 98.26%. Notably, the linear SVM outperforms ANN in terms of both accuracy and speed (tunning and training), with the efficiency of requiring adjustment of only one parameter contributing to faster tuning. The exploration of feature reduction indicates that all 544 features are indispensable for optimal results with the linear SVM.

The article of Laouti, Sheibat-Othman, and Othman (2011) explores the use of SVM for fault detection in wind turbines, addressing challenges such as variable wind speeds and continuous operation. Focusing on online supervision, the study aims to cut maintenance costs and enhance turbine performance. Simulations cover various faults, including stuck positions, gain factor changes, and alterations in pitch actuator parameters. While fixed value faults in pitch position and rotor speed are quickly detected, the study notes challenges in detecting faults in rotor speed sensors and emphasizes the potential for improvement through more

data and better filtering. The research sheds light on SVM's effectiveness in fault detection for different turbine components, identifying successes and areas for further investigation.

Tuerxun, Chang, Hongyu, Zhijie, and Huajian (2021) focuses on utilizing a SVM enhanced by the Sparrow Search Algorithm (SSA) for efficient fault diagnosis in wind turbines. Introduced SSA is a novel optimization algorithm known for its effective search capabilities, particularly in finding optimal parameters like the penalty factor (C) and kernel parameter (g) crucial for SVM model performance. The SSA-SVM model stands out in terms of speed and accuracy, surpassing alternative models like SVM, GA-SVM, and PSO-SVM, with a notable 99.2% accuracy in correctly classifying test samples and superior performance across various metrics, including precision, recall, and F1-score. The study underscores the SSA-SVM model as a superior choice for fault diagnosis in wind turbines, showcasing its speed, accuracy, and overall effectiveness in capturing nuanced performance metrics.

The authors Dhiman, Deb, Muyeen, and Kamwa (2021) assesses various anomaly detection methods and identifies that employing Twin Support Vector Machines (TWSVM) with an adaptive threshold yields the best results. Using SCADA data recorded at 10-minute intervals, the study employs machine learning techniques, including K-means clustering, to pinpoint abnormal patterns indicative of gearbox issues. TWSVM stands out for its efficiency in solving smaller mathematical problems, making it faster than other models tested. The research, focused on gearbox problem detection, introduces an adaptive threshold for time-series data, improving the identification of normal and faulty conditions and achieving a high classification accuracy of 95.94%. TWSVM emerges as the top-performing model, demonstrating superior computation speed compared to other tested classifiers.

Finally, Laouti, Othman, Alamir, and Sheibat-Othman (2014) explores the integration of SVM and a Kalman-like observer for fault detection in wind turbines. SVM, known for its efficacy with limited data, is trained and tested to identify various faults, showcasing its quick detection capabilities. Simultaneously, the observer, designed akin to a Kalman filter, focuses on detecting changes in the pitch position actuator and estimating parameters for fault identification. SVM is robust and works well even with small datasets, but requires a lot of data for training. The Kalman-like observer can detect faults quickly, often within 2 sampling periods, without needing prior training.

2.2.3 Ensemble Learning

As seeing the in the work of Seni and Elder (2010), "Ensemble Learning" is a machine learning approach that combines predictions from multiple individual models to improve overall accuracy and robustness. This technique aims to overcome the limitations of individual models by leveraging their diverse strengths and compensating for their weaknesses. Common ensemble methods include bagging, boosting, and stacking, each employing distinct strategies to amalgamate predictions. The ensemble's collective decision is often more accurate and stable than that of its constituent models.

Zhang et al. (2018) introduces a data-driven approach for fault detection in wind turbines by leveraging the power of ensemble learning models, specifically Random Forests (RF) and extreme gradient boosting (XGBoost). In contrast to traditional methods like SVM, which may struggle with high-dimensional data, the ensemble strategy of RF and XGBoost proves to be highly effective. The combination of RF and XGBoost enhances the fault classifier's performance by utilizing RF to assess and prioritize feature importance and then refining the

classifier using XGBoost with the top features identified by RF. The use of decision trees as base models within the ensemble framework ensures resilience against overfitting, a common challenge with single decision trees. The study's comprehensive feature analysis, incorporating various indicators from sensors without necessitating detailed wind turbine models, further underscores the practicality and robustness of the proposed methodology.

Comparative analysis against SVM and RF models, made by Zhang et al. (2018), reveals the superior performance of the ensemble approach in fault detection, especially for sensor faults. RF's ability to rank features by importance and XGBoost's subsequent refinement contribute to high accuracy in fault detection, outperforming SVM. The ensemble approach's reliability is highlighted by consistently high hit rates, suggesting its potential as an effective and versatile model for wind turbine fault detection across diverse conditions and turbine types. Overall, this research provides valuable insights into the efficacy of ensemble learning for wind turbine fault detection, opening avenues for improved reliability and efficiency in wind energy systems.

In their research, Z. Wu, Wang, and Jiang (2020) introduce an advanced methodology for identifying faults in wind turbines using the ReliefF and XGBoost algorithms, leveraging data collected from a supervisory control and data acquisition (SCADA) system. The "ReliefF + XGBoost" approach outperformed alternative methods, such as radial basis function-Support Vector Machine (rbf-SVM) and Adaptive Boosting (AdaBoost), in accurately diagnosing issues within wind turbines. The ReliefF algorithm, originating in 1992, excels in selecting crucial data features, particularly well-suited for scenarios involving multiple groups. By assessing the proximity of data points within and across groups, ReliefF assigns scores to features, ultimately choosing the most informative ones.

XGBoost, developed by T. Chen and Guestrin (2016), represents a powerful machine learning technique that harnesses the principles of ensemble learning. It employs numerous simple models, such as CART trees, to enhance predictive capabilities. The algorithm focuses on minimizing the difference between predictions and actual results, incorporating a penalty for complex models to prevent overfitting. In the research [Citation – XGBoost], experimental results showcased the superiority of the proposed "ReliefF + XGBoost" algorithm over SVM and AdaBoost, affirming its efficacy in wind turbine fault diagnosis, as evidenced by superior accuracy, macro precision, macro recall, and macro F1-Score (all them equal to 100%).

The research made by M. Tang et al. (2020) introduces an enhanced approach, known as adaptive Light Gradient-Boosting Machine (LightGBM), for online fault detection in wind turbine gearboxes, showcasing improved efficiency compared to traditional methods. The method incorporates a sophisticated feature selection technique to identify crucial parameters from wind turbine data, demonstrating superior fault detection performance with minimized errors. The four-step process, including data preparation, feature selection, model training, and online fault detection, addresses challenges such as missing data and optimally tunes the LightGBM algorithm. The study identifies four gearbox states and utilizes LightGBM for fault diagnosis, treating it as a binary classification problem. Comprehensive evaluations against other fault diagnosis methods like GBDT and XGBoost highlight LightGBM's superior efficiency, attributed to its handling of large datasets, advanced techniques like gradient-based one-side sampling and exclusive feature bundling, and superior hyper-parameter optimization using the Tree-structured Parzen Estimator method. The research provides a robust and efficient solution for online fault detection in wind turbine gearboxes, emphasizing the

significance of LightGBM in enhancing performance metrics such as false alarm rate and missing detection rate.

The study of Yuan, Sun, and Ma (2019) addresses the critical need for robust fault detection methods in wind turbines, given their pivotal role in sustainable energy infrastructure. By leveraging SCADA data and employing a novel approach combining a stacking model with change-point detection algorithms, the research proposes an effective strategy for predicting gearbox issues in wind turbines. Through the integration of three machine learning models – RF, GBDT, and XGBOOST – and the utilization of a specialized distance measure and change-point detection algorithm, the study demonstrates promising results in fault prediction across five turbines. The approach capitalizes on the intricate relationship between gearbox oil temperature and fault occurrence, offering valuable insights into early fault detection and maintenance planning within wind energy systems.

Furthermore, the research of Yuan et al. (2019) underscores the significance of data preprocessing techniques, such as Density-based spatial clustering of applications with noise (DBSCAN) and quartile methods, in enhancing the quality and reliability of SCADA data. By effectively filtering anomalies and selecting pertinent features, the study optimizes the predictive capabilities of the model, facilitating accurate fault diagnosis and prognosis. The ability to anticipate faults well in advance, as demonstrated by the algorithm's detection of faults 13 to 14 days before occurrence, underscores the potential of the methodology to mitigate downtime and enhance the reliability of wind energy systems. Future endeavors will focus on refining prediction intervals to minimize false alarms and extending the methodology to encompass other components of wind turbines, thus advancing the field of fault detection and contributing to the reliability and sustainability of wind energy systems.

2.2.4 Decision Tree

It's explained in the book made by Gama, Carvalho, Faceli, Lorena, and Oliveira (2012) that a decision tree employs a strategic "divide and conquer" approach to solve complex decision problems. It intelligently breaks down a sophisticated problem into simpler subproblems, recursively applying the same strategy. The solutions to these subproblems are then combined into a tree-like structure, resulting in a solution to the original problem. The strength of this approach lies in its ability to partition the instance space into subspaces and adapt each subspace using different models. This fundamental idea forms the basis for decision tree-based algorithms. This inherent flexibility makes decision trees particularly advantageous for handling diverse datasets and accommodating different decision-making scenarios, making them a popular choice in various applications.

In addressing the complex task of wind turbine fault diagnosis, decision trees emerge as a pragmatic choice due to their simplicity and scalability. The study of Abdallah et al. (2018) harnesses decision tree classifiers for automating fault diagnosis and root cause analysis, advocating for a holistic approach. The researchers demonstrate the efficacy of decision trees in linking faults to root causes and introduce a real-time framework designed for big data monitoring in wind turbines. Leveraging cloud computing services like Azure, this framework integrates decision trees and Bayesian networks, efficiently handling extensive data processing and storage in the cloud.

The proposed framework created by Abdallah et al. (2018) is tailored for real-time monitoring, employing cloud-based solutions to facilitate seamless data analysis. Leveraging Hadoop,

Kafka, and Spark, the system ensures reliable and swift storage and processing of large datasets. With over 2.5 million records analyzed, including 980 instances of excessive vibration errors, the Bagged decision tree classifier exhibited a remarkable misclassification rate of less than 1% with a bag size exceeding 10. This underscores the algorithm's effectiveness in detecting a range of issues in wind turbines, from faults and errors to damage patterns, anomalies, and abnormal operation. The integration of decision trees and cloud-based computing sets the stage for advanced, data-driven strategies in wind turbine structural health monitoring.

2.2.5 Bayes Algorithms

As it was saw in the book of Gama et al. (2012), the Bayesian classifiers, a family of probabilistic models, leverage Bayes' theorem to facilitate classification tasks based on probability theory. These classifiers estimate the probability of a given instance belonging to a particular class by combining prior knowledge with observed evidence. By assuming that features are conditionally independent given the class, these classifiers simplify the calculation of probabilities. One notable example is the Naive Bayes classifier, which assumes independence among features, making it computationally efficient. Another variant is the Bayesian network classifier, which captures dependencies among features using a graph structure.

In the study of Joshuva and Sugumaran (2017), focused on blade fault detection in wind turbines, various classifiers were assessed, including Bayes Net (BN), Discriminative Multinomial Naïve Bayes (DMNB), Naïve Bayes (NB), Simple Naïve Bayes (SNB), and Updateable Naïve Bayes (UNB). The decision tree algorithm J48, based on C4.5, emerged as a crucial tool for classifying data in blade condition monitoring. J48 employs a tree structure with branches, a root, nodes for attributes, and leaves representing class labels. Information gain, a measure of how well an attribute separates data, aids in identifying the most significant attribute for classification. The algorithm, by automatically selecting crucial features, reduces reliance on expert knowledge, recognizing important aspects like sum, range, standard deviation, and kurtosis for blade condition representation.

Recorded vibration signals from wind turbine blades, encompassing both normal and faulty conditions, were analyzed using the J48 algorithm, revealing its effectiveness in pinpointing essential features for blade fault identification in the article of Joshuva and Sugumaran (2017). Among the classifiers, the Bayes Net classifier exhibited the highest accuracy at 85.67% for fault detection, employing a simple estimator and simulated annealing technique. The Bayes Net model demonstrated a robust performance with a high kappa statistic (0.828), indicative of good agreement with true classes, as well as low mean absolute error (0.0716) and root mean square error (0.1891), affirming accurate predictions. Moreover, the model's efficiency in real-time monitoring, with a quick build time of 0.56 seconds, underscores its practical suitability for wind turbine condition assessment.

2.3 The Main Unsupervised Learning Techniques for Wind Turbine Fault Detection

2.3.1 Principal Component Analysis

As see in the book of Jolliffe (2002), PCA is a dimensionality reduction technique widely employed for fault detection in various systems, including wind turbines. PCA transforms the original variables into a new set of uncorrelated variables, termed principal components, which capture the maximum variance in the data. By focusing on the most significant features, PCA helps simplify complex datasets and identify patterns associated with normal system behavior. In the context of wind turbine fault detection, PCA analyzes sensor data, highlighting deviations from the learned normal patterns that may indicate faults or anomalies. This method proves effective in recognizing subtle changes in system dynamics and is commonly applied in conjunction with statistical metrics for fault detection.

The article of Wang, Ma, and Qian (2018) presents groundbreaking algorithms designed for efficient wind turbine monitoring by utilizing a condensed yet crucial dataset. The authors introduce a method that strategically selects essential data, ensuring the preservation of key information necessary for identifying fault occurrences, understanding their timing, pinpointing their location, and assessing their severity. The algorithms' effectiveness is thoroughly demonstrated through comprehensive testing with simulations, real wind farm SCADA data, and experimental data from a wind turbine test rig, highlighting their operational efficiency with reduced information.

Central to the study of Wang et al. (2018) is the application of PCA, a method employed to distill the main features of the dataset by transforming them into uncorrelated variables known as principal components (PCs) through Singular Value Decomposition (SVD). This technique allows for a simplified representation without significant information loss regarding data variability. The study accentuates the practical applicability of these methods, successfully applying them to gearbox and generator faults, thus showcasing their potential for practical deployment in real-world scenarios and paving the way for advancements in real-time monitoring. The developed algorithms specifically target the reduction of data required for wind turbine fault detection, introducing a variable selection algorithm adept at choosing crucial variables. These selected variables prove effective in identifying faults, discerning their timing, location, and severity.

2.3.2 Autoencoders

As saw in the book of Goodfellow, Bengio, and Courville (2016b) autoencoders are a type of artificial neural network designed for unsupervised learning, particularly in the realm of feature learning and data compression. Structured as an encoder-decoder architecture, an autoencoder aims to reconstruct its input data, effectively learning a condensed representation or encoding of the input in its hidden layer. This condensed representation, known as the bottleneck or latent space, captures essential features of the input data. By minimizing the difference between the input and the reconstructed output, autoencoders effectively capture salient patterns and features within the data. They find applications in various domains, such as image and signal processing, anomaly detection, and dimensionality reduction. Notable types of autoencoders include denoising autoencoders, variational autoencoders, and sparse

autoencoders. The versatility of autoencoders makes them valuable tools in uncovering intrinsic structures and patterns in complex data.

In the pursuit of identifying problems in wind turbines, researchers face challenges due to the intricate behavior of turbines, external influences, and noisy data. To address this, a novel approach is proposed by Jiang, Xie, He, and Yan (2017), employing a denoising autoencoder (DAE) that leverages past and present data to discern patterns, enhancing the detection of turbine issues. Autoencoders, particularly DAEs, prove effective in learning from noisy data and are superior to older methods, offering flexibility, resilience against noise, and the ability to handle both straight and wavy patterns. The study introduces a sliding window technique to capture temporal relationships in wind turbine data, enabling the DAE model (SW-DAE) to recognize complex patterns within short time frames. The method exhibits significant promise, outperforming traditional techniques like PCA and DPCA, particularly in scenarios involving noisy and complex data, showcasing its potential for robust fault detection in wind turbines.

The fault detection methodology showed by Jiang, Xie, et al. (2017) involves an offline learning phase, where the DAE model comprehends normal behavior, followed by an online phase to identify anomalies during regular turbine operation. Comparative analysis with PCA and Demixed Principal Component Analysis (DPCA) reveals that the SW-DAE method consistently outshines in terms of Receiver Operating Characteristic Curve (ROC Curve), Fault Detection Rate, and Area Under The Curve (AUC) values, for all scenarios of failure, indicating its efficacy in fault diagnosis. Notably, the sliding window approach demonstrates superior performance compared to non-sliding window counterparts, emphasizing the importance of considering temporal information for accurate fault diagnosis in wind turbines.

The paper of Jiang, He, Xie, and Tang (2017) introduces a novel approach named Stacked Multi-Level-Denoising Autoencoders (SMLDAE) for detecting faults in wind turbine gearboxes by analyzing vibrations. SMLDAE employs various noise levels during training, enabling the model to effectively learn fault patterns from raw data, thereby enhancing its diagnostic capabilities. Through a unique training scheme that involves both unsupervised learning and supervised fine-tuning, the method outperforms traditional techniques in identifying gearbox problems when tested on real-world data from a wind turbine gearbox test rig. The SMLDAE approach demonstrates superior fault diagnosis by collecting vibration signals, learning from them through unsupervised learning with multiple noise levels, and subsequently fine-tuning the system for supervised fault classification, showcasing its adaptability to complex data patterns in wind turbine applications.

The study of Jiang, He, et al. (2017) highlights the complexity of analyzing vibration signals from wind turbine gearboxes, emphasizing the challenges associated with uncovering hidden faults. SMLDAE is introduced as a solution to address these challenges by training on varying noise levels to better discern fault patterns in the intricate vibration data. The method exhibits remarkable accuracy, especially when compared to traditional methods such as Stacked Autoencoders (SAE) and Stacked Denoising Autoencoders (SDAE), showcasing its efficacy in learning useful features from the data and significantly improving fault diagnosis. SMLDAE's adaptability to changing conditions is demonstrated through experiments, where it excels at identifying faults in new conditions, solidifying its potential as an advanced tool for gearbox fault detection in wind turbines.

The paper of X. Wu, Jiang, Wang, Xie, and Li (2019) talks about an approach similar to SMLDAE, the multi-level-denoising autoencoder (MLD-AE), designed to enhance wind

turbine fault detection by incorporating various levels of noise during the learning process. In contrast to the conventional denoising autoencoder (DAE), which employs a single noise level, MLD-AE progressively introduces noise, enabling the model to capture both general and detailed features. This method proves effective in understanding complex wind turbine data, where noisy patterns can be challenging to discern. The training mechanism gradually reduces noise levels while updating model parameters, facilitating improved learning and pattern recognition. The MLD-AE model, trained on sensor data from wind turbines, demonstrates its ability to predict normal turbine behavior by identifying differences between expected and actual sensor data, employing robust Mahalanobis distance for reliable fault detection.

The study of X. Wu et al. (2019) evaluates different MLD-AE variations, particularly those incorporating MLD-AE-S2, showcasing their superior performance in terms of fault detection rate (FDR) and false alarm rate (FAR) when compared to traditional methods like PCA and AE. Notably, the MLD-AE variants outperform other methods in detecting faults, providing an advantage in identifying fault initiation and termination, which proves challenging for AE and PCA. The results consistently highlight the robustness and effectiveness of MLD-AE, emphasizing its potential for accurate wind turbine fault diagnosis and maintenance management.

In his paper, Nie, Liu, and Xie (2020) propose a novel model called the Denoising Stacked Feature Enhanced Autoencoder with Dynamic Feature Enhanced Factor (DSFEAE-DF) to improve fault diagnosis in wind turbines. They introduced Gaussian noise during the training phase, which helped the model learn more robust and discriminative features from the noisy input data. This noise-adding strategy, combined with the dynamic feature enhancement process, allowed the autoencoder to focus on important features while filtering out irrelevant ones. As a result, the model demonstrated superior performance in accurately diagnosing faults, even in noisy environments, by effectively reducing overfitting and improving the generalization ability of the autoencoder.

2.3.3 Fast Fourier Transform

As saw in the work of Randall (2021), Fast Fourier Transform (FFT) is commonly employed for fault detection by analyzing frequency domain characteristics in signals. In the context of machinery, including wind turbines, FFT can be applied to vibration or acoustic signals to identify abnormal frequency components associated with faults. Anomalies such as irregularities in gear meshing, bearing defects, or structural damage introduce unique frequency patterns in the signal spectrum. By transforming time-domain signals into the frequency domain, FFT allows engineers to pinpoint specific frequencies associated with faults, providing valuable insights into the condition of mechanical components.

The article of Nejad, Odgaard, Gao, and Moan (2014) presents an innovative approach for detecting problems in wind turbine gears and bearings by monitoring the rotational speed of the gearbox shafts. Utilizing existing sensors and a straightforward methodology, the technique employs a specialized error-checking method that examines changes in the energy of the speed error signals to identify faults. Tested on a 750 kW wind turbine gearbox, the method successfully detected issues in certain bearings, showcasing its efficacy for early problem detection. This method addresses the increasing challenges of maintaining offshore wind turbines by providing a reliable and accessible means of monitoring drivetrain health, crucial for ensuring the continued functionality and longevity of wind turbine systems. The simplicity

and effectiveness of the approach make it a promising tool for proactive maintenance in the wind energy industry, contributing to enhanced turbine reliability.

In both experimental cases of the article made by Nejad et al. (2014), the method demonstrated its capability by revealing distinct changes in the FFT amplitude at specific frequencies, effectively distinguishing between 'Fault Free' and 'Faulty' conditions. The proposed technique relies on the analysis of FFT spectra to discern anomalies, making it a valuable tool for diagnosing faults or monitoring the overall health of wind turbine systems over time. The study underscores the significance of this method in simplifying fault detection processes while maintaining accuracy, marking a noteworthy advancement in the field of wind turbine condition monitoring.

The article of Lu, Qiao, and Gong (2017) presents an innovative approach to detecting gearbox faults in wind turbines by analyzing electrical current signals from the generator. The method involves adapting the signal to a constant value, enabling the identification of fault features through analysis. Two detectors are employed to confirm the presence of faults. This approach proves effective in detecting gearbox issues without the need for additional equipment, offering a cost-efficient and reliable solution for wind turbine monitoring. The research addresses the challenge of changing speeds in wind turbine gearboxes by introducing a normalization process that ensures the method's reliability under varying load conditions.

The method proposed by Lu et al. (2017) processes stator current signals to highlight characteristic frequencies associated with gearbox faults, overcoming the limitations of traditional FFT methods. The research demonstrates the versatility of the approach in identifying different types of gear faults, including broken teeth and wear, by analyzing variations in sidebands in the current signal. By establishing thresholds from healthy gearbox data, the method successfully detected all six gear faults, showcasing its potential for robust and comprehensive fault detection in wind turbine gearboxes.

2.4 Techniques Comparison: Advantages and Disadvantages of Each Approach

The comparison of fault detection methods reveals distinct strengths and considerations for each approach. MSCNN exemplified by Jiang et al. (2018) and the MS-CNN by Zare and Ayati (2021), showcase impressive accuracy, emphasizing the significance of incorporating vibration signals at different scales for effective fault detection. However, the computational demands of CNNs, particularly deep architectures, pose challenges, necessitating substantial resources for training and inference, potentially limiting accessibility.

SVMs present an alternative with notable efficiency and accuracy, outperforming ANNs in terms of both speed and accuracy, as discussed by Santos et al. (2015). SVMs demonstrate effectiveness with moderate-sized datasets and excel in high-dimensional spaces, making them well-suited for tasks with a significant number of features.

Ensemble learning, featuring RF and XGBoost, emerges as a robust strategy, surpassing SVM in fault detection accuracy, as saw in the work of Zhang et al. (2018). The "ReliefF + XGBoost" algorithm of Z. Wu et al. (2020), specifically, showcases superiority over SVM and AdaBoost, achieving exceptional accuracy, precision, recall, and F1-Score.

Autoencoders, particularly the SMLDAE, proposed by Jiang, He, et al. (2017), present promising outcomes, outperforming traditional methods like Principal Component Analysis

and Stacked Autoencoders. The comparison highlights SMLDAE's efficacy in learning crucial features from data, leading to a significant enhancement in fault diagnosis. These findings suggest that opting for the SMLDAE approach could be preferable over the conventional DAE methods, as indicated by Jiang, Xie, et al. (2017).

In summary, the choice of a fault detection method should consider the specific requirements of the application, such as computational resources, dataset characteristics, and the desired level of interpretability. Each method excels in different aspects, and their effectiveness may vary depending on the context of wind turbine fault detection.

Chapter 3

The Problem Studied

3.1 Definition of the Problem

3.1.1 A brief explanation about the wind turbine components

The wind turbine, a sophisticated apparatus designed for harnessing wind energy and converting it into electricity, comprises a series of interconnected components working in harmony. At its base stands the tower, a robust structure providing support and elevating the turbine to heights where it can effectively capture prevailing winds. In the following section we are going to talked about some parts of the default wind turbine, to understand a bit how it work and which are the main components of this equipments.

Mounted atop the tower, the rotor serves as the initial point of interaction with the wind, incorporating both the hub and aerodynamically designed blades. These blades play a pivotal role in capturing kinetic energy from the wind, initiating the process of energy conversion. As the rotor turns, the low-speed shaft transfers the mechanical energy to the gearbox. Positioned strategically within the nacelle, the gearbox is a critical component responsible for increasing the rotational speed of the low-speed shaft to levels conducive for efficient electricity generation. The high-speed shaft, connected to the output of the gearbox, then transfers the enhanced rotational speed to the generator, housed within the nacelle.

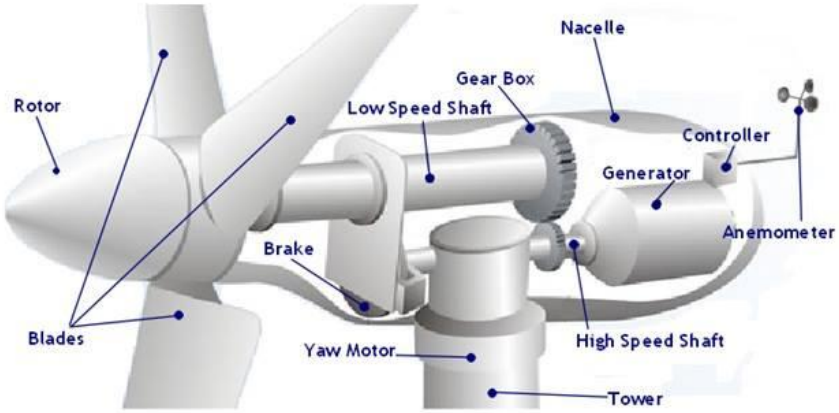

Figure 3.1: Generic HAWT Wind Turbine Components Example

The nacelle, a pivotal housing structure, encapsulates several key components, including

the gearbox, generator, and controller. Positioned atop the tower, it has the capability to rotate horizontally using the yaw motor, ensuring that the rotor consistently faces into the wind for optimal energy capture. The controller, serving as the central brain of the wind turbine, manages the operation of various components. It adjusts the pitch of the blades, controls the yaw system, and monitors overall turbine performance.

The system is further equipped with safety features such as brakes, integrated to control the rotation of the rotor and blades, especially during maintenance or extreme weather conditions. Anemometers, instrumental in measuring wind speed, contribute essential data to the controller, allowing the turbine to adjust its orientation and blade pitch dynamically for optimal efficiency.

In this intricate sequence of operations, each component plays a crucial role in the efficient conversion of wind energy into electricity, underscoring the complexity and sophistication of wind turbine technology.

3.1.2 The main technologies of Wind Turbine

Wind turbine technology encompasses two predominant designs: Horizontal Axis Wind Turbines (HAWT) and Vertical Axis Wind Turbines (VAWT), each distinguished by its unique characteristics and applications.

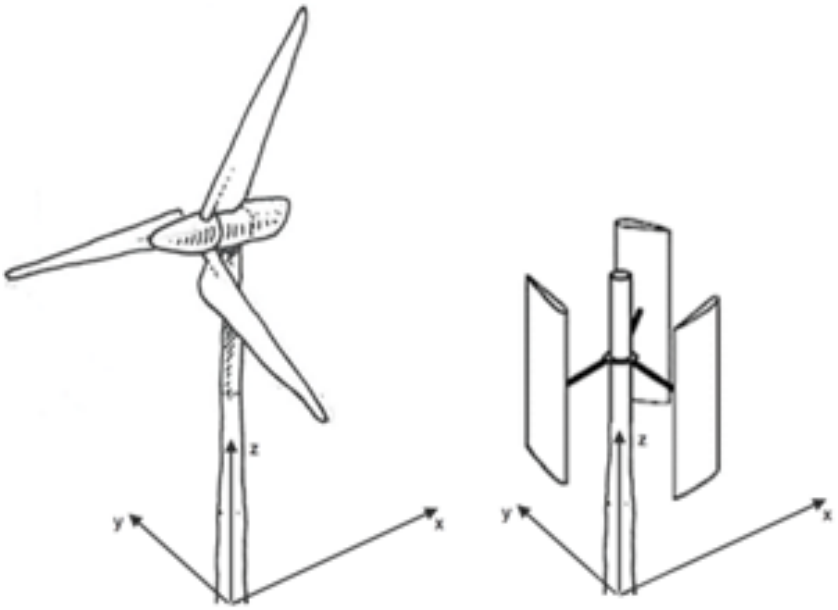

Figure 3.2: The HAWT (left) and VAWT (right) Technologies of Wind Turbines

HAWTs feature a horizontal rotor shaft, and their blades rotate around a horizontal axis. The rotor is positioned to face into the wind, with blades perpendicular to the wind direction. Typically configured with three blades made of materials like fiberglass or carbon composites, HAWTs are renowned for their efficiency and scalability. The three-blade design allows for a high tip speed ratio, contributing to enhanced energy conversion. HAWTs are widely adopted in utility-scale wind farms, ranging from small residential turbines to large megawatt-scale turbines. The mature technology of HAWTs ensures reliability and ease of maintenance.

In contrast, VAWTs have a vertical rotor shaft, and their blades rotate around a vertical axis. VAWTs can be further categorized into drag-based (Savonius) and lift-based (Darrieus) designs. Savonius VAWTs feature scoop-like blades, while Darrieus VAWTs incorporate aerofoil-shaped blades in configurations such as egg-beater or helix shapes. A notable advantage of VAWTs is their omni-directional wind capture, making them suitable for changing and turbulent wind conditions without the need for a yaw mechanism. Some VAWTs also exhibit lower wind start speeds, enabling electricity generation at lower wind speeds. However, VAWTs generally face challenges in achieving high efficiency, especially at larger scales, and their design complexities present engineering challenges.

In summary, the choice between HAWTs and VAWTs depends on specific project requirements and environmental conditions. While HAWTs dominate commercial wind farms due to their efficiency and well-established technology, VAWTs find application in scenarios where omni-directional wind capture and lower wind start speeds are advantageous. Ongoing research aims to refine both designs, contributing to the evolution of wind turbine technology for increased efficiency and versatility.

3.1.3 The Goal

The primary objective of this work is to develop a model capable of predicting failures in wind turbines by leveraging the available SCADA (Supervisory Control and Data Acquisition) data from turbines, which were provided by Enlitia. Specifically, this study focuses on detecting hydraulic failures, which are critical to the safe and efficient operation of wind turbines. By narrowing the scope to hydraulic issues, the aim is to enhance the model's performance in identifying these specific types of failures. Although detecting these failures in advance — before they occur —would be highly advantageous for maintenance planning and minimizing downtime, and would be a plus in this work, the core goal remains the accurate detection of these failures to improve the overall reliability and operational efficiency of the wind turbines.

3.2 Methodology Used

This dissertation employs the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework as its methodological foundation. CRISP-DM is a widely recognized approach in data mining that structures the process into six distinct phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. Each phase involves specific tasks and deliverables, ensuring a thorough and systematic approach to handling data mining projects across various industries, as detailed by Wirth and Hipp (2000).

CRISP-DM's iterative process is designed to allow for continuous refinement and optimization. Revisiting previous phases based on insights gained in later stages ensures that the data mining models are rigorously evaluated and improved. This flexibility is essential for addressing the complexities of real-world data and achieving robust and reliable outcomes. The methodology also emphasizes clear documentation and effective communication, which are crucial for replicating successful projects and fostering best practices.

The practical application of CRISP-DM in this dissertation includes detailed phases tailored to the specific context of the research. The business understanding phase involves defining project objectives and understanding the requirements from a business perspective.

Data understanding focuses on acquiring and exploring the data to identify quality issues and gain initial insights. During data preparation, the data is cleaned and transformed into an appropriate format for analysis. The modeling phase involves selecting and applying various analytical techniques to build predictive models. Evaluation entails assessing the performance of these models to ensure they meet the project's objectives. Finally, the deployment phase involves implementing the developed models in real-world scenarios to derive actionable insights.

Each chapter of this dissertation corresponds to a specific phase of the CRISP-DM methodology. The business understanding phase is discussed in the initial chapters, setting the stage for the research. Data understanding and preparation are covered in subsequent chapters, providing a comprehensive view of the data and the preprocessing steps. The modeling phase is detailed in the middle chapters, where different modeling techniques are explored and applied. Evaluation of the models is addressed in the later chapters, ensuring the models' effectiveness and reliability. The deployment phase will be outlined as part of the future steps by Enlitia, demonstrating the practical application of the research findings.

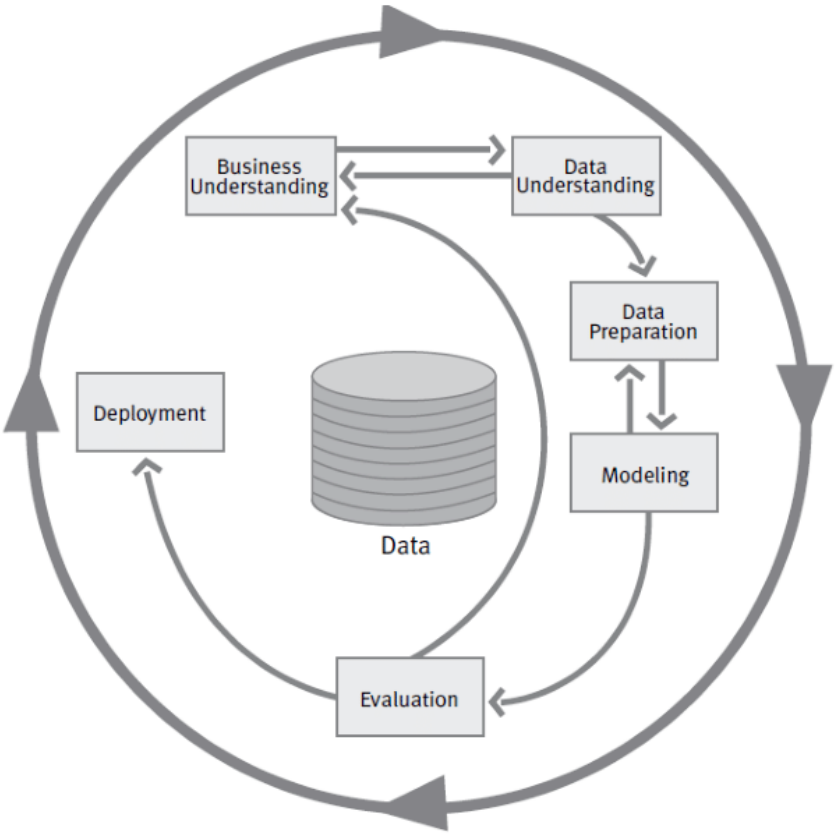

Figure 3.3: Steps of CRISP-DM Methodology

3.3 The data Provided

The dataset utilized in this study originates from wind turbines located in the same geographic space (reduce the variability between their data). Comprising five wind turbine datasets, it is structured into two distinct parts. Firstly, it encompasses a comprehensive array

of variables measuring the functionality of the wind turbines, including but not limited to oil pressure, gearbox RPM, and numerous other operational parameters, totaling more than 40 variables in all. These variables are recorded at 10-minute intervals, providing insights into the operational status and performance of the turbines over time. Secondly, the dataset incorporates a fault log documenting instances of turbine malfunctions and faults. Unlike the operational variables, fault data is logged precisely at the moment of occurrence, capturing real-time fault events automatically through the turbine's monitoring system.

Spanning from the year 2022 to 2024, the dataset offers a longitudinal perspective on wind turbine operations and faults, enabling a comprehensive analysis of operational trends and fault occurrences over time. With its extensive temporal and operational coverage, this dataset serves as a valuable resource for exploring fault detection methodologies and developing predictive models aimed at enhancing the reliability and efficiency of wind turbine operations. In the following sub-chapters we will understand better how they are structured.

3.3.1 Description of the Data

The dataset used in this dissertation is divided into two main parts: the data related to the sensors recorded from the wind turbines and the fault dataset where the faults are recorded.

Variables Dataset - sensors data

The Variables Dataset comprises sensor data from each wind turbine, recorded every 10 minutes. It includes 48 variables in total (but unique variables - some of them are captured as average, minimum, maximum, and standard deviation values, totaling the 48 mentioned). These variables are organized into three sub-datasets: ANA (turbine analogic measurements), TEMP (turbine temperature measurements), and WTG (turbine general measurements). Each sub-dataset uses "wt" as the primary key, which indicates the specific wind turbine. Additionally, two datetime columns are present: "datetime_utc" for the universal time and "datetime_local" for the local time of the wind turbine's location. The "datetime_utc" is primarily used for synchronization with the fault dataset. Below is a detailed table listing the variables, their descriptions, and their units.

In this dataset, some missing values are present, which will be discussed in detail in the Exploratory Data Analysis subchapter. This dataset's variables, along with their respective units and descriptions, can be visualized in Table 3.1. This table provides a comprehensive overview of each variable, the sub-dataset it belongs to, its unit of measurement, and a brief description.

Table 3.1: Sensors Variables, Units and Respective Dataset

Number	Dataset	Description of the Variable	Unit
1	ANA	Busbar voltage	V
2	ANA	Generator phase2 current	A
3	ANA	Grid voltage	V
4	ANA	Stator current	A
5	ANA	Hydraulic oil pressure	bar
6	TEMP	Gearbox bearing temperature	ºC
7	TEMP	Gearbox oil temperature	ºC
8	TEMP	Generator bearing temperature at drive end	ºC
9	TEMP	Generator bearing temperature at non drive end	ºC
10	TEMP	Generator slip ring temperature	ºC
11	TEMP	Hydraulic oil temperature	ºC
12	WTG	Wind speed	m/s
13	WTG	Active power	kw
14	WTG	Generator speed	rpm
15	WTG	External temperature	ºC
16	WTG	Wind direction	ºC
17	WTG	Nacelle temperature	ºC
18	WTG	Pitch angle	ºC
19	WTG	Rotor speed	rpm
20	WTG	Transformer frequency	Hz
21	WTG	Performance ratio	%
22	WTG	Power factor	0
23	WTG	Reactive power	kvar
24	WTG	WTG status	int

Faults Dataset

The Fault Dataset contains records of all faults for each wind turbine. Key information includes the "WTG" identifier (matching the "wt" code from the Variables Dataset), "TimeFrom" indicating when the fault started, "TimeTo" indicating when it ended, "EventText" describing the event that caused the fault, "EventCode" as a code corresponding to the event, and "ComponentDesc" which specifies the component of the wind turbine where the fault occurred. Faults are automatically recorded at the moment they occur, triggered by predefined parameter thresholds. In Table 3.2, two examples of faults and their representation in the dataset are illustrated.

Wind Turbines Used

In the analysis, data from five wind turbine parks located in the same region (due to the determination of the company that shared the data, it is not possible to disclose the location), and they are numbered as 8, 12, 15, 16, and 21 (number 16 do not have fault recording). For the training phase of the model, turbines 8 and 12 were selected due to

their lower incidence of faults, providing cleaner data conducive to effective training. This selection was designed to balance the need for a substantial amount of high-quality, fault-free data with computational efficiency. The decision to limit the training data to approximately two years was aimed at optimizing the model's performance on newer, unseen data while preventing excessively long training durations that could result from using the entire dataset available over several years. By choosing turbines with fewer historical faults and located in the same geographic area, the dataset encompasses a diverse range of operational conditions, facilitating a robust training process that is efficient yet comprehensive enough to generalize well across different operational scenarios.

Table 3.2: Sample of the Fault Dataset

WTG	TimeFrom	TimeTo	EventCode	EventText	Compon.
8	27/02/2022 23:47:00	28/02/2022 00:24:00	222	Very low pressure on hydraulic unit	Nacelle
12	12/07/2022 05:42:00	12/07/2022 05:47:00	203	Low pressure on hidraulic group	Nacelle

The structure of the fault dataset presents a challenge for modeling, as faults are recorded at the exact moment they occur, while sensor data is recorded every 10 minutes. This discrepancy means that fault times may not exactly match the timestamps in the sensor dataset. The solution to aligning these datasets and ensuring accurate model training will be discussed in detail in the "Data Preparation and Treatment" subchapter.

The datasets provide a comprehensive view of the operational parameters and faults of the wind turbines, enabling a detailed analysis and modeling to predict and manage turbine faults effectively. This structured approach to data collection and fault recording ensures that all relevant information is captured, providing a solid foundation for the subsequent analysis and predictive modeling tasks.

3.3.2 Exploratory Data Analysis

The Exploratory Data Analysis (EDA) section is a crucial stage of our investigation, aimed at providing a comprehensive initial look at the dataset before delving into more complex analyses. Through EDA, we assess the general patterns, identify anomalies, test hypotheses, and examine assumptions using both visual tools and summary statistics. This process not only helps in understanding the underlying structure of the data but also guides subsequent analytical decisions, ensuring that the insights and models developed later are built on a solid foundation of understanding.

Energy Production Analysis

In the analysis of the energy production for the wind turbines, we focus on the overall production of the entire wind park, without detailed data for individual turbines. The figure above illustrates the energy production over a two-year period, from January 2022 to May 2024. The data shows significant fluctuations in production, which is expected given the variability of wind conditions. Peaks of energy production can be observed intermittently, reaching values above 40,000 kW, indicating periods of strong wind activity. Conversely, there are moments of lower production, likely due to calm weather conditions or possible maintenance periods. The seasonal variations in production are visible, with generally consistent energy outputs, although there are occasional abrupt drops, potentially suggesting downtime or other operational interruptions. The x-axis presents clear timestamps, enabling a temporal understanding of the production patterns across this period. This analysis provides a comprehensive view of the wind park's operational performance, revealing both its capacity and the challenges associated with energy variability.

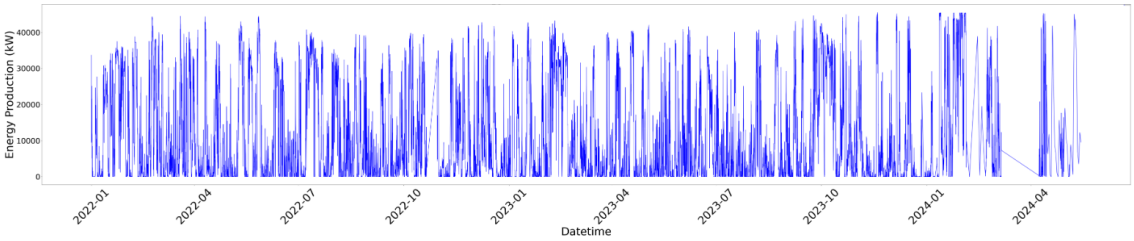

Figure 3.4: Energy Production Over Time

The boxplot below provides a visual summary of the wind park's energy production distribution. The orange line within the box indicates the median production, showing that the majority of production values are skewed toward the lower range, with the median falling slightly below 10,000 MW. The interquartile range (IQR), represented by the box, spans from approximately 0 MW to about 20,000 MW, meaning that 50% of the production values lie within this range. The whiskers extend upwards, with the top whisker reaching just over 40,000 MW, indicating higher but less frequent energy production levels. Notably, there are no visible outliers, suggesting that most of the energy production values fall within expected bounds. This distribution reflects the intermittent nature of wind energy, with a tendency for lower production values, possibly due to variable wind conditions or turbine downtimes, and the presence of outliers may indicate anomalies or faults.

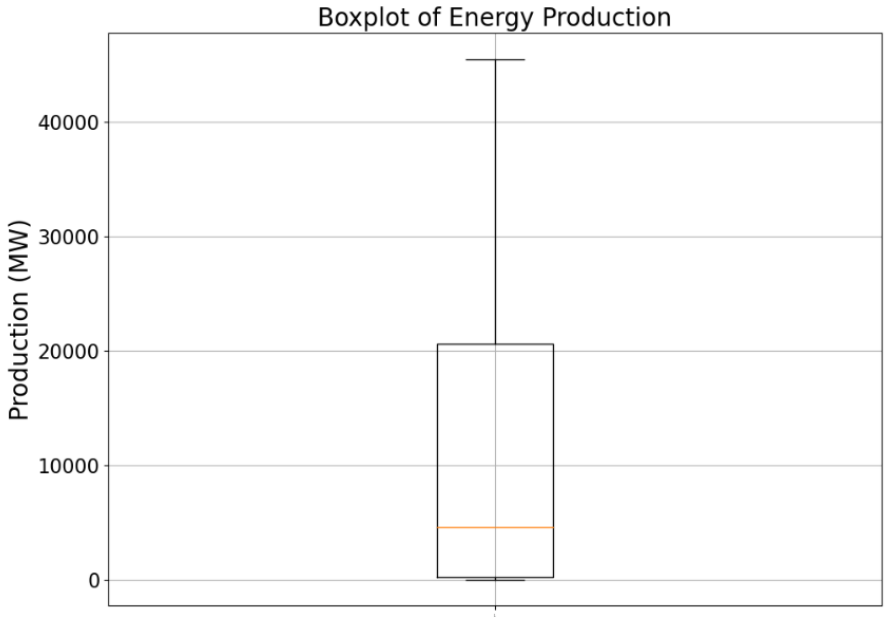

Figure 3.5: Boxplot of Energy Production

Summary and Initial Examination of the Variables

First, we focus on kurtosis and skewness to understand the distribution of our data more deeply. Skewness helps us identify if the data's distribution is symmetrical or if it leans towards higher or lower values, which is crucial for models that assume normality. Kurtosis offers insights into the tail-heaviness of the distribution, indicating the potential presence of outliers or extreme values that could affect statistical tests and model predictions. By analyzing these aspects, we can tailor our preprocessing strategies to mitigate any adverse effects on the model's performance and improve overall data handling.

Additionally, the Kolmogorov-Smirnov Test is employed to evaluate the normality of the dataset—a common assumption in many statistical modeling techniques. This nonparametric test compares the empirical distribution function of our sample with a specified distribution function, allowing us to ascertain the similarity between the observed data and the ideal normal distribution. Understanding whether our data conforms to this assumption is pivotal, as deviations from normality can lead to significant biases in conclusions drawn from statistical tests and analyses. The results from this test guide our decision-making regarding necessary data transformations or adjustments to meet model prerequisites.

Table 3.3: Statistical Properties of Sensor Variables in Dataset

Variable	Normally Distributed	Skewness	Kurtosis
Busbar voltage	No	Negatively Skewed	Tails heavier
Generator phase2 current	No	Positively Skewed	Tails lighter
Grid voltage	No	Negatively Skewed	Tails heavier
Stator current	No	Positively Skewed	Tails lighter
Hydraulic oil pressure	No	Negatively Skewed	Tails heavier
Gearbox bearing temperature	No	Negatively Skewed	Tails heavier
Gearbox oil temperature	No	Negatively Skewed	Tails heavier
Generator bearing temp. at DE	No	Positively Skewed	Tails heavier
Generator bearing temp. at NDE	No	Positively Skewed	Tails heavier
Generator slip ring temperature	No	Positively Skewed	Tails heavier
Hydraulic oil temperature	No	Negatively Skewed	Tails heavier
Wind speed	No	Positively Skewed	Tails heavier
Active power	No	Positively Skewed	Tails lighter
Generator speed	No	Negatively Skewed	Tails lighter
External temperature	No	Positively Skewed	Tails heavier
Wind direction	No	Positively Skewed	Tails heavier
Nacelle temperature	No	Negatively Skewed	Tails lighter
Pitch angle	No	Positively Skewed	Tails heavier
Rotor speed	No	Positively Skewed	Tails heavier
Transformer frequency	No	Negatively Skewed	Tails heavier
Performance ratio	No	Positively Skewed	Tails lighter
Power factor	No	Negatively Skewed	Tails lighter
Reactive power	No	Negatively Skewed	Tails lighter
WTG status	No	Positively Skewed	Tails heavier

The information of the table above reveals that none of the variables are normally distributed. Most variables exhibit significant skewness, with several showing a positive skew, indicating a longer tail on the right side of their distributions, which can affect the mean and pull it towards higher values. Moreover, the kurtosis values indicate heavier tails than a normal distribution for many variables, suggesting the presence of outliers or extreme values. These attributes, particularly the presence of heavier tails in the data, are crucial for the autoencoder's failures detection. High kurtosis values suggest more pronounced outliers, which can be indicative of fault conditions, if the ouliers are related to the failures that we're looking for. The autoencoder, can potentially improve its sensitivity to detect anomalies if trained without these outliers, improving its effectiveness in identifying and predicting failures within the operational data.

In the presented analysis, although only 24 unique sensor variables are described, the actual dataset comprises 49 variables due to the inclusion of various statistical measures for each sensor, such as average, minimum, maximum, and others. This expansion significantly enhances the granularity and detail available for analysis. However, a critical observation from the dataset is the distribution of missing values across these variables. As it's possible to see in the missing values table, notably, the gearbox bearing temperature and hydraulic oil temper-

ature variables exhibit a higher number of missing entries, with counts ranging from 13,413 to 26,889 missing values. This pattern suggests that certain sensor measurements are more susceptible to data loss or transmission issues, which is crucial to address, considering the total dataset contains 98,216 observations. The presence of missing values in these key areas could potentially impact the reliability and accuracy of any predictive maintenance models developed using this dataset, necessitating careful preprocessing to mitigate any biases or errors during model training.

Table 3.4: Missing Values in Variables dataset

Variable	Original Variable	# Missing Values
avg_metwndspd_ms	Wind speed	10.028
avg_wtgp_kw	Active power	10.032
avg_genspd_rpm	Generator speed	10.028
avg_metextmp_degc	External temperature	10.028
avg_metwndspd_dir	Wind direction	10.028
avg_nacelle_degc	Nacelle temperature	10.033
avg_rotpitchanga_deg	Pitch angle	10.028
avg_rotspd_rpm	Rotor speed	10.028
avg_trffrec_hz	Transformer frequency	10.028
avg_wtgcapf_perc	Performance ratio	10.029
avg_wtgcos_phi	Power factor	10.028
avg_wtgq_kvar	Reactive power	10.028
avg_wtgst_int	WTG status	10.046
avg_gbx_bear_degc	Gearbox bearing temperature	13.413
min_gbx_bear_degc	Gearbox bearing temperature	13.659
max_gbx_bear_degc	Gearbox bearing temperature	13.674
desv_gbx_bear_degc	Gearbox bearing temperature	20.476
n_gbx_bear_degc	Gearbox bearing temperature	0
avg_gbx_oil_degc	Gearbox oil temperature	13.429
min_gbx_oil_degc	Gearbox oil temperature	13.652
max_gbx_oil_degc	Gearbox oil temperature	13.649
desv_gbx_oil_degc	Gearbox oil temperature	19.387
n_gbx_oil_degc	Gearbox oil temperature	0
avg_gen_bear_de_degc	Generator bearing temp. at DE	13.412
min_gen_bear_de_degc	Generator bearing temp. at DE	13.616
max_gen_bear_de_degc	Generator bearing temp. at DE	13.662
desv_gen_bear_de_degc	Generator bearing temp. at DE	25.485
n_gen_bear_de_degc	Generator bearing temp. at DE	0
avg_gen_bear_nde_degc	Generator bearing temp. at NDE	13.412
min_gen_bear_nde_degc	Generator bearing temp. at NDE	13.625
max_gen_bear_nde_degc	Generator bearing temp. at NDE	13.611
desv_gen_bear_nde_degc	Generator bearing temp. at NDE	24.077
n_gen_bear_nde_degc	Generator bearing temp. at NDE	0
avg_gen_sring_degc	Generator slip ring temperature	13.413
min_gen_sring_degc	Generator slip ring temperature	13.611
max_gen_sring_degc	Generator slip ring temperature	13.641
desv_gen_sring_degc	Generator slip ring temperature	14.024
n_gen_sring_degc	Generator slip ring temperature	0
avg_hyd_oil_degc	Hydraulic oil temperature	13.413
min_hyd_oil_degc	Hydraulic oil temperature	13.556
max_hyd_oil_degc	Hydraulic oil temperature	13.553
desv_hyd_oil_degc	Hydraulic oil temperature	26.889
n_hyd_oil_degc	Hydraulic oil temperature	0
avg_busbar_v	Busbar voltage	14.211
avg_gen_l2_a	Generator phase2 current	14.211
avg_grid_v	Grid voltage	14.211
avg_rotorcurr_a	Rotor current	14.211
avg_statorcurr_a	Stator current	14.211
avg_hyd_oil_bar	Hydraulic oil pressure	14.211

Correlation Analysis

In the analysis of the dataset, both Spearman and Kendall correlations were initially considered to evaluate the relationships between variables. However, the decision was made to primarily use Spearman correlation for further analyses. This choice was influenced by the findings in the previous section, which demonstrated that the variables do not follow a normal distribution. Spearman correlation is particularly suitable for this context as it is a non-parametric measure that can identify monotonic relationships without assuming normality in the data distribution. While Kendall's correlation was also performed, it was ultimately deemed less practical due to its greater computational demands, especially given the large size of the dataset. Spearman's method, therefore, provides a robust approach for understanding the correlations within this non-normally distributed data, ensuring that the analysis remains efficient and relevant.

The analysis of the dataset revealed several significant correlations between the operational variables and fault occurrences, as determined through Spearman's correlation coefficient. The coefficients ranged from positive to negative, indicating varying degrees of association with fault events. Notably, variables like avg_rotpitchanga_deg exhibited a positive correlation of 0.22, suggesting that as pitch angle increases, the likelihood of a fault occurrence also increases. Conversely, variables such as avg_busbar_v and avg_hyd_oil_bar showed strong negative correlations of -0.221679 and -0.217573, respectively, indicating that higher values in these variables are associated with a lower likelihood of faults.

Due to the extensive number of variables analyzed, the full correlation matrices (Spearman and Kendall) could not be included in the main text of the dissertation and were relegated to the appendix. The tailored selection of variables based on the criteria will be explained after, is placed in the correlation matrix below. .

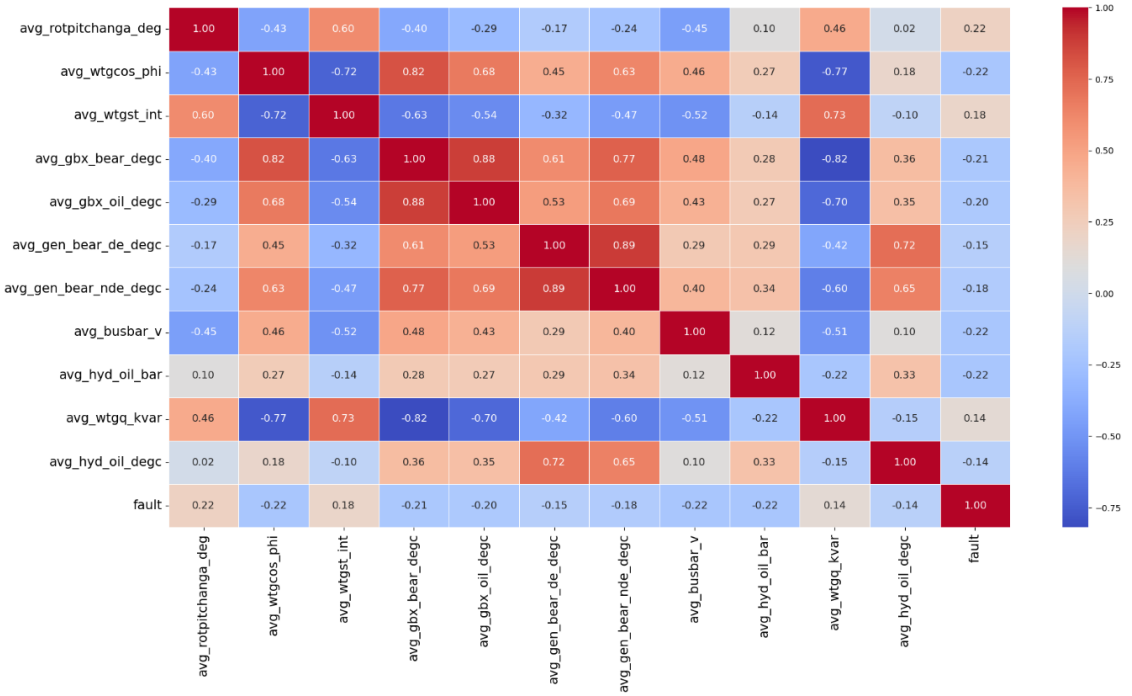

Figure 3.6: Spearman's Correlation Matrix of the Chosen Variables

The Spearman correlation analysis reveals predominantly negative correlations between

the selected variables and the occurrence of faults, with only a few variables like $avg\_rotpitchanga\_deg$ and avg_wtgst_int displaying positive correlations. Most variables are negatively correlated with fault occurrences, suggesting that higher readings in parameters such as avg_busbar_v and $avg\_hyd\_oil\_bar$ generally indicate normal operating conditions, whereas lower values may signal potential faults. The magnitude of these correlations, while significant, does not typically exceed moderate strength, indicating that while there is a clear relationship between these variables and faults, it is not exceptionally strong. This suggests a complex interplay of factors influencing fault occurrences, necessitating a nuanced approach to model training and predictive maintenance. Furthermore, from the 49 types of variables available, only 11 were selected for further analysis based on the criteria that will be explained after, focusing on those with the most significant correlation to faults.

Chapter 4

Modeling

4.1 The Theory of the Algorithm used - LSTM Autoencoder

In this study, we utilized Long Short-Term Memory (LSTM) Autoencoders to model and detect anomalies within time-series data. LSTM Autoencoders, a specialized type of recurrent neural network, are designed to learn efficient representations of sequential data by encoding the input into a compressed form and then reconstructing it. This approach leverages the LSTM architecture's ability to capture long-term dependencies and temporal patterns, making it particularly effective for analyzing time-series data. While the LSTM Autoencoder serves as a robust tool for anomaly detection, we also employed additional techniques to further enhance performance. These complementary methods, which will be discussed in detail in the following sections, were crucial in achieving a more comprehensive and accurate modeling approach.

4.1.1 LSTM

As saw in Wei et al. (2023), LSTM, which stands for Long Short-Term Memory, is an advanced form of Recurrent Neural Networks (RNN). While RNNs are designed with "short-term memory," allowing them to leverage recent information for the current task, LSTMs extend this functionality by introducing a "long-term memory" mechanism. This allows the network to retain and utilize information over much longer sequences, making LSTMs particularly effective for tasks involving time-series data.

An LSTM unit consists of several key components: a cell, an input gate, an output gate, and a forget gate. The cell acts as the memory unit, storing values over varying time intervals, while the gates control the flow of information into and out of the cell, thus determining what information should be remembered, updated, or discarded.

The Cell State in an LSTM unit serves as the network's long-term memory, storing a sequence of past information. Meanwhile, the Previous Hidden State represents the network's output from the previous time step, functioning as its short-term memory. The Input Data at each time step is the current input value being processed by the network. These components work together to manage the flow of information through the LSTM, enabling it to effectively model temporal dependencies in sequential data.

Forget Gate

The forget gate's primary role is to determine which parts of the cell state are important, given the previous hidden state and the new input. This is achieved by feeding these two inputs into a neural network, which produces a vector with elements ranging from [0, 1] using a sigmoid activation function. The forget gate is trained to output values close to 0 for irrelevant information and values closer to 1 for relevant information. These outputs are then multiplied with the previous cell state. The operation of the forget gate can be mathematically expressed as:

$$f_t = \sigma(w_f[H_{t-1}, X_t] + b_f) \quad (1)$$

where σ is the sigmoid function, w_f and b_f represent the weights and biases of the forget gate, and H_{t-1} and X_t denote the hidden state and current input, respectively.

Input Gate

The input gate has a dual function. First, it evaluates whether the new information, derived from the previous hidden state and current input, should be retained in the cell state. Second, it determines what new information should be added. This process involves generating a memory update vector $\tilde{C}_t$ by combining the previous hidden state and new input data. A tanh activation function is used to constrain this vector's elements within the range [-1, 1]. This operation is captured in Equation 2:

$$\tilde{C}_t = \tanh(w_c[H_{t-1}, X_t] + b_c) \quad (2)$$

where w_c and b_c are the input gate's weight matrices and biases.

In parallel, the input gate determines which components of the new input, in the context of the previous hidden state, should be remembered. Like the forget gate, the input gate uses a sigmoid activation function to produce a vector of values in the range [0, 1]. This is expressed in Equation 3:

$$i_t = \sigma(w_i[H_{t-1}, X_t] + b_i) \quad (3)$$

where w_i and b_i are the weight matrices and biases for the input gate. The results from these two operations are then multiplied and added to the cell state, thereby updating the network's long-term memory as described in Equation 4:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (4)$$

Output Gate

Once the long-term memory has been updated, the output gate is responsible for determining the new hidden state. This gate uses the newly updated cell state, the previous hidden state, and the current input to make this decision. The hidden state and input data are passed through a sigmoid activation function to produce a filter vector o_t, as shown in Equation 5:

$$o_t = \sigma(w_o[H_{t-1}, X_t] + b_o) \quad (5)$$

where w_o and b_o are the weight and bias of the output gate. The cell state is then passed through a tanh activation function to squash its values into the range [-1, 1]. This "squished"

cell state is then multiplied by the filter vector to create a new hidden state H_t, as outlined in Equation 6:

$$H_t = o_t \odot \tanh(C_t) \quad (6)$$

The updated cell state C_t and the new hidden state H_t are then passed to the next LSTM unit. This process repeats until all input data across the time series have been processed.

The effectiveness of LSTM in capturing temporal dependencies and handling sequential data has been well-demonstrated, including in studies such as "LSTM-Autoencoder based Anomaly Detection for Indoor Air Quality Time Series Data," where LSTM models were successfully applied for detecting anomalies in time-series data.

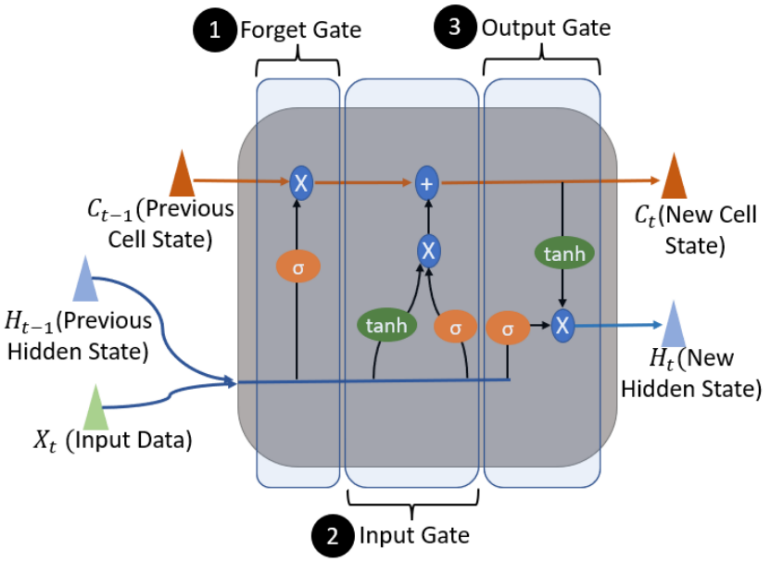

Figure 4.1: How LSTM Works. Source: Wei et al. (2023)

4.1.2 Autoencoder

As discussed by Wei et al. (2023), an autoencoder is an unsupervised neural network designed to learn efficient codings of unlabeled data by compressing the input into a lower-dimensional representation and then reconstructing the original input from this compressed representation. The network is typically composed of three main parts: an encoder, a bottleneck (or latent space), and a decoder.

Encoding

In the encoding stage, the input data $x \in \mathbb{R}^m$ is mapped to a lower-dimensional representation h through a function f_1 involving a weight matrix W_i and a bias vector b_i:

$$h = f_1(W_i x + b_i)$$

Decoding

In the decoding stage, the low-dimensional representation h is mapped back to a reconstruction of the original input $\hat{x}$ using another function f_2, which also involves a weight matrix

W_j and a bias vector b_j:

$$\hat{x} = f_2(W_j h + b_j)$$

Reconstruction Loss

The objective of the autoencoder is to minimize the difference between the input x and the reconstructed output $\hat{x}$. This difference is often quantified using a loss function L, such as Mean Absolute Error (MAE):

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^{n} |\hat{x}_i - x_i|$$

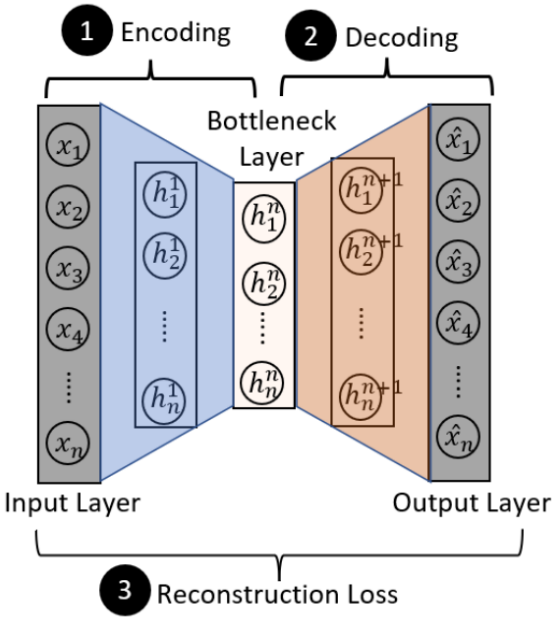

Figure 4.2: How Autoencoder Works. Source: Wei et al. (2023)

4.1.3 LSTM-Autoencoder

In a typical deep learning setup, an Autoencoder is used to learn a compressed representation of the input data by encoding it into a latent space and then decoding it back to reconstruct the original input. While Autoencoders are powerful for tasks like dimensionality reduction and anomaly detection, they are not inherently designed to handle sequences of data where the order and temporal relationships between data points are crucial. This is where the Long Short-Term Memory (LSTM) network comes into play.

LSTM networks are a specialized type of Recurrent Neural Network (RNN) capable of learning long-term dependencies in sequences. They are particularly well-suited for tasks involving time-series data because they can maintain and update a hidden state that reflects information from previous time steps, allowing them to capture patterns over time.

As the Wei et al. (2023) brought, the LSTM-Autoencoder model combines these two architectures, leveraging the strengths of both. The Autoencoder framework provides the

structure for compressing and reconstructing data, while the LSTM units within the encoder and decoder are responsible for handling the sequential nature of the input. In the following, it's going to be possible to understand some advantages of the model.

LSTM as the Encoder

In the LSTM-Autoencoder, the encoder is composed of LSTM units rather than traditional feedforward layers. This allows the encoder to not only compress the data but also to capture the temporal dependencies within the sequence. The LSTM encoder processes the input sequence one time step at a time, updating its hidden state to reflect the information learned from the entire sequence.

Latent Space Representation

The output of the LSTM encoder is a fixed-size vector that represents the entire input sequence in a compressed form. This latent space captures the most important features of the sequence, including the temporal relationships that the LSTM units have learned.

LSTM as the Decoder

The decoder is also composed of LSTM units. It takes the compressed latent space representation and attempts to reconstruct the original sequence. The LSTM decoder generates each time step of the output sequence, using its hidden state to maintain the temporal continuity of the sequence.

Reconstruction Loss and Anomaly Detection

The effectiveness of this combination lies in the reconstruction loss—the difference between the original sequence and the one produced by the decoder. Since the LSTM encoder and decoder have learned to capture and replicate the temporal dependencies in normal sequences, sequences that deviate from this learned normal pattern will result in a higher reconstruction loss. This property is exploited for anomaly detection: if a sequence produces a reconstruction loss above a certain threshold, it is flagged as anomalous.

Advantages of the Approach

By combining LSTM networks with the Autoencoder architecture, the model gains the ability to handle sequential data effectively, making it a powerful tool for tasks like anomaly detection in time-series data. The LSTM units enhance the Autoencoder by enabling it to understand and reconstruct the temporal dynamics within the data, which a standard Autoencoder would struggle to achieve on its own.

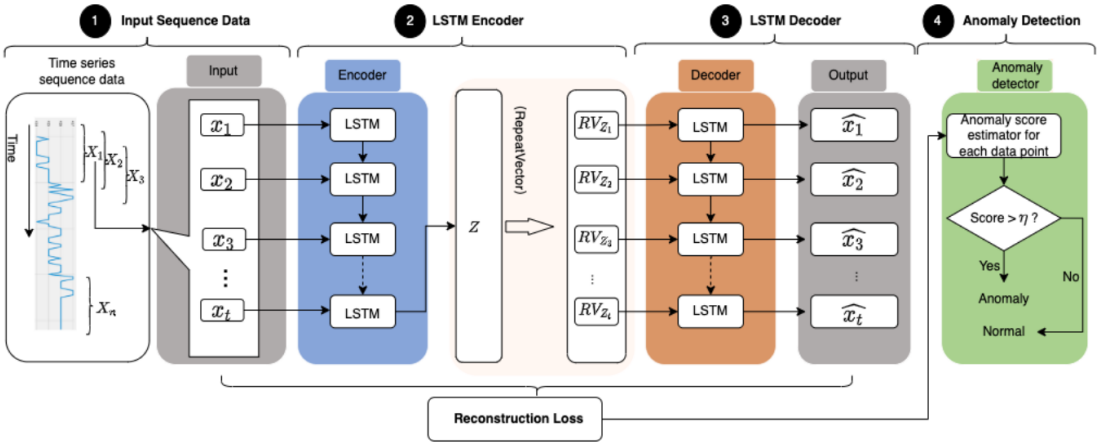

Figure 4.3: LSTM-Autoencoder Schema. Source: Wei et al. (2023)

4.1.4 Other Techniques Used to Outperform LSTM Autoencoder

In the quest to enhance the performance of LSTM Autoencoders, additional techniques have been explored and integrated into the modeling process. These techniques aim to address specific challenges, such as noise in the data and the need for more refined feature extraction, thereby improving the overall robustness and accuracy of the model. Two prominent methods that have shown significant promise in this regard are the introduction of Gaussian noise during training and the application of low-pass filters to preprocess the data. These techniques serve to complement the LSTM Autoencoder, enhancing its ability to learn meaningful patterns from the data while mitigating the effects of noise and irrelevant features. This section will delve into these methods, discussing the theory behind them and how they contribute to the improved performance of the LSTM Autoencoder.

Gaussian Noise

Gaussian noise, also known as normal noise, is a type of statistical noise characterized by a probability density function (PDF) that follows a normal distribution, often referred to as a Gaussian distribution. This means that the noise values are distributed in a way that most of the values are centered around the mean, with the probability of higher deviations decreasing symmetrically as you move away from the mean. Gaussian noise is commonly used in signal processing and machine learning to simulate real-world noise and randomness in data. It can help in testing the robustness of models by introducing random variations that the models must learn to handle, thereby improving their ability to generalize to new, unseen data.

As discussed by Nie et al. (2020), introducing Gaussian noise during the training process helps the model improve its robustness and generalization capabilities. By adding noise to the input signals, the model learns to reconstruct the original signal from a perturbed version, which forces it to focus on the most relevant and discriminative features rather than memorizing the training data. This process acts as a form of regularization, preventing overfitting and enabling the model to perform better on unseen, noisy data. As a result, the model becomes more effective in distinguishing between normal and faulty conditions, even in the presence of noise, which is common in real-world scenarios.

Low Pass Filter

As saw in Ribeiro, Pereira, and Gama (2016), a low-pass filter is a technique used to smooth out abrupt changes in a signal by attenuating high-frequency variations, which helps in reducing noise and focusing on the more stable, underlying patterns in the data. In the context of anomaly detection, the low-pass filter is applied to the output of point-anomaly detection algorithms to detect sequences of anomalies, rather than reacting to individual outliers that may be due to transient noise or irrelevant fluctuations. The filter works by gradually adjusting the output based on the current and previous inputs, with a smoothing parameter that controls how quickly the filter responds to changes in the signal. This process helps in reducing false alarms by filtering out spurious, isolated anomalies and instead highlighting persistent deviations that are more likely to indicate a genuine fault.

4.2 Data Preparation and Treatment

This chapter discusses the steps taken to prepare the data, including variable selection, dataset preparation, and the methods used to ensure the model could be effectively trained and tested. It covers the handling of missing values, the rationale behind the chosen imputation techniques, and the strategies employed to align the datasets for accurate and robust predictive modeling. These preparatory steps are crucial for building a reliable and efficient model for fault detection in wind turbines.

4.2.1 Handling Missing Values

As identified in the Exploratory Data Analysis (EDA), the dataset contains some missing values that need to be addressed before proceeding with modeling. Various techniques can be employed to handle missing data, such as dropping rows with missing values, using machine learning algorithms like k-nearest neighbors (KNN) for imputation, or applying moving averages.

Dropping rows with missing values was considered but deemed impractical due to the significant reduction in data volume it would cause. Many rows contain at least one critical variable with missing values, and removing these rows would greatly diminish the dataset's size, adversely affecting the model's performance.

Using machine learning algorithms to impute missing values was also explored. Although algorithms like KNN can be effective, they did not yield satisfactory results in this case. The imputed values sometimes adversely impacted the prediction of failures. Discussions with Enlitia revealed that they had similar experiences, likely due to insufficient correlations among some variables, which reduced the effectiveness of such imputation techniques.

The chosen approach was to use moving averages to fill the missing values. A Python script was developed to create a loop that first fills missing values using the previous 30 minutes of observations, aiming to capture the growth of any disturbances occurring at that moment. If the previous 30 minutes of data were insufficient to calculate the average, the script used the subsequent 30 minutes of data to fill the gaps. This method effectively addressed all missing values and provided better results than KNN imputation.

In failure detection, outliers are often the most critical data points for identifying faults. Therefore, the average was used instead of the median because the average is more affected by

outliers, providing a better fulfillment of the missing values. Using the median could reduce the impact of outliers and potentially miss important signals related to failures.

Additionally, methods like taking the average of the whole hour or every 30 minutes were tested. However, these approaches significantly reduced the dataset size, which is not ideal for neural networks like LSTM Autoencoders. Consequently, the results from these methods were inferior to those obtained using the moving averages from the previous or future 30-minute intervals.

This solution effectively mitigates the issue of missing data, ensuring a more robust and reliable dataset for the subsequent modeling phase. The detailed approach to handling missing values sets a solid foundation for developing accurate and effective predictive models in the next steps of the project.

4.2.2 Choosing the Faults and Variables

In this section, we outline the process of selecting the specific types of variables and the faults that our model aims to detect.

The Faults

As previously mentioned, the focus of this work is on identifying hydraulic-related faults within the wind turbine dataset. This choice was made to narrow the scope of the analysis and to provide a detailed examination of faults that are critical to the operation of wind turbines.

To achieve this, we specifically targeted the detection of faults associated with the following "EventCode" identifiers: 203 (Low pressure on hydraulic group), 205 (Low level of hydraulic group refrigeration oil), 207 (Hydraulic oil high temperature), 209 (Water level low), 222 (Very low pressure on hydraulic unit), and 223 (Very high temperature of hydraulic unit oil). These events were selected because they directly pertain to the hydraulic systems of the turbines, which are crucial for maintaining operational integrity.

During the testing phase, all other types of faults, associated with different "EventCodes," were excluded from the dataset. This was done to ensure that the model is trained and evaluated solely on the detection of hydraulic faults, without the interference of unrelated events. By focusing on these specific faults, the model can more effectively learn and identify patterns related to hydraulic system failures, aligning with the overall objective of this study.

The Variables

Given the insights from the correlation analysis section, the modeling process focused on selecting variables with a correlation score above 0.12, either positive or negative, to maximize the predictive accuracy of the LSTM autoencoder. Following this, we analyzed the correlation between the selected variables themselves, removing any pairs of variables with a correlation greater than 0.9, while retaining the one most strongly correlated with the fault. This approach was refined by first testing the model using all variables chosen in the first step. However, applying the second step significantly improved the results, not only in terms of recall and precision but also in reducing overfitting, as observed in the comparison of the training and validation curves.

Based on this refined selection process, the following variables were chosen for modeling: "avg_rotpitchanga_deg", "avg_wtgcos_phi", "avg_wtgst_int", "avg_gbx_bear_degc",

”avg_gbx_oil_degc”, ”avg_gen_bear_de_degc”, ”avg_gen_bear_nde_degc”, ”avg_busbar_v”, ”avg_hyd_oil_bar”, ”avg_wtgq_kvar”, and ”avg_hyd_oil_degc”. These variables demonstrated the most significant correlations with fault occurrences, making them the most promising for effective fault prediction.

From an initial pool of 49 variables, this rigorous selection process ultimately reduced the number to just 11 key variables. This streamlined set of features was expected to improve the model's performance by focusing on the most relevant data while reducing noise from less predictive variables.

4.2.3 Connecting the Fault Dataset and the Variables Dataset

In our endeavor to enhance fault detection in turbine operations, this chapter explores the intricate process of integrating variable recordings, taken every ten minutes, with sporadically timed fault data. Given the temporal discrepancies between these datasets, we developed methods to align and clean the data, ensuring the integrity of inputs for training our predictive models. This included rounding fault times to the nearest interval and excluding data around fault periods to prevent the LSTM autoencoder from learning anomalies related to faults. By refining these datasets, we established a solid foundation for detecting anomalies based on normal operational patterns, ensuring the autoencoder is trained on clean and representative data. This section details the methodologies used for data synchronization and the rationale behind establishing a fault exclusion window to optimize model training.

Creating fault identification

In our analysis, we utilized two primary data sources: a dataset of variables recorded every 10 minutes and a fault dataset that documents fault events at precise timestamps. Integrating these datasets presented an initial challenge due to their differing temporal resolutions, as the variables are logged at regular intervals, whereas faults could occur at any moment. Both datasets, however, are synchronized to UTC time, which provided a common ground for alignment.

To synchronize the fault events with the variables dataset, we adopted a strategy of rounding the timestamps of fault events to the nearest previous 10-minute interval. For instance, if a fault was recorded at 10:47:07, we adjusted this timestamp to 10:40:00. This adjustment ensured that each fault event corresponded to a specific entry in the variables dataset.

Another challenge was the duration of fault events, which could span multiple intervals in the variables dataset. To handle this, we marked all variable data falling within the start and end times of a fault as occurring during a fault. This was achieved through a Python loop that checked each timestamp against fault intervals and assigned a binary indicator to a new column named ”fault” within the variables dataset. A value of '1' was assigned to timestamps during a fault event, and '0' was assigned otherwise.

This rigorous process of aligning and marking the data was crucial for our subsequent analyses, particularly for training predictive models that rely on accurately labeled data to detect fault conditions effectively. The models could thus learn from a precisely annotated dataset where each data point was clearly labeled as normal operation or fault, enhancing their ability to generalize and predict future fault occurrences based on observed patterns.

Fault Exclusion Window

In order to effectively train the LSTM autoencoder to detect anomalies in operational data without being influenced by known fault conditions, it was essential to provide the model with a dataset devoid of any fault-related influences. To achieve this, I established a fault exclusion window around each recorded fault incident within the dataset.

This exclusion window was designed to remove not only the data points during the fault but also those immediately before and after the fault occurrence. The rationale behind this was to eliminate any data that could potentially be tainted by the onset or aftermath of the fault, thereby ensuring that the autoencoder learns only from data representative of normal turbine operation.

After experimenting with various durations for these exclusion windows, it was determined that a span of 3 hours before and after each fault event provided the optimal balance. This duration effectively cleansed the training dataset of fault-related anomalies without excessively reducing the amount of data available for model training. This careful selection of the exclusion window maximized the quality of the input data for the autoencoder, enhancing its ability to generalize from genuinely non-fault conditions.

4.2.4 Low Pass Filter Implementation

In the "Low Pass Filter Implementation" section, we detail the strategic use of a low-pass filter to enhance the anomaly detection process in time series data from operational wind turbines. According to Ribeiro et al. (2016), a low-pass filter effectively smooths out rapid fluctuations in the data, which are often attributable to noise, by attenuating high-frequency components beyond a certain threshold. This method is particularly beneficial for emphasizing significant shifts that could indicate potential faults.

The chosen cutoff frequency was set at 0.0003, selected after evaluating a range from 0.0001 to 0.0007. This range was determined to be optimal; frequencies beyond 0.0007 overly dampened the signal, potentially masking pertinent anomalies. The sampling frequency, defined as one sample every 600 seconds (10 minutes), corresponds to the data collection interval, ensuring that the filter's response aligns with the natural data recording cadence.

Applying this filter prior to normalization was a deliberate decision to mitigate the influence of extreme outliers or noise on the scaling process. By smoothing the data first, the normalization is applied to a dataset that more accurately reflects the underlying operational patterns, thus enhancing the effectiveness of subsequent anomaly detection performed by the LSTM autoencoder.

This preprocessing step is crucial for preparing the data in a manner that optimizes the learning and generalization capability of the predictive model, allowing it to focus on the most indicative trends and deviations from normal operational patterns.

4.2.5 Transforming the Data

In the data preparation stage, normalization plays a pivotal role in conditioning the dataset for the LSTM autoencoder. By using the MinMaxScaler from Scikit-learn, we scale each feature X in the dataset to a range between 0 and 1. This scaling is achieved by applying the

following transformation:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

where X_{min} and X_{max} are the minimum and maximum values of the feature across the dataset, respectively. This transformation ensures that all input features contribute equally to the model's learning process, preventing any one feature with larger numerical ranges from dominating the learning process.

Normalization also facilitates faster convergence during training by ensuring that the gradient descent steps are taken in proportion to the scales of the features. It's particularly important when working with deep learning models like LSTMs, which are sensitive to the scale of input data. Maintaining consistent scaling across both training and testing datasets by fitting the scaler only on the training data and then transforming both datasets ensures that the model is not biased or misled by different feature scales during validation or testing phases.

4.3 Implementation of LSTM Autoencoder

This chapter introduces the implementation of an LSTM Autoencoder, designed to identify anomalies in time-series data effectively. The architecture of the model leverages layers of LSTM units in both bidirectional and unidirectional formats to capture temporal dependencies within the data. The core mechanism revolves around encoding data into a compressed representation through a series of LSTM layers and subsequently reconstructing it, enabling the model to highlight deviations from normal patterns. The integration of a GaussianNoise layer at the outset helps to enhance the model's robustness by mimicking real-world disturbances. This setup ensures that the autoencoder is not only adept at handling the inherent noise in operational data but also excels in generalizing across unseen scenarios, making it particularly valuable for predictive maintenance tasks.

4.3.1 The LSTM Autoencoder

The specialized LSTM Autoencoder model developed for this study encapsulates a series of layers that effectively encode and decode time-series data to identify patterns and anomalies. The architecture comprises the following components:

1. **Input Layer**: This initial layer accommodates the input sequences, which are detailed by specific timesteps and features, setting the stage for subsequent processing.

2. **Bidirectional LSTM Layers**: The encoding phase begins with a 64-unit LSTM layer, followed by a 32-unit LSTM layer, both leveraging the 'relu' activation function. These layers are instrumental in capturing intricate dependencies within the data by encoding the input into an abstracted representation.

3. **Bottleneck Layer**: A crucial 16-unit LSTM layer serves as the network's bottleneck, condensing the data into a dense representation. This layer is key to reducing dimensionality and focuses on capturing essential features for effective data reconstruction.

4. **Repeat Vector**: Following the bottleneck, the encoded data is replicated across the required timesteps to facilitate the decoding process.

5. **Decoding LSTM Layers**: The decoding sequence is initiated at a 16-unit LSTM layer and progresses through 32 and 64-unit layers, mirroring the encoding steps but in reverse. This phase reconstructs the temporal data back to its original configuration.

6. **Output Layer**: The reconstruction culminates at the TimeDistributed Dense layer, which applies a fully connected operation to each timestep. This ensures that the output sequence meticulously parallels the input data's structure.

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 1, 15)	0
lstm (LSTM)	(None, 1, 64)	20,480
lstm_1 (LSTM)	(None, 1, 32)	12,416
lstm_2 (LSTM)	(None, 16)	3,136
repeat_vector (RepeatVector)	(None, 1, 16)	0
lstm_3 (LSTM)	(None, 1, 16)	2,112
lstm_4 (LSTM)	(None, 1, 32)	6,272
lstm_5 (LSTM)	(None, 1, 64)	24,832
time_distributed (TimeDistributed)	(None, 1, 15)	975

Total params: 70,223 (274.31 KB)

Trainable params: 70,223 (274.31 KB)

Non-trainable params: 0 (0.00 B)

Figure 4.4: Model Summary of LSTM Autoencoder from Python Code

This architecture allows the LSTM Autoencoder to adeptly discern anomalies and deviations from expected patterns through its sequential and precise reconstruction capabilities. By compressing and then decompressing the input data, the model effectively learns both short-term and long-term dependencies within the dataset, a vital feature for predictive maintenance applications where detecting anomalies early can significantly reduce operational risks and enhance maintenance strategies.

4.3.2 Gaussian Noise Implementation

In the initial stage of the LSTM autoencoder, a GaussianNoise layer with a standard deviation of 0.1 is applied to the input data. This layer injects random noise following a Gaussian distribution into each input feature, effectively simulating the presence of real-world disturbances that the model might encounter in operational data. This process is crucial for preventing the model from overfitting to the noise-free training data and promotes the development of a more generalizable model capable of handling unseen variations in new data.

In the model's first layer, Gaussian noise is added directly to the raw input data. This intentional perturbation helps in training the autoencoder to focus on underlying patterns rather than overfitting to the idiosyncrasies of the training set. The noise level, quantified by a standard deviation of 0.1, strikes a balance between adding sufficient noise to challenge the model and maintaining the integrity of the underlying data structure.

According to research discussed in Nie et al. (2020), introducing Gaussian noise is a recognized technique to enhance the robustness of neural networks. It forces the network to learn to denoise and reconstruct the original input from corrupted versions, thereby honing its ability to identify and disregard irrelevant variations in the data. This is particularly advantageous in anomaly detection tasks where distinguishing between normal fluctuations and true anomalies is critical.

The strategic inclusion of Gaussian noise not only serves as a regularization technique but also aligns with the practical demands of processing real-world data, which often contains random variations and noise. By training with noise-augmented data, the LSTM autoencoder becomes adept at decoding noisy signals, making it highly effective for deployment in environments where data integrity may be compromised by various noise factors.

4.3.3 Tunning of the Model

The tuning process for the LSTM Autoencoder model was conducted in two main phases: the treatment of variables and the fine-tuning of the LSTM Autoencoder itself. This section provides a detailed explanation of the steps taken and the reasoning behind the choices made during the tuning process.

Treatment of Variables

In the treatment of variables, the primary focus was on applying a Low Pass Filter to the dataset. This filter helps to remove high-frequency noise, which can potentially obscure the underlying patterns in the data that are crucial for the model to learn. To find the most effective filter configuration, we experimented with varying the cutoff frequency and the filter order.

- **Cutoff Frequency:** The cutoff frequency of the low pass filter was varied from 0.001 to 0.007. The objective was to determine which frequency value would yield the highest precision in detecting faults. After extensive testing, a cutoff frequency of 0.0003 was found to provide the best results. This frequency was low enough to filter out most of the noise while preserving the important features of the signal.
- **Filter Order:** We also experimented with the filter order, testing values from 1 to 5. The filter order determines the sharpness of the cutoff. The best results were obtained with a filter order of 1. A lower order generally means a smoother filter with a gentler cutoff, which in this case helped retain more of the original signal characteristics while still effectively reducing noise.

Tuning the LSTM Autoencoder

In the second phase, the LSTM Autoencoder model itself was fine-tuned to optimize its performance. Two primary aspects were adjusted: the GaussianNoise layer and the optimizer used during training.

- **GaussianNoise Layer:** The GaussianNoise layer is a regularization technique that adds noise to the inputs of the model during training. This noise forces the model to learn more robust features, making it less likely to overfit the training data. We varied the standard deviation of the GaussianNoise layer from 0.1 to 0.5 to find the optimal level of noise. The best results were obtained with a standard deviation of 0.1. This value struck the right balance, introducing enough noise to improve the model's generalization ability without overwhelming the model with too much randomness, which could have impeded learning.

- **Optimizer Selection:** Another crucial part of the tuning process was determining the best optimizer for the LSTM Autoencoder. We tested three different optimizers: Adam, SGD (Stochastic Gradient Descent), and RMSprop. The Adam optimizer provided the lowest training and validation losses, making it the most effective choice for this particular model configuration. Adam's ability to compute adaptive learning rates for each parameter likely contributed to its superior performance, enabling the model to converge faster and more effectively. In contrast, SGD showed significantly higher losses, likely due to its simpler, non-adaptive approach, which can struggle with complex loss landscapes. RMSprop performed better than SGD but still did not match the performance of Adam.

4.4 Evaluation and Results Obtained

In this section, we present the results of the LSTM Autoencoder model, comparing its performance before and after the application of the Low Pass Filter. We evaluate the model using key metrics such as accuracy, precision, recall, and F1-score. Additionally, we assess the explicability of the model using SHAP, which helps us understand the contribution of each feature to the reconstruction error. This analysis highlights the most important features driving model performance, ensuring both accuracy and interpretability.

4.4.1 Explicability of the Model

As saw in the Mosca, Szigeti, Tragianni, Gallagher, and Groh (2022), SHAP (SHapley Additive exPlanations) method is a powerful tool for model interpretability, grounded in cooperative game theory. Conceptually, SHAP assigns each feature in a model an importance value based on how much it contributes to the model's output. This approach is derived from Shapley values, which quantify the marginal contribution of each player (feature) in a cooperative game. When applied to machine learning models, SHAP values explain the contribution of each feature to an individual prediction by comparing the model's output with and without the feature, averaged over all possible combinations of features. The result is a set of additive feature attributions that sum to the actual output, ensuring consistency and fairness in the feature importance scores. By employing SHAP, we can interpret complex models like LSTM Autoencoders by visualizing how each input feature contributes to the reconstruction error, offering insights into the model's decision-making process and highlighting patterns or anomalies in the data

This SHAP summary plot was constructed by explaining the reconstruction error of the LSTM autoencoder. The reconstruction error measures how well or poorly the autoencoder

can recreate the input data. A higher reconstruction error suggests that the model is struggling to accurately recreate certain aspects of the input, potentially signaling an anomaly or unusual pattern in the data. SHAP values represent the contribution of each feature to the reconstruction error, indicating whether a particular feature increases or decreases the model's performance. Positive SHAP values suggest that a feature is pushing the reconstruction error higher (worse reconstruction), while negative SHAP values indicate that the feature helps lower the reconstruction error (better reconstruction).

In this graph, each feature's contribution to the reconstruction error is visualized along the x-axis, with the color gradient representing the actual value of the feature (red for higher values and blue for lower values). The width of the spread along the x-axis shows how much each feature influences the reconstruction error. For example, "avg_rotpitchanga_deg" has both high positive and negative SHAP values, meaning that both high and low values of this feature can significantly impact the model's ability to reconstruct the input. Specifically, higher values (red) of "avg_rotpitchanga_deg" tend to increase the reconstruction error, while lower values (blue) reduce it, suggesting that this feature plays a major role in determining reconstruction performance. In contrast, "avg_hyd_oil_bar" shows a more concentrated pattern where higher values increase the reconstruction error, while lower values reduce it, indicating a more straightforward relationship.

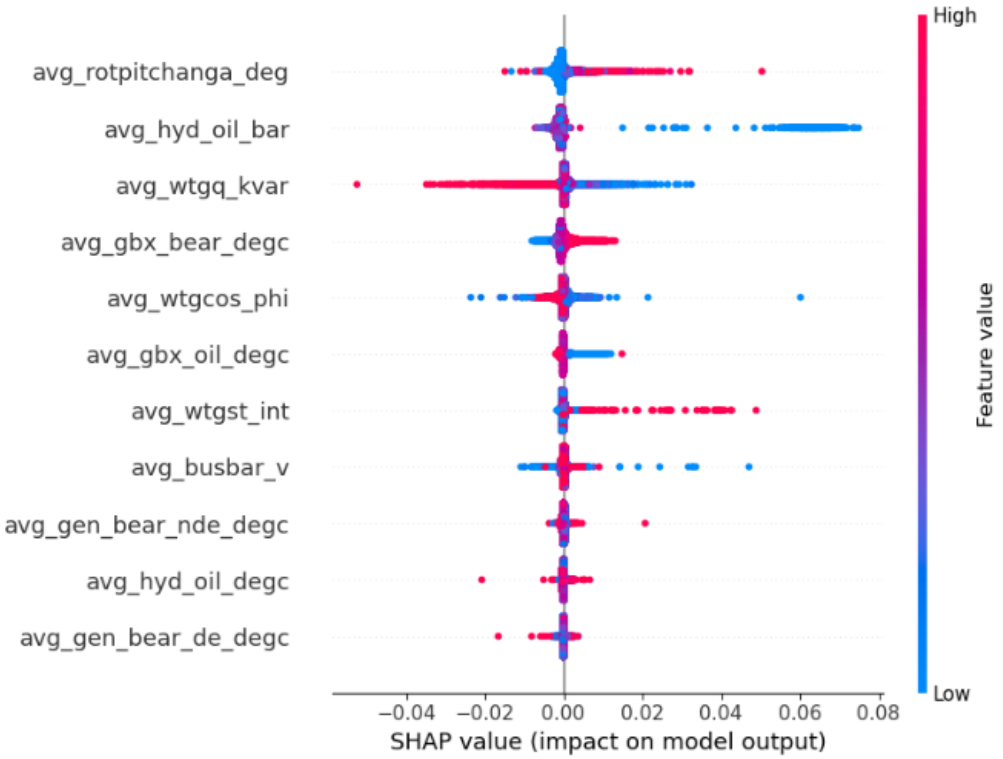

Figure 4.5: SHAP Summary Plot for Model Explanation

The features at the top of the graph, such as "avg_rotpitchanga_deg", "avg_hyd_oil_bar", and "avg_wtgq_kvar", are the most important, meaning they have the greatest impact on the reconstruction error. These features show a wider spread of SHAP values, meaning that changes in their values heavily influence how well or poorly the autoencoder reconstructs the input. On the other hand, features at the bottom, such as "avg_gen_bear_de_degc" and

”avg_hyd_oil_degc”, have a smaller impact on the reconstruction error, as indicated by their narrower SHAP value distributions. These less important features contribute less variability to the model's performance, and their values have a smaller effect on the reconstruction accuracy.

4.4.2 The Process of Training the Model Applied

The training and validation loss curves, illustrated in the two figures, compare the model's performance before and after applying the Low Pass Filter over 100 epochs. In both cases, the losses initially decrease sharply, indicating effective pattern capture. However, after applying the Low Pass Filter, the curves demonstrate greater stability, particularly in the validation set, where fluctuations become minimal. The training loss continues to steadily decline in both figures, but the post-filter figure shows a smoother validation loss trend, suggesting improved generalization and reduced overfitting. In contrast, the pre-filter figure exhibits more significant fluctuations in the validation loss, indicating that the model struggled more with generalization. Overall, the Low Pass Filter led to a more stable and robust training process, enhancing the model's ability to generalize to unseen data.

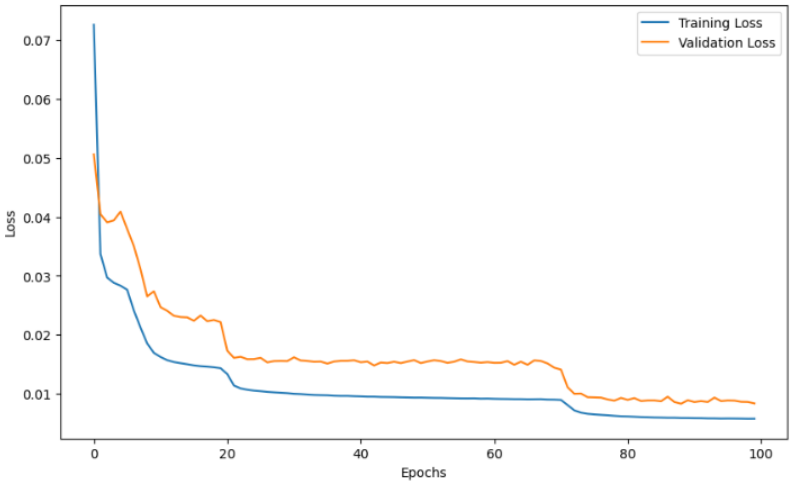

Figure 4.6: Model Training vs Validation Loss Curves Before Low Pass Filter

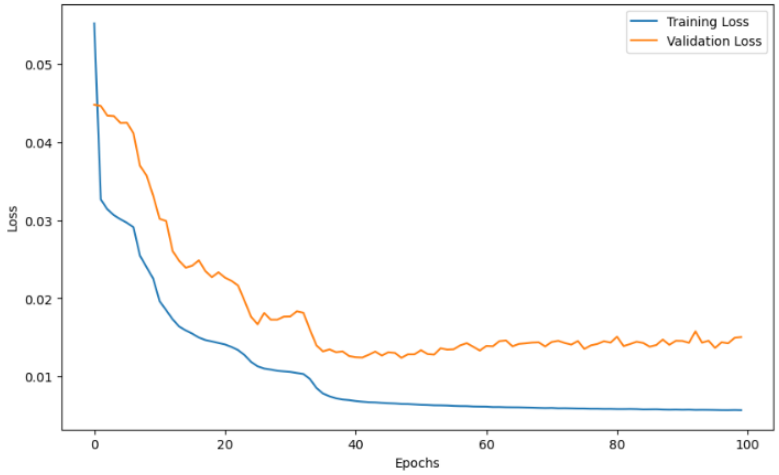

Figure 4.7: Model Training vs Validation Loss Curves After Low Pass Filter

4.4.3 Results Mestrics Before and After Low Pass Filter

The table presents the model's performance across four turbines (8, 12, 15, and 21) when trained using data from turbines 8 and 12, and overall, the results indicate that the model performs well across different turbines. With Precision consistently at or near 100% for all turbines and high Recall values (above 86%), the model demonstrates strong fault detection capabilities while minimizing false positives. Turbine 15 showed the best performance, achieving the highest Recall (94.6%), perfect Precision (100%), and an F1-Score of 97.2%, indicating excellent balance between detecting faults and avoiding unnecessary interventions. Although performance slightly decreases for turbine 21 (with Recall at 86.5%), the model still maintains strong accuracy and a solid F1-Score, which highlights its robustness. This overall performance suggests that the model is indeed transferable and can be effectively applied to turbines it was not specifically trained on, as shown by its strong performance across all turbines tested. The confusion matrix for turbine 15 reinforces this conclusion, showing 20,993 true negatives and 541 true positives, with no false positives and only 31 false negatives, resulting in a high F1-Score and confirming the model's reliability in detecting faults while maintaining accuracy and stability across different turbines.

Table 4.1: Performance Metrics Before Low Pass Filter

Fault Window	WT Data Trained	Turbine Tested	Recall	Precision	F1-Score	Accuracy	Threshold
On the time	8, 12	8	94.0%	100.0%	96.9%	99.8%	3.9 x Std Dev
On the time	8, 12	12	90.2%	100.0%	94.9%	99.9%	4.5 x Std Dev
On the time	8, 12	15	94.6%	100.0%	97.2%	99.9%	3.1 x Std Dev
On the time	8, 12	21	86.5%	94.8%	90.5%	99.7%	5.4 x Std Dev

Figure 4.8: Confusion Matrix Before Low Pass Filter for wt 15

After applying the Low Pass Filter, the overall performance across turbines remained stable or showed slight improvements, particularly in terms of Precision, which consistently stayed at 100% for turbines 8, 12, and 15, with minor variations in Recall and F1-Score. For turbine 21, there was a significant improvement, with Recall increasing from 86.5% to 89.6% and the F1-Score rising from 90.5% to 92.1%, indicating that the model was able to detect more Fault events after the filter was applied. This suggests that the Low Pass Filter effectively reduced noise in the data, allowing the model to better identify true Fault patterns. In turbine 21's case, the data likely contained more high-frequency noise, and the filter helped smooth it out, improving the model's fault detection accuracy. Overall, the Low Pass Filter helped to improve the model's generalization, particularly for turbines where noise may have impacted detection capabilities before filtering.

Table 4.2: Performance Metrics After Low Pass Filter

Fault Window	WT Data Trained	Turbine Tested	Recall	Precision	F1-Score	Accuracy	Threshold
On the time	8, 12	8	93.8%	99.7%	96.6%	99.8%	4.1 x Std Dev
On the time	8, 12	12	90.6%	99.1%	94.7%	99.9%	4.7 x Std Dev
On the time	8, 12	15	94.5%	100.0%	97.2%	99.9%	3.9 x Std Dev
On the time	8, 12	21	89.6%	94.7%	92.1%	99.7%	6.0 x Std Dev

As also confirmed by Ribeiro et al. (2016), by attenuating these rapid, less significant variations, the filter allows the model to focus on the more stable and relevant underlying trends within the data, which are more likely to correlate with actual fault conditions. This reduction in noise helps prevent the model from overfitting to transient (as we already saw in

the validation versus training curves comparison, before and after application of the low pass filter), non-essential features that could lead to a better generalization.

4.4.4 Model Behavior in Predicting Failures in Advance

Finally, the graph below provided plot illustrates the Loss-MAE curve over time, with blue lines indicating the loss values and red vertical lines marking the occurrences of faults. The analysis of the plot reveals that significant spikes in the Loss-MAE values often precede the red vertical lines, which represent actual fault events. This pattern suggests that the LSTM autoencoder is effectively identifying anomalies in the data prior to the occurrence of faults, demonstrating its potential for early fault detection. However, there are also periods where spikes in Loss-MAE do not correspond to any fault, indicating potential false positives. The clustering of red lines in the later period suggests multiple fault events in quick succession, which could be challenging for the model to predict accurately, highlighting the importance of refining the model to improve its precision and reduce false alarms.

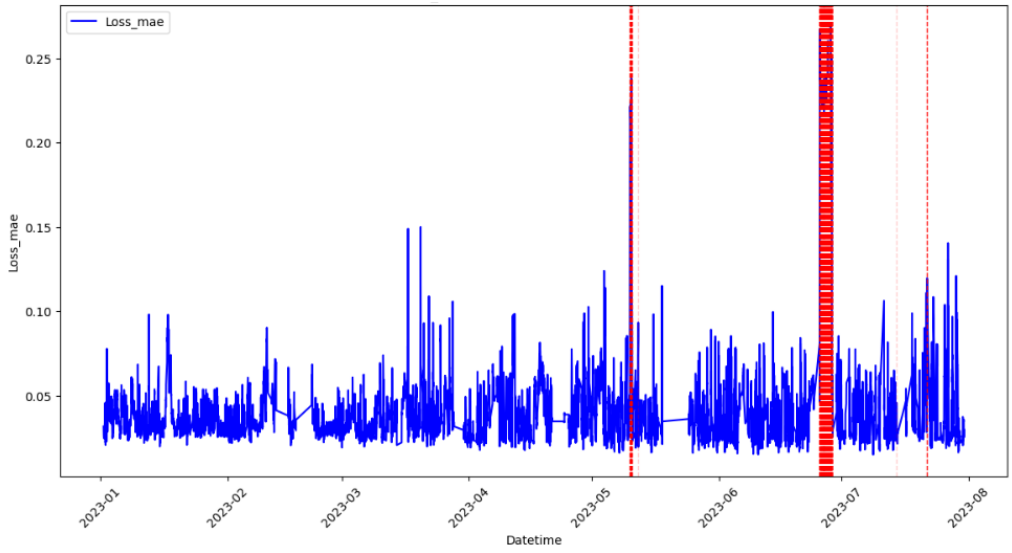

Figure 4.9: Loss-MAE with Fault Stamp for wt 15 - Full Period of Test Dataset After Low Pass Filter

The zoomed-in Loss-MAE curves for turbine 15, both before and after applying the Low Pass Filter, illustrate the challenge of identifying faults in advance (1, 2, or 3 hours before they occur). In both graphs, the loss values remain relatively stable and low until the moments leading directly up to the fault, where there are sharp increases. This pattern shows that the model, even after applying the Low Pass Filter, struggles to detect significant deviations in loss until the fault is imminent. The high-frequency fluctuations in the MAE, particularly visible in the pre-filter graph, indicate noise, which makes it harder to distinguish gradual changes that could signal a future fault. Even though the post-filter graph appears smoother and more stable, the difficulty in predicting faults far in advance persists, as the loss only significantly increases just before the fault occurs. This underscores the challenge of early fault detection, as the subtle changes leading up to a fault may be overshadowed by noise or not distinct enough for the model to capture well in advance.

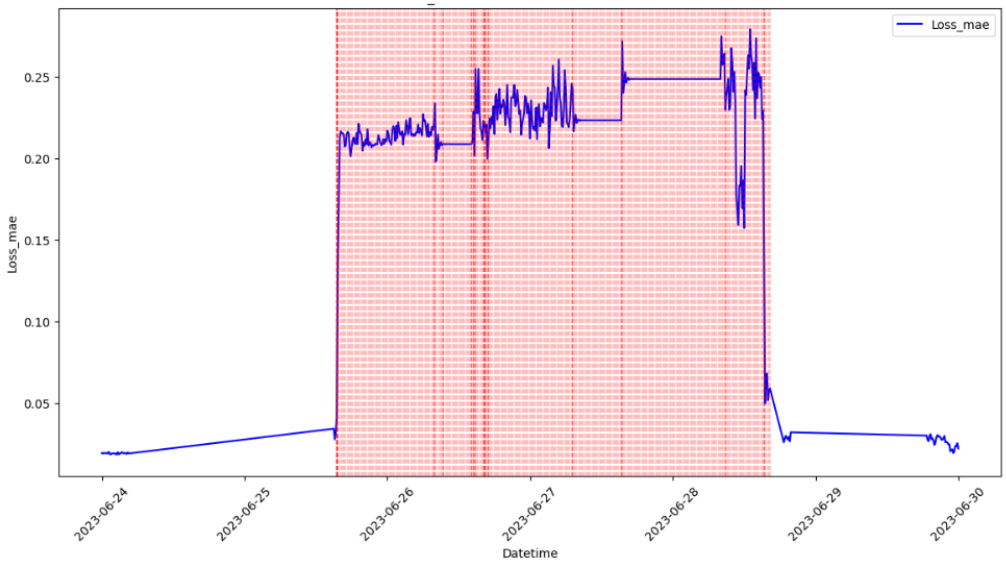

Figure 4.10: Loss-MAE with Fault Stamp for wt 15 - Zoomed Period of Test Dataset Before Low Pass Filter

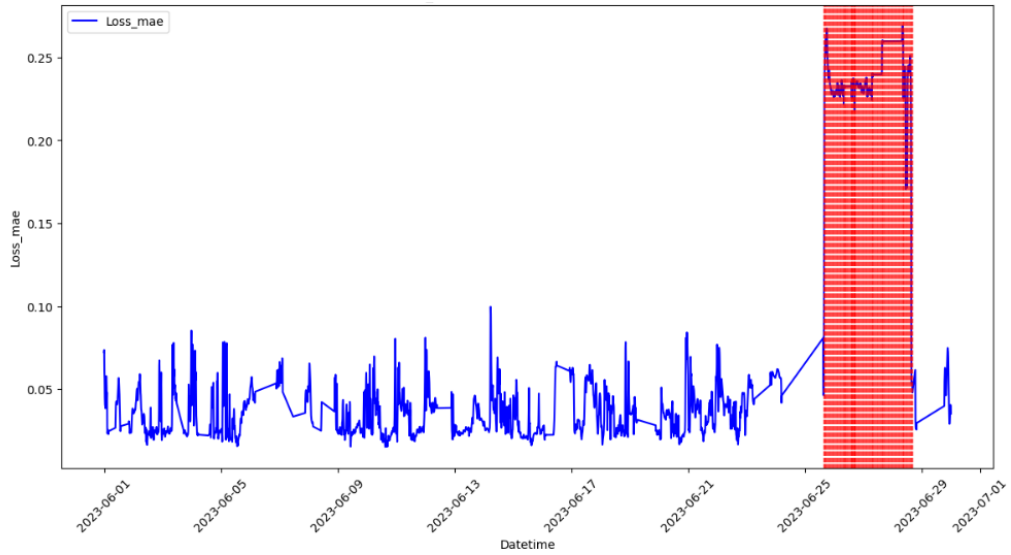

Figure 4.11: Loss-MAE with Fault Stamp for wt 15 - Zoomed Period of Test Dataset After Low Pass Filter

The attempt to detect failures 1 hour in advance, both before and after applying the Low Pass Filter, demonstrates that the model struggles significantly to predict faults early, as seen in the poor Recall, Precision, and F1-Scores across both turbines. This aligns with the behavior observed in the earlier MAE plots, where the model only shows sharp spikes in loss values right before the fault occurs, making it difficult to detect failures well in advance. Despite applying the Low Pass Filter, which improved noise reduction, the predictions remained ineffective, with the model still largely failing to capture early fault signals. These results emphasize that the model is not yet capable of reliably detecting faults before they happen

Table 4.3: Performance Metrics Before Low Pass Filter - Fault Detection in Advance

Fault Window	Turbine Trained	Turbine Tested	Recall	Precision	F1-Score	Accuracy	Threshold
1h before	8, 12	8	1.8%	2.5%	2.1%	99.6%	2.4 x Std Dev
1h before	8, 12	21	4.3%	1.4%	2.1%	98.8%	4.5 x Std Dev

Table 4.4: Performance Metrics After Low Pass Filter - Fault Detection in Advance

Fault Window	Turbine Trained	Turbine Tested	Recall	Precision	F1-Score	Accuracy	Threshold
1h before	8, 12	8	16.7%	0.7%	1.3%	97.6%	2.4 x Std Dev
1h before	8, 12	21	21.7%	2.0%	3.7%	96.7%	3.6 x Std Dev

Chapter 5

Conclusion

5.1 Final Remarks

The application of the LSTM Autoencoder in detecting faults within wind turbine data has proven to be a robust approach, particularly when enhanced by careful preprocessing of the data. The use of a Low Pass Filter significantly contributed to the model's effectiveness, smoothing out noise and highlighting the underlying patterns that are critical for accurate anomaly detection. This preprocessing step along with the strategic selection of relevant variables based on their correlation with fault occurrences, laid a solid foundation for the model's performance.

A key takeaway from this study is the importance of data processing. The integrity and quality of the data play a critical role in model performance. SCADA datasets often come from multiple sensors that record data at different intervals or formats. If these datasets are not properly aligned and preprocessed, the resulting model will struggle to learn meaningful patterns, leading to poor performance. Proper data handling, including cleaning, synchronization, and noise reduction, was essential in achieving the results presented in this dissertation. Without careful attention to the data processing steps, it would have been impossible to obtain reliable and accurate results.

The use of SHAP analysis contributed to the interpretability of the model, allowing for a deeper understanding of the importance of individual features, such as rotor pitch angle and hydraulic oil pressure, in influencing model predictions. This level of transparency is essential for industrial applications, where explainable models are required to guide operational decisions.

Through extensive tuning of the LSTM Autoencoder, including the adjustment of the Gaussian Noise layer and the exploration of various optimizers, we identified that the Adam optimizer, combined with a Gaussian Noise standard deviation of 0.1, yielded the most favorable results. These findings underscore the importance of model optimization in improving the accuracy and reliability of fault detection, particularly in complex, non-linear datasets typical of wind turbine operations.

5.2 Limitations and Future Work

While the LSTM Autoencoder successfully detected faults as they occurred, it struggled to anticipate these faults with the desired lead time. This limitation highlights a critical area for

improvement: enhancing the model's capability to detect faults earlier. Achieving better early detection may require adjustments to the model's configuration, such as experimenting with different architectures or fine-tuning the current model further. Additionally, incorporating new types of variables that have a stronger correlation with faults could provide the model with more predictive power, allowing it to identify anomalies before they escalate into critical issues.

In future work, it would be beneficial to explore alternative modeling approaches, such as ensemble methods or hybrid models that combine the strengths of multiple algorithms. Furthermore, recording and analyzing different types of variables, especially those that might offer early warning signs of faults, could greatly enhance the model's ability to perform predictive maintenance. By addressing these limitations, the model could be transformed into a more proactive tool, capable of not just identifying when a fault has occurred, but predicting when a fault is likely to happen, thereby enabling more timely and effective interventions.

Bibliography

Abdallah, I., Ntertimanis, V., Mylonas, C., Tatsis, K., Chatzi, E., Dervilis, N., … Eoghan, M. (2018). Fault diagnosis of wind turbine structures using decision tree learning algorithms with big data. *Safety and Reliability–Safe Societies in a Changing World*, 3053–3061.

Administration, I. T. (2023). *Renewable energy infrastructure.* Retrieved 2024-07-15, from https://www.trade.gov/country-commercial-guides/brazil-renewable-energy-infrastructure-0

Berry, M. W., Mohamed, A., & Yap, B. W. (2019). *Supervised and unsupervised learning for data science*. Springer.

Brasil, A. (2023). *Capacidade de geração de energia eólica deve bater recorde neste ano.* Retrieved 2024-07-15, from https://agenciabrasil.ebc.com.br/economia/noticia/2023-04/capacidade-de-geracao-de-energia-eolica-deve-bater-recorde-neste-ano

Chen, L., Xu, G., Zhang, Q., & Zhang, X. (2019). Learning deep representation of imbalanced scada data for fault detection of wind turbines. *Measurement*, *139*, 370–379.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273–297.

Dhiman, H. S., Deb, D., Muyeen, S., & Kamwa, I. (2021). Wind turbine gearbox anomaly detection based on adaptive threshold and twin support vector machines. *IEEE Transactions on Energy Conversion*, *36*(4), 3462–3469.

Gama, J., Carvalho, A. C. P. d. L. F. d., Faceli, K., Lorena, A. C., & Oliveira, M. (2012). Extração de conhecimento de dados: data mining.

Goodfellow, I., Bengio, Y., & Courville, A. (2016a). *Deep learning*. MIT press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016b). *Deep learning*. MIT press.

Jiang, G., He, H., Xie, P., & Tang, Y. (2017). Stacked multilevel-denoising autoencoders: A new representation learning approach for wind turbine gearbox fault diagnosis. *IEEE Transactions on Instrumentation and Measurement*, *66*(9), 2391–2402.

Jiang, G., He, H., Yan, J., & Xie, P. (2018). Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox. *IEEE Transactions on Industrial Electronics*, *66*(4), 3196–3207.

Jiang, G., Xie, P., He, H., & Yan, J. (2017). Wind turbine fault detection using a denoising autoencoder with temporal information. *IEEE/Asme transactions on mechatronics*, *23*(1), 89–100.

Jolliffe, I. T. (2002). *Principal component analysis for special types of data*. Springer.

Joshuva, A., & Sugumaran, V. (2017). A comparative study of bayes classifiers for blade fault diagnosis in wind turbines through vibration signals. *Structural Durability & Health Monitoring*, *11*(1), 69.

Laouti, N., Othman, S., Alamir, M., & Sheibat-Othman, N. (2014). Combination of model-based observer and support vector machines for fault detection of wind turbines. *International Journal of Automation and Computing*, *11*, 274–287.

Laouti, N., Sheibat-Othman, N., & Othman, S. (2011). Support vector machines for fault detection in wind turbines. *IFAC Proceedings Volumes*, *44*(1), 7067–7072.

Lu, D., Qiao, W., & Gong, X. (2017). Current-based gear fault detection for wind turbine gearboxes. *IEEE Transactions on Sustainable Energy*, *8*(4), 1453–1462.

Maheshwari, J. A., R. (2024). *Top ai statistics and trends.* Retrieved 2024-07-15, from https://www.forbes.com/advisor/in/business/ai-statistics/

Mosca, E., Szigeti, F., Tragianni, S., Gallagher, D., & Groh, G. (2022). Shap-based explanation methods: a review for nlp interpretability. In *Proceedings of the 29th international conference on computational linguistics* (pp. 4593–4603).

Nejad, A. R., Odgaard, P. F., Gao, Z., & Moan, T. (2014). A prognostic method for fault detection in wind turbine drivetrains. *Engineering Failure Analysis*, *42*, 324–336.

Nie, X., Liu, S., & Xie, G. (2020). A novel autoencoder with dynamic feature enhanced factor for fault diagnosis of wind turbine. *Electronics*, *9*(4), 600.

Randall, R. B. (2021). *Vibration-based condition monitoring: industrial, automotive and aerospace applications*. John Wiley & Sons.

Rego, E. (2023). *Brasil é o terceiro maior gerador de energia elétrica renovável no mundo.* Retrieved 2024-07-15, from https://exame.com/esg/brasil-e-o-terceiro-maior-gerador-de-energia-eletrica-renovavel-no-mundo/

Ribeiro, R. P., Pereira, P., & Gama, J. (2016). Sequential anomalies: a study in the railway industry. *Machine Learning*, *105*, 127–153.

Santos, P., Villa, L. F., Reñones, A., Bustillo, A., & Maudes, J. (2015). An svm-based solution for fault detection in wind turbines. *Sensors*, *15*(3), 5627–5648.

Seni, G., & Elder, J. (2010). *Ensemble methods in data mining: improving accuracy through combining predictions*. Morgan & Claypool Publishers.

Tang, M., Zhao, Q., Ding, S. X., Wu, H., Li, L., Long, W., & Huang, B. (2020). An improved lightgbm algorithm for online fault detection of wind turbine gearboxes. *Energies*, *13*(4), 807.

Tang, S., Zhu, Y., & Yuan, S. (2021). An improved convolutional neural network with an adaptable learning rate towards multi-signal fault diagnosis of hydraulic piston pump. *Advanced Engineering Informatics*, *50*, 101406.

Tang, S., Zhu, Y., & Yuan, S. (2022). Intelligent fault diagnosis of hydraulic piston pump based on deep learning and bayesian optimization. *ISA transactions*, *129*, 555–563.

Tuerxun, W., Chang, X., Hongyu, G., Zhijie, J., & Huajian, Z. (2021). Fault diagnosis of wind turbines based on a support vector machine optimized by the sparrow search algorithm. *Ieee Access*, *9*, 69307–69315.

Wang, Y., Ma, X., & Qian, P. (2018). Wind turbine fault detection and identification through pca-based optimal variable selection. *IEEE Transactions on Sustainable Energy*, *9*(4), 1627–1635.

Wei, Y., Jang-Jaccard, J., Xu, W., Sabrina, F., Camtepe, S., & Boulic, M. (2023). Lstm-autoencoder-based anomaly detection for indoor air quality time-series data. *IEEE Sensors Journal*, *23*(4), 3787–3800.

Wirth, R., & Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery*

and data mining (Vol. 1, pp. 29–39).

Wu, X., Jiang, G., Wang, X., Xie, P., & Li, X. (2019). A multi-level-denoising autoencoder approach for wind turbine fault detection. *Ieee Access*, *7*, 59376–59387.

Wu, Z., Wang, X., & Jiang, B. (2020). Fault diagnosis for wind turbines based on relieff and extreme gradient boosting. *Applied Sciences*, *10*(9), 3258.

Xiang, L., Wang, P., Yang, X., Hu, A., & Su, H. (2021). Fault detection of wind turbine based on scada data analysis using cnn and lstm with attention mechanism. *Measurement*, *175*, 109094.

Xiao, C., Liu, Z., Zhang, T., & Zhang, X. (2021). Deep learning method for fault detection of wind turbine converter. *Applied Sciences*, *11*(3), 1280.

Yuan, T., Sun, Z., & Ma, S. (2019). Gearbox fault prediction of wind turbines based on a stacking model and change-point detection. *Energies*, *12*(22), 4224.

Zare, S., & Ayati, M. (2021). Simultaneous fault diagnosis of wind turbine using multichannel convolutional neural networks. *ISA transactions*, *108*, 230–239.

Zhang, D., Qian, L., Mao, B., Huang, C., Huang, B., & Si, Y. (2018). A data-driven design for fault detection of wind turbines using random forests and xgboost. *Ieee Access*, *6*, 21020–21031.

Appendix A

Appendix

A.1 Other Information

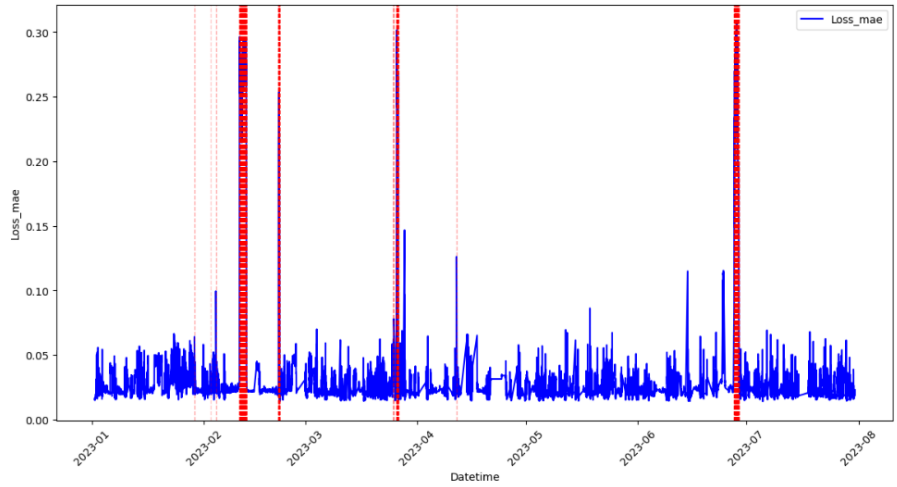

Figure A.1: Loss-MAE with Fault Stamp for wt 8 - Full Period of Test Dataset Before Low Pass Filter

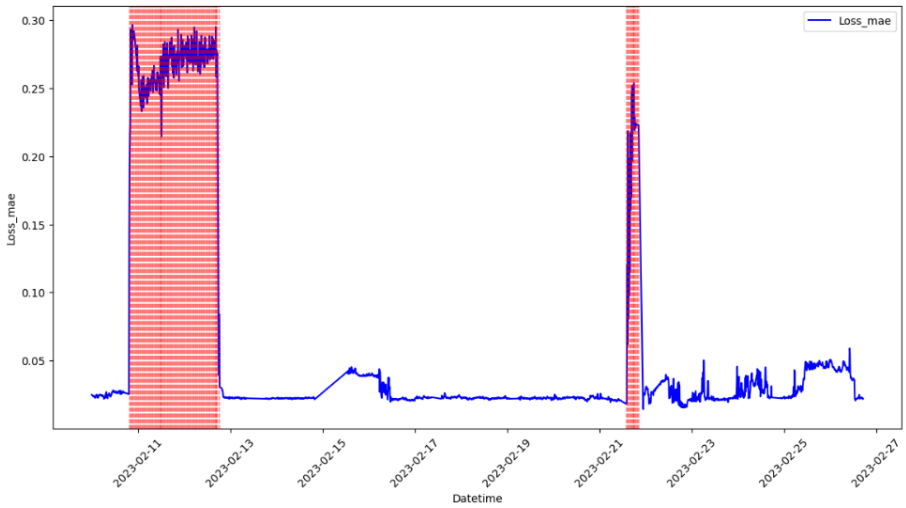

Figure A.2: Loss-MAE with Fault Stamp for wt 8 - Zoomed Period of Test Dataset Before Low Pass Filter

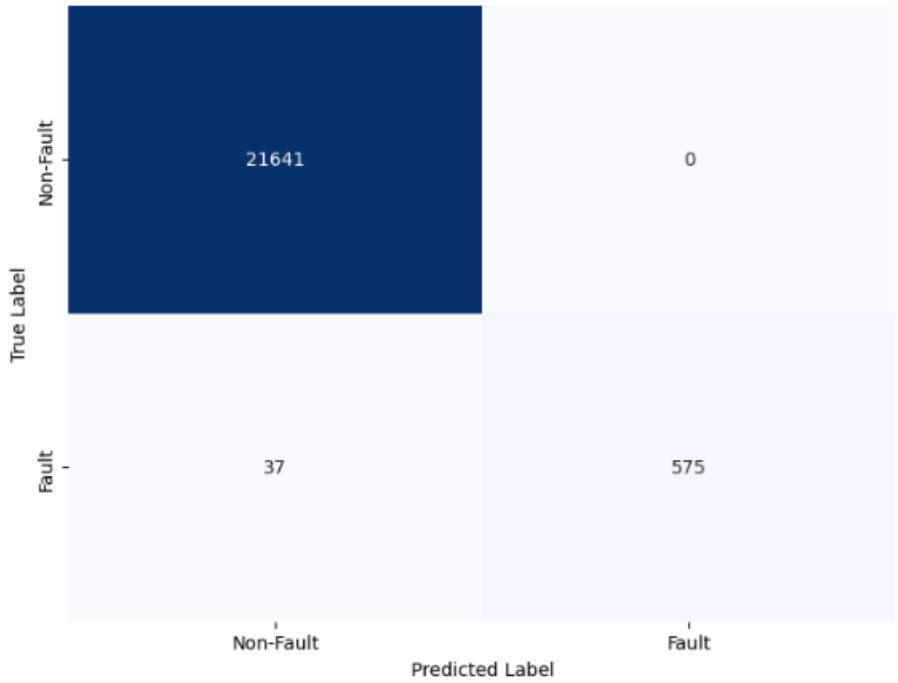

Figure A.3: Confusion Matrix Before Low Pass Filter for wt 8

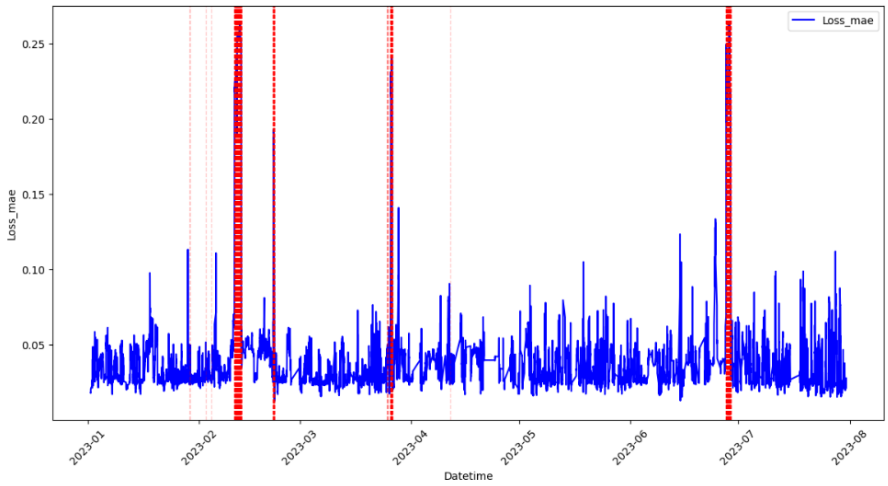

Figure A.4: Loss-MAE with Fault Stamp for wt 8 - Full Period of Test Dataset After Low Pass Filter

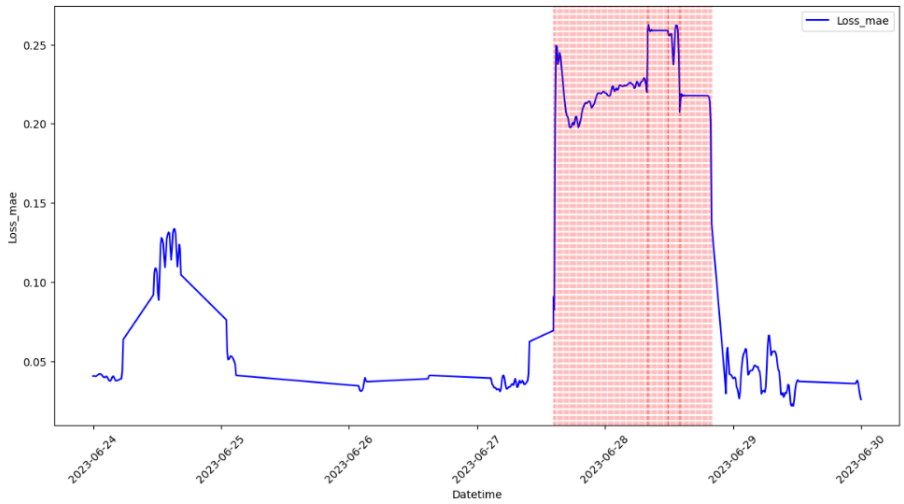

Figure A.5: Loss-MAE with Fault Stamp for wt 8 - Zoomed Period of Test Dataset After Low Pass Filter

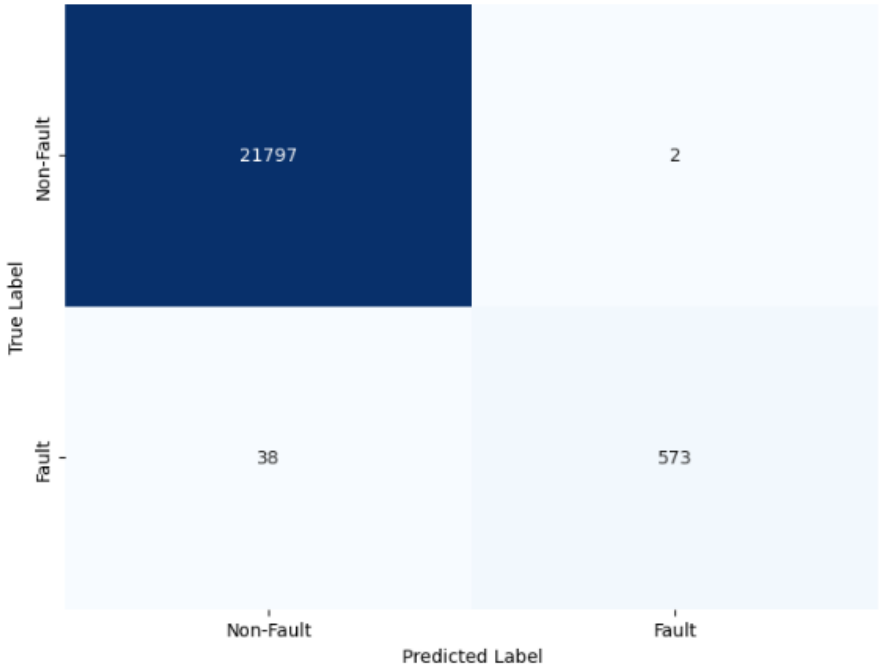

Figure A.6: Confusion Matrix After Low Pass Filter for wt 8

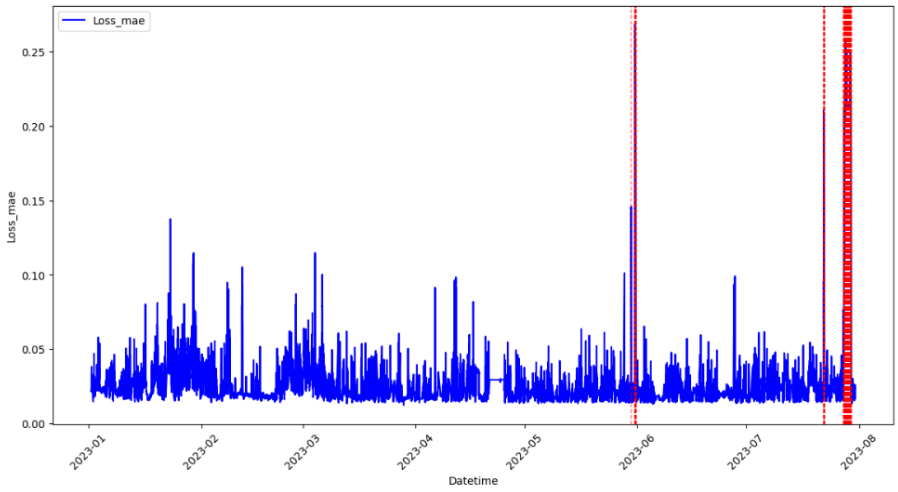

Figure A.7: Loss-MAE with Fault Stamp for wt 12 - Full Period of Test Dataset Before Low Pass Filter

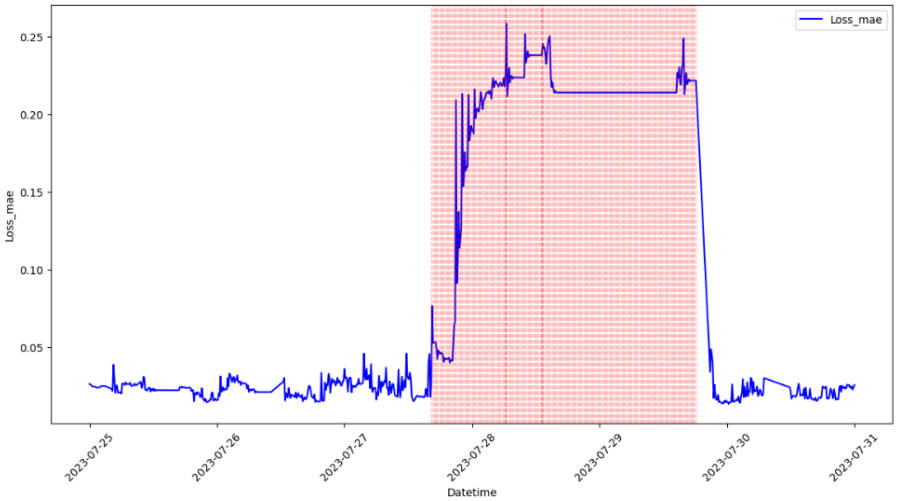

Figure A.8: Loss-MAE with Fault Stamp for wt 12 - Zoomed Period of Test Dataset Before Low Pass Filter

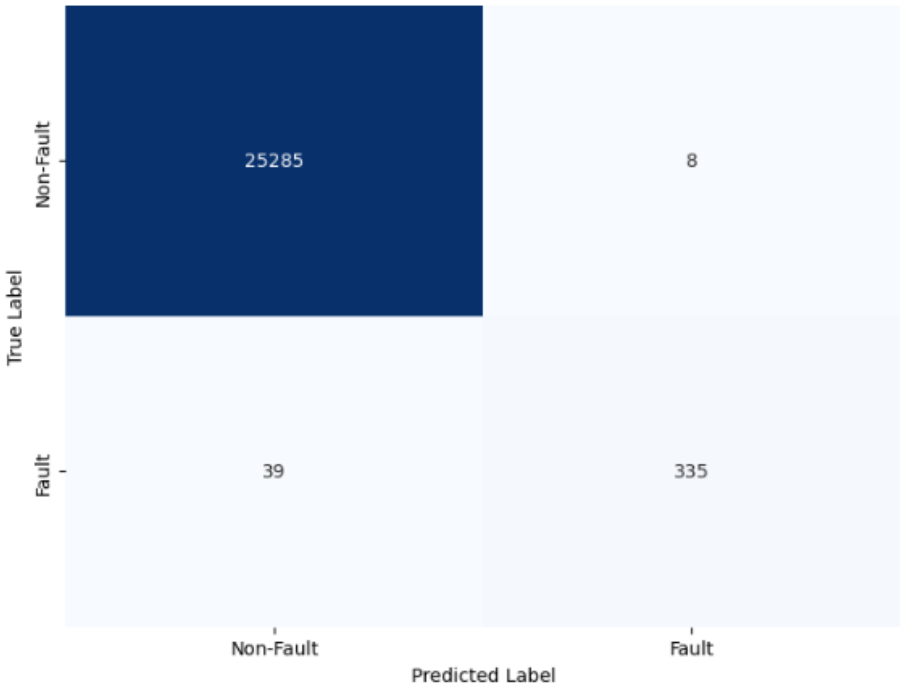

Figure A.9: Confusion Matrix Before Low Pass Filter for wt 12

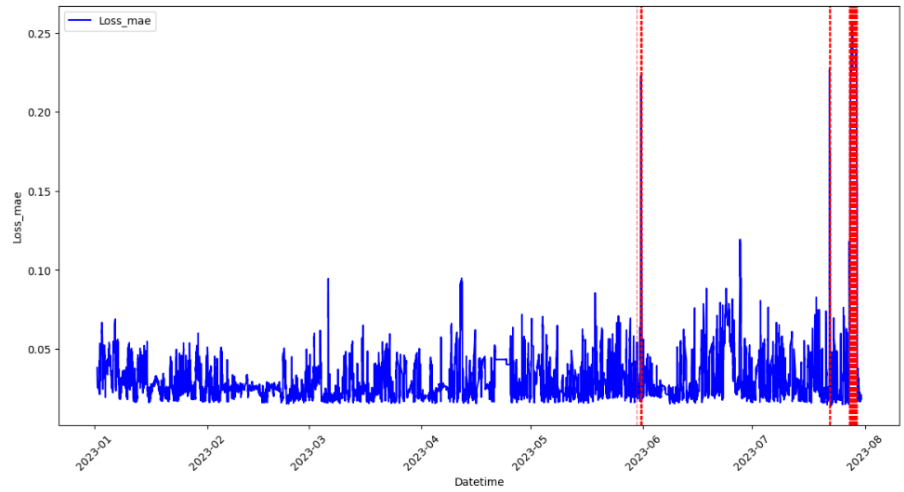

Figure A.10: Loss-MAE with Fault Stamp for wt 12 - Full Period of Test Dataset After Low Pass Filter

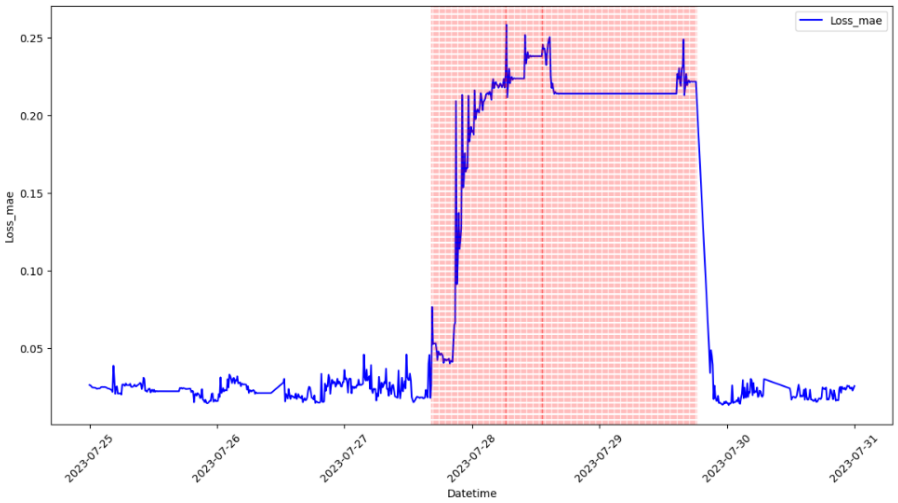

Figure A.11: Loss-MAE with Fault Stamp for wt 12 - Zoomed Period of Test Dataset After Low Pass Filter

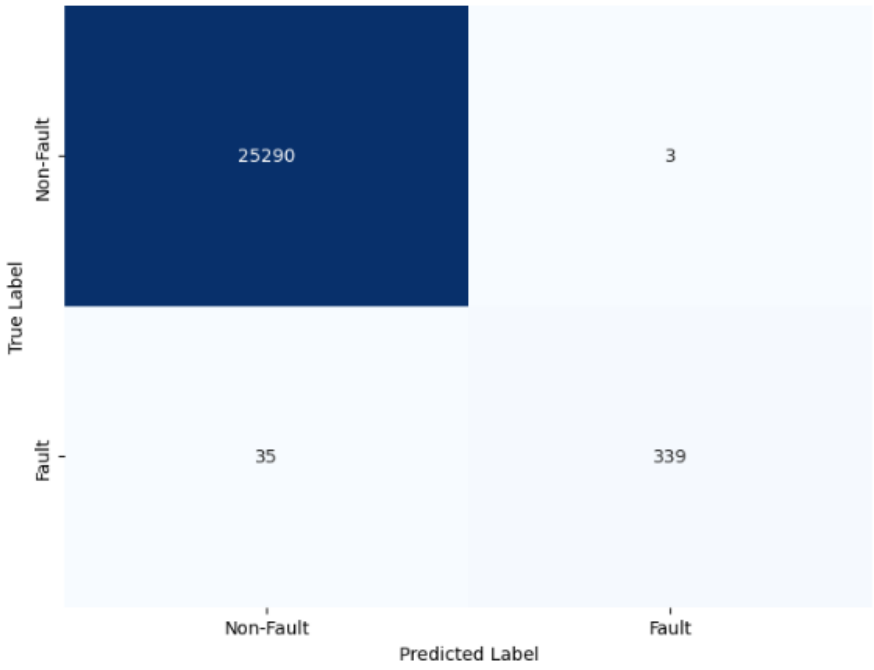

Figure A.12: Confusion Matrix After Low Pass Filter for wt 12

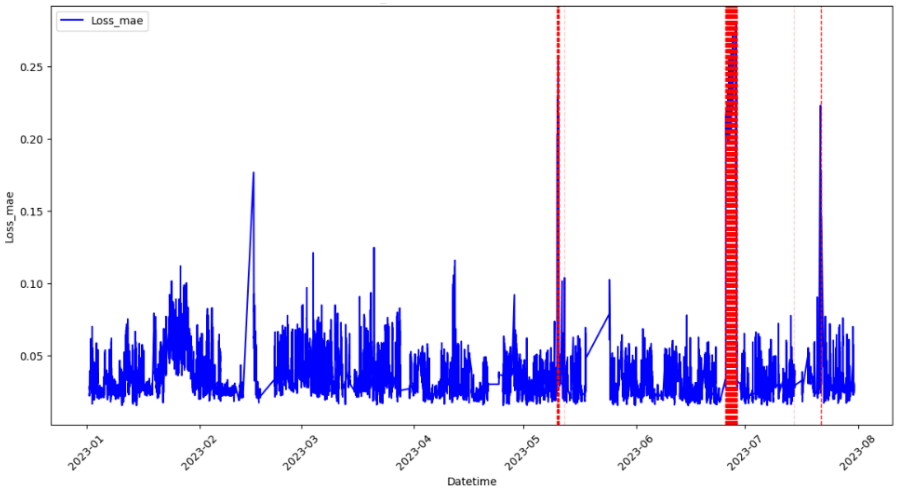

Figure A.13: Loss-MAE with Fault Stamp for wt 15 - Full Period of Test Dataset Before Low Pass Filter

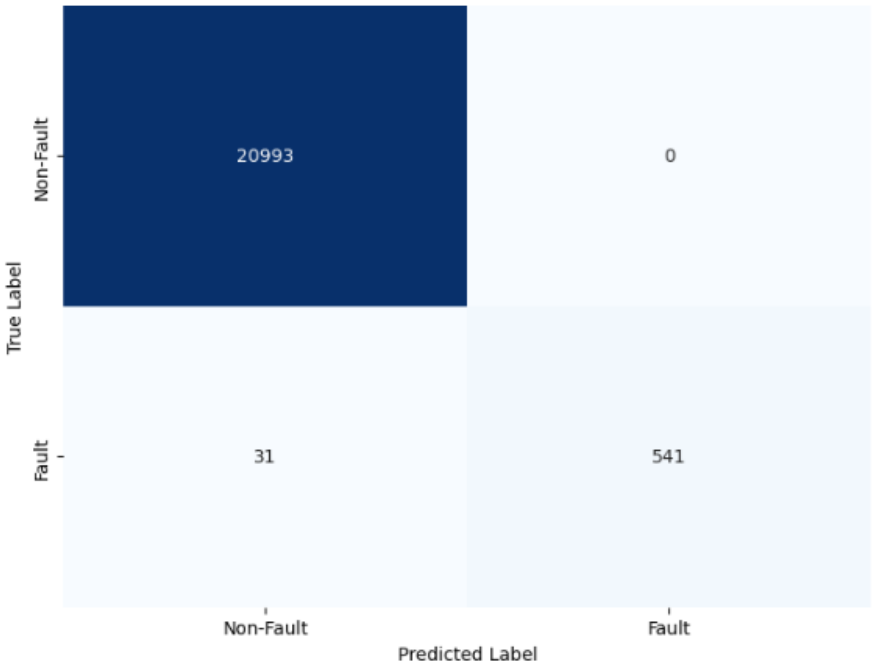

Figure A.14: Confusion Matrix Before Low Pass Filter for wt 15

Figure A.15: Confusion Matrix After Low Pass Filter for wt 15

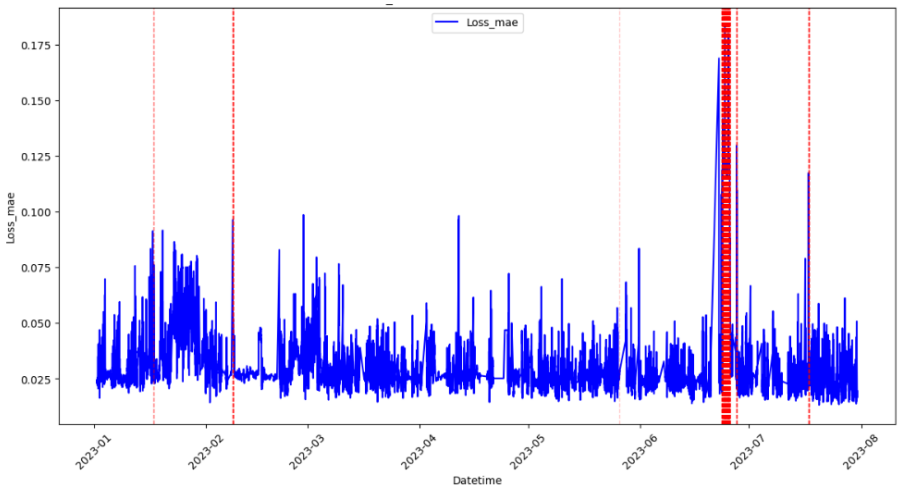

Figure A.16: Loss-MAE with Fault Stamp for wt 21 - Full Period of Test Dataset Before Low Pass Filter

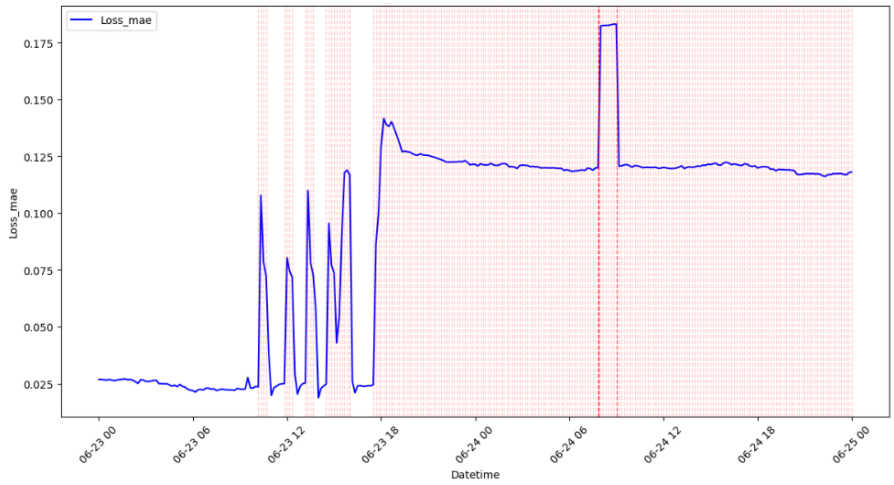

Figure A.17: Loss-MAE with Fault Stamp for wt 21 - Zoomed Period of Test Dataset Before Low Pass Filter

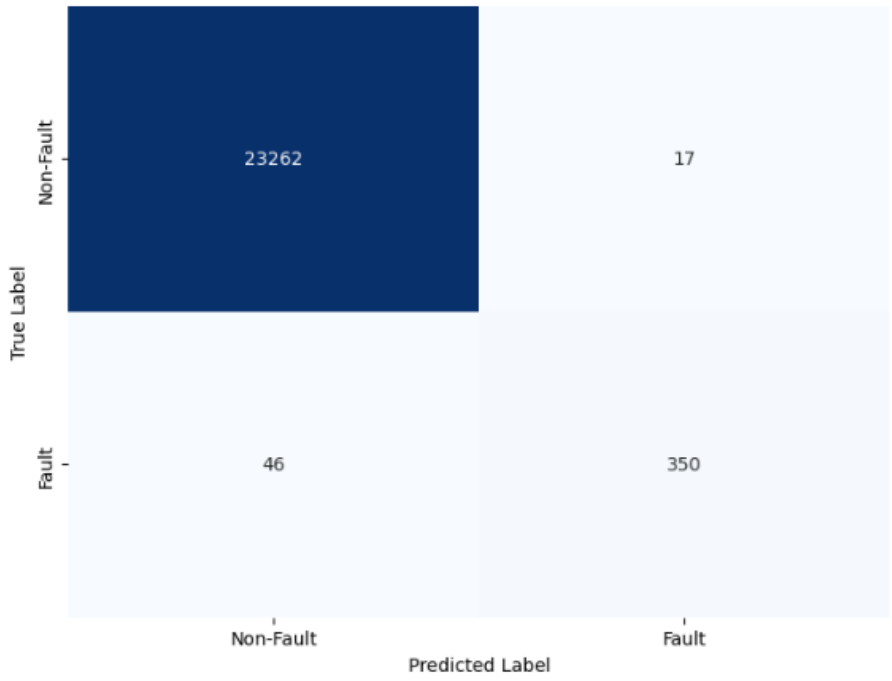

Figure A.18: Confusion Matrix Before Low Pass Filter for wt 21

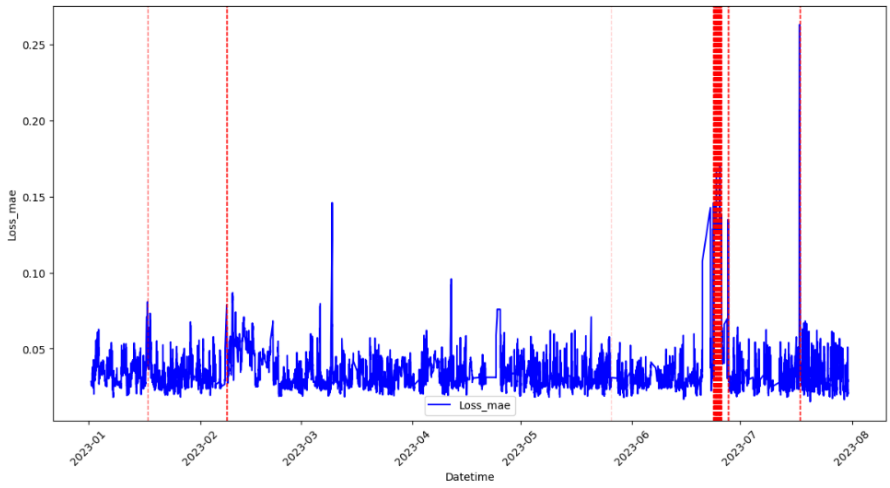

Figure A.19: Loss-MAE with Fault Stamp for wt 21 - Full Period of Test Dataset After Low Pass Filter

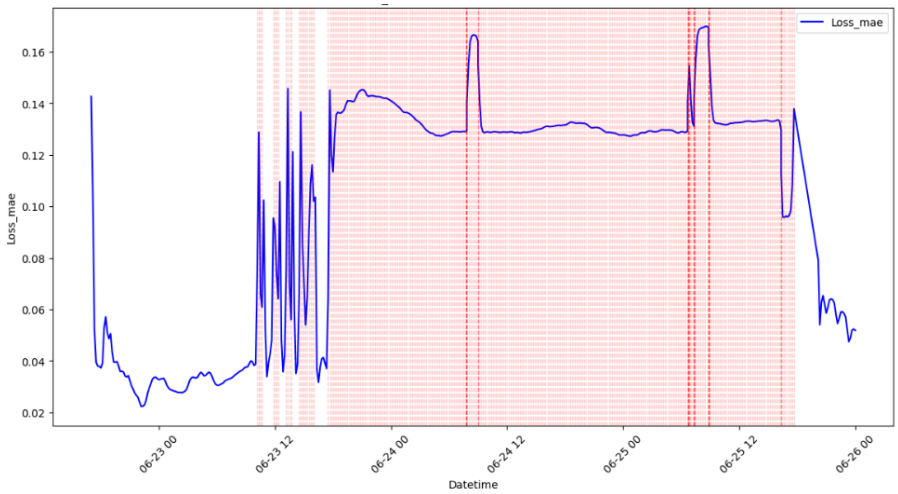

Figure A.20: Loss-MAE with Fault Stamp for wt 21 - Zoomed Period of Test Dataset After Low Pass Filter

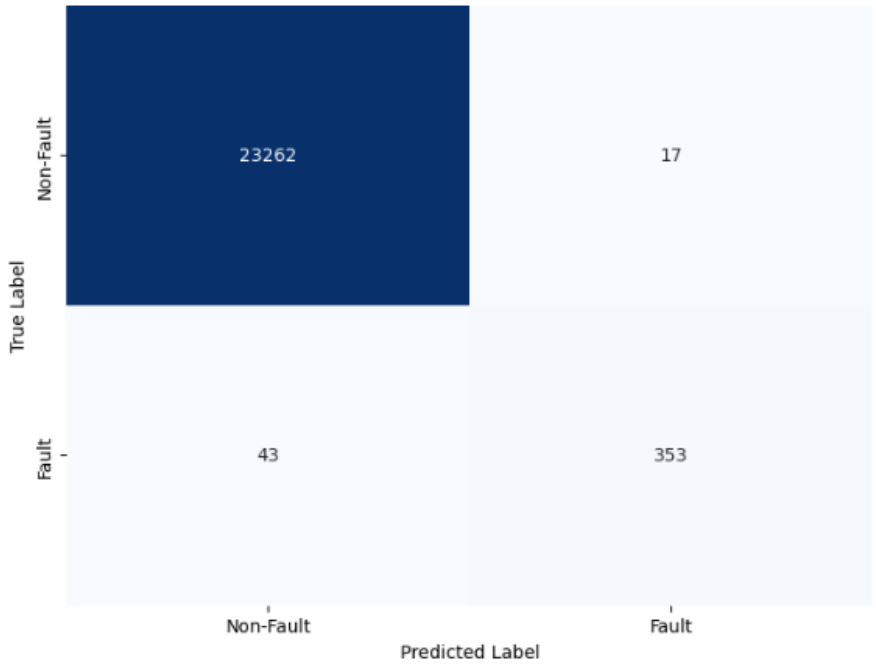

Figure A.21: Confusion Matrix After Low Pass Filter for wt 21

	wt	datetime_utc	datetime_local_x	avg_metwindspd_ms	avg_wtgp_kw	avg_genspd_rpm	avg_metextmp_degc	avg_metwindspd_dir	avg_nacelle_degc	avg_rotpitchanga_deg	avg_rotspd_rpm	avg_trffrec_hz	avg_wtgcapf_perc
8379	8	2022-02-27 07:30:00	2022-02-27 08:30:00	5.69744	312.35325	1177.77692	5.58333	-5.11039	24.02857	1.7	11.64805	50.05	15.61435
8380	8	2022-02-27 07:40:00	2022-02-27 08:40:00	5.9859	158.00519	560.67848	5.55	7.17089	23.8125	65.86923	9.20426	50.0	7.89994
8381	8	2022-02-27 07:40:00	2022-02-27 08:40:00	5.9859	158.00519	560.67848	5.55	7.17089	23.8125	65.86923	9.20426	50.0	7.89994
8382	8	2022-02-27 07:50:00	2022-02-27 08:50:00	6.53846	148.93896	529.14545	5.65	-15.61688	22.6	51.12222	9.64	50.0	7.45513
8383	8	2022-02-27 08:00:00	2022-02-27 09:00:00	5.89114	253.24557	1003.21139	5.55	-11.80253	22.12857	35.12	10.93562	50.0	12.66228
8384	8	2022-02-27 08:10:00	2022-02-27 09:10:00	5.56329	280.11539	1137.52437	5.68571	-10.8641	23.33333	1.7	11.20779	49.95455	14.00577
8385	8	2022-02-27 08:20:00	2022-02-27 09:20:00	6.4	433.34687	1298.95863	6.275	-8.60172	23.76	1.7	12.94468	50.0	21.66734
8386	8	2022-02-27 08:30:00	2022-02-27 09:30:00	6.40519	433.76538	1305.73292	6.275	-9.03924	23.76	1.7	12.89467	50.0	21.66827
8387	8	2022-02-27 08:40:00	2022-02-27 09:40:00	5.97308	150.42152	566.13164	6.325	-3.12152	23.68	47.42353	9.55682	50.0	7.52025
8388	8	2022-02-27 08:50:00	2022-02-27 09:50:00	6.54103	399.35641	1269.25065	6.45	-15.90909	23.14286	1.66429	12.51818	50.0	19.96115
8389	8	2022-02-27 09:00:00	2022-02-27 10:00:00	7.00641	494.37949	1371.25897	6.56	-17.38077	23.175	1.7	13.53718	50.0	24.66532
8390	8	2022-02-27 09:10:00	2022-02-27 10:10:00	6.95065	539.73506	1399.02821	6.5	-14.20513	23.3	1.775	13.73718	49.95	27.01801
8391	8	2022-02-27 09:20:00	2022-02-27 10:20:00	6.3481	429.74557	1301.73975	6.5	-11.76026	23.55	1.75	12.83636	50.0	21.48728
8392	8	2022-02-27 09:30:00	2022-02-27 10:30:00	3.86071	82.52881	1050.04745	6.63333	-8.08361	23.83333	3.0119	10.35192	50.0	4.12644
8393	8	2022-02-27 09:40:00	2022-02-27 10:40:00	3.76154	76.62821	1050.06961	6.63333	-7.10506	23.83333	2.9726	10.34872	50.0	3.83141
8394	8	2022-02-27 09:50:00	2022-02-27 10:50:00	4.99494	168.51645	1071.41924	6.82222	-13.67179	23.95	2.52857	10.57975	50.0	8.42582
8395	8	2022-02-27 10:00:00	2022-02-27 11:00:00	5.31154	229.91795	1096.1141	6.85	-7.82692	24.22857	1.85806	10.78974	50.0	11.48282
8396	8	2022-02-27 10:10:00	2022-02-27 11:10:00	5.43077	256.19103	1116.20513	6.97	-14.26538	24.34	2.15714	11.00385	50.0	12.80955
8397	8	2022-02-27 10:20:00	2022-02-27 11:20:00	4.64684	126.16582	1050.29873	7.45556	-13.24684	24.56	2.0371	10.35696	50.0	6.30829
8398	8	2022-02-27 10:30:00	2022-02-27 11:30:00	4.81923	172.13974	1063.13461	7.43333	-6.05897	24.36571	2.26226	10.46923	50.0	8.59647
8399	8	2022-02-27 10:40:00	2022-02-27 11:40:00	5.86071	360.39	1231.84915	7.6	-4.6	23.9125	1.9875	12.08125	50.0	18.0195
8400	8	2022-02-27 10:50:00	2022-02-27 11:50:00	5.84026	347.56844	1222.0039	7.6	-4.72857	23.9125	1.9875	12.01169	50.0	17.37792
8401	8	2022-02-27 11:00:00	2022-02-27 12:00:00	5.45823	269.38481	1129.07595	7.86667	-5.85316	24.01429	1.77143	11.14533	50.0	13.46709
8402	8	2022-02-27 11:10:00	2022-02-27 12:10:00	5.49231	59.676	241.36753	8.0	17.24416	24.68	66.58462	5.46875	49.95	2.98387
8403	8	2022-02-27 11:20:00	2022-02-27 12:20:00	5.55128	150.76835	717.78861	8.175	-15.98734	24.55	42.25263	8.88167	50.0	7.48778
8404	8	2022-02-27 11:30:00	2022-02-27 12:30:00	4.76282	-6.44571	11.01923	8.32857	-14.00385	23.7375	85.95	0.0	50.0	-0.32268
8405	8	2022-02-27 11:40:00	2022-02-27 12:40:00	5.125	-5.23333	18.41875	8.5	-11.4775	23.77778	86.0	0.18333	50.0	-0.26094
8406	8	2022-02-27 11:50:00	2022-02-27 12:50:00	5.17324	-4.48906	9.78267	8.625	-7.38108	23.83333	86.0	0.15	50.0	-0.225
8407	8	2022-02-27 12:00:00	2022-02-27 13:00:00	5.22692	-4.47692	9.39359	8.625	-7.89744	23.83333	86.0	0.15	50.0	-0.22429
8408	8	2022-02-27 12:10:00	2022-02-27 13:10:00	4.60641	-2.63967	12.21154	9.07778	-28.51667	23.67143	86.0	0.2	49.94	-0.13193
8409	8	2022-02-27 12:20:00	2022-02-27 13:20:00	5.11447	-2.78816	7.10921	8.97778	-44.98052	22.27143	86.0	0.23333	50.0	-0.13934
8410	8	2022-02-27 12:30:00	2022-02-27 13:30:00	5.28974	-2.68947	10.30641	8.98333	-32.47792	22.12857	85.94	0.23333	50.0	-0.13447
8411	8	2022-02-27 12:40:00	2022-02-27 13:40:00	5.32564	-2.55541	12.36026	8.9875	-23.15385	22.53	86.0	0.2	50.0	-0.12784
8412	8	2022-02-27 12:50:00	2022-02-27 13:50:00	5.25443	-2.7359	11.96203	9.04286	-17.76835	22.7875	86.0	0.15	50.0	-0.13673
8413	8	2022-02-27 13:00:00	2022-02-27 14:00:00	4.34902	-2.3814	8.05741	9.15	-28.87593	23.26667	86.0	0.0	50.0	-0.1192
8414	8	2022-02-27 13:10:00	2022-02-27 14:10:00	4.3557	-2.375	8.61013	9.15	-28.37468	23.26667	86.0	0.0	50.0	-0.11888

Figure A.22: A Sample of the Dataset

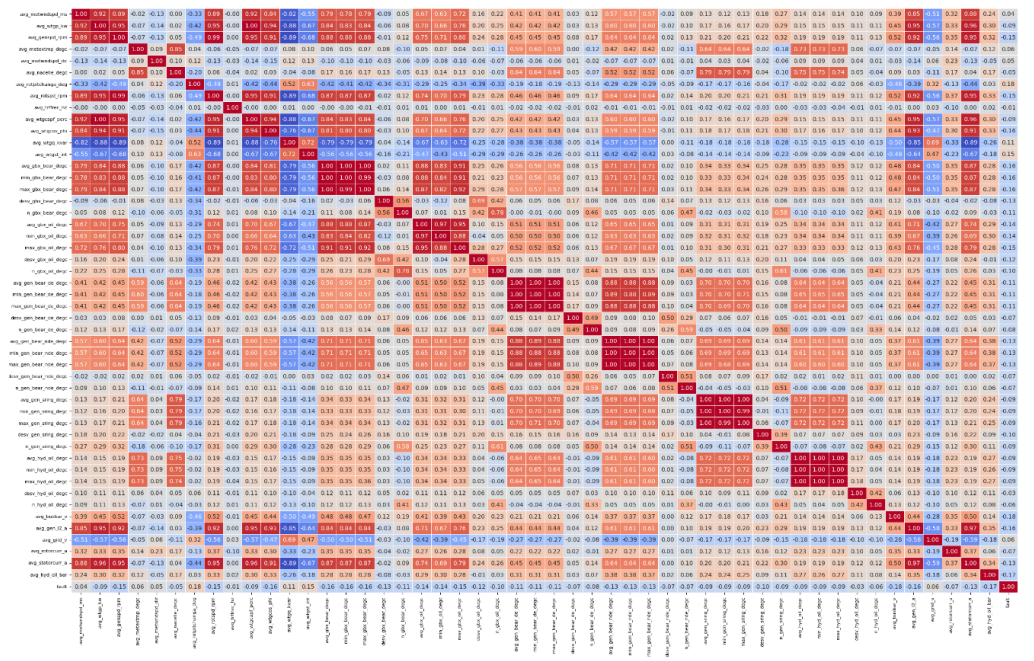

Figure A.23: Spearman's Correlation of the All Variables

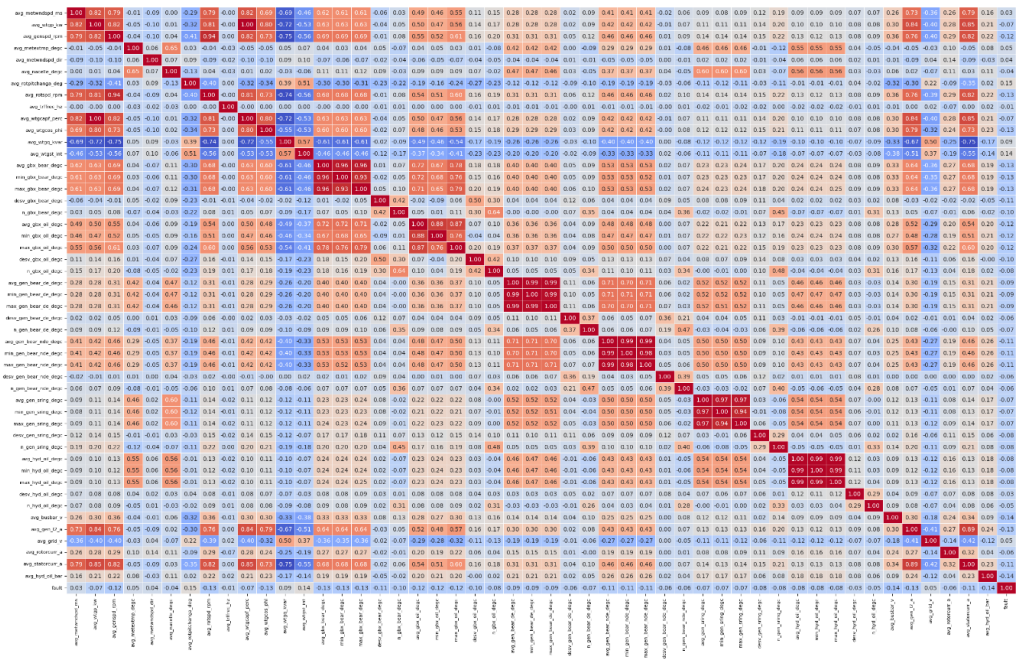

Figure A.24: Kendall's Correlation of the All Variables

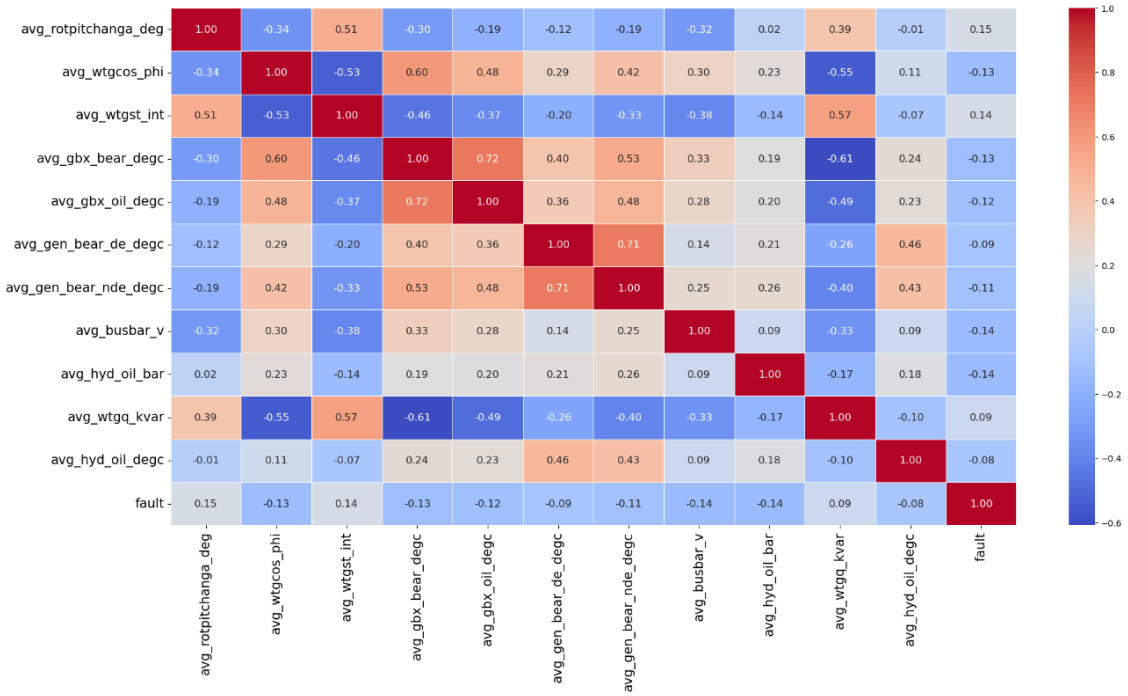

Figure A.25: Kendall's Correlation Matrix of the Chosen Variables