uc3m | Universidad Carlos III de Madrid

University Degree in Mechanical Engineering
Academic Year (e.g. 2022-2023)

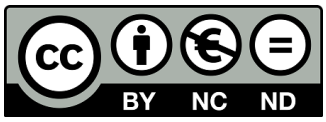
Bachelor Thesis

"Condition Monitoring and Remaining Useful Life Estimation of Wind Turbines by means of Machine Learning Methods"

Artur Adam Habuda

María Belén Muñoz Abella

15th of June 2023



ABSTRACT

The objective of this project is to demonstrate a set of advantages that Supervisory Control And Data Acquisition (SCADA) systems present for condition monitoring of wind turbines. Accurate condition assessment of all components in wind turbines can significantly reduced costs related to reparations and maintenance operations. Being in possession of a proven and reliable condition monitoring method can improve the competitiveness of wind power generation relative to non-renewable energy sources. This fact, could finally bring us closer to a green and clean future.

In this project, SCADA systems are utilized to build several machine learning models, whose mission is to find patterns in the SCADA datasets to predict and identify rises in the generator's bearing temperature levels. Such temperatures raises could suggest potential faults in that component or other peripheric devices. This models are presented in different case studies in which 6 Senvion MM92's wind turbines at the Kelmarsh wind farm in the UK, are tested. From SCADA files of 6 years of operation of each of the turbines, four machine learning models are built to predict bearing temperature levels. In specific a multiple linear regression model(MLR), a long short-term memory model(LSTM), an eXtreme gradient Boosting model(XGboost) and a deep neural network(DNN) are built, for this purpose.

The created models, predict with up to 80% accuracy the temperature levels of the bearings, while additionally triggering a set of alarms that may indicate dangerous or anomalous temperature rises. Not only it is demonstrated that data driven approaches are useful for indicating faults in wind turbine components, but also a reliable and robust methodology is presented to be used for real-world applications. In this work are also presented numerous ways in which the models could be improved and future lines of investigations that could be pursued.

Keywords: condition monitoring, wind turbine, bearing, MLR, LSTM, XGBoost, DNN, machine learning, SCADA system, data-driven

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to Amir Rasekhi Nejad. Without him this bachelor thesis project wouldn't have been possible. Although his name does not appear on the front cover of this thesis, due to formal procedures in my home university, Amir has been the main person supporting this project. He proposed it and agreed to guide me trough its development, without any apparent selfish interests. He was just interested in guiding and introducing a student into a fascinating field of study. I will always be thankful with him for this. Since the very beginning he encouraged me to pursue this work, and constantly showed support and provided guidance when I needed it the most. I couldn't have asked for a better mentor.

I would also like to sincerely thank María Belén Muñoz Abella, my co-supervisor back in my home university in Madrid, she also played a role in making this project possible. She eased every step in this work and always was keen to help and give me advice.

I also want to thank PhD candidate Etienne Purcell, who helped me to find solutions for challenges I found during the development of this project. He provided great advice and shared with me valuable insights.

Last, but not least, I want to acknowledge my parents and sister back in Spain who have always backed me and showed me support. They have made the development of this project in such favourable conditions possible. Also, wanted to thank my friends here in Trondheim who made everything less complicated and fun. They made me see the complicated things trough a different lens.

CONTENTS

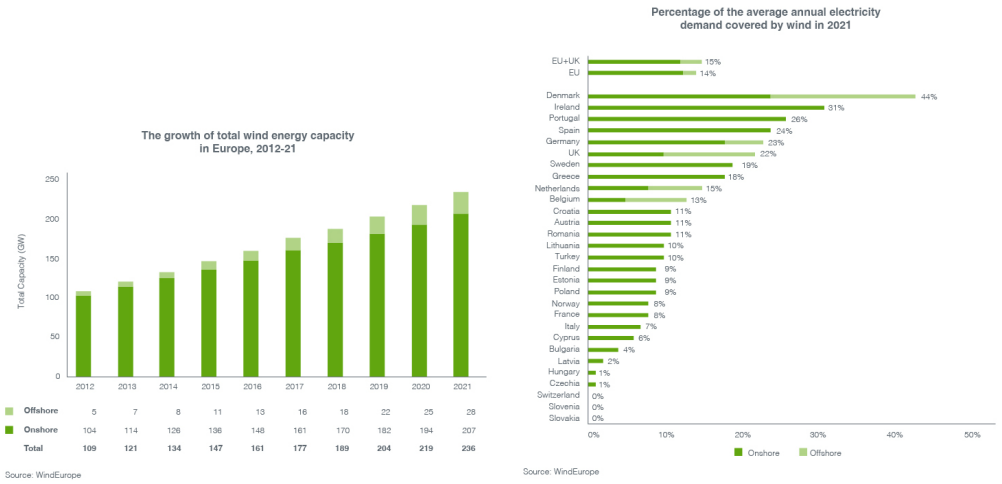
LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation

Wind energy is becoming an important player in today's world energy economy. Specially in Europe, numerous countries are dedicating increasing amounts of resources and efforts towards wind energy and its development. According to [1], since 2008, when the first wind onshore farms began to be installed, the energy share owned by wind generation has increased from 0% to 6,65% of the world's total energy production. And this numbers are even more impressive when analyzed for developed geographic zones such as Europe, where the importance of wind energy generation has been significantly greater. According to a report on the evolution of wind power in Europe by Iberdrola.SA[2], in the year 2021, wind generated 437TWh, enough to cover 15% of the European Union's(EU) electricity demand, of which 12,2% came from offshore wind and 2,8% from onshore wind. Additionally in certain countries such as Spain, wind energy secured it's spot as the leading electricity generation source, with an outstanding 24% share.



(a) Growth of total wind energy capacity in Europe, 2012-2021

(b) Percentage of the average annual electricity demand covered by wind in 2021

Fig. 1.1. Wind energy evolution in Europe [2]

Such efforts are motivated in large by international commitments oriented towards carbon neutrality. This is the case of the European Green Deal[3], a long-term plan mapped by the members of the European union, with objectives such as; a 55% reduction in greenhouse gas emissions by 2030(compared to 1990 levels) and becoming the first neutral continent by 2050. Other international treaties and agreements[4] of similar nature such as the Montreal Protocol, the Kyoto Protocol or the Paris agreement have all motivated this accelerated advancement of wind energy technology.

Wind energy generation has come a long way in the last 30 years, and during this time, wind turbine technology has experienced a tremendous evolution. In the early years, wind turbines were mainly used in remote onshore locations, and they were much smaller and less efficient than the wind turbines we see today. Nowadays wind turbines have grown taller and wider, with rotor diameters that can now reach up to 200 meters(Siemmens Gammesa, SG 11.0-200 DD)[5]. As wind turbines have grown in size, they have also become more efficient, with higher capacity factors that enable them to generate more electricity. Offshore wind farms have also attracted many investors and enthusiasts, who see their potential in allowing for better energy harvesting and reduced disturbance of human activities. Nevertheless, operating in offshore environments, imply harsher conditions for the operating equipment, due to higher air humidity and stronger winds. This increase in dimensions and more demanding environment, has inevitably lead to higher loads and stresses acting on the different components, which makes them more prone to presenting faults. As wind turbine farms grow in number, and the market becomes more competitive, it becomes crucial to reduce the downtimes produced by these faulty states. According to Stehly et al.[6], the operational cost for onshore WTs can range between 32 and 54$/kW/year, whereas offshore WT faults can cause expenditures ranging between 62 and 186$ /kW/year. Xiaohang Jin et Al[7] exemplifies that downtime cost for a 2MW onshore WT, operating at 50% of its rated capacity, and with an electricity price of 0,12$/kWh, would be 2,880$. Additionally, the cost of repairing a failed drivetrain component belonging to a current commercial WT could reach to 250,000$ [8].

A report including failure statistics from several wind farms located in Germany, extracted from 15 years of data of the German '250MW Wind program' report an average availability of about 98% .Data from Swedish turbines between 1997 and 2005 in Sweden, suggest an average failure rate of 0.4 failures per turbine per year. In these reports the electrical and electronic systems were the most prone to failure, whereas generator and gearbox faults caused the longest downtimes. Said cases serve to portrait the frequency at which faults can be expected during normal operation scenarios in typical wind farms. These, and more recent reports on fault statistics will be discussed on posterior sections of this work.

Considering the potential looses originated from faulty states in drivetrains of WTs, it becomes crucial to detect faults long before they are critical to the operation of the whole system, and ideally, estimate the remaining useful life of the WT so that appropriate actions can be taken to palliate the negatives related to the faulty component.

This work will try to collect, analyze and apply different techniques and methods, that allow the early and reliable detection of WT drivetrain faults, as well as create estimations on the remaining useful life of specific components within the drivetrain.

1.2. Main objectives

With the accelerated incursion of wind turbines in the energy market, governments and businesses strive to find new ways to become competitive with wind generation technology. Cutting down on costs, and reducing the financial impact of wind farms results crucial for this purpose. As it has been shown earlier, the rapid detection of faults and an accurate remaining useful life(RUL) estimation of any WT component can be an essential process in reducing downtime and reparation costs, which in turn help the economic viability of wind turbine generation projects. Taking into account the importance of this task, this work aims to summarize and discuss several detection and RUL estimation methods and then select and use this method on real WT sensor data, to test its effectiveness.

1.3. Work outline

After the introduction presented earlier, this work will begin, in chapter 2 "State of the art" by commenting on the State of the Art of several topics related to Wind Turbines: from the wind turbine themselves, to typical loads, main condition monitoring techniques, machine learning algorithms and a deeper description of data-driven approaches for condition monitoring purposes. The objective of this chapter will be to set the foundation of the rest of the work and explain some of the steps and features commonly found and applied in the industry.

Once, defined the state of the art techniques for condition monitoring, the developed work will be presented in the third chapter: "Developed Work". This will consists of several case studies and demonstrations of the results in a clear and structured manner. From the developed work, results, conclusions will be deduced and a set of ideas will be discussed with the purpose of finding the root causes to possible mistakes and further lines of investigation and improvement. This will take place in chapters 4. "Results" and 5. "Conclusions and Future lines of investigation" In chapter 4 there is a section called "Interesting points from the Case Studies". This section could be of special interest for the reader, as novel ideas are discussed and perhaps, interesting conclusions could be present.

To contextualize the utility of the work and obtained results, two chapters on Socioeconomic impact(chapter 6) and Regulatory Framework(chapter 7) will be also included.

To conclude the annex with all the code for the machine learning models built during the development of the project is also incorporated.

2. STATE OF THE ART

In this section a closer look into typical faults in WT drivetrains and current methodologies available for detecting and determining the severity of this faults will be carried out. A comprehensive explanation on Wind turbine, wind turbine main components and typically loads found for this components. Additionally, a deeper delve into machine learning algorithms will be performed, to demonstrate its potential for the achieving the aforementioned objectives. The goal in this section is to end-up with a reliable, effective method for diagnosing a WT, hence several trade-off analysis will be included to compare different available methods. Also, Failure Statistics and technology trends will be commented on, to acquire an overall picture of the current state of the field. This will allow for a selection of diagnosis method aligned with the current common faults seen in the industry and also provide this work with greater value, as it will be more relevant in the foreseeable future.

2.1. Wind Turbines

Wind turbines are in essence relatively simple artefacts that can be subdivided in several key parts. First the tower, that holds the task to support and position the the rest of the WT at an optimal height to take advantage of better wind conditions. Second, the nacelle, storing and protecting the drivetrain, responsible for transmitting and generating power; and the power converter liable of making the generated electric power suitable for the electric grid. Lastly, we find the rotor, compose by the blades and the hub, which are the responsible for harvesting the mechanical power stored in the wind and transmit it to the main shaft. Towers can vary in length and shape depending on their location and intended function.

Although this simple explanation serves to showcase the fundamental working of most commercially available WT it falls short in the context of a deeper analysis aimed to identify faults, which can be originated in any component in the WT, and can lead to catastrophic events that may force the stop of the WTs operation. A more in depth explanation, specifically focused on drivetrains, as it is the main topic of study of this work, will be carried out next.

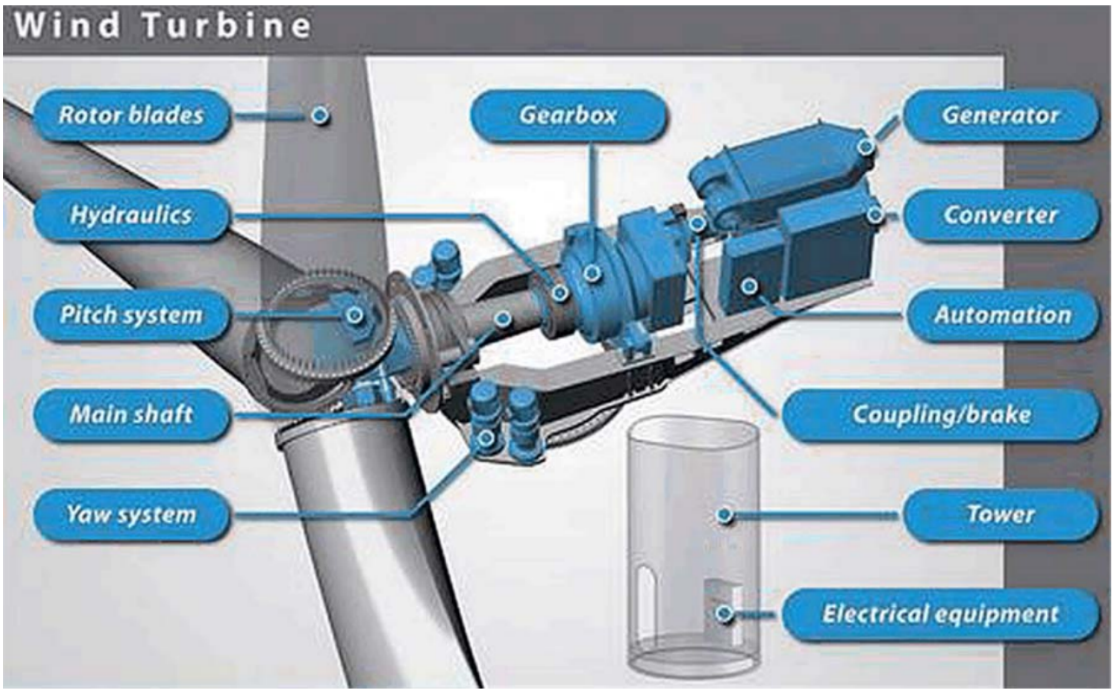


Fig. 2.1. Main parts and systems in typical 3-blade Wind Turbines.[9]

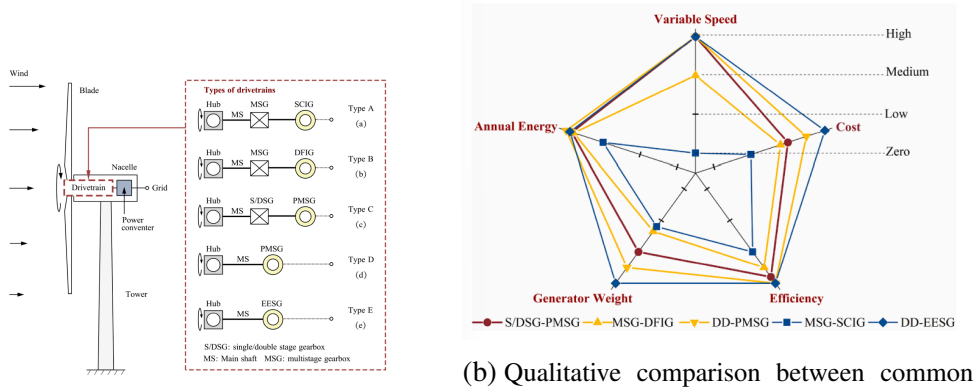
Generators

The main function of WT generators is to transform the mechanical power generated by the blades and the rotor into electrical power that can be utilized in the grid. The basic pyhisics principle behind the working of any electric generator, is electromagnetic induction, which allows for the creation of an electric current through wire coils, when a magnet is moving trough said coil. Inside the generator rotor and stator can be found, in which the first is composed by all elements in the generator that rotate and the letter in all elements that don't. The coil with current can be found in either of this elements, depending on the generator design, but in both cases the current flows from the output of the coil.

Drivetrains of wind turbines could be categorized into geared and gearless or direct drive(DD) types[10]. Geared systems are far more prone to failures and downtimes than gearless WT. Gearboxes constitute one additional component that can be subject to failure and it has one of the longest downtimes among all components within the WT. Gearless drivetrains include hydraulic or direct drive systems. Theses gearless systems require the use of big generators that inevitably increase the size and weight of the WT. Two main configurations are the most common for direct drive systems: permanent magnet synchronous generators(PMSG) and electrically excited synchronous generators(EESG). In general Direct drive systems are inferior relative to geared generators in terms of restrictiveness for the generators, as certain design conditions need to be fulfilled in order to cope with the large torque and low rotational speed provided by the rotor. Stronger generators are needed in this case.

For geared systems, on the other hand, other types of topologies and generators are usually used. Gearboxes reduce the demands on the generator by reducing torque and increasing rotational speed. Some of the most common configurations for geared WT are: Squirrel cage induction generators(SCIG), doubly fed induction generators(DFIG) and permanent magnet synchronous generators(PMSG) are usually the preferred choice.

Currently there is no clear winner between DD and geared configurations although single or double stage gearbox with PMSG seems to be a preferred choice in the industry.[10] A comparison between some of the mentioned configuration is provided in Fig 2.2. In said image some terminology to describe drivetrain configuration is used: (a) the multi-stage gearbox (MSG) with SCIG; (b) the MSG with DFIG; (c) the single/double stage gearbox (S/DSG) with PMSG; (d) the DD with PMSG; (e) the DD with EESG. Multi.stage gearbox refers to a gearbox with a combination of two or more gear pairs. It shall be mentioned that bigger and more complex generators characteristic of DD configurations are inevitably prone to higher failure frequency and longer downtimes.



(a) Common configurations of WT drivetrains

(b) Qualitative comparison between common WT drivetrain configurations

Fig. 2.2. WT drivetrain configuration and comparison[10].

A promising technology for generator manufacture could be high-temperature superconductor generator(HTSG) which allows for lowered weight, size, high efficiency and simple maintenance[11]. The technology is currently being explored by main WT manufacturers.

Possible faults in electrical machines range from electrical faults(stator or rotor insulation damage, open circuit or electrical imbalance) to mechanical faults(broken rotor bar, bearing failure, bent shaft or air gap imbalance). Electrical signals, electric machine vibration, shaft displacement, torque measurements and temperature of the generator are common ways for fault detection.

Power Converters

Before the electric power produced by the WT reaches the grid, a power conversion step has to be still performed. In this step the voltage and current coming from the generator coil is transformed into appropriate values so that it can be connected with optimum

efficiency to the grid.

Common topologies for power converters are partially loaded back-to-back converter(BTB), typically used with DFIG for generators of less than 3MW; fully loaded BTB with PMSG for power rating of 3MW or higher. For power levels of 10MW or more, multiple level BTB converters are typically connected in parallel.

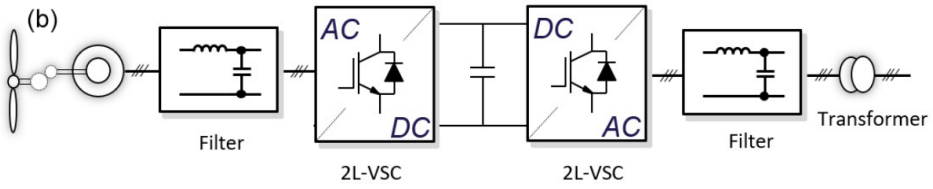


Fig. 2.3. BTB converter topology[12]

The electronic subsystems belongs among the most failure prone subsystems in WT. This is specially true in large capacity WT. Failure in power converters are mainly due to temperature, vibration and humidity. Temperature is the major stressor[9]. Capacitors, PCBs, semiconductors, solder joints or connectors are just a few of the components that can go wrong within the power conversion subsystem of a WT. Figure ... provides an overview of the most failure share of each component.

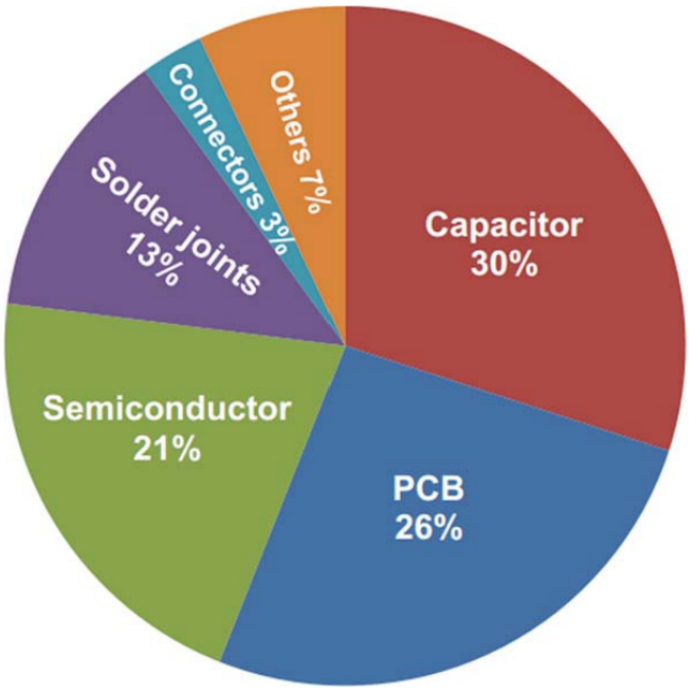


Fig. 2.4. Failure share of different component within the power conversion subsystem[10]

PCBS could potentially have broken buried metal lines, corrosion or crack, board

delamination, component misalignment,etc. Common capacitor failure modes include excessive leakage and shorts, dielectric breakdown and leads separated from the capacitor, to name a few.

Gearboxes

Three types of gears are mainly used in WT gearboxes: spur gears, helical gears and double helical gears. Among the spur gear advantages stand out simplicity and efficiency. Helical gears are characterized by higher operational torque and longer lifetime. Single and double helical gears are used at high speeds.

Among the most common gearbox configurations we can find:

- Simple parallel axis gearbox: intended for the lower end of WTs in terms of power
- Gearbox introducing planetary gears: which enhances power generation abilities of the drivetrain.
- Integrated Gearbox: a compact and lighter design that merges gear and rotor bearing
- Power split gearbox: specially useful for extra high-power applications(Several MW). Another gearbox useful for this purpose is multi-output gearbox.

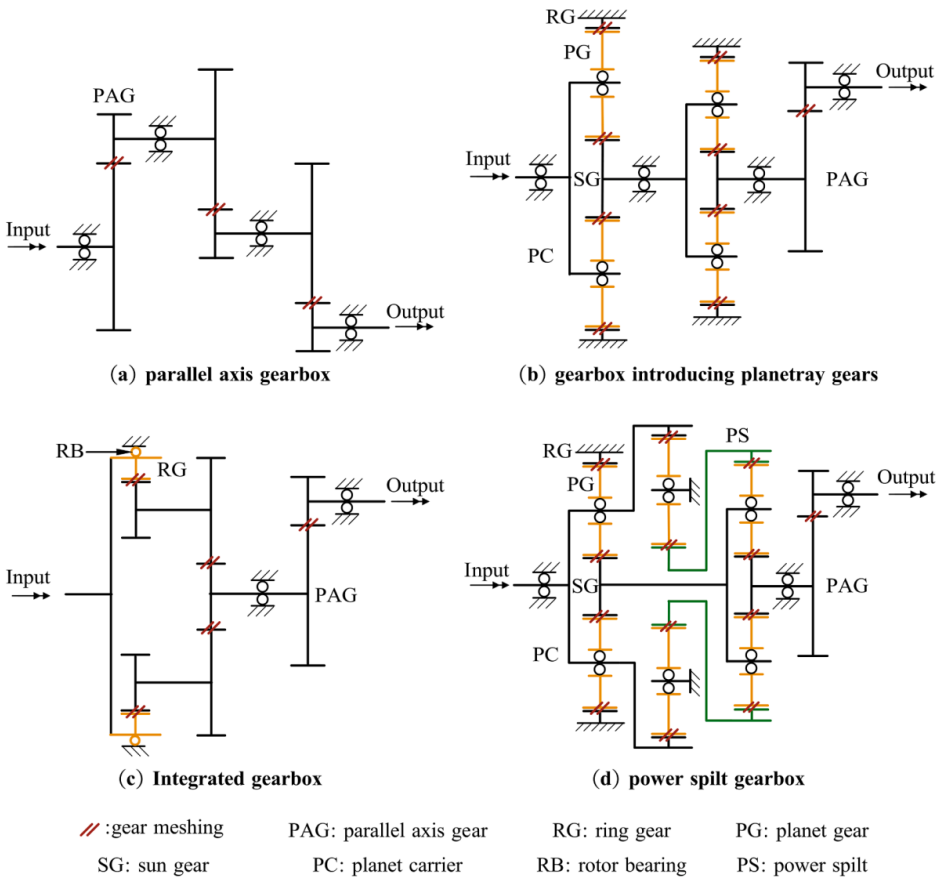


Fig. 2.5. Gearbox common configurations in WT drivetrains.[13]

Another common set of gearbox topologies is described in Figure, in which each topology is also associated with a power rating.

Type of gearbox	Power rate
Spur or Helical gear set gearbox (3-stage)	up to 750 kW
Planetary helical gearbox (3-stage)	750 kW to 2000 kW
Two Planetary helical gearboxes (3-stage)	2000 kW to 4000 kW
Extended Planetary gearbox (4-stage)	2500 kW to 5000 kW
Multi-Duored gearbox (4-stage)	6500 kW up to 12000 kW

Fig. 2.6. Gearbox configurations by power rating.[10]

One aspect to take into account when it comes to gearboxes is their complexity which seems to directly correlate with maintenance difficulty, downtimes and costs.

Most gearbox faults find their origin in design or material defects, manufacturing or installation errors, misalignment, torque overloads, surface wear, fatigue or crack development in certain areas. For example abrasion of gears could be caused by debris generated from bearing failure. Other failures of gears are independent of other components, such as tooth abrasion due to poor lubrication.

Electrical signals from the generator and vibration monitoring are the favored supervision techniques for gearboxes.

Bearing

Main bearings are responsible for holding the rotor in place, transmitting loads and maintaining alignment and structural rigidity of the wind turbine. Although many WTs still make use of ball bearings, the current trend in the industry is shifting towards spherical roller bearings in sub-5MW machines. Tapered roller bearings are also commonly used for this power ratings. For WT of 5MW and above, it is common practice to choose tapered roller main bearing technology. Bearings usually suffer significant variations in load which enhance possibility of suffering damaging events such as roller skidding, wear, abrasion or surface fatigue[14]. Bearing faults initially manifest as wear or surface roughness of certain areas, and slowly develop into more sever faults such as fatigue cracks or breakages of inner and outer races.

Vibration signals, electrical signal and acoustic emissions are the preferred parameters to monitor for bearing fault detection.

Due to the current trend in increased integration of components, the main bearing is even taking the role of supporting generator rotor while maintaining an appropriate generator air gap[12]. These aspect should be taken into consideration for WT reliable design, and even Condition monitoring, as perhaps main bearing could be find guilty of

faults presented by generator.

Other components

Other important components that play crucial roles in the power generation of WT: are pitch and yaw systems, mechanical brakes, hydraulic system, main shaft, secondary bearings, sensors and control subsystems. Although the role of these components is as important as the previous ones within a WT, a trough explanation of each of these is out of the scope of this work, in which drivetrain components are the main focus.

2.2. Loads on Wind Turbines

No fault detection, diagnosis or prognosis method makes sense, without understanding first the magnitude and nature of the loads under which a WT operates. These will affect the intrinsic characteristics of the data provided by sensors within the WT, that will be later used to detect faults and estimate their severity.

According to Xu, Ziyang et. Al [13] it is possible to differentiate between two types of loads acting on a WT, depending on their origin. External loads, that arise from outside factors such as wind, sea current or even gravity. On the other hand we have internal excitations; forces generated within the WT that affect the normal operation of the whole system. Further down the line, external loads can be subdivided mainly into loads in the hub side, input loads via the bed plate and electric disturbances. The loads on the hub side come mainly from the operating torque of the rotor, which is desired, and non-torque loads caused by bending moment on the main shaft of the turbine, that is induced by phenomena such as wind shear,gravity of rotor, tower shadow, control actions, etc. The thrust generated by the rotating blades can be assumed negligible[15]. The operating torque of the turbine depends directly on the speed of wind, and hence is highly variable. Providing enough torsional-dumping becomes difficult, specially for direct drive configurations. It is safe to assume that high speed winds can increase the vibrations in drivetrains. Non-torque loads are highly undesirable and are responsible for increasing the dynamic forces supported by bearings as well as increase the displacements within the gearbox. This can worsen gearbox internal responses such as tooth edge loading, load sharing or gear meshing. Tooth edge loading refers to the concentration of forces on the edges of the gear teeth. Load sharing describes the distribution of the load on each specific gear within the gearbox. Lastly, gear meshing refers to the process by which two gears engage with each other. The disruption of this phenomena can eventually lead to premature gearbox failure, which translates into extended downtimes and expensive reparation or replacement costs. Input loads via the bad plate, depend on several factors such as tower height, foundation type, materials, location or dumping of the nacelle[13]. Specially in offshore locations, platform motions on the surge and pitch directions must be taken into consideration. Said events can produce harmful consequences to drivetrains such as air-gap closure in generators or resonance in lower natural frequency components within the drivetrain. The first

refers to the closure of the air gap separating the stator and the rotor in electric generator which can alter the magnetic field produced in said generator and hence its performance. The latter describes the match of a certain body vibration to its natural frequency. Finally, electric disturbances of the form of voltage disturbance, can also produce fluctuations in the generator's electromagnetic torque that can cause even more serious vibrations than mechanical torque. Electric disturbances impacts negatively the effective damping for the electro-mechanical oscillation[16]. In edge scenarios, short circuits can produce sudden torque change and vibration in mechanical parts. Additionally, grid loss situations can cause important torque variations, since stored torsional energy is released by the drive-trains.[17] The excitation frequencies in power loss events are close to drivetrain natural frequencies, hence resonance could be perceived.

As for internal excitations, this are specially characteristic of gearboxes and generators. In gearboxes, meshing excitations induced by mesh stiffness, dynamic transmission errors, gear backslash and structural damping are the predominant internal excitaitons. For geared drivetrains, mesh frequency and its harmonics are the main focus of concern. Moreover, flexible structures and floating components within the gearbox, further enhance the magnitude of internal excitations. Noise in the gearbox is related to gear meshing and impact[13].

Regarding WT generators, high internal excitations are specially noticeable in direct drive systems. First, electromagnetic force generates noise and vibration, and second the air gap between generator rotor and stator is an important source of internal excitations as well. The generator rotor is subjected to external loads that inevitably influence the separation between stator and rotor. One can be pulled towards the other which deviates the stability of the air gap and generates non-homogeneous magnetic pull[18]. A stable air gap is required for power conversion. Bearings and other damping subsystems meant to isolate stator and rotor from external loads are typically installed in WT. Despite this said measures are also subject to degradation and its performance is not always ideal.

With the current industry trend towards more complex and highly integrated systems, understanding and monitoring internal excitations becomes more important than ever.

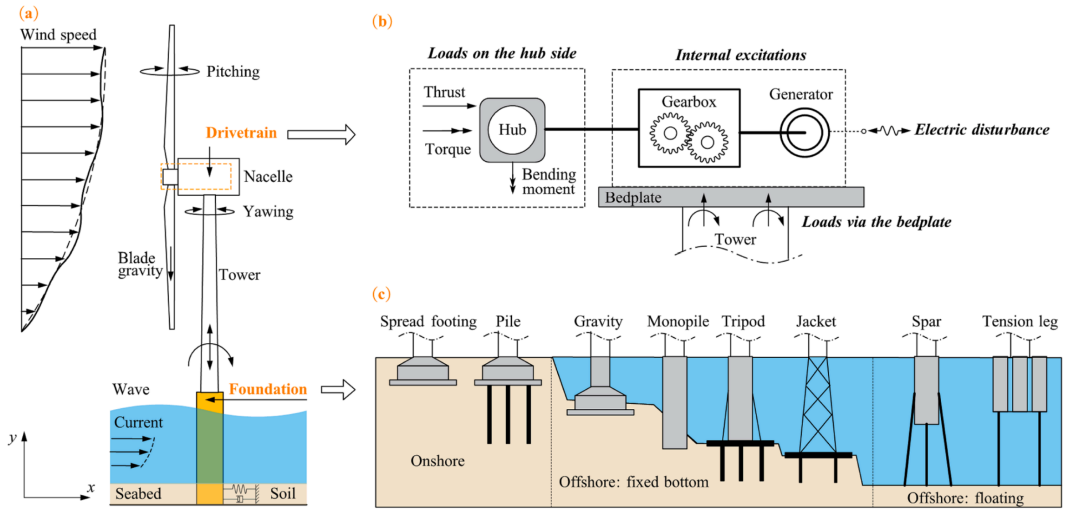


Fig. 2.7. Schematic external loads acting on Wind Turbine.[13]

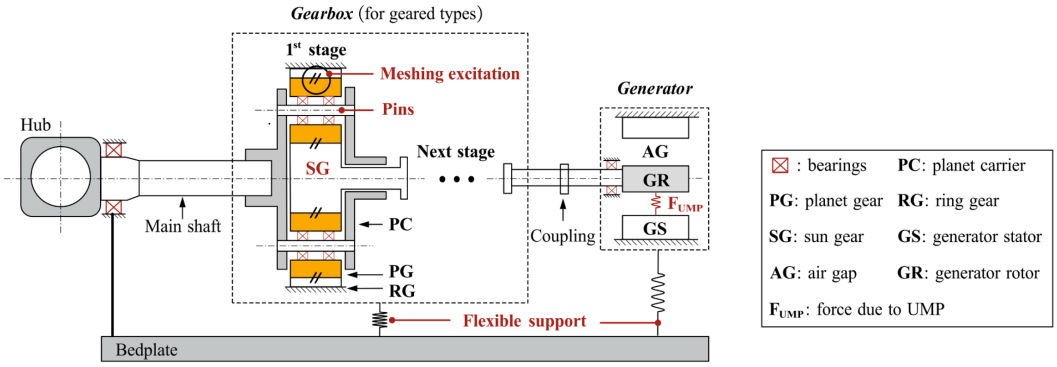


Fig. 2.8. Internal excitaitons drivetrain[13]

2.3. Failure Statistics

In an extensive wind turbine reliability data review conducted by Dao et al.(2019)[19], 90,000 turbine-years from over 18,000 WTs belonging to 18 data bases were analyzed to determine common trends in failure statistics among WT. This study observed that in terms of failure rates, electrical, control, blades and hub, and pitch systems are the most critical subassemblies for onshore wind turbines as well as offshore ones. In terms of downtime, gearbox, generator, blades and hub, and drivetrain were the most critical. It was also found that offshore wind farms posses higher average failure rates than onshore ones, as well as downtimes of offshore installations being approximately double than that of onshore installations.These discrepancies were attributed mainly due to the difficulties in repair/maintenance accessibility and harsh operating environment of offshore WTs. It is relevant to mention that this study pointed out that generally failure statistics on WTs posses significant variations in failure and downtime rates of different data sources. It is concluded that data volume, collection duration, location and WT power rating are all

influencing factors in failure trends in failures of these systems.

A different paper by Tautz-Weinert et al.[20] found out, from an evaluation of 15 years of German '250 MW' WT and more than 95% of all the WT operating in Sweden between 1997 and 2005, that electrical control systems were the most failure prone, and that the gearbox was the responsible for the longest downtimes. In the same study, Danish statistics for WT were provided and it was found that the most affected system was the yaw-system. In all the studied cases in the mentioned paper, the longest downtimes were caused by the gearbox.

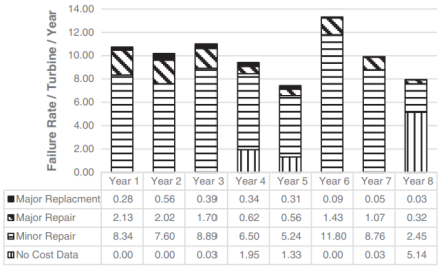
In Wilkinson et al.[21] a failure survey was analyzed, on the basis of 35,000 downtime events from 350WTs. It was concluded that subsystems characterized with higher failure rates were the power module assembly, the rotor module, control system, nacelle and drivetrain, in descending order. The subbasemblies with most frequent failure occurrence were pitch system, frequency converter and yaw system. The downtime hierarchy was very similar.

The National Renewable Energy Laboratory in the US[22], published a report, based on 289 failure events, in which approx. 70% of gearbox failures were caused by bearing faults and approx. 26% by gear teeth faults.

Perhaps the most relevant work due to its scope and relative novelty, is the one from Carroll et al.[23]. He noticed, from a study on approx. 350 offshore wind turbines, that the average failure rate per wind turbine is about 10. Conclusions were extracted from 350 WTs, with all turbines ranging from 3 to 10 years of operation on 10 wind farms throughout Europe(1768 turbine years).With dramatic replacements(more than 10k Euro) accounting for 2.5% of this failures, major repairs(classified was those having an expense between 1 and 10k Euro) accounted for 17.5% and small repairs(of less than 1k Euro) that were responsible for around 80% of the yearly failures. In this same study, the pitch/hydraulic subsystems, generator and power converter as the most prone to failure. The last two are also more incline to present failures in offshore conditions than in onshore ones. This disparity has been hypothesised to exist due to higher winds, bigger size of offshore turbines and harsher conditions. A clear correlation was deduced from failure data and wind speed, indicating that in fact a direct correlation between both variables existed, making higher wind speed condition less favourable for longevity and health of WT components. Hub, blades and gearbox had the highest repair times, repair costs and number of technicians needed for the repair/replacement operation in offshore WTs. However the hub and blades failure rate is much lower compared to the gearbox, which makes the letter a major contributor to overall O&M costs. Another differentiating element between onshore and offshore turbines is the lower failure rate of the onshore ones. The authors propose that apart from the reasons mentioned before, the easier accessibility to onshore WTs, inevitably causes them to be maintained to a better standard.

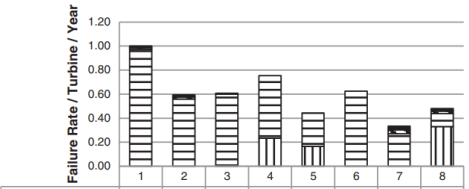
An interesting topic treated in this study is the analysis done on failure rate of WTs per year of operation. In the image below, a representative progression of the failure rate

of WT in the course of 8 years. In said image fairly high values are observed in the first 3 years, followed by a drop off in all type failures until year 6, followed by a peak in years 6 and consecutive reduction of failures in the following analyzed years. This points out a downwards trend in faults for all turbines, that has been already discussed in the literature. Some studies have made the comparison of the evolution of failures of WT with the bathtub curve[24] . Despite this, the paper argues that the systems that follow the bathtub curve(i.e pitch and hydraulic system) are outnumbered by the systems that do not(i.e. converter and electrical components). This challenges some of the common agreed concepts regarding trend of faults within the CM context and behaviour for scheduled maintenance operations, and perhaps opens up future possibilities for research. Keeping on that theme, it is worth mentioning that different O&M approaches should be adopted accordingly to the longevity of the system, as not only failure rate varies with time, but also the available data for training data driven models is significantly reduced in the early stages of a WT operation.



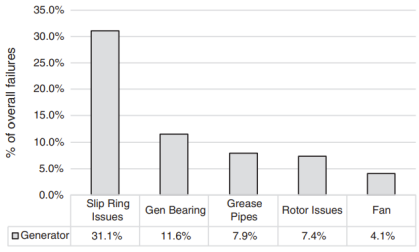
	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7	Year 8
Major Replacment	0.28	0.56	0.39	0.34	0.31	0.09	0.05	0.03
Major Repair	2.13	2.02	1.70	0.62	0.56	1.43	1.07	0.32
Minor Repair	8.34	7.60	8.89	6.50	5.24	11.80	8.76	2.45
No Cost Data	0.00	0.00	0.03	1.95	1.33	0.00	0.03	5.14

(a) Failure rate per year for WT



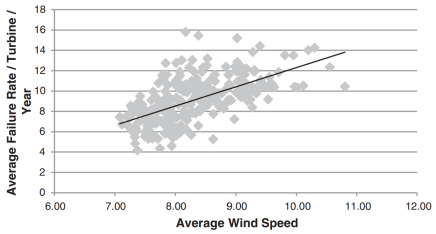
	1	2	3	4	5	6	7	8
Major Replacement	0.01	0.02	0.00	0.00	0.00	0.00	0.03	0.02
Major Repair	0.01	0.02	0.00	0.00	0.00	0.00	0.03	0.02
Minor Repair	0.97	0.56	0.59	0.52	0.28	0.63	0.27	0.11
No Cost Data	0.00	0.00	0.01	0.23	0.17	0.00	0.00	0.33

(b) Failure rate per year for Generator



	Slip Ring Issues	Gen Bearing	Grease Pipes	Rotor Issues	Fan
Generator	31.1%	11.6%	7.9%	7.4%	4.1%

(c) Generator failure modes



(d) Wind speed to Failure Rate relation

Fig. 2.9. Failure statistic representations[23]

From all the above discussed studies some common ground can be identified:

- Electrical, control, pitch and yaw systems, blades and hub are usually the most failure-prone assemblies.
- Gearbox, generator, blades and hub as well as
- Offshore WTs, in terms of failure rates and downtimes, are for almost every subassembly and component higher than for onshore ones.

- The vast majority of faults belong to small and medium repairs, which are also responsible for the majority of expenses of O&M operations.
- Failure rate is dependent on wind turbine lifespan, with varying but identifiable behaviour for different components

2.4. Condition Monitoring of Wind turbines

Condition monitoring(CM) is an integral part of the operation and maintenance(O&M) of a wind turbine. It is used as a mean of assessing the health of a component within the wind turbine. The literature[13] identifies 3 types of maintenance for wind turbines:

- Reactive maintenance(fault-based): refers to an approach consisting of replacing a component whenever a defect occurs or a set of defects accumulates on a given component. It does not utilize CM and it is the most expensive approach as it requires a full component replacement and potentially other components replacement whose normal operation has been altered by the defective component.
- Preventive maintenance(time-based): consists of a maintenance practice in which components are substituted before coming to a faulty state. A CM strategy can be adopted in these type of maintenance to asses the likelihood of defect appearance in a given component and repair it or replace it upon next intervention.
- Predictive maintenance(condition-based): in this approach, the condition of a given component or system, constitutes the driving parameter for maintenance operations. It is seen as the middle ground between time-based and fault-based approaches, where the moment of action is timed at a point in time where a fault is still not taking place but also it makes the most sense to act, according to other parameters such as low speed wind, availability of wind turbine reparation vessels(in the case of offshore installations),etc.

In the letter approach, CM plays a major role, as it is the responsible for determining the moment of action.

Monitoring can be viewed from different perspectives. Firstly depending on the physical impact of the monitoring technique upon the component being monitored. From this viewpoint Intrusive and non-intrusive monitoring can be distinguished.

- Intrusive monitoring: englobing methods that impose physical damage on the component being monitored such as oil debris monitoring, vibration monitoring, shock pulse, etc.
- Non-Intrusive Monitoring: composed of techniques that don't cause damage of the component being monitored. Some examples of this kind of monitoring are ultrasonic testing, power signal analysis or thermography.

On the other hand another differentiation can be done in the basis of the purpose. CM can be used for fault in real-time fault detection or in the future.

- CM for fault detection:a fault is identified at the moment of occurrence.
- CM for fault diagnosis: in which based on historical faulty data, a chosen method finds patterns in the input data to create predictions on future failure states of the components.

Another important function of CM is Remaining Useful life estimation(RUL) of any given component. This step is additional to CM for prognosis, as not only the fault is predicted to exist in the future but also an estimate on the temporal occurrence of the fault is produced.

2.4.1. Signals for Condition monitoring of Wind Turbines

Different signals are commonly used in the industry for condition monitoring purposes. Understanding they advantages and drawbacks is a key step for deciding which data to use and to get an overall picture of the different case scenarios in which each dataset could fit best.

Temperature

CM trough temperature control in WT is typically approached by supervising the output from a set of thermocouples placed in specific components in the WT, and watching that said output doesn't surpass certain values, that are considered to be in the range of normal operation. Spikes in temperature can be caused by degradation of gear, bearing or other mechanical components but also because of generator winding shortcircuits or excessive rotor speed. To give an example, power converters are usually monitored by using coolant temperature and case and junction temperatures of semiconductor modules. This displays the usefulness of utilizing temperature readings form different sensors in different locations for identifying faults in one given component.

Some of the arguments against and in favour of using this type of technology are:

- Temperature CM is considered a reliable and mature technology, that has proven to be cost effective for detecting faults in WT.
- Nevertheless, temperature raises, which are the basis behind the working of this approach, can be caused by meany events. It can be challenging to identify what is the root cause of these spikes. Additionally temperature sensors are intrusive and are prone to failure in harsh environments.

Acoustic emissions

Sound waves may be generated by materials subjected to stress, called acoustic emissions(AE).AE sensors are placed on structures of interest to detect and diagnose possible faults. AE waveform parameters such as rise time or amplitude can be use to estimate where and when damage could happen, and how this damage would progress. Examples of CM using this technology are shown in [25],[26] and [27], where gearboxes, bearings and blade faults were respectively detected and diagnosed.

AE condition monitoring techniques have both positive and negative parts with respect to other technologies:

- Noise signals have high SNRs and hence are suitable for highly disturbed environments. SNR is a measure used to compare the level of desired signal to the level of background noise, produced by disturbances to the signal's generating device. Additionally noise signals have higher frequencies compared to other signals, and therefore are effective in diagnosing incipient faults.

- On the other hand, condition monitoring trough this approach requires a large number of acoustic sensors trough the WT, which translates into elevated costs and complexity. Moreover, theses sensors can be of difficult access during the operation of the WT. High frequency data makes signal processing complex and computationally heavy as well.

Oil parameters

Lubrication trough oil is commonly used in rotating machinery in WTs. Common practice in the industry is to monitor oil characteristics such as water content, particle count, temperature, level or pressure[28]. These parameters give clues to contamination or degradation processes, which indicate the health of the WT. Two main types of oil monitoring for WT exist[29]:

- Offline oil monitoring: done by periodic offline sample analysis by extracting the sample and transporting it into analyses facilities. The main drawback of this method is that faults occurring between consecutive samples may not be detected timely.

- Online oil monitoring: measures oil parameters trough sensors installed in the WT. Measurements require additional information transmission systems that further increase costs of this type of monitoring. Although overcoming the key disadvantage of offline monitoring, increased costs and equipment complexity constitute important drawbacks. Another negative aspect is the complexity in interpreting the information provided by the inline oil sensors. Moreover, not all oil parameters can be monitored with this approach.

Electrical Signals

Motors and generators within the WT produce certain voltages and currents at its terminals that can be used for condition monitoring of wind turbines. Inherently, these type of signals are mainly utilized for analysis of generator or other electrical components faults. Typically these faults are studied from the harmonics of the produced electrical signals. Faults in an induction generator of a WT(Popa et al.[30], trough stator and rotor currents were identified. In Yang et al.[31] rotor electrical imbalance of an induction generator was detected and studied. Although electrical signals are mainly used for electrical components CM, it is also possible to identify mechanical faults with this type of signal. Thanks to the electromechanical coupling between many components and the generator, mechanical faults expressed in the form of vibration anomalies translate into modulation of the electrical signal produced by the generator. For example in Jeffries et al.[32], a significant reduction in stiffness of blades was detected trough monitoring changes in the power spectral density of the WT electrical power. In [33] and [34] they were able to identify bearing and gearbox faults respectively, from generator current signals.

The main advantages of electrical signal analysis:

- Necessary equipment already in use in WT: no need for added sensors. Hence lower capital expenditure and easier implementation than other signals, as vibration or acoustic sensors.
- Electrical signals are easily accessible and readable.
- Well developed instrumentation and understanding of its use. In general, electrical signal analysis is considered a reliable method for CM.

As for the disadvantages:

- Despite being able to detect faults in coupled components to the measured electrical signal producing devices, they present important limitations in terms of detecting mechanical faults of other components.
- low signal to noise ratio(SNR).
- Fault associated characteristics in the electrical signal are modulated trough its intrinsic harmonic components, which are proportional to the non-stationary WT shaft rotating speeds. Therefore complex signal processing models are needed for identifying fault signatures in the non stationary electrical signals.

Vibration data

Vibratory data has been historically of great interest for condition monitoring and health determination of WTs. It is by far the most used signal for CM. Usually vibration data is extracted by means of accelerometers, velocity sensors or displacement sensors, being the first ones the preferred choice due to their high sampling rate: 1 to 30kHz. By

analyzing the vibration signal in different domains, such as time or frequency domain, faults can be detected, and by analyzing the magnitude or frequency of certain components within the signal the severity of the faults can be assessed.

Gearbox[35], bearing[36], blades and rotor[37] are some examples of mechanical components that can be studied trough this kind of analysis. Possible analysis can be also carried out in tower, shafts, breaks, etc. The wide spectrum of components that can be supervised for anomalies, makes this technology very appealing for CM purposes.

Vibration data allows for the earliest fault or anomaly detection in WTs. On the other hand, vibration data is expensive both in terms of equipment in the form of sensors and computationally, due to the high sampling frequency of this kind of data. Additionally is is not capable of detecting electrical faults.

In Ibrion et al.[38] it is described the utility of different signals; such as vibrations, acoustic signal, lubrication particles and temperature; for detection of faults at different points in time. Vibrations are discussed to be the signal with highest capability for early detection of WT and temperature as the lowest one. This results reinforces the previously discussed ideas; in which temperature changes are often times associated with secondary effects of faults.

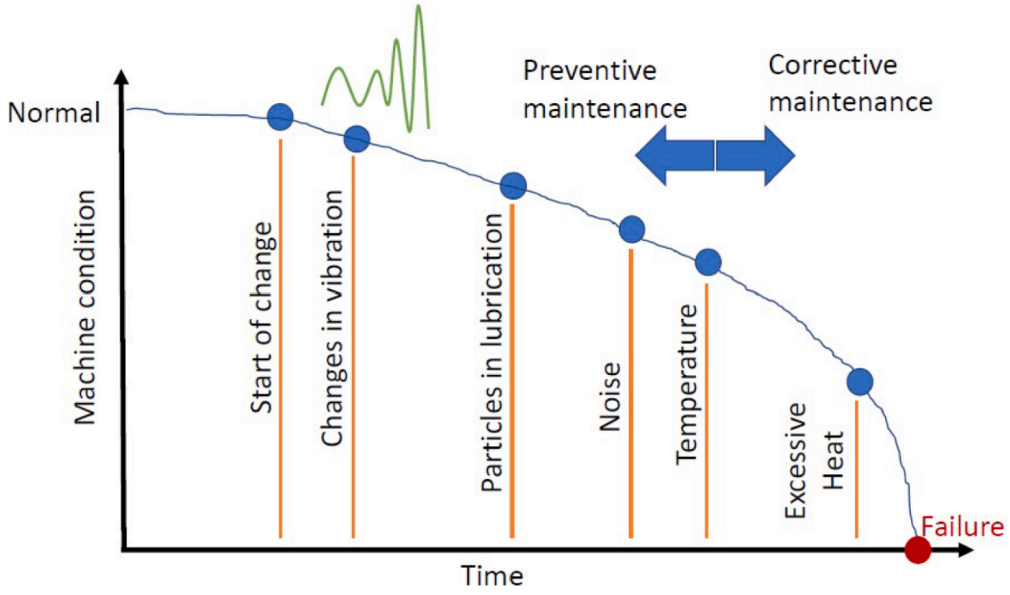


Fig. 2.10. Overview of incipient failure identification at different stages[38]

Merizalde et al.(2019)[39] proposed the predominance of vibration data for CM over SCADA and electrical signal pointing out some key advantages of this type of data:

- Every mechanical component with rotational, linear or reciprocating movement within the WT produces some kind of vibration that will be transmitted to all coupled structures. The vibration magnitude can be indicative of the equipment's health.

- Vibration signals are the earliest proof of faults
- It has less limitations than other signals for detecting faults in most WT components.
- It is a well matures and standardized technology

Regarding the disadvantages of this type of signal with respect to others:

- Vibration monitoring requires additional sensors and equipment which makes it more expensive,
- The installation of the sensors is intrusive and sometimes can result challenging. Additionally vibration sensors are delicate instruments which can be subject to failure.
- Vibration signal have a low SNR when used to diagnose incipient faults.
- High sampling rate of this kind of signal, brings big data challenges.
- Cannot detect electrical faults

SCADA data

Nowadays, wind turbines have multiple data sources available to make predictions on condition and health estimation of the different turbine components. Nevertheless, there is one set of data that is available in most common wind turbines. SCADA stands for Supervisory Control And Data Acquisition. These systems are built into WTs and provide input about a range of physical data relative to the turbine, typically in the form of 10 min intervals averages and statistics. SCADA systems are not originally meant for CM but rather for remote supervision and control purposes. Despite these, the high availability of these data, their economic appeal and the presence of extensive records along years of wind farm operation, makes this dataset a focus of research and technology development in the industry. Although other data could provide more accurate and reliable information about the state of various WT components, this are not standard among WT manufacturers, they lack a history record to be studied, and require additional costs to be installed and maintained.

As mentioned before SCADA data are typically averaged every 10 min and sampled at around 1 Hz. Although not all SCADA parameters are useful for CM applications, many are quite revealing on the state of different WT components. In Nejad et al.[12] they propose a classification of SCADA data according to its belonging to Environmental, Electrical, Control variable or Temperature. In Kusiak et al.[40], the proposed classification of SCADA parameters is wind parameters(direct measurement of wind), Energy conversion parameters(related to energy conversion process such as power output, pitch

angle, rotor speed...), Vibration parameters(both from nacelle and tower) and Temperature parameters(temperature measured at turbine components).

Category	SCADA parameter
Environmental	Wind speed, wind direction, ambient temperature, nacelle temperature
Electrical	Active power output, power factor, reactive power, generator voltages, generator phase current, voltage frequency
Control variables	Pitch angle, yaw angle, rotor shaft speed, fan speed and status, generator speed, cooling pump status, number of yaw movements, set pitch angle and deviation, number of starts and stops, operational status code
Temperatures	Gearbox bearing, gearbox lubricant oil, generator winding, generator bearing, main bearing, rotor shaft, generator shaft, generator slip ring, inverter phase, converter cooling water, transformer phase, hub controller, top controller, converter, controller, grid busbar

Fig. 2.11. Summary of basic SCADA input variables [14]

Condition monitoring based on SCADA data targets, in most cases, secondary effects of the fault. Typically SCADA based CM, uses WT under performance or anomalous overheating of specific components to detect faults within the wind turbine.

Another problem found with SCADA data, is the time averaging performed to the sensed signals. As mentioned before, SCADA data is usually collected as statistical data from 10 minutes of operation of the WT. This sampling rate hinders the possibilities of this type of data for incipient fault detection, as much time is needed for any decision autonomous or experience based approach for CM to get enough data to draw any conclusion on the health of a given component.

One of the main advantages of this type of data is its potential to be utilized by data driven algorithms for condition monitoring. Some of these algorithms thrive with large amounts of data at their disposal, and the long history of available SCADA data from different wind farms around the world, make this dataset ideal for this application. 'Data-hungry' ML models are currently booming on today's day an age, and new and exciting research is being developed in this area. Hence SCADA data is the best bet for taking the most advantage out of this effort. SCADA is a large, standardized set of parameters present in almost every modern wind turbine, so developing CM methods on top of this resource makes a lot of sense from a practical standpoint. It shall be mentioned that the access to this kind of data from research institutions is limited, and most available data is restricted to private corporations. Although competitiveness and business advantages are important factors for companies, the disclosure of this kind of data could be very beneficial for the research community, as more rich data pools would greatly enhance the pace of progress and discovery from the academic side of the industry; and consequently to the entire wind generation industry.

2.4.2. Fault Detection

This is the first and most basic step in any condition monitoring approach. As explained before, fault detection consists on determining whether a fault has happened or not. It involves keeping track of various parameters within the WT to asses its performance, identify 'normal' behaviour, and consequently recognize when anomalies happen. This anomalies or deviations from normal operation are finally judged to conclude if they correspond to a fault or not. Normal behaviour is considered as optimal or in line with the expected values projected for the given WT. Any behaviour out of this norm is consider anomalous. Fault detection also involves determining the severity of a given fault. This is achieved by empirically measuring the deviation of the anomalous behaviour relative to the defined normal.

Fault detection is an indispensable step in diagnosis and life estimations of components, but the level of uncertainty in this area of research is much lower than in the other cases. For this reason less attention is given to this topic in the recent literature. Nevertheless, several methods, based on human expertise on the topic, data driven approaches, or a combination of both, already are capable of reliably and accurately determine if a fault is taking place. In Ling Xiang et al.[41] a data driven approach for detecting faults is presented, in which a data driven approach based on Convolutional neural network(CNN) and attention mechanism(AM) is utilized to extract useful patterns from SCADA data and distinguish between 'healthy' and 'faulty' states. Terms such as CNN or AM belong to a series of data-driven approaches that will be treated with more detail in section 2.3 of this work.Another set of examples of fault detection methods is presented in Bebars et al.[42], in which a review for doubly-fed induction generators(DFIG) is carried out. The paper discusses methods that can be used to detect internal electrical faults in a DFIG stator, rotor or both.

2.4.3. Fault Diagnosis

Once a fault is detected, fault diagnosis focuses on identifying the root cause or specific type of fault within the wind turbine. This step is key for good planning and timing and good planing of maintenance operations. In general fault diagnosis methods can be classified in knowledge-based methods, analytical models and data-driven methods(Mingzhu Tang et al.[43]). Knowledge based methods rely on experts experience. These methods accuracy and reliability largely depend on the quality and amount of the expert's knowledge and understanding of the topic. The main drawback of this approach is the poor self-learning abilities and error recognition from previous diagnosis, relatively to the other two methods.

Model-based methods rely on mathematical models of the turbine system, to analyze the WT and achieve real-time diagnosis of the faults. These diagnosis are hence very dependent on the model parameters and the accuracy of the internal WT system mod-

elling dictates the accuracy of the outputs. The modelling of the aforementioned system can become highly complex due to elements such as gearboxes, in which complex contact mechanics between gears require simplifications that give rise to residual errors and approximate values. Often, a trade off between model accuracy and computational expenditure has to be done(R.Nejad et.al[12]).

Data-driven approaches, such as principal component analysis, support vector machines, artificial neural networks, and deep learning algorithms, leverage historical data and patterns to classify and identify the fault type.

2.4.4. Fault prognosis

Fault Prognosis is concerned with estimating time related information about faults. In specific, a subset within fault prognosis, is Remaining Useful life estimation(RUL), that relates to the prediction of the remaining operational life of a wind turbine or its components. This information is crucial for planning maintenance activities, managing spare parts inventory, and minimizing downtime. RUL estimation methods can be broadly classified into data-driven, model-based, and hybrid approaches. Data-driven methods, such as regression analysis, Kalman filters, particle filters, and recurrent neural networks like LSTMs, use historical data to predict future performance and failure times. Model-based methods rely on physics-based or empirical models, while hybrid approaches combine data-driven and model-based techniques for improved accuracy.

2.5. Machine Learning Algorithms

Two simple distinctions can be made based on the level of human supervision required for the successful deployment of the machine learning method.

Supervised learning: It draws conclusions based on labeled input data. Each data point is connected to a known output or goal variable. These algorithms develop a mapping function that they can use to forecast the results for brand-new, unexplored data points. A rule or connection that maps inputs to outputs is referred to as a mapping function. A mapping function is used in the context of machine learning to explain the connection between the input data (independent variables) and the output data (dependent variables), allowing predictions to be made for new, unforeseen data points. Such mapping function can be denoted by the formula f(x) = y, where x denotes the input, y the output, and f the function denoting the connection between the two. To learn the best approximation of this function from a given collection of training data is the objective of many machine learning algorithms.

In supervised learning tasks, such as classification and regression, the mapping function is learned based on the input-output pairs in the labeled training dataset. Once the function is learned, it can be used to make predictions on new, unseen input data points.

Regression models predict a numerical output when the system is operating at what is called "normal" or "healthy" state. For example, a regression model of the power curve of a specific wind turbine can be built (power generated by the turbine vs wind speed), and hence a more accurate representation of the power output of the turbine in different external conditions can be obtained, instead of relying on the power curve given by the WT manufacturer, which is obtained in one specific environmental condition. Outliers or points that deviate greatly from the trend depicted by the regression model can indicate a fault in one of the components of the WT, which can be very useful for CM purposes. Regression models can posses different levels of complexity

Classification models, on the other hand find relationships between independent variables within sets of data. This sets of data are often times assigned to predefined categories identified by labels.

Machine learning models use various techniques to learn and represent mapping functions. Some examples include linear regression (where the mapping function is a linear combination of input features), decision trees (where the mapping function is represented as a tree-like structure of decisions based on input features), and neural networks (where the mapping function is represented as a composition of multiple layers of interconnected nodes or neurons).

The choice of the mapping function depends on the specific problem, the data available, and the desired level of complexity or interpretability of the model.

Some interesting supervised learning models include:

- Artificial Neural Network(ANN): This kind of models consist of three layers, each composed by a set of neurons: input, output and hidden layer. Signals travel from the input layer to the output layers, passing trough the hidden layers, in which this signals are transformed by mathematical functions chosen by the creator of the net to optimize for the desired outputs. Each neuron processes the signal received trough the connection with another neuron, which then forwards the transform signal into another neuron. This process iterates until reaching the output layer. Each neuron get a weight assign, which automatically adjust according to its contribution in yielding a correct prediction in the output layer. Commonly used neural network methods are: self-organizing maps neural networks(SOM), radial basis functions(RBF) and Adaptive resonance theory(ART). ANN provides good robustness and precision but it also requires numerous parameters and extensive training time. It also works well with non-linear data is performs exceptional good in contexts in which incomplete or poorly categorize data exists, as it can learn an generalize with ease. This kind of technique may not be optimal for non data-rich contexts and it can make it difficult to interpret and explain learned relationships.

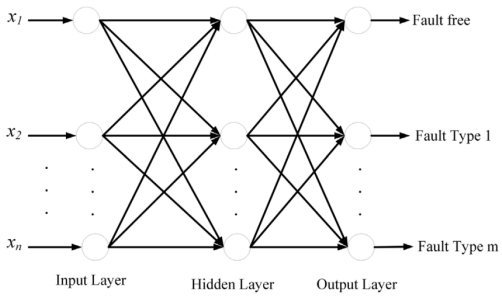
- Support Vector Machine (SVM): This type of algorithms is used both in regression and classification tasks in which the objective is to fin two hyperplanes to separate

two sets pf data in a multi-dimensional space and maximize the margin between the hyperplanes. In simpler terms, given a dataset, a decision plane separating all datapoints in two groups, is tried to be defined, in order to maximize the distance from the datapoints of each group to the decision hyperplane. SVMs are effective in high-dimensional spaces and robust to outliers. On the other hand they are computationally expensive for large datasets.

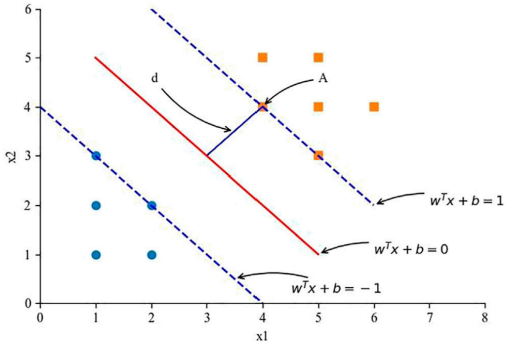
- Decision Tree(DT): this machine learning models work by recursively partitioning the input data into subsets based on feature values, yielding a tree-like structure. A set of conditions is established at different levels of the learning process, and datapoints are assigned to certain categories or processed further than the line, depending if they fulfill or not his conditions. Some advantages of this method, is that it is relatively simple to interpret, as it mimics human decision-making; it doesn't require much data pre-processing; and has fast training and prediction times. On the other hand, it is prone to overfitting, can be very sensitive to small changes in the training data and often is not as accurate as SVM or ANN.

- Ensemble Learning: The basic idea behind Ensemble Learning is to merge and train several base learning models into a greater model that has better performance on average than any other single member.Some frequently employed ensemble learning techniques are bagging and boosting. Bagging consist in dividing the original data sets into smaller datasets and training different models with each sub-dataset. Then, the prediction of all the methods is combined to produce a more reliable estimation. A popular bagging algorithms is random forest(RF). In boosting algorithms, similarly to bagging ones, different machine learning models are used to make predictions on the dataset, but in this case the training is not performed in parallel among different sub-datasets but each model is trained on the same original dataset, and the weight given to each prediction is given according to the correctness of the models prediction. Popular boosting algorithms are XGBoost and LightGBM. Among the positive points of Ensemble Learning models are: good performance and generalization and reduced overfitting by averaging the outputs of different models. The negative aspects are the increased complexity compared to single-model approaches, computational expenditure and difficult interpretation.

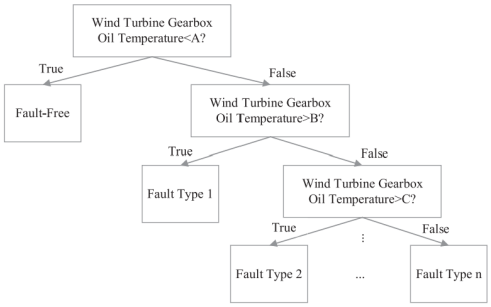
- Deep learning: this basically refers to deeper ANNs, composed by multiple layers of hidden layers. Configurations such as deep belief nets(DBNs), deep autoencoder(DAE) networks and convolutional neural netwroks(CNNs) are commonly employed. Deep learning algorithms are capable of working with hierarchical and abstract representations of data and work really well with large datasets. Nevertheless they are computationally heavy, prone to overfitting and difficult to interpret and tune.



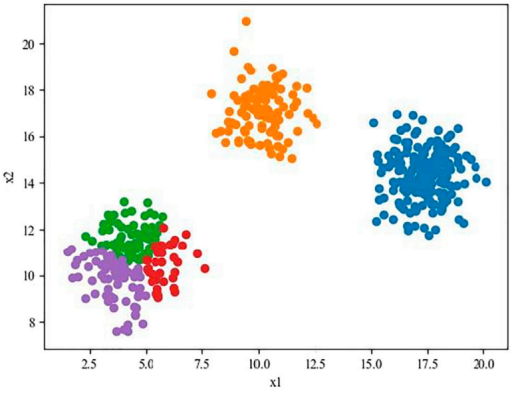
(a) Artificial Neural Network scheme for fault detection and diagnosis[43]



(b) Representation of Support Vector Machine.



(c) Decision Tree schematic for Fault diagnosis



(d) K-means clustering

Fig. 2.12. Schematics for different machine learning methods[43]

Unsupervised learning: provides output predictions based on unlabeled input data, by means of recommender systems, clustering algorithms, etc A very representative technique in unsupervised learning is clustering. Most methods in this kind of learning are based on clustering. A data set is organized into clusters or groups according to their similarity in terms of certain parameters which the algorithm is trying to optimize for. K-means, fuzzy C-means(FCM) and Gaussian mixture model are some typical examples of clustering algorithms.

- K-Means: this algorithm tries to divide the initial dataset composed by n observations into k clusters. An ideal data point or mean is chosen for each cluster and then all datapoints or observations are assigned to one of the clusters. K-means is simple in its implementation and performs well for fault diagnosis purposes. Despite this, the initial cluster center or 'ideal' data point of each cluster can be difficult to choose and often times is the source of problems in the case of WT fault monitoring

- Fuzzy C-Means(FCM): for this algorithm each data point can belong to more than one cluster. The goal is to maximize the similarity between points in a cluster while minimizing the similarity between points in different clusters. One of the drawbacks of this type of algorithm, similarly to other unsupervised methods commented in this section, is its inability to create effective predictions from large, almost fault-free datasets.

- Hierarchical Clustering: in this method a hierarchical structure of clusters is created. This hierarchy can be pictured as a tree-structure, in which the leaves are the original observations are slowly the model converges into fewer branches until reaching the root or main branch in which a definitive cluster is identified. Gaussian Mixture models are time consuming and therefore unsuitable for large WT datasets, due to the need of calculating large proximity matrices between data points.

- Gaussian Mixture Model(GMM): this models assumes that all observations fit to the Gaussian distribution, and is created from a mixed number of statistical models with unknown parameters. By conforming a linear mixture of Gaussian distribution functions, data distribution can be performed. GMM perform positively in big volume datasets, but is computationally heavy and time consuming.

Tang et al.[43] proposes an additional categorization of computer learning technique called Semi-supervised Learning Methods. This method recognizes some shared characteristics between labeled and unlabeled data samples to assist in determining the model's characteristics and to transfer labels from labeled to unlabeled data. The main idea behind semi-supervised learning models comes from the principle that it is easier and cheaper to train a dataset from unlabeled data than from labeled one. To improve the accuracy of the prediction labeled data is added.

- S3VM: this method takes the basic idea behind SVM models. It defines a hyperplane with the unlabeled data and then improves the prediction performance by adding labeled data until successful fault diagnosis rate is sufficient. The main drawback of such model could rely on the uncertainty in determining how large should be the WT data-sample to create an effective model.

- Generative Models: these are based on the likelihood that unlabeled observations belong to a given category. An expectation maximization algorithm is used to make the likelihood estimate.These models are robust, but have shown low accuracy and long training times.

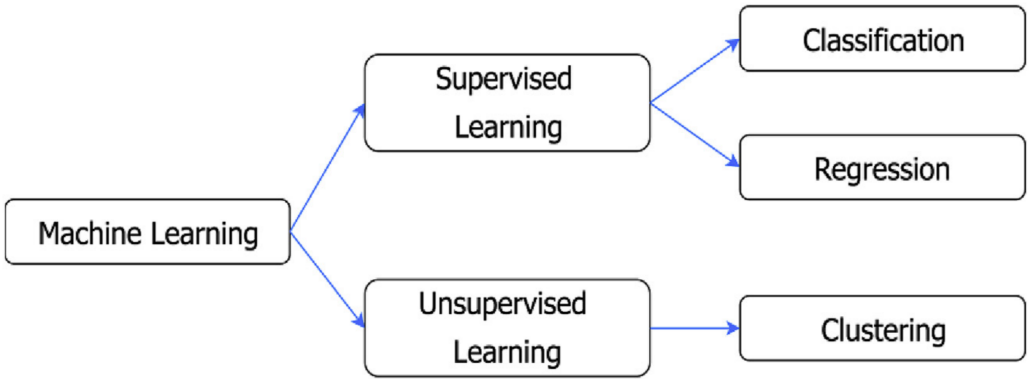


Fig. 2.13. Taxonomy of ML models [13]

2.6. Data Driven Condition Monitoring

2.6.1. Data acquisition

The initial step of any data driven condition monitoring approach is to collect the data. This is usually done by means of sensors placed on the wind turbines measuring all sorts of parameters of interest: vibration, temperature, acoustic emissions, electric signals,etc. Typically the data of interest is the described in the section of this work: "Signals for Condition monitoring of Wind Turbines". These sensors provide a comprehensive picture of the turbine's performance and health.

2.6.2. Data Pre-processing

Raw data from sensors can oftentimes be noisy and contain outliers. This step involves outlier identification, which consists on discarding extreme or likely impossible values originated from measurement errors, extreme environmental conditions or some other anomaly in the habitual operation of the WT.

It is interesting to mention a study from Puig et al.[44], in which the effectiveness of purely data driven models for filtration and elimination of outliers of data, seems to perform poorly. More specifically, it was experimentally demonstrated that although performing well in train data set, most of the models decreased their predictive performance when faced with new data. This was attributed to the removal of many outliers that were indeed fault states of the WT. They argue that a more intelligent strategy would be to manually define ranges(absolute and relative) for the variables to be analyzed. Manufacturers and human experts asses the normal operation ranges for the WT and said figures should make sure that the selected ranges are broad enough to contain values of healthy and damaged wind turbines, excluding real outliers.

2.6.3. Feature selection

Due to the size an amount of data available for post processing, in the context of condition monitoring, feature extraction becomes a critical mechanism for reducing the computational load of the entire process as well as removing possible outliers than can hinder the performance of the predictions done by the system. By keeping the main characteristics of the signals produced by the sensors, noise can be effectively discarded, and speed of training model is enhanced.

Feature selection consists on the process of selecting variables that relate to the outcome we wish to analysis, understand or predict. This process can be done with the help of human expertise, choosing to keep certain type of data and discarding other, on the basis of the experience of the supervisor; or automatically using an appropriate method. Some of these methods are:

- Wrapper method: this technique sees ML algorithms as black boxes which are given a series of features as inputs. Upon these subset of inputs, a certain prediction will be made, and on the basis of the performance of this prediction, the relative usefulness of the subset of variables will be judged. The user has a high degree of control over the method, as he is responsible for selecting the algorithms to use, the strategy for selecting features and the performance measure of the model.

- Embedded methods; in this methods, feature extraction is performed as a part of the training of the ML model, in which the algorithm/algorithms responsible for executing the prediction adjust the selected features accordingly to the veracity of these predictions.

- Filter methods: In this category, feature extraction is separate from the model itself. They create a significance test between features and their outcome and then rank them. The user is responsible afterwards, for selecting n number of features, based on the performance they provide with a given model.

The main difference between all models relies on the degree of involvement of the user in the feature extraction methodology, and the dependence or integration of the feature extraction mechanism within the ML model itself.

2.6.4. Feature extraction

At this stage high-dimensional time series are compressed while maintaining the relevant characteristics of the signal intact. Hence correlations and noise are the objects to be discarded[45]. There are some preferred techniques to perform feature extraction:

- Statistics: simple to calculate and most efficient method. Mean, maximum, minimum, skewness, kurtosis, peak-to-peak, root mean square,etc; these are just some of the commonly employed statistical techniques allowing for extraction of meaningful features from the signal.

- Time-Frequency domain properties: these methods convert signals in time domain to signals in the frequency domain. These strategies enjoy a great deal of popularity in the industry, due to their capabilities. First, it makes easier the identification of characteristic frequencies, which are associated with faults of some components in the WT. Second, noise reduction is also achieved, resulting in a cleaner signal for future analysis. And third, Frequency-domain analysis can provide more compact representation of the data,as it allows for identification of dominant frequencies and discarding less important frequency components. This brings computational advantages.

- Parameters or fitted time series: these are parameters obtained from fitted time series models, that describe the relationship between datapoints collected over time.

Examples are ARIMA models[46], a technique used for forecasting that combines autoregression, diferencing and moving average to capture temporal dependencies in the time-series data and can help identify patterns on this data; Autocorrelation coefficients[47], which are a measure of the similarity between one signal at different moments, and helps identifying structures in the data; Stochastic processes, which are mathematical models describing the evolution of random variables over time; and Hidden Markov Models [48], a type of stochastic process that assumes the system being modeled as a Markov process. A Markov process is a sequence of random variables in which the future state of the process only depends on the current state and not any other past states.

2.6.5. Condition assessment: fault detection, diagnosis & prognosis

Following feature extraction, by revealing the relevant features from the preprocessed data; machine learning algorithms combined with statistical techniques are utilized to identify patterns within the data that are indicative of faults or degradation of a component in the WT. The methods discussed in the fault detection, diagnosis and prognosis sections fall into this category. The fault is detected and its severity is assessed, the fault is diagnosed; meaning that estimations on the location, nature and time related issues about the fault are generated; and lastly a prediction on the future behaviour of the fault and a remaining useful life estimation are computed.

Typically one(or several) of the ML models explained earlier(either regression or classification based) is chosen to perform the predictory actions. This model will work based on a dataset representing the behaviour of the WT and containing information about the health status of it. Some examples that can result useful to understand the principles behind the assessment of these faults involve miscorrelation or anomalous behaviour of certain SCADA parameters. For instance, miscorrelation between wind speed and generator active power can indicate a fault in a wind turbine[49]. Miscorrelation between generator speed and generator active power could potentially indicate a fault in a WT generator[50]. Anomalous behaviour of temperature caused by heat transfer can serve for identifying the trajectory of a generator fault[51].

This dataset is split into training and testing sets with typical split of 70:30 ratio in favour of the training group. Usually unbalance will exist in the dataset as healthy data dominates over anomalies. For that reason balancing techniques are oftentimes utilized. Under-sampling works by removing instances in the class that overpopulates the dataset. Although this method is prone to discarding useful data[52]. On the other hand oversampling adds datapoints to the minority class. In this case, overfitting can be a potential issue. Often times, several ML models are tested to asses which one fits the best the working environment and machinery involved. For that reason metrics are needed to judge the performance of a model. Before discussing these models some terminology shall be first showcased:

- True positive (TP): this refers to a prediction of healthiness when a component was in fact healthy. A high TP value increases the reliability of the model as ir reveals correctly the component's health status.
- True negative (TN): it refers to a prediction of unhealthiness when the component was actually unhealthy. A high TN value increases the reliability of the model as well.
- False positive(FP): a healthy prediction when in reality the instance was unhealthy
- False negative(FN): a prediction of unhealthiness when the instance was in fact healthy.

Some of the most popular performance assessment metrics are:

- Accuracy(ACC): $\frac{TP+TN}{TP+TN+FP+FN}$ Accuracy gives a measurement of how often a model is correct in its predictions. It is oftentimes the first parameter to look at when comparing the performance of different models.
- Recall, True positive rate(TPR) or Sensitivity: $\frac{TP}{TP+FN}$ Recall is the ratio of correctly predicted positive observations to all observations in actual class. It represents the model's ability to find all the positive samples. High recall means that an algorithm returned most of the relevant results. Although ACC may describe the performance of a model in a broad way, it does not take into account the missclasification cost, which may be relevant in an situations where each mistake can potentially mean great economic losses for the wind farm responsibles.
- Specificity or True negative rate(TNR): $\frac{TN}{TN+FP}$ Similarly to TPR it allows to take into account the misclassification costs.
- Precision: $\frac{TP}{TP+FP}$ This metric is the ratio of correctly predicted positive observations to the total predicted positives. In other words, it represents the model's ability not to label a negative sample as positive. High precision means that an algorithm returned substantially more relevant results than irrelevant ones.
- F1: $2 * \frac{Precision*Recall}{Precision+Recall}$ F1 is the harmonic mean of precision and recall, and it tries to find the balance between both parameters. This allows you to easily optimize for one value or the other.

The larger ACC, TPR and TNR are, the better the performance of the model, and hence its predictions.

2.6.6. Decision-making and maintenance scheduling:

The last step in data-driven condition monitoring processes is to formulate insights from the previous steps to make an informed decision on the maintenance and potentially repair operations needed. Based on the condition assessment of the WT, maintenance personnel can prioritize certain activities over others. This is specially important in situations in which access to the WT is exceptionally challenging, such as in offshore operations. In these cases being well prepared pays off, as each visit to the wind farm place, involves great movement of resources and efforts, which inherently increases WT O&M costs.

2.6.7. Review of SCADA useful parameters for CM

In this section a table where SCADA parameters can be comprehensively overviewed depending on their utility for fault detection, diagnosis or Remaining useful life estimation. These parameters have been chosen according to available literature on the topic. To further increase the understanding of SCADA data analysis in the context of wind turbine condition monitoring, this literature will be reviewed, highlighting relevant aspects. In order to ease the visualization of the functionality of different SCADA parameters for fault detection, diagnosis and prognosis; table 2.1 will be set up. Additionally each parameter will be color coded, according to the component or subsystem it is meant to asses. The color coding works in the following way:

Generator, gearbox

In [53] a model based approach is chosen to create a fault diagnosis method from 126 operating SCADA parameters and state information recorded every 10 min. In said paper; Active power, Wind speed, Rotated speed of gearbox, Yaw angle, Bearing temperature of gear box, Oil temperature of gear box, Cabin Temperature and External temperature. Trough Nonlinear Multivariate State Estimate Technique(NMSET) a normal behaviour model of the temperature within the gearbox is constructed. By setting thresholds and judging the variation of the temperature with respect to this thresholds, faults in gearboxes can be identified. The approach is described as simple and very appropriate for complex and random processes.

In Jin et al.[54] an ensemble approach is conducted in order to identify and diagnose faults in a Wind Turbine doubly-fed induction generators(DFIG). A model of normal behaviour of the WT is built trough a Mahalanobis reference space[55] in order to observe deviations. The SCADA parameters used in order to build the model are: Generator output power, Average wind speed(over 30second time intervals), generator rotor speed, generator winding temperature, generator drive-end bearing temperature, generator nondrive-end bearing temperature, temperature inside the nacelle. An interesting approach followed in this study is the dynamic generation of the Mahalanobis space every 3 months. This allows for a more updated and flexible assessment of the reference operation state of the WT, and for having a model in line with the stage of life to which

the WT belongs. The authors concluded that the model proved to be effective in diagnosing a specific failure before the need to repair in two different generators. Moreover, they argued that no information on fault states was needed, and therefore it made their approach specially useful for practical applications in health monitoring of WTs.

In Udo et al.[56] a scan over recent literature on SCADA valuable inputs for monitoring of critical components such as gearbox or generator was carried out. In this analysis Nacelle Temperature, Rotor Speed, Active power, Outdoor temperature and Gearbox oil sump temperature were identified as useful sensor data for gearbox monitoring. Nacelle temperature, Active power, Generator Speed and Generator stator temperature were the valuable parameters for Generator monitoring. Following the discovered information, the researchers in the study tried to build a healthy state of various wind turbines. They based their methodology in four steps: Data cleaning, features selection, model processing(involving training and testing) and post-processing. The underling principles behind these processes has been already commented on at earlier section of this work. Nevertheless, it is interesting to highly some of the key aspect of each of these steps. In Data cleaning, expert knowledge based criteria and thresholds were established, such as the removal of instances where the power generated was zero but the wind speed wasn't, or instances in which one or more values lied outside the normal range. For features selection, the team relied on literature review to understand and select the best variable combinations required to monitor system behaviour of components such as generator and gearbox. The selected variables were extracted form the pool described earlier, tailored for both generator and gearbox. The chosen ones were generator bearing temperature, for WTs generator health monitoring and gearbox bearin temeprature for gearbox health status monitoring. Regression models were employed, specifically multiple linear regression(MLR); as a base model, and two non-linear regression models: extreme gradient boosting(XGBoost) and long short-term memory(LSTM). Again, these algorithms were carefully selected in accordance to the relevant scientific literature. The dataset was divided into training and testing with a 70:30 ratio. Additionally K-fold cross validation was used in order to ensure that each modeled produced was robust and accurate. K-K-Fold cross validation is a statistical model used for estimating the performance of ML models when making predictions. It is important to understand how it works:

- Splitting the data into K parts.
- Once data is split, one of this parts will be used to test the model and the remaining parts for training the model.
- After achieving this task, the process is repeated but this time using a different part for testing and the rest for training. This process is repeated until all parts have been used for testing.
- At the end a set of results is produced from each test. By averaging these results a more robust measure of the model's performance is obtained.

It is easy to assume, that this extended testing and training procedure is useful when working with small datasets. After testing each model the model's accuracy is assessed by comparing deviations between measured and predicted values. It s observed that LSTM algorithm performed better when identifying fault states in the generator whereas XGBoost better fitted the gearbox's behaviour. It is worth mentioning that the models were tested without failure logs, which makes this work specially interesting for novel wind farms without much data available. Another important conclusion derived by the authors is the identification of non-linear models as best suited for wind turbine fault detection and diagnosis.

A RUL estimation procedure is built by Fan et al.[57], in which a 1.5 MW WT gearbox fault is estimated to happen in thirty days.

Another study related to RUL estimation and fault diagnosis for 1,5 MW WT generators is the one performed by Zhao et al.[58]. This study showed promising results, as it was able to predict wind turbine generator RUL with about 80% accuracy 18 days ahead and diagnose generator faults with 94% accuracy when they occur, just from SCADA data and status information. The model was tested in two different wind farms in China. The algorithm chosen for the prediction models was a SVM classifier, supported by SMOTE[59] to work around unbalanced datasets, characteristic of SCADA data and status logs, in which healthy or normal behaviour samples greatly overpopulate the data pool. The reasoning behind selecting this algorithm is again an extensive literature review performed by the authors. These selected SVM, KNN ANN and naive Bayesian algorithms as the most relevant ones according to the up to date literature. They tested models using each of these algorithms and observed that SVM performed the best. As for the selected SCADA features, they identified the following values as the most prevalent in research papers on the topic: wind speed, ambient temperature, rotor speed, generator rotational speed, generator active power, generator reactive power, temperature of main bearing, temperature of low-speed shaft and temperature of high speed shaft. Nevertheless, the authors went a step further and pointed out that some of these relevant parameters were closely related. Their solution consisted in combining similar features into a single feature when these presented a Pearson correlation coefficient[60] larger than 0.98. In this way they achieved an optimal trade off between redundancy of datapoints and maintenance of information of original features.

Table 2.1. SCADA PARAMETERS FOR DIFFERENT CM PURPOSES OF WTS, COLOR CODED BY COMPONENT OR SUBSYSTEM

Purpose	**Useful SCADA parameters**
Fault Detection	Active power, Wind speed, Rotated speed of gearbox, Yaw angle, Bearing temperature of gear box, Oil temperature of gear box, Cabin Temperature, External temperature, Nacelle Temperature, Rotor Speed, Active power, Outdoor temperature and Gearbox oil sump temperature Generator output power, Average wind speed, generator rotor speed, generator winding temperature, generator drive-end bearing temperature, generator nondrive-end bearing temperature, temperature inside the nacelle, Nacelle temperature, Active power, Generator Speed and Generator stator temperature
Fault Diagnosis	Active power, Wind speed, Rotated speed of gearbox, Yaw angle, Bearing temperature of gear box, Oil temperature of gear box, Cabin Temperature, External temperature Generator output power, Average wind speed, generator rotor speed, generator winding temperature, generator drive-end bearing temperature, generator nondrive-end bearing temperature, temperature inside the nacelle, Nacelle temperature, Active power, Generator Speed and Generator stator temperature
RUL	wind speed, ambient temperature, rotor speed, generator rotational speed, generator active power, generator reactive power, temperature of main bearing, temperature of low-speed shaft, temperature of high speed shaft

3. DEVELOPED WORK

As a natural continuation of the work and knowledge that has been discussed until now, a data-driven approach will be followed to try to predict the presence of a specific type of fault within a WT. Some of the machine learning models will be utilized in order to test their effectiveness in fault detection applications within the context of wind turbine generation. In order to make this work more meaningful, some parameters have been strategically chosen, to actually provide some value to the field and demonstrate certain features of the approaches discussed up to now. These will be explained in the following section: Description and Case Studies.

3.1. Description

During the entire work some points from the currently available technologies have been highlighted. Now the importance given to this points will make more sense. First of all, SCADA data will be used, due to its efficacy, demonstrated in the literature, and its usefulness and potential when combined with machine learning techniques. On top of this the developed case studies will be focused on detecting faults in electrical components of the Wind turbine as other approaches such as vibrations or acoustic emissions monitoring have already proven to be more effective when detecting early faults in mechanical components. Additionally, developing work with tools and data that is arguably available for the entire wind power generation industry makes the most sense, in an intent to develop the state of the art of the technology and further the common understanding of the topic. Las but not least, Machine learning approaches, which have seen incredible advancements in the last decade, will inevitably inflict a great deal of impact on the sector. Watching this trend, using machine learning models for this type of problem, just adds more significance to this work.

Four models were created based on the case studies provided by [56] and [61]. The purpose of all this models was to predict generator bearing temperature of a Wind Turbine with high enough accuracy to be able to detect potential incoming faults or in other words, temperature peaks, that indicate abnormal operation of the component.Note that his can not only indicate faults in that bearing but also other adjacent and closely correlated abnormalities that could have generated this unhealthy behaviour of the bearing. The wind turbine SCADA data was extracted from a dataset provided by [62] in which 10-minute SCADA and events data from 6 Senvion MM92's onshore wind turbines at Kelmarsh wind farm in the UK, grouped by year from 2016 to mid-2021, were included. Not all signals are available for the entire period but the data provided in this time periods was rich enough to serve as the foundation for the models built. The SCADA data, which was the main source of input in this study, contained over 280 SCADA parameters

were included in each csv file, ranging from actual 10 minute averaged sensor values, to statistical values of interest such as minimums or maximums.

Fig. 3.1. Senvion MM92 Wind Turbine[63]

Table 3.1. SENVION MM92 SPECIFICATIONS

Rated power	2,050.0 kW
Cut-in wind speed	3.0 m/s
Rated wind speed:	12.5 m/s
Cut-out wind speed:	24.0 m/s
Rotor Diameter	92.5 m
Rotor max speed	15.0 U/min
Number of blades	3
Rotor material	GFK
Gearbox type	spur/planetary
Gearbox ratio	1:120
Generator	Double Fed Asynchronous
Max Speed	1,800.0 U/min
Grid Frequency	50.0 Hz
Voltage	690.0 V
Onshore	Yes
Offshore	No

In order to look for solutions to to problems and bugs found during the development,

several resources and internet forums such as stackoverflow, GitHub or reddit were visited. . First a Multiple linear regression model was built due to its simplicity and flexibility when working with data. Lack of experience in this area of study, which is data science, also inclined the balance in favour of a simple but potent method such as MLR. The basic architecture of the model was build based on information found on the papers mentioned earlier in this paragraph. On top of that, two other models proposed in the aforementioned studies were also tested. These were chosen as the reference models in terms of performance becasue they were the top performance in the other studies. The results found in this work, don't quite match the conclusions found in udo and santolamazza, but this will be commented on further in the results section. Finally, as an additioonal contribution to the models already tested in other studies, and to provide further novelty to this work, a Deep Neural Netwrok was also built.

3.2. Case Studies

In this section different models will be 'resented for predicting generaots bearing temeprature levels and potentially setting thresholds for alarm triggering.

3.2.1. MLR model

The code was build intending to create a learning pipeline that predicts the temperature of generator bearings in a wind turbine based on several features. The features were selected based on relevant literature review, mainly from the aforementioned reference papers that had already used a similar approach. The model was build as follows.

1. **Importing libraries and packages**: Various libraries are imported such as pandas, numpy, matplotlib, sklearn and scipy to provide necessary functions for data manipulation, visualization, and machine learning.

2. **Defining wind turbines to be analyzed and file paths**: definiiton of which turbine to analyze. Each turbine has its own data stored in separate CSV files for different years.

3. **Defining normal ranges for the features**: This is done for data cleaning purposes. Any data point that falls outside of these specified ranges will be considered an outlier and removed. This part was done based on datasheet of the wind turbine and knowledge based input from literature.

4. **Reading and concatenating all CSV files**: The code reads all the CSV files corresponding to different years for a specific turbine and concatenates them into a single dataframe. This is done because data is splitted and it is desirable to use all the available data to train the model.

5. **Data Cleaning**: At this step, missing values are dropped and data points are filtered based on the specified normal ranges. The Power is filtered to be above 0 and Wind speed is filtered to be above 3 m/s, assuming a cut-in wind speed of 3 m/s (i.e., the minimum speed at which the turbine will generate power).

6. **Data Visualization**: The script then plots the wind turbine power curve, before and after data cleaning, showing power vs wind speed. This gives a visual representation of the turbine's performance.

7. **Preprocessing for Machine Learning**: The script then divides the data into features (X) and target (y) for machine learning. The target is the 'Generator bearing rear temperature' that we want to predict, and the features are the other columns that have been used as inputs.

8. **Train-Test Split**: The data is then split into a training set (70% of the data) and a test set (30% of the data). The training set is used to train the machine learning model, and the test set is used to evaluate its performance.

9. **Outlier detection and removal**: Outliers are detected and removed from the training data based on the Mahalanobis distance. The method removes top 10% of outliers most deviated from the mean of features included in the equation. In this case power and wind were the only features included.

10. **Second data Visualization**: After outliers removal, the wind turbine power curve is plotted again. This is to visualize the effect of the cleaning and outliers removal. This is plotted on top of the previous curve to clearly see the effect of the cleaning procedure.

11. **Correlations plot** : This part plots the relationship between each of the inputs and the output temperature. It provides both a scatterplot and the Pearson, Spearman, and Kendall correlation coefficients to understand how each input feature is related to the output.

12. **Data Scaling**: The features are then standardized (mean = 0, standard deviation = 1) using StandardScaler. This is done because the features are measured in different units, and standardizing them allows the model to more easily find patterns.

13. **Model Training**: A linear regression model is trained on the scaled training data. The model tries to find a linear relationship between the features and the target.

14. **Cross-Validation**: Cross-validation is performed on the training data to get a better estimate of the model's performance. It splits the training data into several parts, trains the model on some parts and tests it on the remaining parts. This process is repeated several times with different parts used for training and testing.

15. **Model testing** : The trained model is used to predict the output for both the training and test sets. The performance of the model is evaluated using various metrics including R2 score, Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE).

16. **Visualization of model predictions**: The actual vs predicted generator temperatures for the test set are plotted over time.

17. **Control Chart (Deviations Plot)**: The deviations of the predicted values from the actual values are calculated and plotted over time. The Upper Control Limit (UCL) and Lower Control Limit (LCL), which are defined as three standard deviations away from the mean, are also plotted.

18. **Model testing on new data**: A function is defined to pre-process new data (in this case, from a different turbine) in the same way as the original data. The preprocessed data is then used to test the trained model.

This process can be easy visualized in the flow diagram in Fig..3.3.

In summary, this code builds a multiple linear regression model to predict the generator bearing temperature of a wind turbine based on several other parameters. The model's performance is evaluated, and the predictions are visualized on a control chart to help detect potential faults.

The model predicts the generator bearing rear temperature (in ºC) based on several parameters including nacelle temperature, power, rotor speed, stator temperature, and wind speed. If a fault or abnormal condition occurs in the bearing, it often results in an increase in friction and hence an increase in temperature. Therefore, abnormal temperatures can be an indication of a possible fault in the bearing.

After predicting the temperatures, the model calculates the deviations of the actual temperatures from the predicted ones. These deviations are then used to create a control chart. A control chart is a useful tool for monitoring the quality of a process, and it can be used here to monitor the health of the wind turbine generator.

The control chart includes a central line representing the average of the deviations, and two control limits: the Upper Control Limit (UCL) and the Lower Control Limit (LCL). These are typically set at the mean ± 3 standard deviations of the deviations. If the deviations of the actual temperatures from the predicted ones stay within these limits, it means the generator bearing is operating under normal conditions. But if the deviations exceed the UCL or fall below the LCL, it means there is an abnormal condition, which might be due to a fault in the bearing.

In summary, by continuously monitoring the deviations and whether they stay within the control limits,it is possible to identify potential bearing faults. When an out-of-control point (a point outside of the UCL or LCL) is detected, an alarm could be triggered for further investigation or maintenance. In fact, in the code included in the appendix a simple

mechanism for printing dates of potential faults is implemented. In this mechanism the top 1% of most deviated predictions are displayed to check if a fault in fact occurred or not. This method is not accurate nor really studied and is meant to just serve as base for a more complex method. It's also worth mentioning that this method doesn't pinpoint what kind of fault has occurred, only that an abnormal condition likely exists. Other diagnostic measures would be needed to determine the exact type and location of the fault.

3.2.2. XGBoost model

1. **Outlier Filtering**: In this case and all the following built ML algorithms, Mahalobis distance was calculated based on all input features. This seemed to have somewhat of a positive effect on the testing results, specially in the XGBoost algorithm. The range of outliers defined was still kept at 10%.

2. **XGBoost Model Definition:** In the new implementation, an XGBoost Regressor model is used for prediction instead of a Multiple Linear Regression model. XGBoost is a gradient boosting algorithm that can model complex non-linear relationships.

3. **Hyperparameter Grid Definition:** A grid of hyperparameters for the XGBoost model is defined. This grid includes parameters such as the learning rate ('eta'), maximum depth of the trees ('max_depth'), number of estimators ('n_estimators'), minimum sum of weights needed in a child ('min_child_
weight'), subsample ratio of the training instances ('subsample'), and ratio of column sampling ('colsample_bytree'). The performance of the XGBoost model can significantly depend on the tuning of these hyperparameters.

4. **Grid Search Cross Validation:** The GridSearchCV method is used for finding the best hyperparameters for the XGBoost model. This method fits the model with every combination of hyperparameters defined in the grid and chooses the combination that provides the best performance, as measured by a specified scoring metric.

5. **Model Training with Optimal Parameters:** Once the optimal hyperparameters are obtained via GridSearchCV, the XGBoost model is re-defined using these parameters and then trained using the training data.

   An important remark here is that due to hardware constrains were calculated on just one year of wind turbine operation, and just one turbine. Specifically it was 2016 of WT6. This hyperparameters are optimized taking into account data size and features as well, so perhaps the results of this model could have been improved by running the optimization method in every calculation.

   The obtained hyperparameters are defined in Fig.3.2

```
'learning_rate': [0.01],
'max_depth': [7],
'n_estimators': [1000],
'min_child_weight': [1],
'subsample': [0.5],
'colsample_bytree': [1.0],
```

Fig. 3.2. Hyperparameters found from optimization process

6. **Performance Metrics Calculation:** Performance of the model is assessed using R-squared, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE), for both the training and testing sets.
7. **Control Chart for Deviations:** Again, a control chart for deviations between the actual and predicted values is plotted together with the Upper Control Limit (UCL) and Lower Control Limit (LCL).

The flow diagram in this case is outline in Fig.3.4.

3.2.3. LSTM model

Similarly as in the XGBoost case, just the unique steps of this model will be outlined, as the majority of the process remains the same.

1. **No Manahalobis filtering**: For this algorithm no improvement was noticed when applying this type of data filtering, neither to power curve components nor to all inputs. Therefore it was skipped.
2. **Reshape data for LSTM:** after standardizing the data reshape the data to a 3D array to fit the input requirement of the LSTM model.
3. **Define LSTM model:** Define the LSTM model using Keras. The model consists of multiple LSTM layers and a Dense layer at the end. The hyperaparemeters used for the the LSTM model were chose based on advice from the literature review performed in the reference papers:

Hyperparameter	Description
Number of LSTM Layers	There are three LSTM layers in the model.
Number of Neurons in Each Layer	The first and second LSTM layers have 50 neurons each, the third LSTM layer has 25 neurons.
Activation Function	The activation function used in the LSTM layers is the Rectified Linear Unit (ReLU) function.
Return Sequences	For the first two LSTM layers, return sequences is set to true. For the last LSTM layer, the return sequences is set to false.
Optimizer	The optimizer used for training the network is Adam.
Loss Function	The loss function used is Mean Squared Error.
Batch Size	The batch size in the fit function is 5. This is the number of samples per gradient update.
Epochs	The number of epochs is set to 5.
Validation Split	During training, 20% of the data is used as the validation data.
Early Stopping Monitor	Early stopping is implemented with monitoring on the validation loss.
Early Stopping Mode	The mode is set to 'min', which means training will stop when the quantity monitored has stopped decreasing.
Early Stopping Patience	The patience is set to 10, which is the number of epochs with no improvement after which training is stopped.

Table 3.2. HYPERPARAMETERS OF THE LSTM MODEL

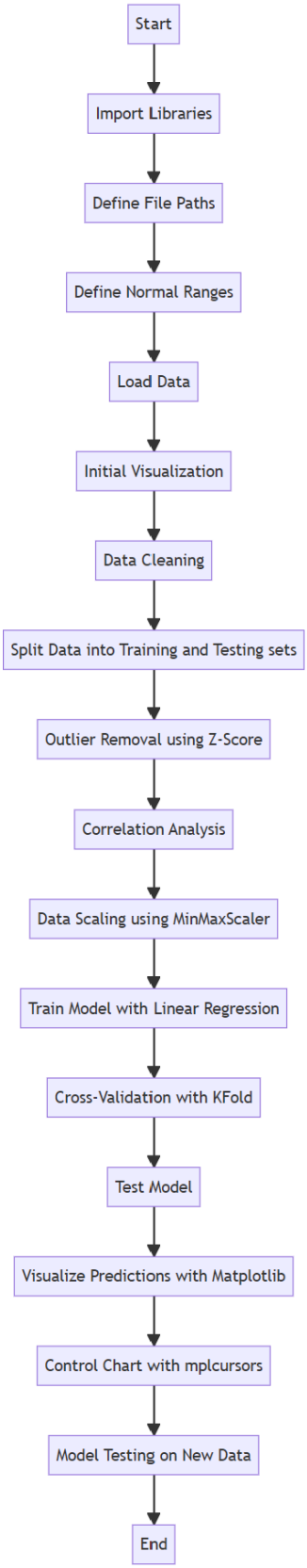
3.2.4. ANN model

Sure, here is the list with titles for each step:

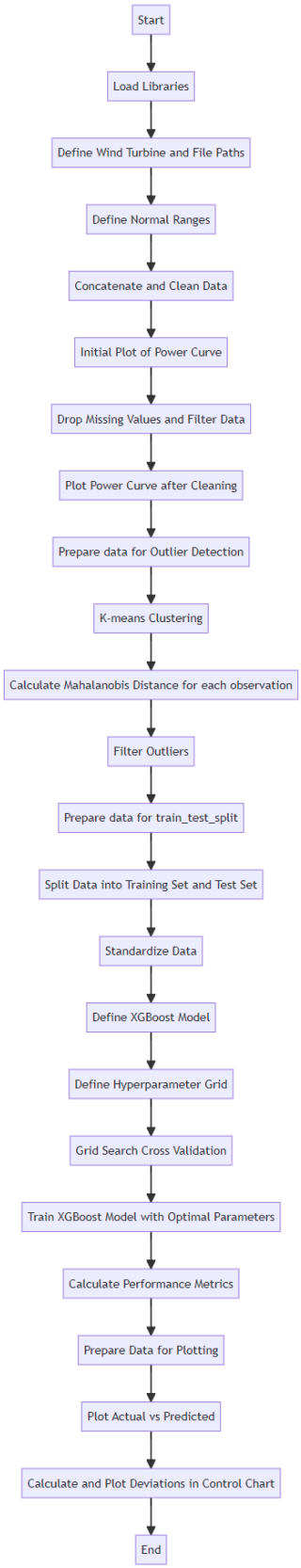
1. **Import Libraries and Modules:** In this case Manahalobis distance is neither used as it didn't serve to improve the performance metrics of the model. At least not with the model hyperparameters used.

2. **Define Neural Network Model:** Define the structure of the deep neural network model using the Keras Sequential API. The model consists of multiple dense (fully connected) layers with dropout for regularization. The number of neurons and activation functions are chosen based on relevant literature and hardware constraints.

3. **Compile and Train Model:** Compile the model with mean squared error as the loss function and Adam as the optimizer, then train the model on the training data for a certain number of epochs.

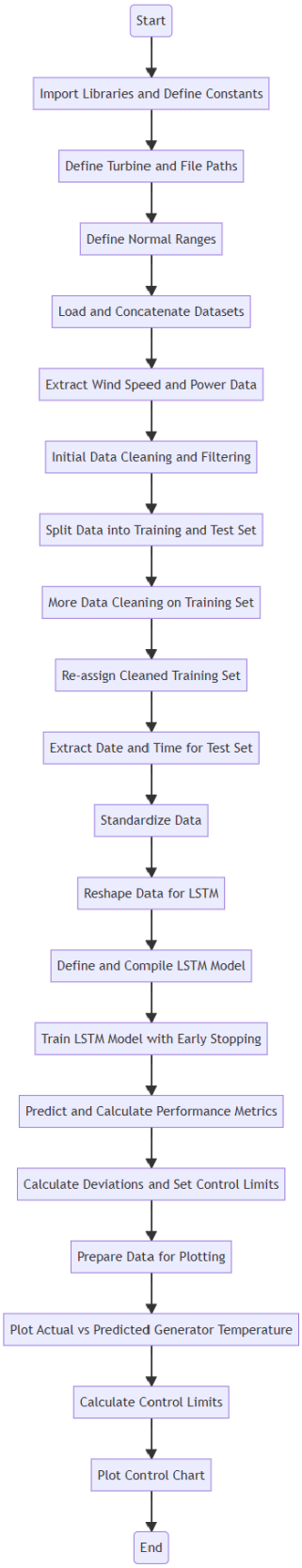


(a) MLR model

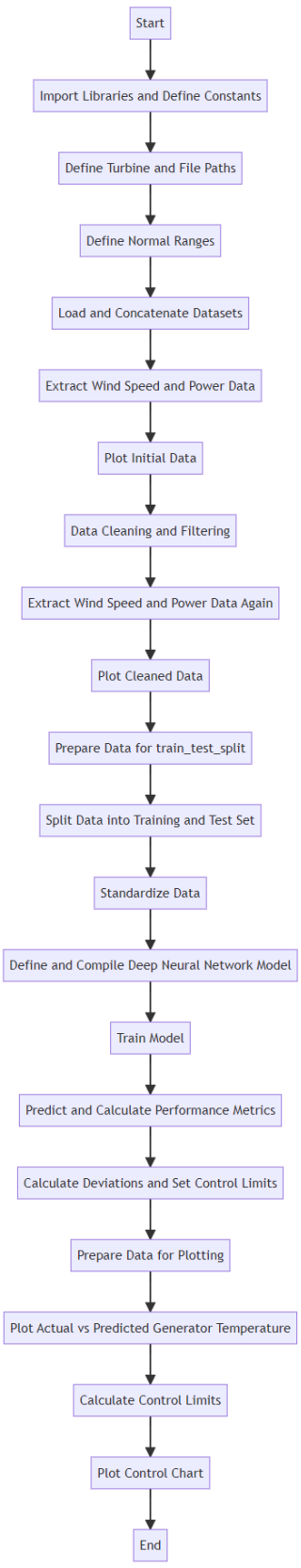


(b) XGBoost model

Fig. 3.3. Flow diagrams for case-studies of machine learning models_1



(a) LSTM model



(b) ANN model

Fig. 3.4. Flow diagrams for case-studies of machine learning models_2

4. RESULTS

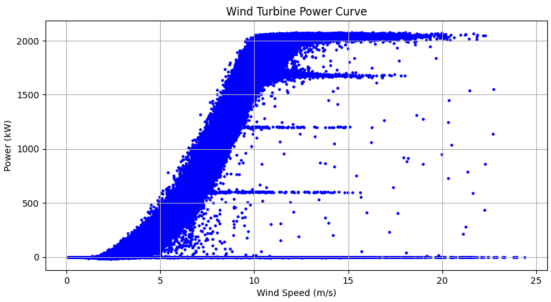
4.0.1. Power curve

The data cleaning section in all case studies played a major role in the predictive ability of the ML models. Two separate procedures were applied to the the models. First a knowledge-base normal range definition was done for the input values of the SCADA parameters. Every data-point outside of this ranges was removed. The cleaned power curve after this preliminary cleaning is presented in Fig.4.2.b. The ranges were defined on the basis of the data-sheet of the wind turbine and literature review about normal range definition in machine learning models for this purposes. Additionally a cut in wind speed of 3m/s was defined and instances were power was 0 or missing values were preset, were eliminated form the dataset.

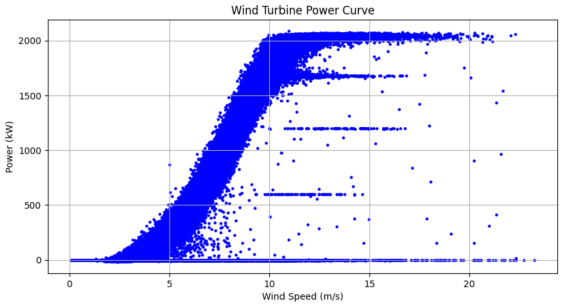
```
# Define normal ranges for SCADA parameters based on knowledge
normal_ranges = {
    'Nacelle temperature (Â°C)': (-10, 50),
    'Power (kW)': (0, 2000),
    'Rotor speed (RPM)': (0, 73),
    'Stator temperature 1 (Â°C)': (0, 150),
    'Wind speed (m/s)': (0, 24),
    'Generator bearing rear temperature (Â°C)': (0, 100),
}
```

Fig. 4.1. Normal ranges defined for ML models

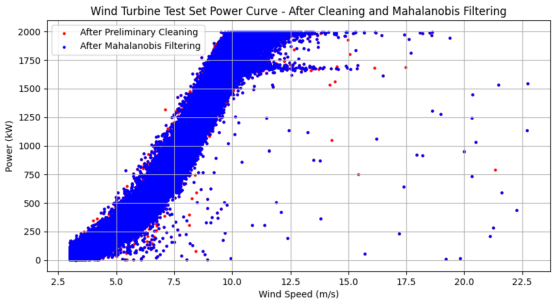
Once this cleaning was completed, some of the Models were further filtered based on the Manahalobis distance method described in the reference paper. In this Method a set of multidimensional data is assumed to follow a Gaussian distributions and outliers deviated a certain distance from the normal are eliminated. The threshold for elimination was set at 10% for the MLR and the XGBoost models. This percentage was selected based on trial and error, trying to look for better performance parameters. The LSTM and ANN models didn't see an improvement with this filtering. Additionally, this filtering was just performed on the training data, applying it on testing data would not be of interest, as some of this removed outliers could be potential faults. A visual representation of the effect of this filtering can be appreciated on Fig.4.2.c & d.



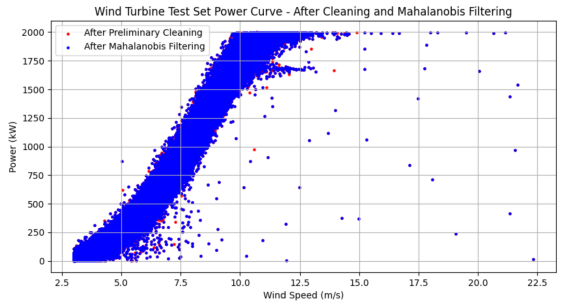
(a) Power curve of WT 1



(b) Power curve of WT 6



(c) Power curve of WT 1 after preliminary cleaning and filtering



(d) Power curve of WT 6 after preliminary cleaning and filtering

Fig. 4.2. Data cleaning, power curve and correlation analysis

Table 4.1. CLEANING PERFORMANCE MLR

Wind Turbine	**Before cleaning**	**After cleaning**	**Cleaning Reduction**	**Training data after cleaning**	**Training data after filtering**	**Filtering Reduction**
WT1	288864	216250	25.14 %	151375	136237	10.0 %
WT2	288864	219164	24.13 %	153414	138072	10.0 %
WT3	288864	219160	24.13 %	153412	138070	10.0 %
WT4	288864	220208	23.77 %	154145	138730	10.0 %
WT5	288864	216250	25.14 %	151375.0	136237	10.0 %
WT6	288864	214022	25.91 %	149815	134833	10.0 %

4.0.2. Correlation parameters

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related or correlated. When we speak about correlation, usually we refer to Pearson's correlation. However, there are other measures of correlation, including Spearman's rank correlation and Kendall's Tau, that can reveal other insights on the data. Here is a short overview of the parameters:

Pearson correlation is a measure of the linear relationship between two continuous

random variables. It can take a range of values from -1 to 1, where -1 indicates a perfect negative linear relationship, 0 indicates no linear relationship, and 1 indicates a perfect positive linear relationship. Pearson correlation assumes that the data is normally distributed and the relationship between the variables is linear. It also assumes that the variances of the individual variables are similar (homoscedasticity)[64].

Unlike the Pearson correlation, Spearman's correlation does not assume that both datasets are normally distributed. Instead, it measures the strength and direction of the monotonic relationship between two ranked variables. This means it looks at how changes in one variable are associated with changes in another, but it doesn't necessarily have to be a linear relationship. The correlation ranges from -1 to 1, with -1 indicating a perfect negative monotonic relationship, 0 indicating no monotonic relationship, and 1 indicating a perfect positive monotonic relationship.[65]

Like Spearman's rank correlation, Kendall's Tau is a non-parametric measure of relationships between columns of ranked data. The Tau correlation coefficient returns a value of 0 to 1, where 0 indicates that the data are uncorrelated and 1 indicates that the data are perfectly monotonically correlated. Kendall's Tau has been considered as a better measure of correlation in many cases as it's good at dealing with small datasets and it's better at handling ties (i.e., when the ranks for both variables are the same) compared to Spearman's[66].

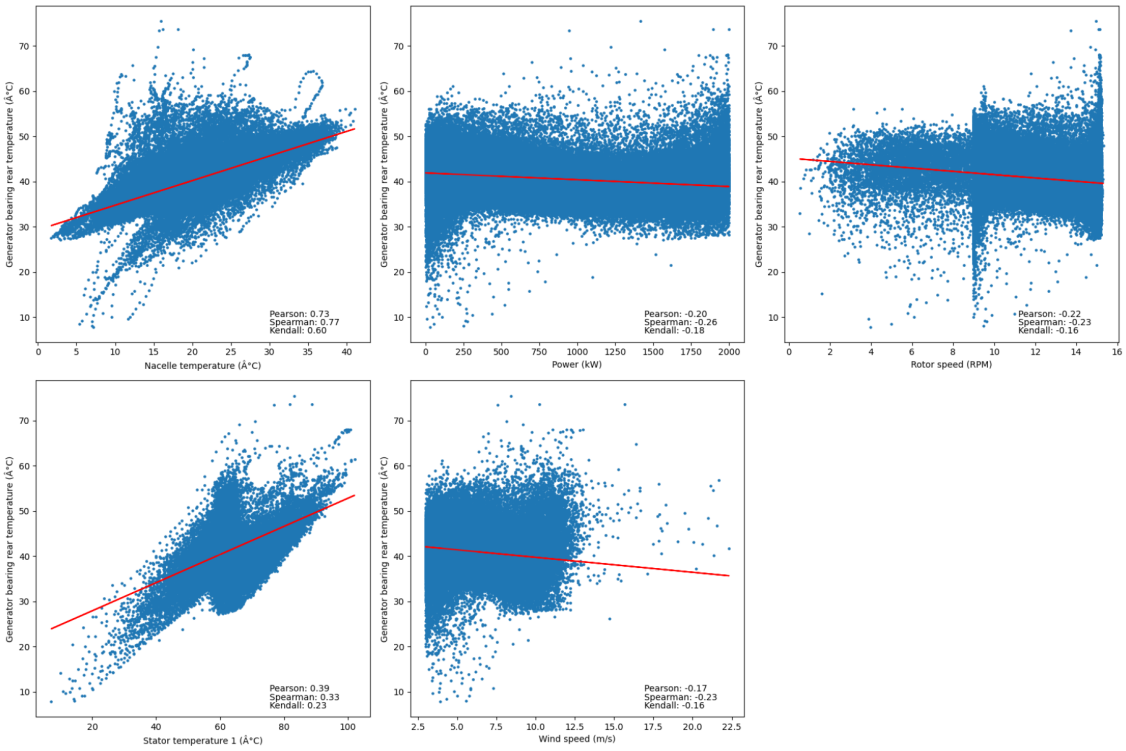


Fig. 4.3. Correlation input-output WT6 with correlation parameters

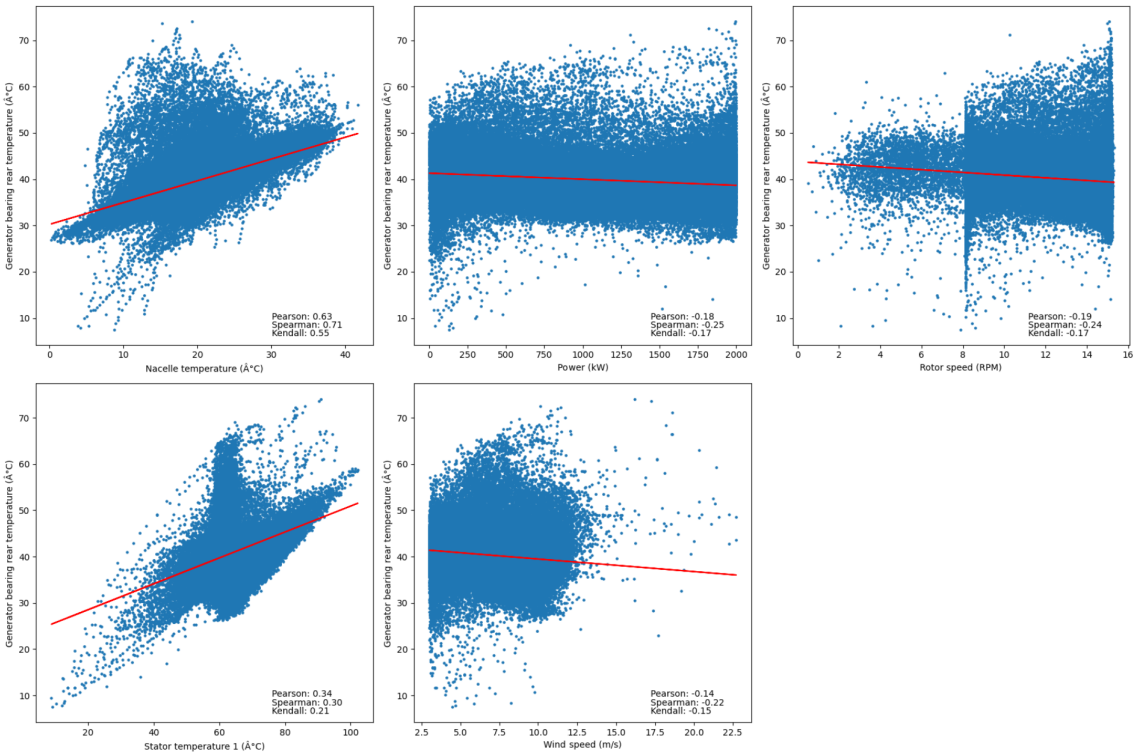


Fig. 4.4. Correlation input-output WT1 with correlation parameters

For each wind Turbine slightly different correlation parameters have been obtained, although they are similar enough to be able to represent the overall correlation input-output with just one wind turbine case. A "healthy" wind turbine, WT6(on the basis of general good model performance)has been chosen to represent said parameters. Next, each of the correlations will be commented.

Nacelle temperature vs. Generator bearing rear temperature: Pearson: 0.73: This indicates a strong positive linear relationship between the two variables. Spearman: 0.77: This indicates a strong positive monotonic relationship, suggesting that as one value increases, so does the other, but not necessarily at a constant rate. Kendall: 0.60: This also indicates a positive monotonic relationship, similar to the Spearman's correlation, but Kendall's Tau is considered more robust to small samples sizes and non-parametric assumptions.

Power vs. Generator bearing rear temperature: Pearson: -0.21, Spearman: -0.28, Kendall: -0.19: All three coefficients are negative, suggesting a weak inverse relationship between Power and Generator bearing rear temperature. However, the strength of the correlation is not strong.

Rotor speed vs. Generator bearing rear temperature: Pearson: -0.24, Spearman: -0.26, Kendall: -0.18: Similar to the Power correlation, these also suggest a weak inverse relationship. As the Rotor speed increases, the Generator bearing rear temperature tends to decrease and vice versa.

Stator temperature 1 vs. Generator bearing rear temperature: Pearson: 0.36,

Spearman: 0.31, Kendall: 0.22: These show a weak to moderate positive relationship. As the Stator temperature increases, the Generator bearing rear temperature also tends to increase.

Wind speed vs. Generator bearing rear temperature: Pearson: -0.18, Spearman: -0.25, Kendall: -0.17: These values suggest a weak negative correlation. As the Wind speed increases, the Generator bearing rear temperature tends to decrease slightly.

4.0.3. Results

Table 4.2. WIND TURBINE 1

ML model	R-squared	RMSE	MAE	MAPE
MLR	0.5571	2.6685	1.5502	3.7325
XGBoost	0.5888	2.5713	1.4228	3.4101
LSTM	0.54908	2.6928	1.6799	4.1342
DNN	0.5452	2.7043	1.4874	3.5565

Table 4.3. WIND TURBINE 2

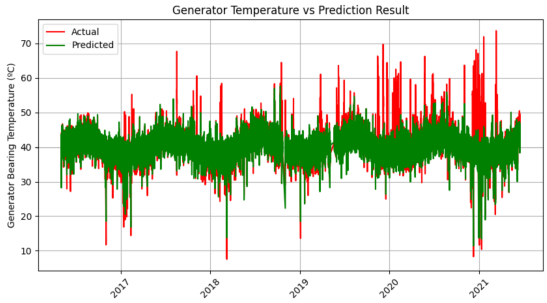
ML model	R-squared	RMSE	MAE	MAPE
MLR	0.7944	1.5831	1.0691	2.7027
XGBoost	0.8150	1.5017	0.9648	2.4269
LSTM	0.8075	1.5318	0.9822	2.4710
DNN	0.7518	1.7395	1.1545	2.8868

Table 4.4. WIND TURBINE 3

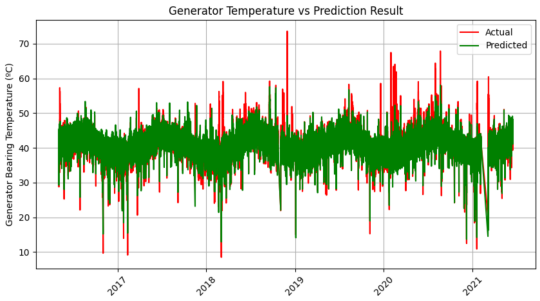
ML model	R-squared	RMSE	MAE	MAPE
MLR	0.7912	1.6588	1.1869	2.9787
XGBoost	0.8229	1.5277	1.02487	2.5572
LSTM	0.7194	1.9230	1.5178	3.8500
DNN	0.7508	1.8122	1.2282	3.0203

Table 4.5. WIND TURBINE 4

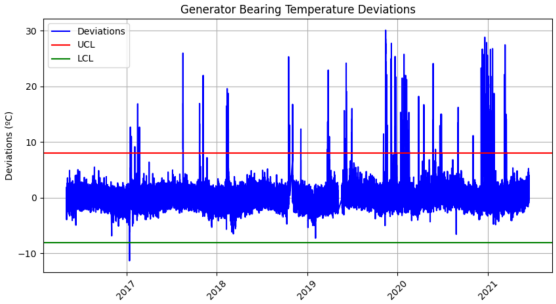
ML model	R-squared	RMSE	MAE	MAPE
MLR	0.6208	2.4661	1.3691	3.2657
XGBoost	0.6375	2.4111	1.3001	3.0873
LSTM	0.5870	2.5736	1.5438	3.7347
DNN	0.5224	2.7677	1.6099	3.7769



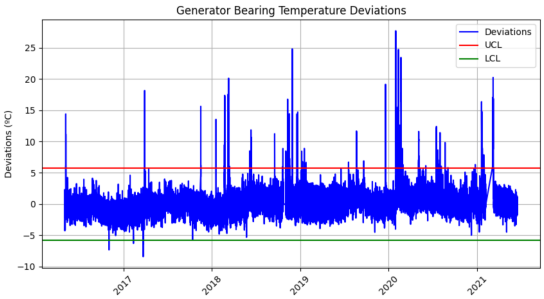
(a) Actual vs Predicted values WT 6



(b) Actual vs Predicted values WT 6



(c) Deviations plot WT 1



(d) Deviations plot WT 6

MLR Algorithm,

Fig. 4.5. Actual vs Predicted Temperatures, together with deviations for WT1 and WT6

Table 4.6. WIND TURBINE 5

ML model	R-squared	RMSE	MAE	MAPE
MLR	0.7689	1.6691	1.1830	3.0150
XGBoost	0.7958	1.5688	1.0838	2.7528
LSTM	0.7870	1.6024	1.1055	2.8038
DNN	0.7263	1.8166	1.2913	3.2424

Table 4.7. WIND TURBINE 6

ML model	R-squared	RMSE	MAE	MAPE
MLR	0.7425	1.9141	1.3201	3.2180
XGBoost	0.7649	1.8291	1.2385	3.007
LSTM	0.7451	1.9045	1.2587	3.0387
DNN	0.7080	2.0385	1.3769	3.3806

4.0.4. MLR Results

[H]

From the performance of the Multiple Linear Regression (MLR) model on each metric:

1. **R-squared**: This is the proportion of the variance in the dependent variable that is predictable from the independent variables. The R-squared value for the MLR model is relatively high for all the wind turbines, but it is consistently lower than the R-squared value obtained from the XGBoost model. This suggests that while MLR is capable of capturing a good amount of variance in the data, it is not as effective as the XGBoost model.

2. **RMSE**: This is a measure of the differences between the values predicted by a model and the values observed. The Root Mean Squared Error for the MLR model is generally higher than that of the XGBoost model. A higher RMSE value means the model's predictions are less accurate. Therefore, the MLR model is less accurate than the XGBoost model in predicting wind turbine outputs.

3. **MAE**: This is the average magnitude of the errors in a set of predictions, without considering their direction. The Mean Absolute Error for the MLR model is also higher than the XGBoost model across all wind turbines. This suggests that on average, the absolute prediction errors of the MLR model are larger, pointing to less accuracy in its predictions.

4. **MAPE**: This calculates the average of the absolute percentage differences between the predicted and actual values. The Mean Absolute Percentage Error is again higher for the MLR model than the XGBoost model. This means that, in percentage terms, the MLR model's predictions are less accurate.

Overall, while the MLR model demonstrates a decent level of performance in predicting wind turbine output, it is consistently outperformed by the XGBoost model across all metrics and wind turbines. It is still a viable model, but there may be complex relationships and patterns in the data that MLR, being a linear model, cannot capture as effectively as a more complex model like XGBoost.

4.0.5. XGBoost Results

Focusing specifically on the results for the XGBoost model, let's analyze each metric:

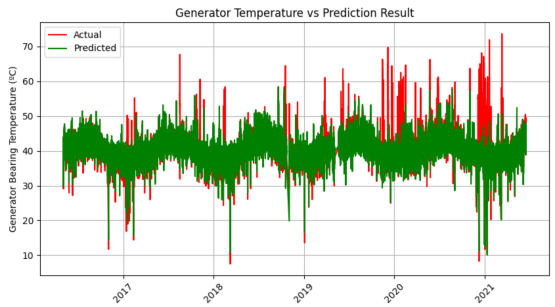
1. **R-squared**: A higher R-squared indicates a better fit of the model to the data. For all wind turbines, XGBoost has the highest R-squared, suggesting it is the best model for capturing the variance in the data.

2. **RMSE**: . A lower RMSE indicates a better fit of the model. Again, for all wind turbines, XGBoost has the lowest RMSE, meaning it was the most accurate model.

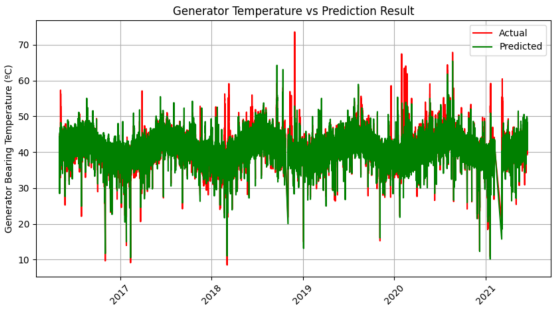
3. **MAE**: . A lower MAE indicates a better fit of the model. Yet again, XGBoost performed the best, with the lowest MAE across all wind turbines.

4. **MAPE**: . This metric is particularly useful when you want to understand the prediction error in percentage terms. Similar to the other metrics, a lower MAPE suggests a better fit of the model. XGBoost consistently yielded the lowest MAPE for all turbines, implying it was the most accurate model in percentage terms as well.

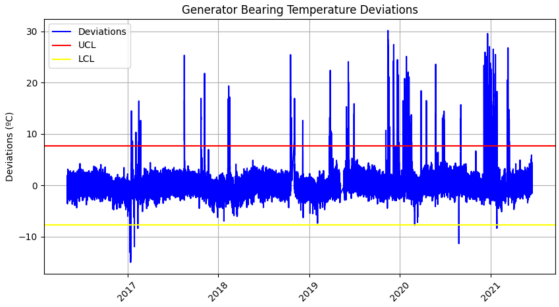
In conclusion, the XGBoost model performed the best among all the models evaluated for predicting wind turbine output, regardless of the turbine. Its performance was consistently superior across all evaluation metrics (R-squared, RMSE, MAE, and MAPE), implying that XGBoost was able to capture the underlying patterns in the wind turbine data more effectively than the other models. This would suggest that for further work in this area, the XGBoost model could be prioritized over the others for wind turbine output prediction. On top of that some hyperparameters could be even further optimized for the specific data and data size, as explained in the case study description of the XGBoost model.



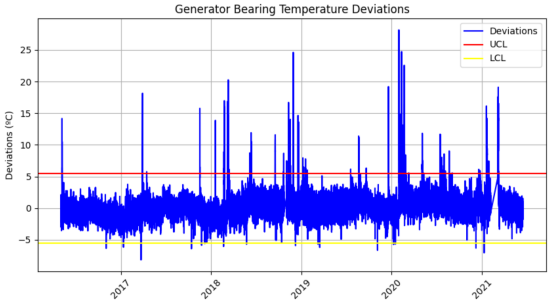
(a) Actual vs Predicted values WT 6



(b) Actual vs Predicted values WT 6



(c) Deviations plot WT 1



(d) Deviations plot WT 6

XGBoost Algorithm,

Fig. 4.6. Actual vs Predicted Temperatures, together with deviations for WT1 and WT6

4.0.6. LSTM Results

The Long Short-Term Memory (LSTM) model is a type of recurrent neural network that is commonly used in the prediction of time series data. Let's analyze its performance:

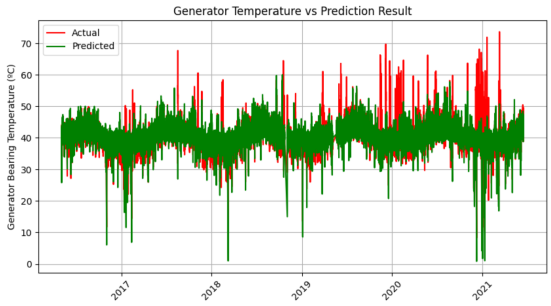
1. **R-squared**: The R-squared value for the LSTM model varies across the different wind turbines. In some cases, such as for Wind Turbine 2, its performance is close to the XGBoost model. However, for others like Wind Turbine 3, its performance is significantly lower. This indicates that LSTM is able to capture a fair amount of the variance in the data, but not consistently across all turbines.

2. **RMSE**: The Root Mean Squared Error for the LSTM model is generally higher than that for the XGBoost model across all wind turbines, which indicates that the predictions of the LSTM model are less accurate than those of the XGBoost model.

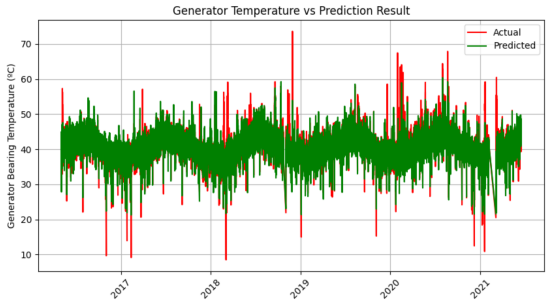
3. **MAE**: The Mean Absolute Error for the LSTM model is also typically higher than that of the XGBoost model. This suggests that on average, the absolute prediction errors of the LSTM model are larger, which again points to lower prediction accuracy.

4. **MAPE**: The Mean Absolute Percentage Error of the LSTM model is also higher than that of the XGBoost model. This means that, in percentage terms, the LSTM model's predictions are less accurate.

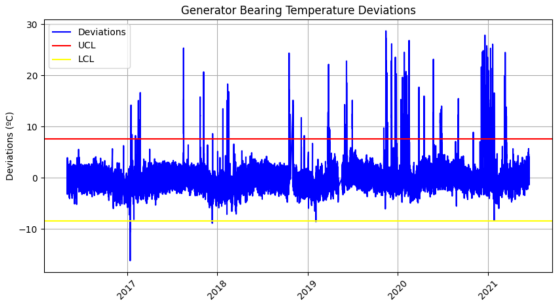
Overall, the LSTM model's performance is inconsistent and generally less accurate than the XGBoost model across all the metrics. This may be due to the LSTM model's sensitivity to the sequence of the data and the potentially non-sequential nature of the dataset. Additionally, the LSTM model may require more complex and careful parameter tuning, or it may need a larger dataset to effectively learn and make predictions. Although the model hyperparameters were based on the suggestions of the reference papers, perhaps this should be further investigated for the wind turbines studied in this work, as their LSTM models were the best performers in many areas.



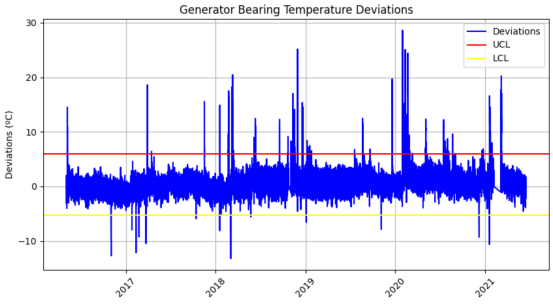
(a) Actual vs Predicted values WT 6



(b) Actual vs Predicted values WT 6



(c) Deviations plot WT 1



(d) Deviations plot WT 6

LSTM Algorithm,

Fig. 4.7. Actual vs Predicted Temperatures, together with deviations for WT1 and WT6

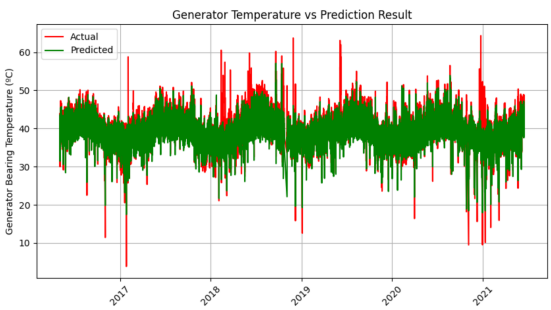
4.0.7. DNN Results

A Deep Neural Network (DNN) is a type of Artificial Neural Network with multiple layers between the input and output layers. They can model complex patterns and systems, making them powerful tools for predictions. Let's analyze its performance on the given dataset:

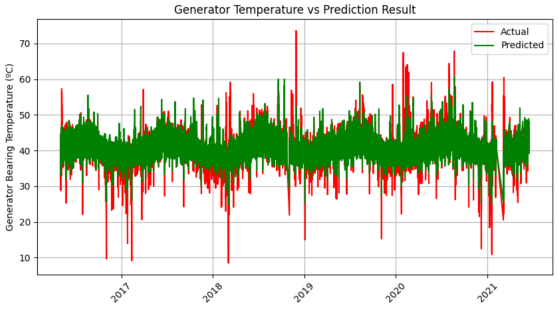
1. **R-squared**: The R-squared values for the DNN model are consistently lower than those for the XGBoost model across all the wind turbines. The highest R-squared value for DNN is 0.7518 (Wind Turbine 2), while the lowest is 0.5224 (Wind Turbine 4). This indicates that DNN model is not capturing as much of the variance in the data as the XGBoost model does.

2. **RMSE**: The Root Mean Squared Error (RMSE) values for the DNN model are generally higher than those for the XGBoost model, which suggests that the DNN model's predictions are less accurate.

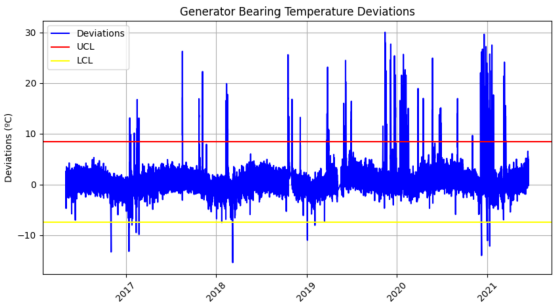
3. **MAE**: Similarly, the Mean Absolute Error (MAE) for the DNN model is typically higher than that of the XGBoost model across all the wind turbines. This indicates that the DNN model's predictions, on average, deviate more from the actual values.



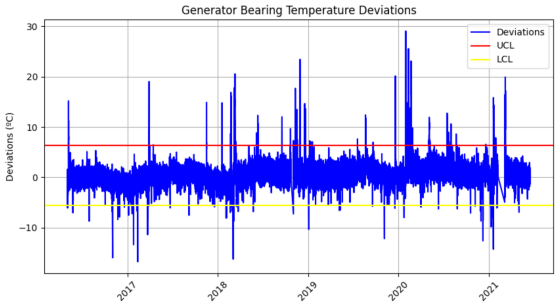
(a) Actual vs Predicted values WT 6



(b) Actual vs Predicted values WT 6



(c) Deviations plot WT 1



(d) Deviations plot WT 6

DNN Algorithm,

Fig. 4.8. Actual vs Predicted Temperatures, together with deviations for WT1 and WT6

4. **MAPE**: The Mean Absolute Percentage Error (MAPE) values of the DNN model are also higher than those of the XGBoost model, signifying that the DNN model's predictions are less accurate in terms of percentages.

In summary, the DNN model's performance, as measured by these metrics, is generally weaker than the XGBoost model across all the wind turbines. This could be due to various factors, such as the architecture of the DNN, the optimization method used, the lack of sufficient training data, or the presence of noisy data. Despite their ability to model complex patterns, DNN models require careful design, tuning, and sufficient data to achieve high performance. Due to inexperience in working with machine learning models, proper model tuning have most likely played the major role in the underpefromace of this model.

4.0.8. Interesting points from the Case Studies

Linearity of data over time

An interesting pattern identified when working with the models, was the decrease in model performance as more years of wind turbine operation were added into the machine learning dataset. To exemplify this, data for WT 3 is displayed in Table4.8. It can be appreciated that performance is kept roughly constant in the first 2 years of operation and

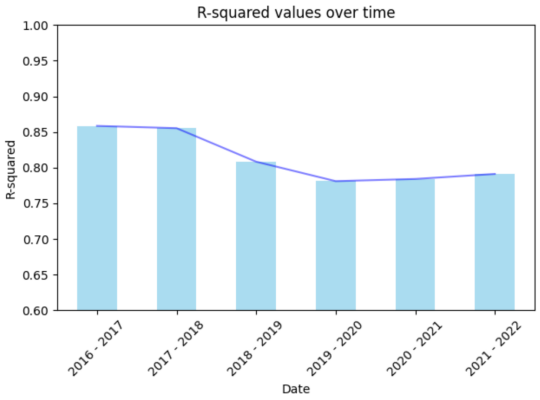
flattens in the following years , with a slight upturn in year 6. The trend can be clearly appreciated in Fig. 4.9a. This pattern is followed by all wind turbines and suggests that the temperature behaviour of the WT generator's bearing presents a high linear correlation with the inputs used in the model, and said linearity sees a drop with consecutive years until year 6. This fact could result potentially useful in formulating new approaches to condition monitoring in alike components in WT. An adaptable formulation that adjusts its hyperparmeters and boundaries accordingly to the operation time of the WT could present noticeable advantages in predictive ability.

This could be analogous to the "Bathtub curve" described in section "Failure statistics" in this work. This idea proposes that a wind turbine does follow a bathtub curve in terms of its likelihood of failure over time. The likelihood of failure consists of three periods: an initial period of high failure rate in the first 2-3 years of operation (infant mortality), a period of drop-off and constant failure rate (useful life) up to year 6 , a peak again in failures around this year and again a period of decreased and constant failure rate from that point on (wear-out).

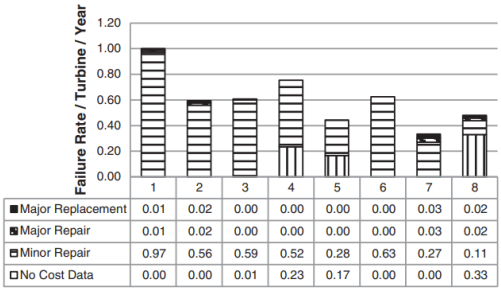
It seems that the R-squared value follows a direct relation with the failure rate described by the bathtub curve. While the R-squared and hence the linearity between inputs and bearings temperature, is high in the first years, the failure rate is high too. Both linearity and failure rate drop off in the following years and around year 6 they seem to spike up again. As mentioned before, a monitoring approach could be benefited from this correlation.

Table 4.8. WIND TURBINE 3, MLR ALGORITHM

DATE	**R-squared**	**RMSE**	**MAE**	**MAPE**
2016 - 2017	0.8586	1.1710	0.9516	2.3639
2017 - 2018	0.8554	1.2763	1.0266	2.6052
2018 - 2019	0.8084	1.6138	1.1749	2.9358
2019 - 2020	0.7811	1.7038	1.2050	2.9939
2020 - 2021	0.7842	1.6777	1.1892	2.9716
2021 - 2022	0.7912	1.6588	1.1869	2.9787



(a) R-squared evolution of generator's bearing WT6



	1	2	3	4	5	6	7	8
Major Replacement	0.01	0.02	0.00	0.00	0.00	0.00	0.03	0.02
Major Repair	0.01	0.02	0.00	0.00	0.00	0.00	0.03	0.02
Minor Repair	0.97	0.56	0.59	0.52	0.28	0.63	0.27	0.11
No Cost Data	0.00	0.00	0.01	0.23	0.17	0.00	0.00	0.33

(b) Failure rate per year for Generator

Model Extrapolation

One of the ideas that came to mind during the development of the project, was to try to see how well the trained models extrapolated to different wind turbines of the same characteristics. For that reason, a considered healthy specimen (WT6)was used to train a machine learning model and then test its predictive abilities on a highly faulty wind turbine. This resulted in a model that very poorly performed on the new data. The performance metrics indicated that the model was not even able to predict the variation in actual bearing temperatures with 50% accuracy. This result was slightly worse than the model trained on WT1 data, which got an R-squared value of 0.5571. It must be said that the training data set and testing data set was heavily unbalanced, as the training set was cons and corrective measures for this issue would have possibly improved the performance of this strategy. Despite this several conclusions can be yielded from this result:

- Although the performance was slightly lower, the reduction was small enough to consider the extrapolated model, of usefulness and relevancy for condition monitoring purposes of other wind turbines of similar characteristics.
- Perhaps the linearity of WT6 is higher than the one presented in WT1. This can play a crucial role in the results obtained with this model, and the consequent underperformance. Moreover, this approach was tested for other wind turbine pairs such as WT6 and WT5 and the model trained with WT6 data performed considerably good with WT5. There was a decrease in performance but just of 0.03 points in R-squared value. On top of that, the model saw even further improvements when testing just the first years of operation of the WTs and consecutive detriment in performance when increasing the lifespan of the WT. This could suggest the special usefulness of these models to predict unknown data on relatively new systems, in terms of time in operation time.
- After testing, this approach with other models capable of predicting better in non-linear environments, the reduction of performance was significantly lower, suggest-

ing that in general WT bearing temperature behaviour follows a non-linear correlation with the input variables.

- usefulness of Manahalobis distance filtering - By apply Manahalobis distance filtering on on-linear algorithms such as LSTM a slight improvement on the performance of the model was appreciated, although it wasn't very substantial: from 0.79927 in the non filtered model to 0.80756 in the filtered data with 10% threshold.

Table 4.9. PERFORMANCE OF MODEL WT6 ON WT1

ML model	**R-squared**	**RMSE**	**MAE**	**MAPE**
MLR	0.4876	2.8846	1.8792	4.5754

Comparison with reference papers

The built models underperfomed in all wind turbines the case studies described in the reference papers. This could be due to several reasons. First, is models fine-tuning and hyperamarameters definition. This has been commented already in the results section but it is probably one of the main reasons behind this lower metrics. Their models have been optimized to work with their specific dataset and wind turbine parameters while in the case of this case study, the tuning method has been mainly done doing literature review and in the case of XGBoost with limited hyperparameter optimization. The second reason could be the intrinsic character of the wind turbine behaviour. A more "unhealthy" data set from wind turbines presenting more faults, would definitely worsen the predictive results of these ML models, whose essence relies on modeling healthy behaviour to detect outliers. The third, but not least important reason, could be data cleaning. Although the defined normal ranges were defined based on available data sheet of the wind turbine and careful research of usual working parameters in the industry, maybe this ranges were wrongly estimated, and they were to wide. On top of this, the Manahalobis distance filtering, wasn't found to be very useful in the case studies, whereas it plays a major role in the reference papers. It is possible that the distance filtering was wrongly applied while building the model. A sign that gives more weight to this idea is the filtered power curve plots yielded by the models, which still presents outliers.

5. CONCLUSION AND FUTURE LINES OF INVESTIGATION

5.0.1. Closing Statement

This work, has achieved its goal to demonstrate that even with limited resources and very little experience effective condition assessment techniques for wind turbines can be set up in place. Some interesting point have been discussed such as linearity of typical input-output datasets in the context of wind turbines and effectiveness of machine learning models in combination with SCADA data. The developed models were able to predict with up to 80% accuracy the temperatures of a specific component, which seems quite impressive considering the described limitations. Furthermore the trained models also performed relatively well in new, unseen data from other wind turbines. This shows the value of these approach for early and new wind farms, without much data at their disposal. More advanced research and further commitment to develop complex and detailed models for wind turbines can mean the first definitive step for the establishment of renewable technologies in today's world.

SCADA data condition monitoring approaches could become the leading ones in the following years. Although it seems that other technologies such as vibrations or acoustic monitoring present meaningful advantages in terms of capabilities for early detection and accurate diagnosis of mechanical faults, machine learning models could reveal uncovered ways to improve SCADA data performance, up to a comparable level with the other technologies. This would definitely aid the adoption of green energies in modern economies and help build a more sustainable future for the upcoming generations.

5.0.2. Future lines of investigation:

To start, changes in the done work will be discussed. Many things could have been improved in the process of building the models. Some of them have been already discussed in the description and results of the case studies.

1. First, the preliminary cleaning could have been further improved, by investigating the **normal and expected working parameters** of the studied Wind Turbines.

2. Second further research on Manahalobis distance filtering and even changing the model completely could improve the performance of all the models. After filtering for 10% of outliers in the wind speed-power input pair, the curve still looks slightly scattered and with many outliers outside of the main cluster. Perhaps the data don't follow a normal Gaussian distribution and **other filtering manners** could be better employed for this cases.

3. Despite the encouraging performance of the XGBoost model, there's room for further refinement. **More complex or different types of models**, such as ensemble learning or reinforcement learning, could be explored to see if they can further improve the prediction accuracy. Additionally, fine-tuning of hyperparameters, using techniques like GridSearch or RandomSearch, might improve the models' performance. This point alone, is believed to be responsible for most of the deviations between the references studies and the developed case studies.

4. The current models make predictions based on the provided dataset, but there may be other, unexplored variables that could contribute to the model's effectiveness. Investigation into additional features, such as weather conditions, turbine operational history, and load demand, might enhance the model's ability to predict bearing temperature accurately. Effective **Feature engineering** could uncover the best possible input parameters for optimal predictive performance.

In terms of improvements and other lines of investigation outside of the current models, the following could be implemented:

- Adding more weight to certain environmental SCADA parameters in the input datasets of the ML models, accordingly to the location of the studied wind farms.

- The models developed in this project have been tested on specific wind turbines. Investigating how these models perform on a broader range of wind turbines, possibly from different manufacturers or under varying operating conditions, could improve their scalability and generalization.

- While predicting the bearing temperature is useful, predicting faults directly could be even more valuable for preventative maintenance. Further research could focus on identifying patterns that precede faults, enabling the development of models that can predict faults before they occur.

- Taking advantage of higher resolution SCADA data(5min, 10s, 1s, 100HZ...). This could potentially resolve the gap between other condition monitoring techniques in terms of capability for early fault detection.

- Combining SCADA data with other data sources such as vibration, current signature analysis. This approach has already been explored in the field, and promising performance has been discovered. By combining the features from all these technologies, more robust, reliable and adaptable models can be built.

By following these or related lines of investigation, we can continue to refine and expand the application of ML for wind turbine condition monitoring, potentially contributing significantly to the efficiency and sustainability of wind energy production.

6. SOCIOECONOMIC IMPACT

The application of ML models for condition monitoring of wind turbines can have significant socioeconomic impacts.

- **Job Creation**: The development, implementation, and maintenance of these ML models require skilled labor, potentially leading to the creation of high-quality jobs in data science, machine learning, and renewable energy sectors.
- **Cost Efficiency**: Predictive maintenance, enabled by these models, can lead to significant cost savings. By predicting faults before they occur, costly repairs or replacements can be avoided, and the downtime of turbines can be minimized.
- **Energy Security and Sustainability**: Improving the efficiency and reliability of wind turbines contributes to a more resilient and sustainable energy infrastructure. This aligns with global goals of transitioning towards renewable energy and can strengthen energy security in many regions. By increasing the financial competitiveness of wind power energy generation, green energy overtake could accelerate significantly.
- **SCADA data revaluation** : From all the condition monitoring approaches discussed earlier, SCADA data could potentially stand out as the most relevant one of them. If further research proofed that reliability and accuracy of SCADA data was good enough to serve as the go to health level monitoring technique, significant costs and complexity could be saved.
- **Machine Learning definitive incorporation into the industry**: with the current boom of large language models and intelligent and compute-efficient IA systems, it would be wise to adopt this technology and its features into this area of study as soon as possible. Data rich dataset such as SCADA could be the perfect complementary piece for this technologies.

6.1. Budget

The budget for implementing ML models in wind turbine condition monitoring can vary significantly based on the specific scope and requirements. Major cost components could include:

Data Acquisition and Preparation: This involves the collection, cleaning, and preprocessing of data from the wind turbines. The cost here would depend on whether new sensors need to be installed, or existing data can be used. As mentioned previously,

SCADA systems are typically installed in every wind turbine installed nowadays, therefore the costs from the data acquisition point of view are practically negligible.

Model Development and Validation: This includes the cost of the data scientists and engineers working on the model, computational resources, and any software licenses required. For the models proposed in the developed case studies, not many resources would be needed. A small teams of Data scientists or engineers familiar with this type of work could easily take on the task of building, managing and fine-tuning models capable of performing the tasks described. Moreover, a team centered on this task solely, could easily implement some of the features described in the "future lines of investigation" section and achieve greater results. Expenses would mainly depend on the typical salaries of the country where wind turbines would be deployed and the expertise of the personnel. Moreover, this task could be done remotely, and personnel could be working from anywhere in the world, thanks to the data transmission capabilities existing in SCADA systems.

Implementation and Maintenance: Once the model is developed, there will be costs associated with integrating it into existing systems, potential cloud storage and computing fees, as well as ongoing costs for model updates and maintenance. Hardware for data processing could be a major expense if the compute power is decided to be kept in house.

A detailed budget should be prepared based on the specific needs and scale of the project. It's also important to factor in potential cost savings from improved maintenance practices, which can offset some of the upfront and ongoing costs. For example maintenance visits to the wind turbines could be accurately scheduled, to prevent expensive failure situations in the wind turbine. Early, replacements could potentially save many other components that could result affected from the failure.

7. REGULATORY FRAMEWORK

The implementation and application of Machine Learning (ML) models for condition monitoring of wind turbines must be done within an appropriate regulatory framework. This is crucial to ensure that the practices are not only technically sound but also ethically and legally compliant. This project isn't place specific, as it treats a general approach for predicting system faults of any wind turbine. In most jurisdictions, the following aspects are of paramount importance:

Data Privacy and Security: The data used for training and testing the ML models can sometimes contain sensitive information, such as precise geolocation of the wind turbines or proprietary technical details. Therefore, all the procedures must comply with local and international data protection regulations, such as the General Data Protection Regulation (GDPR) in the European Union**European**.

Safety Standards: The ML models are used to predict the bearing temperature in wind turbines, a critical component related to the safe operation of the equipment. Any use of these predictions for maintenance and operations should therefore comply with established safety standards for wind turbines, such as those outlined by the International Electrotechnical Commission (IEC)**2023-06-09_2023-06-08_2023-06-07_2023**.

Model Transparency and Fairness: Given the increasing demand for explainability in machine learning, it's important to ensure that the models used are as transparent and understandable as possible. Furthermore, any automated decision-making process, if employed, must be fair and non-discriminatory.

BIBLIOGRAPHY

[1] P. bynbsp; Lucía Fernández and F. 8, *Global wind energy share in electricity mix 2021*, Feb. 2023. [Online]. Available: https://www.statista.com/statistics/1302053/global-wind-energy-share-electricity-mix/.

[2] Iberdrola, Dec. 2017. [Online]. Available: https://www.iberdrola.com/sustainability/wind-power-evolution-europe#:~:text=EVOLUTION%5C%20OF%5C%20WIND%5C%20ENERGY%5C%20This%5C%20news%5C%2C%5C%20which%5C%20is,a%5C%20total%5C%20of%5C%2017.4%5C%20GW%5C%20of%5C%20new%5C%20installations.

[3] [Online]. Available: https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/european-green-deal.

[4] [Online]. Available: https://www.cfr.org/backgrounder/paris-global-climate-change-agreements.

[5] [Online]. Available: https://www.siemensgamesa.com/products-and-services/offshore/wind-turbine-sg-11-0-200-dd.

[6] T. Stehly, P. Beiter, and P. Duffy, "2019 cost of wind energy review," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2020.

[7] X. Jin, Z. Xu, and W. Qiao, "Condition monitoring of wind turbine generators using scada data analysis," *IEEE Transactions on Sustainable Energy*, vol. 12, no. 1, pp. 202–210, 2020.

[8] J. Van Rensselar, "The elephant in the wind turbine," *Tribology & Lubrication Technology*, vol. 66, no. 6, p. 38, 2010.

[9] W. Qiao and D. Lu, "A survey on wind turbine condition monitoring and fault diagnosis—part i: Components and subsystems," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, pp. 6536–6545, 2015.

[10] E. Taherian-Fard, R. Sahebi, T. Niknam, A. Izadian, and M. Shasadeghi, "Wind turbine drivetrain technologies," *IEEE Transactions on Industry Applications*, vol. 56, no. 2, pp. 1729–1741, 2020.

[11] K. Kails, Q. Li, and M. Mueller, "A modular and cost-effective high-temperature superconducting generator for large direct-drive wind turbines," *IET Renewable Power Generation*, vol. 15, no. 9, pp. 2022–2032, 2021.

[12] A. R. Nejad *et al.*, "Wind turbine drivetrains: State-of-the-art technologies and future development trends," *Wind Energy Science*, vol. 7, no. 1, pp. 387–411, 2022.

[13] Z. Xu *et al.*, "A state-of-the-art review of the vibration and noise of wind turbine drivetrains," *Sustainable Energy Technologies and Assessments*, vol. 48, p. 101 629, 2021.

[14] E. Hart, "Developing a systematic approach to the analysis of time-varying main bearing loads for wind turbines," *Wind Energy*, vol. 23, no. 12, pp. 2150–2165, 2020.

[15] J. Keller, Y. Guo, W. LaCava, H. Link, and B. McNiff, "Gearbox reliability collaborative phase 1 and 2: Testing and modeling results," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2012.

[16] G. P. Prajapat, N. Senroy, and I. Kar, "Modeling and impact of gear train backlash on performance of dfig wind turbine system," *Electric Power Systems Research*, vol. 163, pp. 356–364, 2018.

[17] J. Helsen, C. Devriendt, W. Weijtjens, and P. Guillaume, "Experimental dynamic identification of modeshape driving wind turbine grid loss event on nacelle testrig," *Renewable Energy*, vol. 85, pp. 259–272, 2016.

[18] L. Sethuraman, V. Venugopal, A. Zavvos, and M. Mueller, "Structural integrity of a direct-drive generator for a floating wind turbine," *Renewable energy*, vol. 63, pp. 597–616, 2014.

[19] C. Dao, B. Kazemtabrizi, and C. Crabtree, "Wind turbine reliability data review and impacts on levelised cost of energy," *Wind Energy*, vol. 22, no. 12, pp. 1848–1871, 2019.

[20] J. Tautz-Weinert and S. J. Watson, "Using scada data for wind turbine condition monitoring–a review," *IET Renewable Power Generation*, vol. 11, no. 4, pp. 382–394, 2017.

[21] M. Wilkinson *et al.*, "Measuring wind turbine reliability-results of the reliawind project," *Wind Energy*, vol. 35, no. 2, pp. 102–109, 2011.

[22] S. Sheng, "Report on wind turbine subsystem reliability-a survey of various databases (presentation)," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2013.

[23] J. Carroll, A. McDonald, and D. McMillan, "Failure rate, repair time and unscheduled o&m cost analysis of offshore wind turbines," *Wind Energy*, vol. 19, no. 6, pp. 1107–1119, 2016.

[24] M. S. Alvarez-Alvarado and D. Jayaweera, "Bathtub curve as a markovian process to describe the reliability of repairable components," *IET Generation, Transmission & Distribution*, vol. 12, no. 21, pp. 5683–5689, 2018.

[25] Y. Qu, E. Bechhoefer, D. He, and J. Zhu, "A new acoustic emission sensor based gear fault detection approach," *International Journal of Prognostics and Health Management*, vol. 4, pp. 32–45, 2013.

[26] S. R. Saufi, Z. A. B. Ahmad, M. S. Leong, and M. H. Lim, "Low-speed bearing fault diagnosis based on arssae model using acoustic emission and vibration signals," *IEEE Access*, vol. 7, pp. 46 885–46 897, 2019.

[27] X. Pan *et al.*, "Early warning of damaged wind turbine blades using spatial–temporal spectral analysis of acoustic emission signals," *Journal of Sound and Vibration*, vol. 537, p. 117 209, 2022.

[28] S. Sheng, "Investigation of oil conditioning, real-time monitoring and oil sample analysis for wind turbine gearboxes (presentation)," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2011.

[29] W. Qiao and D. Lu, "A survey on wind turbine condition monitoring and fault diagnosis—part ii: Signals and signal processing methods," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, pp. 6546–6557, 2015.

[30] L. M. Popa, B.-B. Jensen, E. Ritchie, and I. Boldea, "Condition monitoring of wind generators," in *38th IAS Annual Meeting on Conference Record of the Industry Applications Conference, 2003.*, IEEE, vol. 3, 2003, pp. 1839–1846.

[31] W. Yang, P. Tavner, C. Crabtree, and M. Wilkinson, "Research on a simple, cheap but globally effective condition monitoring technique for wind turbines," in *2008 18th International Conference on Electrical Machines*, IEEE, 2008, pp. 1–5.

[32] W. Jeffries, J. A. Chambers, and D. G. Infield, "Experience with bicoherence of electrical power for condition monitoring of wind turbine blades," *IEE Proceedings-vision, image and signal processing*, vol. 145, no. 3, pp. 141–148, 1998.

[33] W. Yang, R. Court, and J. Jiang, "Wind turbine condition monitoring by the approach of scada data analysis," *Renewable energy*, vol. 53, pp. 365–376, 2013.

[34] D. Lu, X. Gong, and W. Qiao, "Current-based diagnosis for gear tooth breaks in wind turbine gearboxes," in *2012 IEEE Energy Conversion Congress and Exposition (ECCE)*, IEEE, 2012, pp. 3780–3786.

[35] O. I. Owolabi, N. Madushele, P. A. Adedeji, and O. O. Olatunji, "Fem and ann approaches to wind turbine gearbox monitoring and diagnosis: A mini review," *Journal of Reliable Intelligent Environments*, pp. 1–21, 2022.

[36] A. Hu, L. Xiang, and L. Zhu, "An engineering condition indicator for condition monitoring of wind turbine bearings," *Wind Energy*, vol. 23, no. 2, pp. 207–219, 2020.

[37] M.-H. Wang, S.-D. Lu, C.-C. Hsieh, and C.-C. Hung, "Fault detection of wind turbine blades using multi-channel cnn," *Sustainability*, vol. 14, no. 3, p. 1781, 2022.

[38] M. Ibrion, N. Paltrinieri, and A. R. Nejad, "Learning from failures in cruise ship industry: The blackout of viking sky in hustadvika, norway," *Engineering Failure Analysis*, vol. 125, p. 105 355, 2021.

[39] Y. Merizalde, L. Hernández-Callejo, O. Duque-Pérez, and V. Alonso-Gómez, "Diagnosis of wind turbine faults using generator current signature analysis: A review," *Journal of Quality in Maintenance Engineering*, vol. 26, no. 3, pp. 431–458, 2019.

[40] A. Kusiak and W. Li, "The prediction and diagnosis of wind turbine faults," *Renewable energy*, vol. 36, no. 1, pp. 16–23, 2011.

[41] L. Xiang, P. Wang, X. Yang, A. Hu, and H. Su, "Fault detection of wind turbine based on scada data analysis using cnn and lstm with attention mechanism," *Measurement*, vol. 175, p. 109 094, 2021.

[42] A. D. Bebars, A. A. Eladl, G. M. Abdulsalam, and E. A. Badran, "Internal electrical fault detection techniques in dfig-based wind turbines: A review," *Protection and Control of Modern Power Systems*, vol. 7, no. 1, p. 18, 2022.

[43] M. Tang *et al.*, "Review and perspectives of machine learning methods for wind turbine fault diagnosis," *Frontiers in Energy Research*, p. 596, 2021.

[44] P. Marti-Puig, A. Blanco-M, J. J. Cárdenas, J. Cusidó, and J. Solé-Casals, "Effects of the pre-processing algorithms in fault diagnosis of wind turbines," *Environmental modelling & software*, vol. 110, pp. 119–128, 2018.

[45] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial intelligence*, vol. 97, no. 1-2, pp. 245–271, 1997.

[46] R. H. Shumway, D. S. Stoffer, R. H. Shumway, and D. S. Stoffer, "Arima models," *Time Series Analysis and Its Applications: With R Examples*, pp. 75–163, 2017.

[47] B. Amirataee, M. Montaseri, and H. Sanikhani, "The analysis of trend variations of reference evapotranspiration via eliminating the significance effect of all autocorrelation coefficients," *Theoretical and Applied Climatology*, vol. 126, pp. 131–139, 2016.

[48] D. W. Stroock, *An introduction to Markov processes*. Springer Science & Business Media, 2013, vol. 230.

[49] Y. Zhao *et al.*, "Fault prognosis of wind turbine generator using scada data," in *2016 North American Power Symposium (NAPS)*, IEEE, 2016, pp. 1–6.

[50] W. Yang, R. Court, and J. Jiang, "Wind turbine condition monitoring by the approach of scada data analysis," *Renewable energy*, vol. 53, pp. 365–376, 2013.

[51] Y. Qiu, Y. Feng, J. Sun, W. Zhang, and D. Infield, "Applying thermophysics for wind turbine drivetrain fault diagnosis using scada data," *IET Renewable Power Generation*, vol. 10, no. 5, pp. 661–668, 2016.

[52] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[53] Z. Yongjie, W. Dongfeng, Z. Junying, and H. Yuejiao, "Research on early fault diagnostic method of wind turbines," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 5, pp. 2330–2341, 2013.

[54] X. Jin, Z. Xu, and W. Qiao, "Condition monitoring of wind turbine generators using scada data analysis," *IEEE Transactions on Sustainable Energy*, vol. 12, no. 1, pp. 202–210, 2020.

[55] E. Ghasemi, A. Aaghaie, and E. A. Cudney, "Mahalanobis taguchi system: A review," *International Journal of Quality & Reliability Management*, 2015.

[56] W. Udo and Y. Muhammad, "Data-driven predictive maintenance of wind turbine based on scada data," *IEEE Access*, vol. 9, pp. 162 370–162 388, 2021.

[57] X. Fan, X. Yang, X. Li, and J. Wang, "A particle-filtering approach for remaining useful life estimation of wind turbine gearbox," in *International conference on chemical, material and food engineering*, Atlantis Press, 2015, pp. 198–200.

[58] Y. Zhao *et al.*, "Fault prediction and diagnosis of wind turbine generators using scada data," *Energies*, vol. 10, no. 8, p. 1210, 2017.

[59] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863–905, 2018.

[60] I. Cohen *et al.*, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

[61] A. Santolamazza, D. Dadi, and V. Introna, "A data-mining approach for wind turbine fault detection based on scada data analysis using artificial neural networks," *Energies*, vol. 14, no. 7, p. 1845, 2021.

[62] C. Plumley, *Kelmarsh wind farm data*, Feb. 2022. [Online]. Available: https://zenodo.org/record/5841834#.ZGo6en3P1D_.

[63] L. Bauer, *Senvion mm92*. [Online]. Available: https://en.wind-turbine-models.com/turbines/889-senvion-mm92.

[64] K. Pearson, "Notes on the history of correlation," *Biometrika*, vol. 13, no. 1, pp. 25–45, 1920.

[65] C. Spearman, "The proof and measurement of association between two things," *The American journal of psychology*, vol. 100, no. 3/4, pp. 441–471, 1987.

[66] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

A. ANNEX: SCRIPTS FOR MLR, XGBOOST, LSTM AND DNN MODELS

A.1. MLR Script

```
1
2  import pandas as pd
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from sklearn.model_selection import train_test_split,
       cross_val_score
6  from sklearn.linear_model import LinearRegression
7  from sklearn.preprocessing import StandardScaler
8  from sklearn.metrics import r2_score, mean_squared_error,
       mean_absolute_error
9  from scipy import stats
10 from sklearn.cluster import KMeans
11 #Remove Matplotlib Warnings/outdated interpretter
12 import warnings
13 import matplotlib.cbook
14 warnings.filterwarnings("ignore",category=matplotlib.cbook.
       mplDeprecation)
15
16 # Define wind turbine to be analyzed
17 u = 3
18
19 # Define your file paths here
20 file_paths = [
21     'Turbine_Data_Kelmarsh_'+str(u)+'_2021-01-01_-_2021-07-01_228.
          csv',
22     'Turbine_Data_Kelmarsh_'+str(u)+'_2020-01-01_-_2021-01-01_228.
          csv',
23     'Turbine_Data_Kelmarsh_'+str(u)+'_2019-01-01_-_2020-01-01_228.
          csv',
24     'Turbine_Data_Kelmarsh_'+str(u)+'_2018-01-01_-_2019-01-01_228.
          csv',
25     'Turbine_Data_Kelmarsh_'+str(u)+'_2017-01-01_-_2018-01-01_228.
          csv',
26     'Turbine_Data_Kelmarsh_'+str(u)+'_2016-01-03_-_2017-01-01_228.
          csv'
27 ]
28
29 # Turbine to be analyzed
30 j = 6
31 new_data_file_path = [
```

```
        'Turbine_Data_Kelmarsh_'+str(j)+'_2021-01-01_-_2021-07-01_228.csv',
        'Turbine_Data_Kelmarsh_'+str(j)+'_2020-01-01_-_2021-01-01_228.csv',
        'Turbine_Data_Kelmarsh_'+str(j)+'_2019-01-01_-_2020-01-01_228.csv',
        'Turbine_Data_Kelmarsh_'+str(j)+'_2018-01-01_-_2019-01-01_228.csv',
        'Turbine_Data_Kelmarsh_'+str(j)+'_2017-01-01_-_2018-01-01_228.csv',
        'Turbine_Data_Kelmarsh_'+str(j)+'_2016-01-03_-_2017-01-01_228.csv'
    ]




    # Define your normal ranges here
    normal_ranges = {
        'Nacelle temperature (  C  )': (-10, 50),
        'Power (kW)': (0, 2000),
        'Rotor speed (RPM)': (0, 73),
        'Stator temperature 1 (  C  )': (0, 150),
        'Wind speed (m/s)': (0, 24),
        'Generator bearing rear temperature (  C  )': (0, 100),
    }

    # Concatenate data from all files
    dataframes = [pd.read_csv(file_path, skiprows=9, encoding='ISO-8859-1') for file_path in file_paths]
    data = pd.concat(dataframes, ignore_index=True)

    data = data[['# Date and time', 'Nacelle temperature (  C  )', 'Power (kW)', 'Rotor speed (RPM)', 'Stator temperature 1 (  C  )', 'Generator bearing rear temperature (  C  )', 'Wind speed (m/s)']]

    # Extract wind speed and power data
    wind_speed = data['Wind speed (m/s)']
    power = data['Power (kW)']

    # Initial number of data points
    data_1 = len(data)

    # Plot
    plt.figure(figsize=(10, 5))
    plt.scatter(wind_speed, power, s=5, color='blue')
    plt.title('Wind Turbine Power Curve')
    plt.xlabel('Wind Speed (m/s)')
    plt.ylabel('Power (kW)')
    plt.grid()
```

```
plt.show()

# Drop missing values
data = data.dropna()

# Filter based on the normal ranges and other conditions
for col, (min_val, max_val) in normal_ranges.items():
    data = data[data[col].between(min_val, max_val)]
data = data[data['Power (kW)'] > 0]
data = data[data['Wind speed (m/s)'] > 3]  # Assuming cut-in wind speed as 3 m/s

# Extract the input and output columns from data
input_features = ['Nacelle temperature (  C  )', 'Power (kW)', 'Rotor speed (RPM)',
                  'Stator temperature 1 (  C  )', 'Wind speed (m/s)']
output_feature = 'Generator bearing rear temperature (  C  )'
time_data = data['# Date and time']

# Compute and print the correlation coefficients


# Prepare the data for train_test_split
X = data[input_features]
y = data[output_feature]

data_2=len(data)
data_3= len(data)*0.7


# Split data into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Extract wind speed and power data again after cleaning
wind_speed_after_cleaning = X_train['Wind speed (m/s)']
power_after_cleaning = X_train['Power (kW)']


from sklearn.covariance import EmpiricalCovariance

# Define the number of clusters for K-means
n_clusters = 3

# Apply K-means clustering on wind speed and power only
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
X_train_clustered = kmeans.fit_predict(X_train[['Wind speed (m/s)', 'Power (kW)']])

```

```
# Calculate Mahalanobis distance for each observation based on wind speed and power
distances = np.zeros(X_train.shape[0])
for i in range(n_clusters):
    cluster_center = kmeans.cluster_centers_[i]
    cluster_points = X_train[X_train_clustered == i][['Wind speed (m/s)', 'Power (kW)']].values
    cov_inv = np.linalg.pinv(np.cov(cluster_points, rowvar=False))
    for j in range(cluster_points.shape[0]):
        diff = cluster_points[j, :] - cluster_center
        distances[j] = np.sqrt(diff.T @ cov_inv @ diff)

# Determine threshold for outliers
threshold = np.percentile(distances, 90)  # Exclude top 15% as anomalies

# Filter out the outliers from the training data
mask = distances < threshold
X_train = X_train[mask]
y_train = y_train[mask]

# Extract wind speed and power data after cleaning and splitting
wind_speed_train = X_train['Wind speed (m/s)']
power_train = X_train['Power (kW)']


# First clenaing
print("Data before cleaning: ", round(data_1, 2), " after: : ", round(data_2, 2), " --->  Reduction of: ", (round(((data_1-data_2))/data_1*100, 2)),"%")
# Filtering
print("Training data before Mahalanobis filtering: ",round(data_3, 2), "after: ", round(len(X_train), 2), "Reduction of ---> ", round((data_3-len(X_train))/data_3*100, 2), "%")


# Plot after cleaning and splitting
# Plot after cleaning and Mahalanobis distance filtering
plt.figure(figsize=(10, 5))
plt.scatter(wind_speed_after_cleaning, power_after_cleaning, s=5, color='red', label='After Preliminary Cleaning')
plt.scatter(wind_speed_train, power_train, s=5, color='blue', label='After Mahalanobis Filtering')
plt.title('Wind Turbine Test Set Power Curve - After Cleaning and Mahalanobis Filtering')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Power (kW)')
plt.legend(loc='upper left')
plt.grid()
plt.show()

```

```
159
160 # Beginning of  correlations plot -------
161 # Compute and print the correlation coefficients
162 correlations = {}
163 for input_feature in input_features:
164     correlations[input_feature] = {
165         'Pearson': data[input_feature].corr(data[output_feature],
              method='pearson'),
166         'Spearman': data[input_feature].corr(data[output_feature],
              method='spearman'),
167         'Kendall': data[input_feature].corr(data[output_feature],
              method='kendall'),
168     }
169
170 # New code to plot the relation between each of the inputs and the
      output temperature.
171 plt.figure(figsize=(18, 12))
172 for i, input_feature in enumerate(input_features, 1):
173     plt.subplot(2, 3, i)
174     plt.scatter(X[input_feature], y, s=5)
175     plt.xlabel(input_feature)
176     plt.ylabel('Generator bearing rear temperature (  C  )')
177
178     # Add a trendline
179     x = X[input_feature]
180     y_trend = y
181     slope, intercept, r_value, p_value, std_err = stats.linregress(x
          , y_trend)
182     plt.plot(x, intercept + slope * x, 'r', label='fitted line')
183
184     # Add the correlation coefficients to the subplot
185     plt.text(0.7, 0.075, f"Pearson: {correlations[input_feature]['
          Pearson']:.2f}", transform=plt.gca().transAxes)
186     plt.text(0.7, 0.05, f"Spearman: {correlations[input_feature]['
          Spearman']:.2f}", transform=plt.gca().transAxes)
187     plt.text(0.7, 0.025, f"Kendall: {correlations[input_feature]['
          Kendall']:.2f}", transform=plt.gca().transAxes)
188
189 plt.tight_layout()
190 plt.show()
191
192 # End of correlations plot -------
193
194 # Extract date and time for test set before scaling
195 date_test = time_data[X_test.index]
196
197 # Standardize the data
198 scaler = StandardScaler()
199 X_train = scaler.fit_transform(X_train)
200 X_test = scaler.transform(X_test)
201
```

```
# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Perform cross-validation
scores = cross_val_score(model, X_train, y_train, cv=5)

print('Cross-Validation Scores:', scores)


y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)

rmse_train = np.sqrt(mean_squared_error(y_train, y_pred_train))
rmse_test = np.sqrt(mean_squared_error(y_test, y_pred_test))

mae_train = mean_absolute_error(y_train, y_pred_train)
mae_test = mean_absolute_error(y_test, y_pred_test)

mape_train = np.mean(np.abs((y_train - y_pred_train) / y_train)) *
    100
mape_test = np.mean(np.abs((y_test - y_pred_test) / y_test)) * 100

print('Train Metrics:\nR2:', r2_train, 'RMSE:', rmse_train, 'MAE:',
    mae_train, 'MAPE:', mape_train)
print('Test Metrics:\nR2:', r2_test, 'RMSE:', rmse_test, 'MAE:',
    mae_test, 'MAPE:', mape_test)

deviations = y_test - y_pred_test

UCL = deviations.mean() + 3*deviations.std()
LCL = deviations.mean() - 3*deviations.std()

# Prepare data for plotting
plotting_data = pd.DataFrame({'Date': date_test,
                              'Actual': y_test, 'Predicted':
                                 y_pred_test})
plotting_data['Date'] = pd.to_datetime(plotting_data['Date'])
plotting_data.sort_values('Date', inplace=True)

# Plot 1: Actual vs Predicted
plt.figure(figsize=(10, 5))
plt.plot(plotting_data['Date'], plotting_data['Actual'], label='
    Actual', color='red')
plt.plot(plotting_data['Date'], plotting_data['Predicted'], label='
    Predicted', color='green')
plt.title('Generator Temperature vs Prediction Result')
plt.xlabel('Date')
```

```
247 plt.ylabel('Generator Bearing Temperature ( C )')
248 plt.legend()
249 plt.xticks(rotation=45)
250 plt.grid()
251 plt.show()
252
253 # Calculate Deviations
254 plotting_data['Deviations'] = plotting_data['Actual'] -
        plotting_data['Predicted']
255
256 # Calculate standard deviation for the Deviations
257 std_dev = plotting_data['Deviations'].std()
258
259 # Set UCL and LCL (usually set at mean    3*std_dev for control
        charts)
260 plotting_data['UCL'] = plotting_data['Deviations'].mean() + 3*
        std_dev
261 plotting_data['LCL'] = plotting_data['Deviations'].mean() - 3*
        std_dev
262
263 # Plot 2: Control Chart
264 plt.figure(figsize=(10, 5))
265 plt.plot(plotting_data['Date'], plotting_data['Deviations'], label='
        Deviations', color='blue')
266 plt.axhline(y=plotting_data['UCL'].values[0], label='UCL', color='
        red')  # UCL as straight line
267 plt.axhline(y=plotting_data['LCL'].values[0], label='LCL', color='
        green')  # LCL as straight line
268 plt.title('Generator Bearing Temperature Deviations')
269 plt.xlabel('Date')
270 plt.ylabel('Deviations ( C )')
271 plt.legend()
272 plt.xticks(rotation=45)
273 plt.grid()
274 plt.show()
275
276 # Added code to test the trained model with healthy data on to
        another unhealthy turbine -----------
277
278
279
280 def preprocess_new_data(file_paths, normal_ranges, scaler):
281     # Read and concatenate all files
282     new_dataframes = [pd.read_csv(file_path, skiprows=9, encoding='
            ISO-8859-1') for file_path in file_paths]
283     new_data = pd.concat(new_dataframes, ignore_index=True)
284
285     new_data = new_data[['# Date and time', 'Nacelle temperature (
              C  )', 'Power (kW)', 'Rotor speed (RPM)',
286                          'Stator temperature 1 (  C  )', 'Generator
                             bearing rear temperature (  C  )', '
```

```
                                Wind speed (m/s)']]

    # Drop missing values
    new_data = new_data.dropna()

    # Filter based on the normal ranges and other conditions
    for col, (min_val, max_val) in normal_ranges.items():
        new_data = new_data[new_data[col].between(min_val, max_val)]
    new_data = new_data[new_data['Power (kW)'] > 0]
    new_data = new_data[new_data['Wind speed (m/s)'] > 4]  # Assuming cut-in wind speed as 4 m/s

    # Extract date and time before scaling
    time_data = new_data['# Date and time']

    # Extract actual generator bearing rear temperatures before scaling
    new_actuals = new_data['Generator bearing rear temperature (C  )'].values

    new_data = new_data[input_features]

    # Standardize the data
    new_data = scaler.transform(new_data)

    return new_data, time_data, new_actuals

new_data, new_time_data, new_actuals = preprocess_new_data(new_data_file_path, normal_ranges, scaler)

# Predict on new data
new_predictions = model.predict(new_data)

# Calculate metrics and plot results as before
r2_new = r2_score(new_actuals, new_predictions)
rmse_new = np.sqrt(mean_squared_error(new_actuals, new_predictions))
mae_new = mean_absolute_error(new_actuals, new_predictions)
mape_new = np.mean(np.abs((new_actuals - new_predictions) / new_actuals)) * 100

print('Test on New Data:\nR2:', r2_new, 'RMSE:', rmse_new, 'MAE:', mae_new, 'MAPE:', mape_new)

new_deviations = new_actuals - new_predictions

UCL_new = new_deviations.mean() + 3*new_deviations.std()
LCL_new = new_deviations.mean() - 3*new_deviations.std()

# Prepare data for plotting
plotting_new_data = pd.DataFrame({'Date': new_time_data,
```

```
                                        'Actual': new_actuals, 'Predicted'
                                            : new_predictions})
plotting_new_data['Date'] = pd.to_datetime(plotting_new_data['Date'
    ])
plotting_new_data.sort_values('Date', inplace=True)

# Plot: Actual vs Predicted
plt.figure(figsize=(10, 5))
plt.plot(plotting_new_data['Date'], plotting_new_data['Actual'],
    label='Actual', color='red')
plt.plot(plotting_new_data['Date'], plotting_new_data['Predicted'],
    label='Predicted', color='green')
plt.title('New Data: Actual vs Predicted')
plt.xlabel('Date and Time')
plt.ylabel('Generator Bearing Rear Temperature')
plt.legend()
plt.grid()
plt.show()


# Calculate Deviations
plotting_new_data['Deviations'] = plotting_new_data['Actual'] -
    plotting_new_data['Predicted']

# Calculate standard deviation for the Deviations
std_dev = plotting_new_data['Deviations'].std()

# Set UCL and LCL (usually set at mean    3*std_dev for control
    charts)
plotting_new_data['UCL'] = plotting_new_data['Deviations'].mean() +
    3*std_dev
plotting_new_data['LCL'] = plotting_new_data['Deviations'].mean() -
    3*std_dev

# Plot 2: Control Chart
plt.figure(figsize=(10, 5))
plt.plot(plotting_new_data['Date'], plotting_new_data['Deviations'],
     label='Deviations', color='blue')
plt.axhline(y=plotting_new_data['UCL'].values[0], label='UCL', color
    ='red')  # UCL as straight line
plt.axhline(y=plotting_new_data['LCL'].values[0], label='LCL', color
    ='yellow')  # LCL as straight line
plt.title('Generator Bearing Temperature Deviations')
plt.xlabel('Date')
plt.ylabel('Deviations ( C )')
plt.legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()

# User defined input for percentage of top deviations
```

```
top_deviation_percentage = 1

# Filtering out the data points which surpass the UCL
over_UCL_data = plotting_data[plotting_data['Deviations'] > UCL]

# Calculate the number of data points to select
num_top_points = int(len(over_UCL_data) * top_deviation_percentage / 100)

# Get the top N% data points with highest absolute deviations
top_deviations = over_UCL_data.nlargest(num_top_points, 'Deviations', keep='all')

# Print the date and time for these points
print("For wind turbine:", str(u))
print(top_deviations[['Date']])
```

A.2. LSTM Script

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras.optimizers import Adam
from keras.losses import MeanSquaredError
from keras.activations import relu
from keras.callbacks import EarlyStopping
#Remove Matplotlib Warnings/outdated interpretter
import warnings
import matplotlib.cbook
warnings.filterwarnings("ignore",category=matplotlib.cbook.mplDeprecation)

# Define wind turbine to be analyzed
i = 6

# Define your file paths here
file_paths = [
    'Turbine_Data_Kelmarsh_'+str(i)+'_2021-01-01_-_2021-07-01_228.csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2020-01-01_-_2021-01-01_228.csv',
```

```
    'Turbine_Data_Kelmarsh_'+str(i)+'_2019-01-01_-_2020-01-01_228.csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2018-01-01_-_2019-01-01_228.csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2017-01-01_-_2018-01-01_228.csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2016-01-03_-_2017-01-01_228.csv'
]


# Define  normal ranges here, based on knowledge on the WT
normal_ranges = {
    'Nacelle temperature (  C  )': (-10, 50),
    'Power (kW)': (0, 2000),
    'Rotor speed (RPM)': (0, 73),
    'Stator temperature 1 (  C  )': (0, 150),
    'Wind speed (m/s)': (0, 24),
    'Generator bearing rear temperature (  C  )': (0, 100),
}
# Concatenate data from all files
dataframes = [pd.read_csv(file_path, skiprows=9, encoding='ISO-8859-1') for file_path in file_paths]
data = pd.concat(dataframes, ignore_index=True)

data = data[['# Date and time', 'Nacelle temperature (  C  )', 'Power (kW)', 'Rotor speed (RPM)', 'Stator temperature 1 (  C  )', 'Generator bearing rear temperature (  C  )', 'Wind speed (m/s)']]

# Extract wind speed and power data
wind_speed = data['Wind speed (m/s)']
power = data['Power (kW)']

# Print initial number of data points
print("Number of data points before cleaning: ", len(data))


# Plot Power curve
plt.figure(figsize=(10, 5))
plt.scatter(wind_speed, power, s=5, color='blue')
plt.title('Wind Turbine Power Curve')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Power (kW)')
plt.grid()
plt.show()

# Drop missing values
data = data.dropna()

# Filter based on the normal ranges and other conditions
```

```
for col, (min_val, max_val) in normal_ranges.items():
    data = data[data[col].between(min_val, max_val)]
data = data[data['Power (kW)'] > 0]
data = data[data['Wind speed (m/s)'] > 3]  # Assuming cut-in wind speed as 3 m/s

time_data = data['# Date and time']

# Extract the input and output columns from data
input_features = ['Nacelle temperature (  C  )', 'Power (kW)', 'Rotor speed (RPM)',
                  'Stator temperature 1 (  C  )', 'Wind speed (m/s)']
output_feature = 'Generator bearing rear temperature (  C  )'

# Prepare the data for train_test_split
X = data[input_features]
y = data[output_feature]

# Split data into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Combine X_train and y_train back into training data
data_train = pd.concat([X_train, y_train], axis=1)

# Drop missing values in training data
data_train = data_train.dropna()

# Filter based on the normal ranges and other conditions in training data
for col, (min_val, max_val) in normal_ranges.items():
    data_train = data_train[data_train[col].between(min_val, max_val)]
data_train = data_train[data_train['Power (kW)'] > 0]
data_train = data_train[data_train['Wind speed (m/s)'] > 3]  # Assuming cut-in wind speed as 3 m/s

# Extract wind speed and power data again after cleaning
wind_speed_train = data_train['Wind speed (m/s)']
power_train = data_train['Power (kW)']

# Print number of data points after cleaning
print("Number of data points after cleaning: ", len(data_train))

# Plot after cleaning
plt.figure(figsize=(10, 5))
plt.scatter(wind_speed_train, power_train, s=5, color='blue')
plt.title('Wind Turbine Power Curve - After Cleaning')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Power (kW)')
```

```
plt.grid()
plt.show()

# Re-assign cleaned X_train and y_train
X_train = data_train[input_features]
y_train = data_train[output_feature]

# Extract date and time for the test set before scaling
date_test = time_data[X_test.index]

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Reshape the data for LSTM (Samples, Time steps, Features)
X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

# Define the LSTM model
model = Sequential()
model.add(LSTM(50, activation='relu', return_sequences=True,
    input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(50, activation='relu', return_sequences=True))
model.add(LSTM(25, activation='relu'))
model.add(Dense(1))

# Compile the model
model.compile(optimizer=Adam(), loss=MeanSquaredError())
# Apply Early Stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1,
    patience=10)

# Train the model. Epochs, batch size  and similar metrics can be
    augmented to refine model performace. These values have been
    choosen as a way to reduce computional complexity.
history = model.fit(X_train, y_train, validation_split=0.2, epochs
    =5, batch_size=5, verbose=1, callbacks=[es])

# Perform prediction and obtain performance metrics
y_pred_train = model.predict(X_train).flatten()
y_pred_test = model.predict(X_test).flatten()

r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)

rmse_train = np.sqrt(mean_squared_error(y_train, y_pred_train))
rmse_test = np.sqrt(mean_squared_error(y_test, y_pred_test))

mae_train = mean_absolute_error(y_train, y_pred_train)
mae_test = mean_absolute_error(y_test, y_pred_test)
```

```
159
160 mape_train = np.mean(np.abs((y_train - y_pred_train) / y_train)) *
        100
161 mape_test = np.mean(np.abs((y_test - y_pred_test) / y_test)) * 100
162
163 print('Train Metrics:\nR2:', r2_train, 'RMSE:', rmse_train, 'MAE:',
        mae_train, 'MAPE:', mape_train)
164 print('Test Metrics:\nR2:', r2_test, 'RMSE:', rmse_test, 'MAE:',
        mae_test, 'MAPE:', mape_test)
165
166 deviations = y_test - y_pred_test
167
168 UCL = deviations.mean() + 3*deviations.std()
169 LCL = deviations.mean() - 3*deviations.std()
170
171 # Prepare data for plotting
172 plotting_data = pd.DataFrame({'Date': date_test,
173                               'Actual': y_test, 'Predicted':
                                      y_pred_test})
174 plotting_data['Date'] = pd.to_datetime(plotting_data['Date'])
175 plotting_data.sort_values('Date', inplace=True)
176
177 # Plot 1: Actual vs Predicted
178 plt.figure(figsize=(10, 5))
179 plt.plot(plotting_data['Date'], plotting_data['Actual'], label='
        Actual', color='red')
180 plt.plot(plotting_data['Date'], plotting_data['Predicted'], label='
        Predicted', color='green')
181 plt.title('Generator Temperature vs Prediction Result')
182 plt.xlabel('Date')
183 plt.ylabel('Generator Bearing Temperature ( C )')
184 plt.legend()
185 plt.xticks(rotation=45)
186 plt.grid()
187 plt.show()
188
189 # Calculate Deviations
190 plotting_data['Deviations'] = plotting_data['Actual'] -
        plotting_data['Predicted']
191
192 # Calculate standard deviation for the Deviations
193 std_dev = plotting_data['Deviations'].std()
194
195 # Set UCL and LCL (usually set at mean    3*std_dev )
196 plotting_data['UCL'] = plotting_data['Deviations'].mean() + 3*
        std_dev
197 plotting_data['LCL'] = plotting_data['Deviations'].mean() - 3*
        std_dev
198
199 # Plot 2: Control Chart
200 plt.figure(figsize=(10, 5))
```

```
plt.plot(plotting_data['Date'], plotting_data['Deviations'], label='Deviations', color='blue')
plt.axhline(y=plotting_data['UCL'].values[0], label='UCL', color='red')  # UCL as straight line
plt.axhline(y=plotting_data['LCL'].values[0], label='LCL', color='yellow')  # LCL as straight line
plt.title('Generator Bearing Temperature Deviations')
plt.xlabel('Date')
plt.ylabel('Deviations ( C )')
plt.legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()
```

A.3. XGBoost Script

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras.optimizers import Adam
from keras.losses import MeanSquaredError
from keras.activations import relu
from keras.callbacks import EarlyStopping
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
#Remove Matplotlib Warnings/outdated interpretter
import warnings
import matplotlib.cbook
warnings.filterwarnings("ignore",category=matplotlib.cbook.mplDeprecation)

# Define wind turbine to be analyzed
i = 6

# Define your file paths here
file_paths = [
    'Turbine_Data_Kelmarsh_'+str(i)+'_2021-01-01_-_2021-07-01_228.csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2020-01-01_-_2021-01-01_228.csv',
```

```
    'Turbine_Data_Kelmarsh_'+str(i)+'_2019-01-01_-_2020-01-01_228.csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2018-01-01_-_2019-01-01_228.csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2017-01-01_-_2018-01-01_228.csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2016-01-03_-_2017-01-01_228.csv'
]


# Define normal ranges for SCADA parameters based on knowledge
normal_ranges = {
    'Nacelle temperature (  C  )': (-10, 50),
    'Power (kW)': (0, 2000),
    'Rotor speed (RPM)': (0, 73),
    'Stator temperature 1 (  C  )': (0, 150),
    'Wind speed (m/s)': (0, 24),
    'Generator bearing rear temperature (  C  )': (0, 100),
}
#Look up temeprature vestas

# Concatenate data from all files
dataframes = [pd.read_csv(file_path, skiprows=9, encoding='ISO-8859-1') for file_path in file_paths]
data = pd.concat(dataframes, ignore_index=True)

data = data[['# Date and time', 'Nacelle temperature (  C  )', 'Power (kW)', 'Rotor speed (RPM)', 'Stator temperature 1 (  C  )', 'Generator bearing rear temperature (  C  )', 'Wind speed (m/s)']]

# Extract wind speed and power data
wind_speed = data['Wind speed (m/s)']
power = data['Power (kW)']

# Initial number of data points
print("Number of data points before cleaning: ", len(data))

# Plot Power curve before data cleaning
plt.figure(figsize=(10, 5))
plt.scatter(wind_speed, power, s=5, color='blue')
plt.title('Wind Turbine Power Curve')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Power (kW)')
plt.grid()
plt.show()

# Drop missing values
data = data.dropna()

```

```
# Filter based on the normal ranges and additional conditions such as cut-in wind speed and negative power
for col, (min_val, max_val) in normal_ranges.items():
    data = data[data[col].between(min_val, max_val)]
data = data[data['Power (kW)'] > 0]
data = data[data['Wind speed (m/s)'] > 3]  # Assuming cut-in wind speed as 4 m/s

# Prepare data for train_test_split
input_features = ['Nacelle temperature (  C  )', 'Power (kW)', 'Rotor speed (RPM)',
                  'Stator temperature 1 (  C  )', 'Wind speed (m/s)']
output_feature = 'Generator bearing rear temperature (  C  )'
time_data = data['# Date and time']


# Extract wind speed and power data again after cleaning
wind_speed = data['Wind speed (m/s)']
power = data['Power (kW)']

# Print number of data points after cleaning
print("Number of data points after cleaning: ", len(data))

# Plot Power curve after cleaning
plt.figure(figsize=(10, 5))
plt.scatter(wind_speed, power, s=5, color='blue')
plt.title('Wind Turbine Power Curve - After Cleaning')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Power (kW)')
plt.grid()
plt.show()

#New script Manhalobis
from scipy.spatial import distance
from sklearn.cluster import KMeans

# Prepare data for outlier detection
X_clustering = data[input_features]

# K-means clustering to divide the dataset into smaller groups
kmeans = KMeans(n_clusters=3, random_state=0).fit(X_clustering)
labels = kmeans.labels_
data['Cluster'] = labels

# Calculate Mahalanobis Distance for each observation in each cluster and mark outliers
dist_threshold = {}  # To store the threshold value for each cluster
for cluster in set(labels):
    cluster_data = data[data['Cluster'] == cluster][input_features]
    cov_matrix = np.cov(cluster_data, rowvar=False)
```

```
    inv_cov_matrix = np.linalg.inv(cov_matrix)
    mean_values = cluster_data.mean().values
    mdist_values = []
    for index, row in cluster_data.iterrows():
        mdist_values.append(distance.mahalanobis(row, mean_values,
            inv_cov_matrix))
    mdist_values = np.array(mdist_values)
    # Setting a threshold value of the distance MD to consider about
        10-15% of the points as anomalous
    dist_threshold[cluster] = np.percentile(mdist_values, 90)
    data.loc[data['Cluster'] == cluster, 'MahalanobisDist'] =
        mdist_values

# Filter outliers based on calculated threshold for each cluster
for cluster in set(labels):
    data = data[~((data['Cluster'] == cluster) & (data['
        MahalanobisDist'] > dist_threshold[cluster]))]

#End new code--------

# Prepare data for train_test_split
X = data[input_features]
y = data[output_feature]

# Split data into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.3, random_state=42)
# Extract date and time for the test set before scaling
date_test = time_data[X_test.index]
# Standardize data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define  XGBoost model
model = xgb.XGBRegressor()

# Define the hyperparameter grid. In order to activate search for
    optmial hypermarameter of the XGBoost model, this section should
     be commented out
#param_grid = {
#    'learning_rate': [0.01, 0.1, 0.2, 0.3],
#    'max_depth': [3, 5, 7, 10],
#    'n_estimators': [100, 200, 500, 1000],
#    'min_child_weight': [1, 3, 5],
#    'subsample': [0.5, 0.7, 1.0],
#    'colsample_bytree': [0.5, 0.7, 1.0],
#}
#Once found define directly optimal hyperparameters. In case search
    wants to be performed for optimal hyperparameters, this section
    should be commented and previous one commented out
```

```
param_grid = {
    'learning_rate': [0.01],
    'max_depth': [7],
    'n_estimators': [1000],
    'min_child_weight': [1],
    'subsample': [0.5],
    'colsample_bytree': [1.0],
}

# Perform Grid Search CV
gs_cv = GridSearchCV(model, param_grid, n_jobs=-1, cv=5, verbose=2)
gs_cv.fit(X_train, y_train)

# Get best parameters
best_params = gs_cv.best_params_
print("Best parameters: ", best_params)

# Define the model with the best parameters
model = xgb.XGBRegressor(**best_params)

# Train the model
model.fit(X_train, y_train)

# Perform prediction
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

#Performace metrics

r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)

rmse_train = np.sqrt(mean_squared_error(y_train, y_pred_train))
rmse_test = np.sqrt(mean_squared_error(y_test, y_pred_test))

mae_train = mean_absolute_error(y_train, y_pred_train)
mae_test = mean_absolute_error(y_test, y_pred_test)

mape_train = np.mean(np.abs((y_train - y_pred_train) / y_train)) *
    100
mape_test = np.mean(np.abs((y_test - y_pred_test) / y_test)) * 100

print('Train Metrics:\nR2:', r2_train, 'RMSE:', rmse_train, 'MAE:',
    mae_train, 'MAPE:', mape_train)
print('Test Metrics:\nR2:', r2_test, 'RMSE:', rmse_test, 'MAE:',
    mae_test, 'MAPE:', mape_test)

deviations = y_test - y_pred_test

UCL = deviations.mean() + 3*deviations.std()
LCL = deviations.mean() - 3*deviations.std()
```

```
207
208 # Prepare data for plotting
209 plotting_data = pd.DataFrame({'Date': date_test,
210                               'Actual': y_test, 'Predicted':
                                     y_pred_test})
211 plotting_data['Date'] = pd.to_datetime(plotting_data['Date'])
212 plotting_data.sort_values('Date', inplace=True)
213
214 # Plot 1: Actual vs Predicted
215 plt.figure(figsize=(10, 5))
216 plt.plot(plotting_data['Date'], plotting_data['Actual'], label='
    Actual', color='red')
217 plt.plot(plotting_data['Date'], plotting_data['Predicted'], label='
    Predicted', color='green')
218 plt.title('Generator Temperature vs Prediction Result')
219 plt.xlabel('Date')
220 plt.ylabel('Generator Bearing Temperature ( C )')
221 plt.legend()
222 plt.xticks(rotation=45)
223 plt.grid()
224 plt.show()
225
226
227 # Calculate Deviations
228 plotting_data['Deviations'] = plotting_data['Actual'] -
    plotting_data['Predicted']
229
230 # Calculate standard deviation for the Deviations
231 std_dev = plotting_data['Deviations'].std()
232
233 # Set UCL and LCL (usually set at mean    3*std_dev for control
    charts)
234 plotting_data['UCL'] = plotting_data['Deviations'].mean() + 3*
    std_dev
235 plotting_data['LCL'] = plotting_data['Deviations'].mean() - 3*
    std_dev
236
237 # Plot 2: Control Chart
238 plt.figure(figsize=(10, 5))
239 plt.plot(plotting_data['Date'], plotting_data['Deviations'], label='
    Deviations', color='blue')
240 plt.axhline(y=plotting_data['UCL'].values[0], label='UCL', color='
    red')  # UCL as straight line
241 plt.axhline(y=plotting_data['LCL'].values[0], label='LCL', color='
    yellow')  # LCL as straight line
242 plt.title('Generator Bearing Temperature Deviations')
243 plt.xlabel('Date')
244 plt.ylabel('Deviations ( C )')
245 plt.legend()
246 plt.xticks(rotation=45)
247 plt.grid()
```

```
plt.show()
```

A.4. Deep Neural Network Script

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error,
    mean_absolute_error
from keras.models import Sequential
from keras.layers import Dense, Dropout
#Remove Matplotlib Warnings/outdated interpretter
import warnings
import matplotlib.cbook
warnings.filterwarnings("ignore",category=matplotlib.cbook.
    mplDeprecation)

# Define wind turbine to be analyzed
i = 6

# Define your file paths here
file_paths = [
    'Turbine_Data_Kelmarsh_'+str(i)+'_2021-01-01_-_2021-07-01_228.
        csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2020-01-01_-_2021-01-01_228.
        csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2019-01-01_-_2020-01-01_228.
        csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2018-01-01_-_2019-01-01_228.
        csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2017-01-01_-_2018-01-01_228.
        csv',
    'Turbine_Data_Kelmarsh_'+str(i)+'_2016-01-03_-_2017-01-01_228.
        csv'
]


# Define normal ranges
normal_ranges = {
    'Nacelle temperature (  C  )': (-10, 50),
    'Power (kW)': (0, 2000),
    'Rotor speed (RPM)': (0, 73),
    'Stator temperature 1 (  C  )': (0, 150),
    'Wind speed (m/s)': (0, 24),
    'Generator bearing rear temperature (  C  )': (0, 100),
```

```
}

# Concatenate data from all files, change enconding and row count
    according to format of your  csv file
dataframes = [pd.read_csv(file_path, skiprows=9, encoding='ISO
    -8859-1') for file_path in file_paths]
data = pd.concat(dataframes, ignore_index=True)

data = data[['# Date and time', 'Nacelle temperature (  C  )', '
    Power (kW)', 'Rotor speed (RPM)', 'Stator temperature 1 (  C  )'
    , 'Generator bearing rear temperature (  C  )', 'Wind speed (m/s
    )']]

# Extract wind speed and power data
wind_speed = data['Wind speed (m/s)']
power = data['Power (kW)']

# Print initial number of data points
print("Number of data points before cleaning: ", len(data))

# Plot
plt.figure(figsize=(10, 5))
plt.scatter(wind_speed, power, s=5, color='blue')
plt.title('Wind Turbine Power Curve')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Power (kW)')
plt.grid()
plt.show()

# Drop missing values
data = data.dropna()

# Filter based on the normal ranges and other conditions
for col, (min_val, max_val) in normal_ranges.items():
    data = data[data[col].between(min_val, max_val)]
data = data[data['Power (kW)'] > 0]
data = data[data['Wind speed (m/s)'] > 3]  # Assuming cut-in wind
    speed as 3 m/s

# Extract wind speed and power data again after cleaning
wind_speed = data['Wind speed (m/s)']
power = data['Power (kW)']

# Print number of data points after cleaning
print("Number of data points after cleaning: ", len(data))

# Plot after cleaning
plt.figure(figsize=(10, 5))
plt.scatter(wind_speed, power, s=5, color='blue')
plt.title('Wind Turbine Power Curve - After Cleaning')
plt.xlabel('Wind Speed (m/s)')
```

```
plt.ylabel('Power (kW)')
plt.grid()
plt.show()

#Prepare data for train_test_split
input_features = ['Nacelle temperature (  \ C )', 'Power (kW)', '
    Rotor speed (RPM)',
                  'Stator temperature 1 (  C  )', 'Wind speed (m/s)'
                     ]
output_feature = 'Generator bearing rear temperature (  C  )'
time_data = data['# Date and time']

# Prepare data for train_test_split
X = data[input_features]
y = data[output_feature]

# Split the data into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.3, random_state=42)

# Extract date and time for the test set before scaling
date_test = time_data[X_test.index]

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Define the deep neural network model. Adjust neuron density and
    activation function. These values were choosen based on relevant
     literature and hardware constrains.
model = Sequential()
model.add(Dense(512, input_dim=X_train.shape[1], activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1))

# Compile  model
model.compile(loss='mean_squared_error', optimizer='adam')

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=64, verbose=1)
# Perform prediction
y_pred_train = model.predict(X_train).flatten()
y_pred_test = model.predict(X_test).flatten()

r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)

```

```
rmse_train = np.sqrt(mean_squared_error(y_train, y_pred_train))
rmse_test = np.sqrt(mean_squared_error(y_test, y_pred_test))

mae_train = mean_absolute_error(y_train, y_pred_train)
mae_test = mean_absolute_error(y_test, y_pred_test)

mape_train = np.mean(np.abs((y_train - y_pred_train) / y_train)) * 100
mape_test = np.mean(np.abs((y_test - y_pred_test) / y_test)) * 100

print('Train Metrics:\nR2:', r2_train, 'RMSE:', rmse_train, 'MAE:', mae_train, 'MAPE:', mape_train)
print('Test Metrics:\nR2:', r2_test, 'RMSE:', rmse_test, 'MAE:', mae_test, 'MAPE:', mape_test)

deviations = y_test - y_pred_test

UCL = deviations.mean() + 3*deviations.std()
LCL = deviations.mean() - 3*deviations.std()

# Prepare data for plotting
plotting_data = pd.DataFrame({'Date': date_test,
                              'Actual': y_test, 'Predicted':
                                 y_pred_test})
plotting_data['Date'] = pd.to_datetime(plotting_data['Date'])
plotting_data.sort_values('Date', inplace=True)

# Plot 1: Actual vs Predicted
plt.figure(figsize=(10, 5))
plt.plot(plotting_data['Date'], plotting_data['Actual'], label='Actual', color='red')
plt.plot(plotting_data['Date'], plotting_data['Predicted'], label='Predicted', color='green')
plt.title('Generator Temperature vs Prediction Result')
plt.xlabel('Date')
plt.ylabel('Generator Bearing Temperature ( C )')
plt.legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()

# Calculate Deviations
plotting_data['Deviations'] = plotting_data['Actual'] - plotting_data['Predicted']

# Calculate standard deviation for the Deviations
std_dev = plotting_data['Deviations'].std()

# Set UCL and LCL (usually set at mean    3*std_dev for control charts)
```

```
plotting_data['UCL'] = plotting_data['Deviations'].mean() + 3*
    std_dev
plotting_data['LCL'] = plotting_data['Deviations'].mean() - 3*
    std_dev

# Plot 2: Control Chart
plt.figure(figsize=(10, 5))
plt.plot(plotting_data['Date'], plotting_data['Deviations'], label='
    Deviations', color='blue')
plt.axhline(y=plotting_data['UCL'].values[0], label='UCL', color='
    red')  # UCL as straight line
plt.axhline(y=plotting_data['LCL'].values[0], label='LCL', color='
    yellow')  # LCL as straight line
plt.title('Generator Bearing Temperature Deviations')
plt.xlabel('Date')
plt.ylabel('Deviations ( C )')
plt.legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()
```