UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE INGENIERÍA Y DE ARQUITECTURA

MÁSTER UNIVERSITARIO EN INGENIERÍA AERONÁUTICA

TRABAJO FIN DE MÁSTER

PREDICTING CO AND NO$_x$ EMISSIONS FROM GAS TURBINES

Author: Laura MEIZOSO GARCíA
Tutor: Alan DOMÍNGUEZ MONTERO

January 2024

to obtain the degree of Master of Science in Aeronautical Engineering

at the Universidad Europea de Madrid

This study focuses on predicting aircraft emissions, specifically targeting carbon monoxide (CO) and nitrogen oxides (NO_x), originating from gas turbine engines. This prediction is based on a comprehensive flue gas emissions dataset, which incorporates environmental and process variables spanning a duration of five years.

Motivated by global concerns about environmental degradation due to escalating energy demands, this research delves into delineating the current state of emissions generated by gas turbines within the aeronautical industry, understanding the combustion process's influence on shaping emissions, exploring data analysis methodologies applicable in aviation and implementing predictive models for emissions estimation. Emphasis is placed on Predictive Emission Monitoring Systems (PEMS) due to their widespread use in emissions monitoring.

Exploratory Data Analysis (EDA) reveals intricate relationships between ambient variables and gas turbine parameters, emphasizing substantial correlations with emissions. Robust predictive models, integrating feature engineering techniques like outlier treatment and variable selection, significantly enhance emissions prediction, recognizing the need for filtering due to extensive data dispersion.

Normalized selected variables, including ambient temperature and humidity, turbine inlet temperature and compressor discharge pressure, are utilized for CO and NO_x emissions prediction. Employing GRU and 1D CNN models demonstrates 1D CNN's superior performance in predicting pollutants, contrary to conventional recommendations, with GRU outperforming in predicting NO_x compared to CO.

To summarize, this research aims at predicting emissions while advocating for responsible technological innovation to foster a more sustainable future in aviation. As aptly expressed by Albert Einstein: "We cannot solve our problems with the same thinking we used when we created them".

Este estudio se centra en predecir emisiones de aeronaves, específicamente el monóxido de carbono (CO) y los óxidos de nitrógeno (NO_x), originadas en motores de turbinas de gas. La predicción se basa en un conjunto de datos exhaustivo de emisiones de gases de combustión, que abarca variables ambientales y de proceso durante cinco años.

Motivada por las preocupaciones globales sobre la degradación ambiental debido al aumento de la demanda energética, esta investigación explora el estado actual de las emisiones generadas por turbinas de gas en la industria aeronáutica, entendiendo la influencia del proceso de combustión en la formación de emisiones, analizando metodologías de análisis de datos aplicables en aviación e implementando modelos predictivos para estimar emisiones. Se destaca el uso de Sistemas de Monitoreo de Emisiones Predictivas (PEMS) debido a su amplio uso en el monitoreo de emisiones.

El Análisis Exploratorio de Datos (EDA) revela relaciones complejas entre variables ambientales y parámetros de turbinas de gas, resaltando correlaciones significativas con las emisiones. Modelos predictivos sólidos, que incorporan técnicas de feature engineering como tratamiento de outliers y selección de variables, mejoran notablemente la predicción de emisiones, reconociendo la necesidad de filtrado debido a la amplia dispersión de datos.

Se utilizan variables seleccionadas normalizadas, como temperatura y humedad ambiental, temperatura de entrada de la turbina y presión de descarga del compresor, para predecir las emisiones de CO y NO_x. El uso de modelos GRU y CNN 1D muestra el rendimiento superior de CNN 1D en la predicción de contaminantes, en contraposición a las recomendaciones convencionales, con GRU destacando en la predicción de NO_x en comparación con el CO.

Resumiendo, este estudio busca predecir emisiones mientras promueve la innovación tecnológica responsable para un futuro más sostenible en la aviación. Como dijo Albert Einstein: "No podemos resolver nuestros problemas con el mismo pensamiento que usamos cuando los creamos".

Acknowledgements

I would like to express my deepest gratitude to my Thesis supervisor, Alan, whose invaluable guidance and expertise were instrumental in shaping this research.

Heartfelt thanks are due to my parents for their unwavering love, encouragement and understanding during this academic endeavor. Their continuous support and belief in my abilities have been the cornerstone of my success.

I am profoundly grateful for Jose, for his patience, encouragement and unwavering belief in me. His constant motivation and support were instrumental in navigating through the challenges faced during this journey.

I am grateful to all those who supported and contributed in various ways, directly or indirectly, to the completion of this work. Your encouragement and assistance have been invaluable. Thank you.

Laura Meizoso García
Madrid, January 2024

Contents

List of Figures

List of Tables

Acronyms

ACF	Autocorrelation
ADAM	Adaptive Moment Estimation
ADS-B	Automatic Dependent Surveillance–Broadcast
ADF	Augmented Dickey Fuller
AFDP	Air Filter Difference Pressure
AFR	Air-Fuel Ratio
AH	Ambient Humidity
AI	Artificial Intelligence
AIAA	American Institute of Aeronautics and Astronautics
AP	Ambient Pressure
ARIMA	Autoregressive integrated moving average
AT	Ambient Temperature
ATM	Air Traffic Management
CAEP	Committee on Aviation and Environmental Protection
CCA	Canonical Correlation Analysis
CDP	Compressor Discharge Pressure
CEMS	Continuous Emission Monitoring Systems
CNN	Convolutional Neural Network
CO	Carbon monoxide
CO_2	Carbon dioxide
CORSIA	Carbon Offsetting and Reduction Scheme for International Aviation
CRN	Chemical Reactor Network
CVRs	Cockpit Voice Recorders
DCS	Distributed Control System
EASA	European Aviation Safety Agency
EDA	Exploratory Data Analysis
EHM	Engine Health Monitoring
ELM	Extreme Learning Machine
EPA	Environmental Protection Agency
ETS	Emissions Trading System

FDRs	Flight Data Recorders
GRU	Gated Recurrent Unit
GTEP	Gas Turbine Exhaust Pressure
HC	Hydrocarbons
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
ICCT	International Council on Clean Transportation
IIR	Infinite Impulse Response
IQR	Interquartile Range
KNN	K-Nearest Neighbors
KPSS	Kwiatkowski-Phillips-Schmidt-Shin
LB	Lean Burn
LOWESS	Locally Weighted Scatterplot Smoothing
LSTM	Long Short-Term Memory
LTAG	Long-term global aspirational goal
LTO	Landing and Take-Off
MAE	Mean Absolute Error
MSE	Mean Squared Error
NN	Neural Network
NO	Nitric oxide
NO_x	Nitrogen oxides ($NO + NO_2$)
nvPM	Non-volatile particulate matter
O_3	Ozone
PACF	Partial Autocorrelation
PEMS	Predictive Emission Monitoring Systems
PM	Particulate Matter
PP	Philips Perron
RMS	Root Mean Square
RPK	Revenue passenger kilometer
RQL	Rich-burn, Quick-quench, Lean-burn
RNN	Recurrent Neural Network
SAF	Sustainable Aviation Fuel
SARPs	Standards and Recommended Practices
SES	Single European Sky

SFC	Specific fuel consumption
SHM	Structural Health Monitoring
SN	Smoke number
TAT	Turbine After Temperature
TEY	Turbine Energy Yield
TIT	Turbine Inlet Temperature
UHC	Unburned hydrocarbons
UNFCCC	United Nations Framework Convention on Climate Change
UV	Ultraviolet

1 Introduction

The aim of this Thesis is to provide comprehensive insights into the prediction of aircraft emissions through the application of data analysis methods.

This research is motivated by the pressing global challenge posed by the escalating demand for energy, which has considerably impacted the environment in recent years. This surge in energy demand has led to growing concerns regarding environmental issues, notably the growth in deforestation rates and the concurrent increase in carbon and flue gas emissions.

The aviation industry's impact on the environment is significant due to pollutant emissions, with carbon monoxide (CO) and nitrogen oxides (NO_x) being some of the key aircraft pollutants. This comprehensive review delves into the current state of understanding of aviation emissions, recent efforts to mitigate their effects and ongoing projects that specifically target the reduction of CO and NO_x emissions.

Moreover, considering the criticality of monitoring nitrogen oxides and carbon monoxide pollutants during combustion processes within power plants, the development of effective monitoring solutions has emerged as imperative. In response, two primary solutions have been developed to address this need: Continuous Emission Monitoring Systems (CEMS) and Predictive Emission Monitoring Systems (PEMS).

In this study, particular attention will be given to the exploration of PEMS due to its widespread adoption and relevance in environmental monitoring. PEMS has gained significant traction in the realm of emissions monitoring for its capability to estimate and predict pollutants by utilizing historical data and process variables. This method's prominence arises from its practicality in real-time assessment, making it a prevalent choice across various industrial sectors, including aviation. By relying on statistical models and expert systems, PEMS not only offers cost-effective solutions but also presents an efficient mean to anticipate and regulate emissions, especially in situations where continuous monitoring systems might be impractical or resource-intensive. Through an in-depth analysis and examination of the PEMS framework, this research aims to unravel its effectiveness and limitations, providing a nuanced understanding of its role in the context of aviation emissions and its potential for facilitating more sustainable environmental practices within the industry.

1.1. Motivation

Undertaking a Thesis focused on predicting pollutant levels from an aircraft gas turbine entails multifaceted motivations within the realm of aviation and environmental stewardship. Such a Thesis aligns with the imperative to minimize the ecological footprint of aviation, a critical consideration given the industry's significant impact on air quality. Research in this domain not only contributes to advancing predictive models but also holds promise in enabling proactive measures to mitigate environmental pollution caused by aircraft emissions.

The aviation sector's growing emphasis on sustainability drives the urgency for accurate prediction models that anticipate and monitor pollutant emissions from gas turbine engines. A Thesis concentrating on forecasting contaminant levels from these engines aims to enhance the understanding of emission patterns and their subsequent impact on the atmosphere. This endeavor not only augments academic knowledge but also serves as a crucial step in devising strategies for emission reduction and regulatory compliance, aligning with the aviation industry's commitment to eco-friendly operations.

Figure 1.1: Aircraft engine emitting soot [1]

Furthermore, the motivation behind such a Thesis stems from the complex interplay between technological advancements and environmental responsibility. Exploring predictive modeling techniques in this context empowers researchers to delve into cutting-edge methodologies, incorporating elements of machine learning, data analytics and aerospace engineering altogether. The combination of these disciplines fosters innovation in predictive modeling, offering potential solutions to monitor, manage and optimize the performance of gas turbine engines in a manner that minimizes environmental impact.

Moreover, a Thesis centered on predicting pollutant levels from aircraft gas turbines presents an opportunity to address critical challenges concerning air quality and emissions control. The pursuit of accurate predictive models enhances the understanding of emission dynamics, aiding in the development of more efficient and eco-conscious aviation systems. This research avenue underscores the fusion of technological innovation with environmental consciousness, paving the way for advancements that promote sustainable aviation practices and reduce the ecological footprint of air travel.

1.2. Scope

The scope of this project is the prediction of CO and NO_x pollutant emissions through the analysis of a novel flue gas emission dataset. Thus, the following steps will be followed:

1. The initial chapter 2 predominantly focuses on delineating the current state of the art pertaining to emissions generated by gas turbines within the aeronautical industry. Emphasis is placed on elucidating how the combustion process intricately influences the emitted pollutants, highlighting the pivotal role played by combustion in shaping emissions. The discussion encompasses an in-depth analysis of CO and NO_x emissions, focusing on their environmental impact, and scrutinizes the pertinent regulations associated with aircraft emissions, providing a holistic understanding of the regulatory landscape concerning pollutant emissions from aviation. Subsequently, attention is directed towards explaining ongoing projects and research endeavors aimed at mitigating emissions from aircraft engines.

2. Additionally, this Thesis delves into the realm of data analysis and its relevance within the aeronautical industry. A concise yet insightful overview is presented, emphasizing the utility and applicability of data analysis methodologies in facilitating informed decision-making and mitigating environmental impacts in aviation. Special focus is directed towards CEMS and PEMS, elucidating their significance as predictive tools in estimating and regulating pollutant emissions from gas turbine engines, offering a comparative analysis of their methodologies, advantages and limitations. This chapter 3 also provides

a mathematical explanation of the different analytical models and methods that will be employed later in the emissions data analysis.

3. Moving forward, a subsequent chapter 4 outlines the core problem addressed in this research: implementing a PEMS model to predict pollutant emissions. The approach is based on a comprehensive dataset encompassing environmental variables, process variables and pollutant data collected over a period of five years from a gas turbine. Thus, this chapter meticulously presents an Exploratory Data Analysis (EDA) of the flue gas emission dataset.

4. The following chapter 5 proceeds to detail the process of feature engineering and the implementation of two distinct models: Gated Recurrent Unit (GRU) networks and one-dimensional Convolutional Neural Networks (CNNs). The comprehensive process involves a thorough examination of the models used and their application in predicting emissions, marking a significant stride toward comprehending and predicting pollutant emissions from gas turbines.

5. Finally, the culmination of this Thesis encapsulates the concluding chapter 6, presenting overarching conclusions drawn from the research findings and suggesting potential avenues for future research and development. This final segment offers reflections on the efficacy of the implemented models, the implications of the findings and outlines prospective directions for advancing the predictive capabilities of PEMS models in addressing gas turbine emissions.

2 Combustion and pollutant generation

This chapter delves into the intricate relationship between combustion processes in aircraft engines and the resultant emissions, focusing notably on carbon monoxide (CO) and nitrogen oxides (NO_x). It explores the evolving regulatory landscape governing aviation emissions and outlines future initiatives aimed at innovating technologies and practices to substantially reduce emissions from aircraft engines, fostering a greener aviation industry.

2.1. Aircraft pollutant emissions

When fast jet planes appeared around 80 years ago they made it easier for people to travel, but they also caused loud noise and air pollution issues. Nowadays, several regulations aim to control these aircraft emissions, making newer aircraft versions better for the environment. Nevertheless, as time progresses, there are more flying aircraft, so the efforts to reduce noise and pollution remain important until these days.

Initially, jet engines, despite their inefficiency, replaced propellers due to their capacity for high-speed thrust with low weight. Efficiency was then not an environmental concern, with fuel cost being the sole incentive for enhancements. But now, with the recognition of carbon dioxide (CO_2) from fossil fuel combustion as the primary driver of climate change, there's increasing societal pressure to reduce aviation's environmental impact. Consequently, the adverse environmental impacts of widespread air travel are under greater scrutiny than ever before [2].

The scientific literature encompasses numerous studies related to this field, with the consensus being that aircraft emissions yield significant volatile and non-volatile species [3]:

- Non-volatile emissions are produced in the combustor and consist of refractory materials like soot present in the particulate phase at high temperatures near the engine exit.
- Volatile emissions are created from substances existing as gaseous and vapour-phase pollutants, such as CO_2, CO, NO_x, SO_2, O_3 and many organic compounds, in the hot exhaust.

Aircraft using kerosene-type fuels primarily generate carbon dioxide and water, which are directly linked to the burned fuel. Variations in the carbon-to-hydrogen ratio of the fuel account for minor deviations in emissions [4].

The fuel flow in common aircraft engines is nearly directly proportional to engine thrust settings. High-temperature oxidation in engine combustors prompts the formation of nitrogen oxides from atmospheric nitrogen [5], while trace elements in fuels and non-ideal combustion conditions contribute to the formation of additional by-products such as sulfur oxides, unburned hydrocarbons and particulate soot. Aircraft exhaust can also contain species from burned lubricant oils and mechanical component wear.

Therefore, the real (simplified) combustion scheme in aircraft engines is the following [6]:

$$C_nH_m + N_2 + O_2 + S \rightarrow CO_2 + N_2 + H_2O + O_2 + CO + SO_x + NO_x + HC + soot$$

Reports estimate that a vast majority of commercial engine exhaust (approximately 99.5 to 99.9%) comprises nitrogen, oxygen, carbon dioxide and water vapor [5]. A detailed breakdown illustrates that CO_2 and water vapor constitute the major portion of aircraft exhaust, with the remainder accounting for less than 1% of aircraft emissions. The bulk of emissions (about 90%) occur during cruising at altitude, with the remainder emitted during takeoff, landing and ground-level operations, impacting the environment and human health.

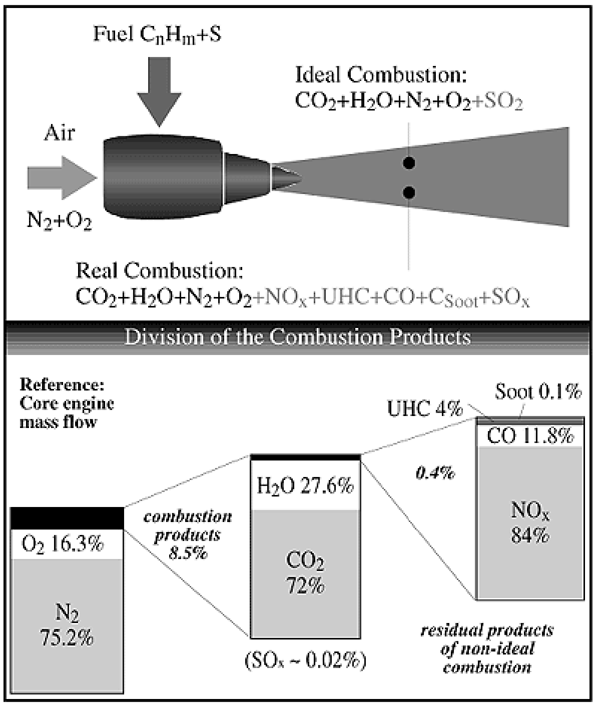


Figure 2.1: Decomposition of aircraft emissions [7]

The volume of exhaust emissions depends on combustor temperature, pressure, fuel-to-air ratio, atomization and air mixture [4]. On the other side, variability in pollutant amounts is observed based on engine technology, model and, especially, thrust, while hydrocarbon emissions are subject to engine type, use, maintenance history and fuel composition, highlighting the complexities in emission determination and their environmental impact.

Poor air quality captures public attention due to its established link to various air pollutants and their detrimental impact on both short-term and long-term human health. Additionally, air pollution not only hampers visibility but also causes damage to buildings, cultural heritage and exerts direct and indirect influences on the climate. While developing nations face heightened air pollution concerns because of population expansion, increased energy needs and economic growth, developed countries have notably reduced the concentrations of several air pollutants in the past decade [8].

The rapid increase in air travel as well as its expected growth to meet future capacity demands has made that aircraft emissions gain substantial attention in the last decades. Emission standards for new aircraft engines have been established by the International Civil Aviation Organization (ICAO) since the late 1970s. These standards, found in Volume II of Annex 16 to the Chicago Convention, outline protocols for measuring carbon monoxide, nitrogen oxides, unburned hydrocarbons (UHC) and smoke number (SN) for new engines [9].

While these regulations have significantly improved engine and fuel efficiency, the projected growth in the aviation industry and subsequent rise in airport traffic may counteract the achieved reductions in aircraft emissions over the last two decades [10].

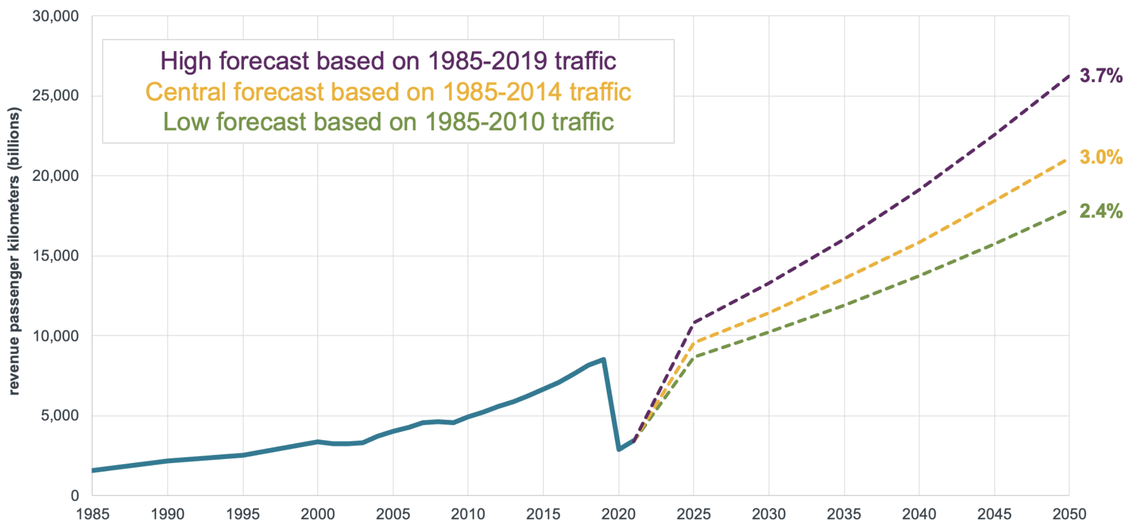


Figure 2.2: Long-term passenger traffic forecast, 2019-2050 [11]

As previously highlighted, the aviation industry plays a pivotal role in global transportation, yet it significantly contributes to pollutant emissions. Therefore, the initial emphasis will be on understanding the combustion process in gas turbines and how pollutants are generated within them. Subsequently, the focus will shift to two key pollutants: carbon monoxide and nitrogen oxides. Their examination will encompass the current comprehension of these pollutants, while later sections will focus on recent measures implemented to mitigate their emissions and potential future strategies for their further reduction.

2.1.1. Combustion and generation of pollutants in gas turbines

The emphasis of this work is focused on the generation of pollutants in gas turbines, for which an exploration of the underlying physics involved in their formation is required, meaning an explanation of the combustion process in gas turbines is needed.

The combustion process in a gas turbine involves the controlled burning of a fuel-air mixture within the combustion chamber. This process is essential for producing the high-temperature, high-pressure gases that power the turbine.

This process consists of the following steps:

1. Fuel injection: the fuel, which can be natural gas, kerosene or liquid/gaseous fuels, is injected into the combustion chamber.

2. Mixing with air: the fuel mixes with a precisely controlled amount of compressed air.

3. Ignition and combustion: once the fuel-air mixture is ignited, combustion occurs, releasing a significant amount of thermal energy. This rapid combustion generates extremely high temperatures (up to thousands of degrees Celsius) and increases the pressure of the gases.

4. Expansion: the hot gases produced expand rapidly, pushing against turbine blades and causing them to spin.

The combustor receives high-pressure air at considerable velocities, meticulously undergoing controlled deceleration to minimize pressure losses before the fuel introduction phase. Inside the combustion chamber, designed intricately, there's enough space and time for the efficient mixing of air and fuel, ensuring a thorough combustion process before reaching the turbine stages. The critical design of the combustor dictates intricate combustion processes, influencing chemical reaction completeness and the engine's emission nature and scale. As a result, the combustor significantly shapes an aircraft's environmental impact on the climate [7].

To meet the diverse operational requirements of aircraft engines, combustors must perform efficiently under a wide range of pressures and temperatures. This includes [7]:

- Reigniting at high altitudes after unexpected shutdowns in cold and low-pressure conditions.
- Sustaining stable combustion across various air velocities and fuel-to-air ratios to prevent flameout during engine deceleration.
- At elevated pressures and temperatures, these combustors need to burn fuel effectively to ensure consistent smooth temperature profiles and protect turbine components from damage, thereby extending their durability.

However, reconciling these varied demands presents a notable engineering challenge due to the conflicting solutions needed at different operational extremes.

Future combustors are anticipated to encounter more demanding requirements as manufacturers strive to improve fuel efficiency. Although higher cycle pressure ratios enhance engine fuel efficiency, the increase in compressor delivery and combustor inlet air temperatures reduce air cooling capacity. Consequently, a greater proportion of total airflow is needed to cool the hottest engine parts, limiting airflow available for primary combustion and dilution. This reduction complicates the regulation of turbine inlet temperature profiles and emission levels [7]. Addressing these challenges needs of continuous advancements in cooling techniques, devices, materials or a combination thereof to resolve these issues.

Now, the focus will be directed towards how the combustion process in gas turbines influences the formation of pollutants.

The combustion chamber in gas turbine engines is subject to a set of requirements delineated in the figure below. These criteria operate independently and have evolved over time through design improvements, aiming to enhance them while also considering their impact on emissions.

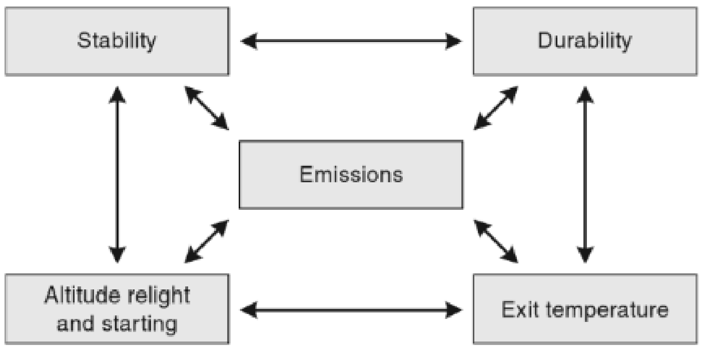


Figure 2.3: Combustor performance requirements [12]

Gas turbine engines operate based on the Brayton cycle, and although an ideal cycle involves isentropic compression, constant pressure heat addition and isentropic expansion through the turbine, real engines experience performance losses leading to stagnation pressure loss in the combustor. Nevertheless, the conversion efficiency of fuel chemical energy into thermal energy remains notably high, typically exceeding 99.9% under most power conditions [12].

Under optimal conditions, combustion of kerosene-type fuels primarily results in the production of carbon dioxide and water vapor (H_2O), the proportions of which vary based on the specific fuel's carbon-to-hydrogen ratio. During cruise conditions, combustion products constitute only about 8.5% of the total mass flow exiting the engine. Among these products, a small fraction (approximately 0.4%) arises from non-ideal combustion processes (soot, HC and CO), as well as nitrogen oxidation (NO_x) [7].

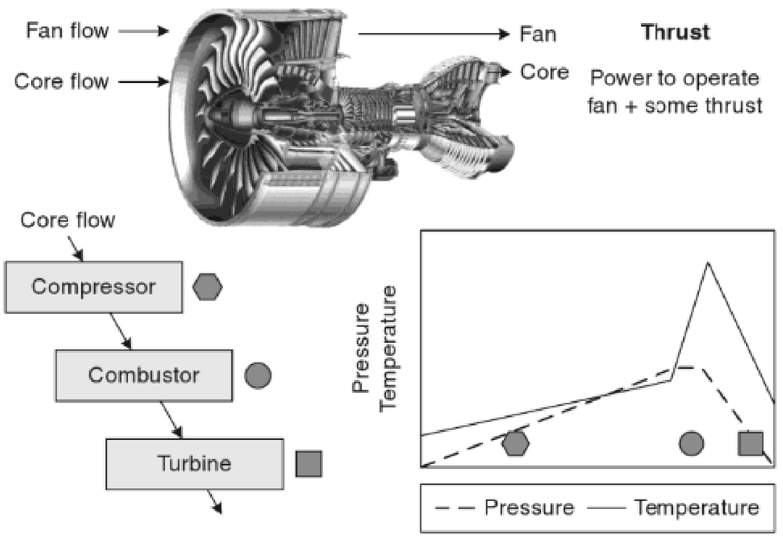


Figure 2.4: Gas turbine components characteristics [12]

The primary pollutants emitted by engines include carbon monoxide, unburned hydrocarbons, nitrogen oxides and Particulate Matter (PM or smoke). At lower power settings, inlet combustor pressure and temperature are relatively low, leading to reduced reaction rates for kerosene-type fuels. Here, liquid fuel undergoes atomization, evaporation and combustion, necessitating sufficient residence time at elevated temperatures for fuel conversion into CO_2. In instances where the flow permits fuel vapor to exit the combustor without complete reaction or if prematurely exposed to lower temperature due to mixing with colder airstreams, incomplete or quenched reactions result in CO production. Conversely, at higher power conditions characterized by elevated air pressures and temperatures, rapid reactions lead to near-zero CO and UHC emissions while NO_x and PM emissions become more pronounced [12].

NO_x emissions, particularly thermal NO_x, result from fast reactions described by the extended Zeldovich mechanism:

$$O_2 \rightarrow 2O$$

$$N_2 + O \rightarrow NO + N$$

$$N + O_2 \rightarrow NO + O$$

$$N + OH \rightarrow NO + H$$

Techniques to reduce thermal NO_x involve limiting the time spent by the flow at high temperatures or controlling maximum flame temperatures through stoichiometric adjustments. Fuel-rich regions at high pressures and temperatures in the combustor can generate small carbon particles through complex chemical processes, undergoing multiple transformations such as surface growth, agglomeration and oxidation before exiting the engine as PM or soot [12]. The term particulate matter encompasses these emissions, which include invisible soot particles and aerosol soot precursors in modern engines.

The relationship between engine power conditions and emission production reveals that UHC and CO levels are highest at low power and decline significantly with increased thrust. Conversely, NO_x and PM emissions escalate with engine power, usually peaking at maximum power conditions [12].

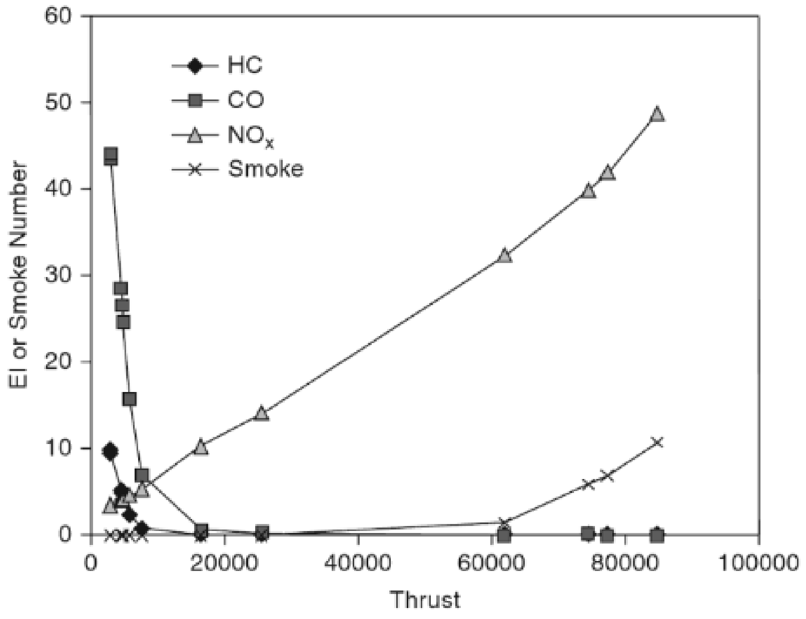


Figure 2.5: Emissions vs. power level for PW4084 [12]

Thus, it can be stated that pollutants are generated as follows [7] [12] [13]:

- CO_2, H_2O and SO_xO are directly linked to engine fuel consumption. In contrast, emissions like NO_x, CO, HC and soot significantly fluctuate due to diverse variables, especially engine power settings and ambient inlet conditions.
- Incomplete combustion leads to elevated CO and HC levels, particularly at low power settings with inefficient fuel atomization and mixing processes.
- Carbon dioxide is generated as a product of complete combustion when carbon in the fuel combines with oxygen.
- Carbon monoxide is formed when there's incomplete combustion due to insufficient oxygen. Inadequate mixing or insufficient time for complete combustion can lead to CO formation.
- Nitrogen oxides are formed due to the high temperatures in the combustion chamber, where nitrogen in the air combines with oxygen. At these high temperatures, nitrogen and oxygen react to form NO_x compounds. The majority of NO_x emissions originate in the highest-temperature zones of the combustor, primarily the primary combustion area, before dilution. The processes behind NO_x formation are dependent on local combustion temperature, pressure and duration, closely associated with the combustor inlet conditions.
- Depending on the fuel composition and combustion conditions, solid particles can be formed from incomplete combustion or from impurities in the fuel. Soot, comprising mainly carbonaceous material resulting from incomplete combustion, is primarily formed in the fuel-rich primary zone before oxidation in higher-temperature areas.

Understanding and controlling the formation of these pollutants is crucial for reducing emissions in gas turbines. Advanced combustion technologies, such as staged combustion and lean-burn systems, are developed to mitigate pollutant formation by optimizing fuel-air mixing and combustion conditions.

The evolution of combustion technology aimed to manage regulated pollutants alongside safety and operational imperatives. With the increasing relevance of non-volatile particulate matter (nvPM) emissions, combustor design must now consider both NO_x and nvPM emissions while optimizing fuel efficiency. This must occur within the constraints of safety and operability, encompassing factors such as altitude relight, turbine inlet temperature, combustion efficiency and thermal load, that shape primary design decisions [14].

Currently, the predominant environmental concern revolves around nvPM and NO_x emissions. Recent advancements in combustion design have aimed to minimize their impact, notably through technologies such as Rich-burn, Quick-quench, Lean-burn (RQL), and Lean Burn (LB).

RQL technologies employ specific combustion chamber designs. In RQL chambers, the fuel undergoes atomization, forming a swirling flow, and maintains a fuel-rich primary flame zone. The quick dilution of this zone into a lean mixture helps restrict NO_x formation by swiftly passing through the high-temperature area [14]. Consequently, RQL designs have notably reduced NO_x emissions compared to earlier chambers.

nvPM typically originates in the primary zone near the fuel spray, where incomplete fuel-air mixing occurs. To counter this, intermediate zones introduce a small amount of air to lower gas temperatures, facilitating the complete oxidation of CO and soot particles. However, managing the oxidation of soot particles in RQL combustors, while simultaneously controlling NO_x, presents challenges due to the coexistence of high NO_x formation and soot particle oxidation zones [14].

LB combustors, characterized by lower fuel-to-air ratios than RQL types, result in decreased peak temperatures and subsequent lower NO_x emissions. Higher excess air leads to reduced nvPM production. Yet, maintaining stability with hydrocarbon fuels under low fuel-air ratios necessitates a pilot zone for stability during specific operational conditions. Despite potential pollutant generation in the pilot zone, downstream areas in lean-burn regions facilitate particle burnout, minimizing nvPM levels [14].

In the subsequent sections, a greater emphasis will be placed on carbon monoxide and nitrogen oxides emissions, as they will be the pollutants under study in this Thesis.

2.1.2. Carbon monoxide (CO) emissions

Atmospheric carbon monoxide primarily results from the photochemical oxidation of methane, non-methane hydrocarbons and direct emissions through various human activities like vehicular exhaust, heating, industrial processes, and biomass burning.

In the troposphere, CO has a chemical lifetime of 30 to 90 days, mainly reduced by oxidation via hydroxyl radicals [15].

It contributes to air quality degradation and poses health risks, particularly in densely populated areas. Its binding with hemoglobin to form carboxyhemoglobin can have adverse effects on human health, causing reduced oxygen-carrying capacity in the blood. While high exposure leads to asphyxia, even low levels may affect neuropsychological performance and pose risks for those with cardiovascular diseases [16].

CO is primarily generated within aircraft engines due to incomplete jet fuel combustion, with it being regulated by international standards. Over the last four decades, advancements in engine technology have notably reduced CO emissions during the LTO (Landing and Take-Off) cycle, with most new engines emitting less than 10kg of CO per complete LTO cycle [8].

CO emissions are highest at low power settings due to lower combustor temperatures and pressures [17], representing challenges in air quality around airports during idle and taxi phases. Chemical transformations of emitted CO in the troposphere result in the formation of carbon dioxide, nitrogen dioxide and ozone.

Even though this topic will be further presented in the next sections, it is important to highlight that aircraft currently being produced are already complying with environmental directives whose application deadline is still far away in time. As an example, it can be observed from the figure below that since 2016 aircraft are already complying with ICAO's CO_2 standard, which takes effect in 2028.

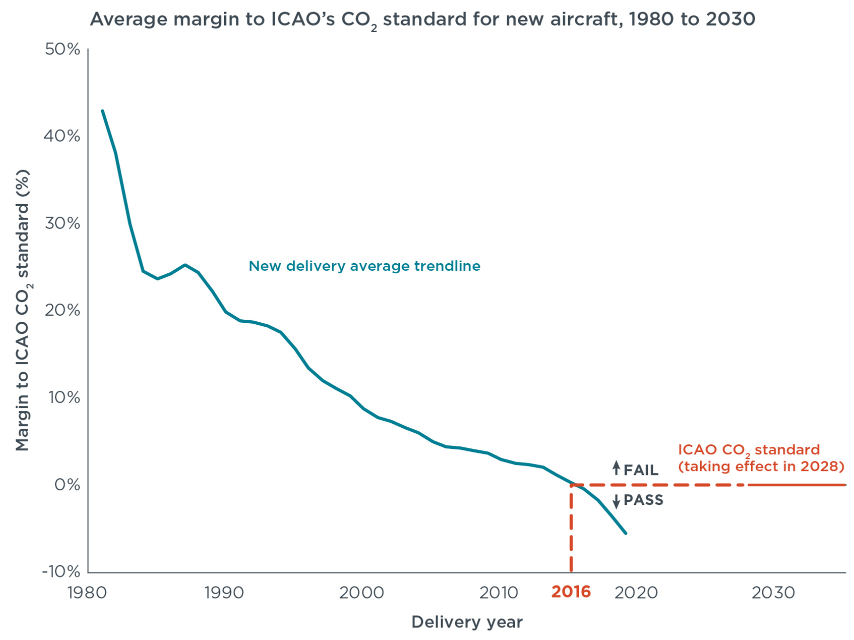


Figure 2.6: Average margin to ICAO's CO_2 standard for new aircraft, 1980 to 2030 [18]

2.1.3. Nitrogen oxides (NO_x) emissions

In urban environments, nitrogen oxides, composed of nitric oxide (NO) and nitrogen dioxide (NO_2), are predominantly produced from fossil fuel combustion.

The extended Zeldovich mechanism, which has been presented in the previous section, describes the rapid reactions involving nitrogen and oxygen to form nitrogen oxides. The reaction involved in forming NO_2 is rapid, with NO reacting with ambient ozone (O_3) or radicals:

$$NO + O_3 \rightarrow NO_2 + O_2$$

These compounds have significant implications for atmospheric chemistry and human health, contributing to respiratory ailments and increased mortality rates. Particularly, NO_2, known as a strong respiratory irritant, has been extensively studied and is associated with cardiovascular and respiratory diseases.

In the context of aircraft engines, the high-temperature zones within combustors play a fundamental role in the formation of nitrogen oxides. Their formation is highly sensitive to parameters such as combustor pressure, temperature, flow rate and engine geometry [17]. Additionally, the thermal oxidation of atmospheric nitrogen in the combustor results in the primary production of NO_x.

Studies have shown that, unlike many other forms of combustion, modern high-bypass turbofan engines primarily emit NO_2 at idle, while NO dominates at high power regimes [19]. They have revealed varying proportions of NO_2 and NO_x ratios based on different engine settings and aircraft types, indicating the complexity of emissions characterization [20].

The intricacies of aircraft emissions, especially in relation to NO_x, demonstrate the challenges in attributing and quantifying their contributions to ambient air quality.

NO_x emissions are major contributors to ozone depletion and smog formation:

- Ozone depletion: it leads to increased levels of ultraviolet (UV) radiation reaching the Earth's surface, which can result in higher rates of skin cancer, cataracts and weakened immune systems in humans. Additionally, it can harm plant life, disrupting ecosystems and impacting agriculture. The ozone layer's depletion also affects marine organisms and can potentially cause damage to materials like plastics and rubber. Overall, it poses significant risks to both human health and the environment.

- Smog: it is a type of air pollution that results from the interaction of pollutants in the atmosphere, often a mix of smoke and fog. It typically forms in urban areas with high levels of vehicle emissions, industrial processes and other combustion activities. The consequences of smog include respiratory problems, such as aggravated asthma, bronchitis and other lung diseases. It can also cause eye irritation and various cardiovascular issues. Long-term exposure to smog is linked to serious health conditions and can even lead to premature death. Additionally, smog can harm vegetation, corrode buildings, and have detrimental effects on the overall environment.

Before presenting the current existing regulation related to aircraft emissions, typical emissions values for engine operating regimes will presented in the table below, as they will be useful later:

Species	Idle	Take-off	Cruise
CO_2	3160	3160	3160
H_2O	1230	1230	1230
CO	25 (10-65)	1	1-3.5
HC	4 (0-12)	0.5	0.2-1.3
NO_x (short haul)	4.5 (3-6)	32 (20-65)	7.9-11.9
NO_x (long haul)	4.5 (3-6)	27 (10-53)	11.1-15.4
SO_xO	1.0	1.0	1.0

Table 2.1: Typical emission index [g/kg] levels for engine operating regimes [7]

2.2. Associated regulation and current policies

This section delves into the intricate landscape of current aviation policies and standards, primarily focusing on mitigating the environmental impacts of aircraft emissions, particularly nitrogen oxides, non-volatile particulate matter, carbon dioxide and other pollutants influencing both air quality and climate change.

As already mentioned, regulation is becoming stricter regarding aircraft emissions, due to the multiple impacts they have on people's health. Furthermore, greater awareness in society means that the measures are also increasingly restrictive, growing exponentially in recent years.

Previous to the COVID-19 pandemic, global aviation (including domestic and international; passenger and freight) accounted for [21]:

- 1.9% of greenhouse gas emissions (which includes all greenhouse gases, not only CO_2).
- 2.5% of CO_2 emissions.

As of 2022, emissions levels close to 80% of those previous to the COVID-19 pandemic had already been reached [22]. Despite doubling since the mid-1980s, aviation's emissions have paralleled the overall CO_2 emission growth, maintaining a relatively stable share of global emissions, hovering between 2% to 2.5% [21].

Over the past 50 years, aviation has seen significant advancements in efficiency, which result in a slower growth in aircraft emissions. In 2018, approximately 125 grams of CO_2 were emitted per revenue passenger kilometer (RPK), marking a considerable improvement from the eleven-fold higher rate in 1960 and twenty-fold higher in 1950. These enhancements stem from aircraft design and technology improvements, larger aircraft sizes accommodating more passengers and the rise in passenger load factor, which increased from 61% in 1950 to 82% in 2018 [21]. This factor measures the actual distance traveled by paying customers in comparison to the distance if every flight operated at full capacity, highlighting the strides made in maximizing airline efficiency.

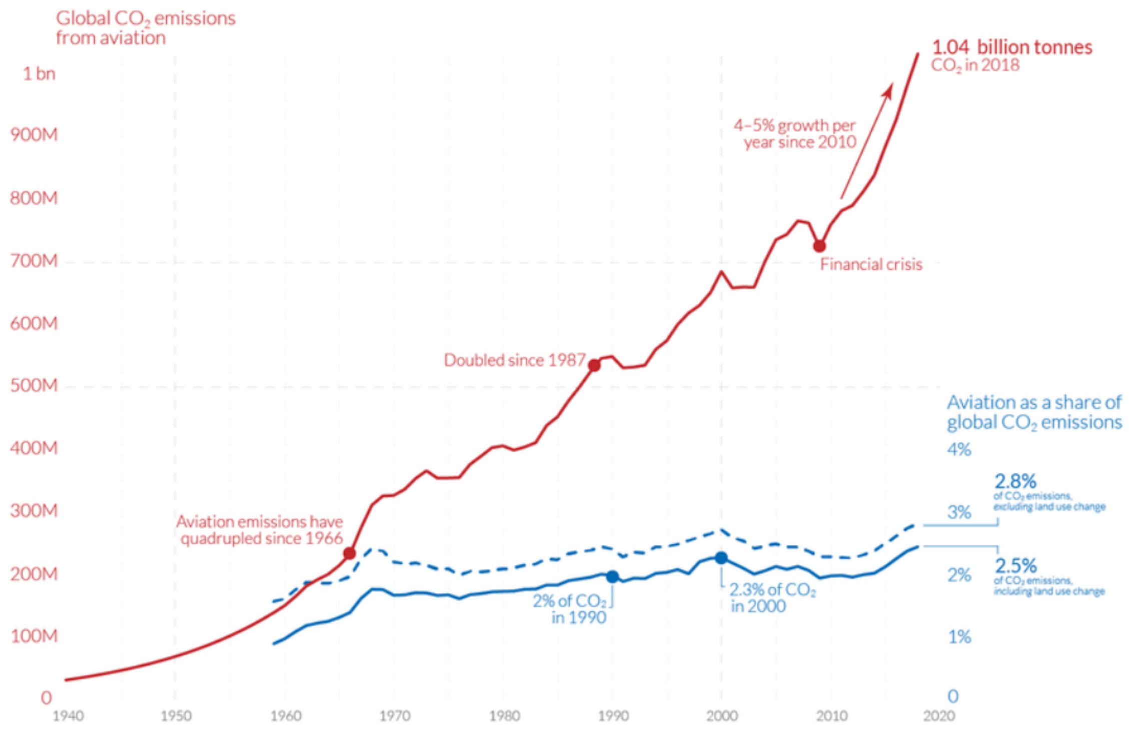


Figure 2.7: Share of global emissions calculated based on total CO_2 data from the Global Carbon Project [21]

As aircraft production rates increase, and considering their significantly long lifespan of approximately 40 years, the gradual rise in the number of aircraft in service occurs over time. This implies that to maintain or decrease the aviation industry's contribution to global pollution, there's a growing necessity for increasingly efficient aircraft.

Early in the aviation era, the primary focus was on minimizing operational costs, predominantly centered around the price of fuel. The introduction of fast jets helped reduce operational time while consuming less fuel due to their lighter weight. However, initially, factors such as efficiency, noise or pollution weren't part of the equations to enhance aircraft versions.

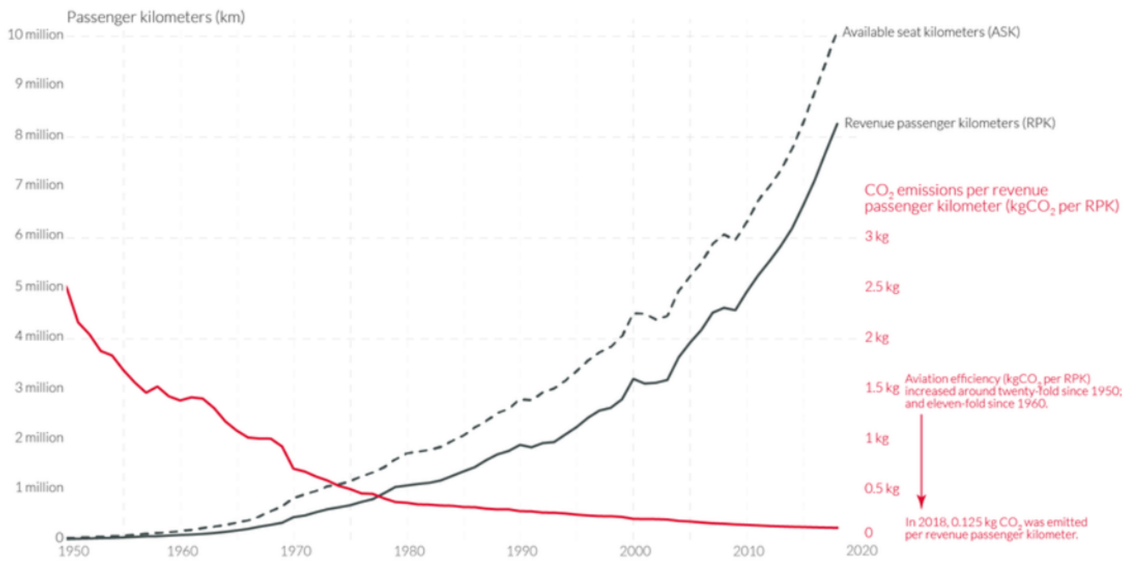


Figure 2.8: Aviation efficiency calculated based on global aircraft traffic data from the ICAO via airlines.org [21]

Between 1965 and 1975, low-smoke combustors were introduced in early commercial jet engines, reducing visible smoke trails from aircraft. Later, higher bypass ratio engines in the late 1960s and early 1970s notably improved fuel efficiency, reducing CO_2 and water vapor emissions. These engines also emitted lower levels of pollutants like HC and CO, especially at low power (idle) settings, due to enhanced fuel/air mixing. Additionally, improved combustor designs minimized take-off smoke. From 1975 to 1985, advancements in combustor design further decreased emissions of HC and CO by enhancing fuel atomization and employing staged fuel injection at idle. The ongoing shift toward higher bypass engines aims to enhance fuel efficiency, responding to both commercial demands and environmental concerns, particularly regarding CO_2 impact [7].

The current policies related to aircraft emissions concentrate on diverse aspects. These policies encompass a multifaceted approach aimed at addressing various factors influencing emissions from aviation. They include measures targeting specific emission types such as CO_2, NO_x and other pollutants originating from aircraft operations.

The policies are designed to regulate and mitigate these emissions, considering their significant impacts on air quality, climate change and atmospheric composition. Moreover, these policies often involve standards, market-based mechanisms and changes in air traffic management procedures to effectively manage and reduce the environmental footprint of aviation activities. Overall, the current policies associated with aircraft emissions reflect a comprehensive strategy aimed at tackling the complex issue of emissions generated by aviation.

These efforts collectively underscore the aviation industry's commitment to advancing technologies to reduce emissions, enhance air quality and mitigate environmental impacts. The pursuit of these goals is driven by a combination of regulatory pressure and commercial incentives, which continually push for innovations that align with evolving environmental standards and global objectives for sustainable aviation.

There are various climate impacts arising from aviation emissions, including CO_2, NO_x, water vapor, SO_2, soot particles, contrails, cirrus formation, aerosol-cloud interactions, O_3 formation and alterations in CH_4 lifetime in the atmosphere. Among these, CO_2, NO_x emissions, contrails and cirrus formation are recognized as having the most significant radiative forcing effects, though uncertainties persist regarding aerosol-cloud interactions' magnitude and effects [23].

Multiple interdependencies among these impacts exist, where changes aiming to mitigate one or more may lead to synergies or trade-offs. For instance, avoiding ice-supersaturated air to reduce contrails and cirrus formation might increase fuel consumption, subsequently elevating CO_2 emissions. Similarly, efforts to lower NO_x emissions might prompt the development of engines consuming more fuel and subsequently emitting more CO_2 [23].

Short-term trade-offs and synergies involve constant technology, while long-term considerations factor in technological advancements. For instance, introducing standards for aircraft cruise-NO_x emissions may not create immediate trade-offs, but over the long term, as future engines reduce NO_x at the expense of higher fuel consumption and CO_2 emissions, potential conflicts might arise. Similarly, implementing charges on specific emissions or fuel components could impact emissions differently in the short and long term based on evolving technology and industry practices.

This study emphasizes the need to consider overall climate impacts when formulating policies, recognizing that certain impacts (like CO_2) remain independent of specific technologies or engine types. The aim is to strike a balance between reducing aviation's climate impacts while also addressing potential trade-offs and synergies between various emissions and operational considerations.

The currently enforced measures focus on the following aspects: [23]:

- Technology-design standards: set by the ICAO environmental committee (CAEP) and enforced by Certification Authorities, the European Aviation Safety Agency (EASA) adheres to stringent certification standards for aircraft engine emissions. These standards, rooted in ICAO Annex 16 Volume II and Volume III, encompass various emissions such as NO_x, nvPM, CO, UHC and smoke. Their primary objective isn't to favor particular technologies but rather to elevate the overall environmental performance of aircraft globally over time by regulating and curbing emissions during the flight cycle.

- Operational regulatory instruments: despite several operational regulations under the Single European Sky (SES) that primarily focus on performance indicators related to fuel efficiency and CO_2 emissions, there's a notable absence of specific directives targeting non-CO_2 impacts like NO_x, nvPM and contrail-cirrus formation.

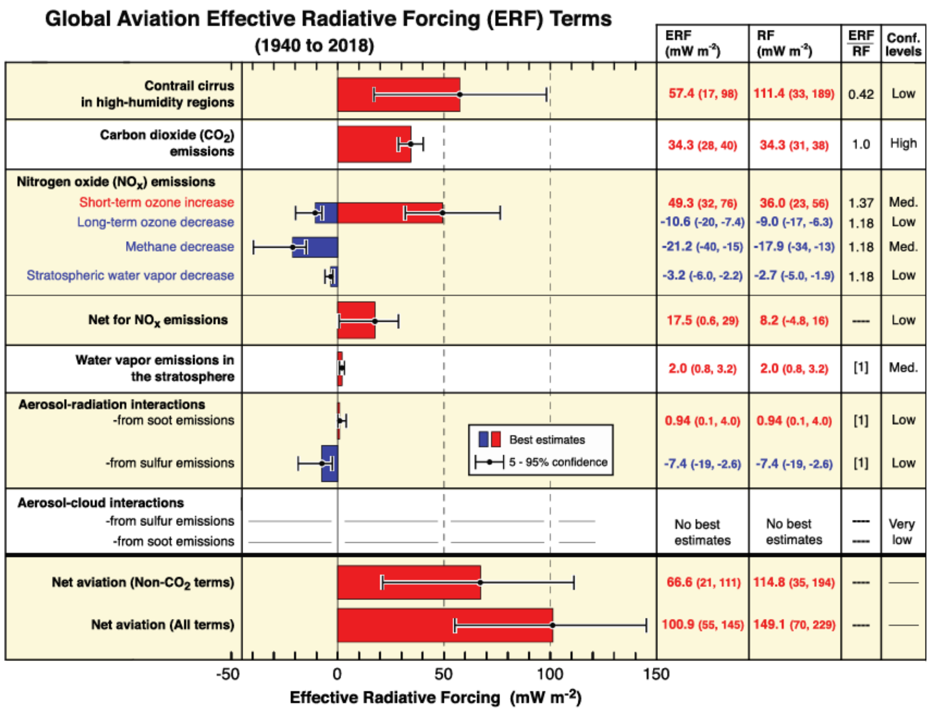
- Fuel standards: jet fuel standards, encompassing DEF STAN 91-91 and ASTM D1655, predominantly ensure compliance with safety and operational requirements. However, these standards might not explicitly align with environmental concerns, except for certain prescribed limits on chemical compositions, affecting emissions such as nvPM and vPM.

- Other policies: market-based measures, including the EU Emissions Trading System (ETS) and the ICAO Carbon Offsetting and Reduction Scheme for International Aviation (CORSIA), are fundamental strategies intended to curtail CO_2 emissions and incentivize the reduction of aviation-related greenhouse gases.

Despite progressive changes in standards, global NO_x emissions from the aviation fleet haven't significantly decreased due to various factors like increased use of engines producing more NO_x, fleet expansion and slow fleet turnover.

CO_2 standards, integrated into european legislation, mandate aeroplane-level certifications, which encompass various fuel efficiency technologies linked to aeroplane design, such as propulsion, aerodynamics and structural modifications, to effectively reduce emissions. The focus is on continually enhancing fuel efficiency to curtail CO_2 emissions over the coming years [23].

For nvPM emissions, new and more stringent standards have been instituted to control ultrafine particle emissions at the engine exit, aiming to enhance air quality and human health. The standards also facilitate a pathway for phasing out the older smoke number regulation [23].



	ERF (mW m^{-2})	RF (mW m^{-2})	ERF/RF	Conf. levels
Contrail cirrus in high-humidity regions	57.4 (17, 98)	111.4 (33, 189)	0.42	Low
Carbon dioxide (CO_2) emissions	34.3 (28, 40)	34.3 (31, 38)	1.0	High
Nitrogen oxide (NO_x) emissions				
Short-term ozone increase	49.3 (32, 76)	36.0 (23, 56)	1.37	Med.
Long-term ozone decrease	-10.6 (-20, -7.4)	-9.0 (-17, -6.3)	1.18	Low
Methane decrease	-21.2 (-40, -15)	-17.9 (-34, -13)	1.18	Med.
Stratospheric water vapor decrease	-3.2 (-6.0, -2.2)	-2.7 (-5.0, -1.9)	1.18	Low
Net for NO_x emissions	17.5 (0.6, 29)	8.2 (-4.8, 16)	----	Low
Water vapor emissions in the stratosphere	2.0 (0.8, 3.2)	2.0 (0.8, 3.2)	[1]	Med.
Aerosol-radiation interactions				
-from soot emissions	0.94 (0.1, 4.0)	0.94 (0.1, 4.0)	[1]	Low
-from sulfur emissions	-7.4 (-19, -2.6)	-7.4 (-19, -2.6)	[1]	Low
Aerosol-cloud interactions				
-from sulfur emissions	No best estimates	No best estimates	----	Very low
-from soot emissions			----	
Net aviation (Non-CO_2 terms)	66.6 (21, 111)	114.8 (35, 194)	----	——
Net aviation (All terms)	100.9 (55, 145)	149.1 (70, 229)	----	——

Figure 2.9: Best-estimates for climate forcing terms from global aviation from 1940 to 2018 [23]

The International Civil Aviation Organization (ICAO) has been dedicated to addressing aviation emissions' impact on local air quality since the late 1970s. They've developed a set of measures focused on curbing emissions from aircraft engines around airports and relevant airport sources. Annex 16 Volume II to the Convention on International Civil Aviation outlines standards for aircraft engine emissions, supported by guidance and technical documentation.

In their continuous efforts, ICAO recently achieved success with the adoption of the CAEP/10 nvPM Standard, based on visibility criteria. This was followed by CAEP/11's agreement on an nvPM mass and number Standard, which the ICAO Council adopted in March 2020, becoming effective from January 2021 [24].

The provisions by ICAO for local air quality encompass various areas, addressing liquid fuel venting, smoke (expected to be replaced by the nvPM Standard) and primary gaseous exhaust emissions such as hydrocarbons (HC), nitrogen oxides and carbon monoxide from jet engines [24]. ICAO not only aims to mitigate the negative impact of civil aviation on the global climate but also formulates policies, updates Standards and Recommended Practices (SARPs) and conducts outreach activities.

The 41st Session of the ICAO Assembly in 2022 saw the adoption of Resolution A41-21, a comprehensive statement consolidating ICAO policies and practices on environmental protection, notably on climate change. This resolution introduced the long-term global aspirational goal for international aviation of achieving net-zero carbon emissions by 2050, aligning with the Paris Agreement's temperature targets [25].

To attain these aspirational goals and foster sustainable growth in international aviation, ICAO is pursuing a range of measures, including advancements in aircraft technology, operational enhancements, sustainable aviation fuels and market-based mechanisms such as CORSIA (Carbon Offsetting and Reduction Scheme for International Aviation) [25]. These efforts collectively aim to steer international aviation toward a more environmentally sustainable future.

Further delving into CORSIA, the Carbon Offsetting and Reduction Scheme for International Aviation was globally adopted by governments in 2016, aiming to stabilize net CO_2 emissions from international aviation starting in 2021. It's been in effect since January 1st, 2019, requiring airlines to report their CO_2 emissions annually, with offsetting obligations kicking in from January 1st, 2021 [26].

It's important to mention that CORSIA isn't meant to replace efforts aimed at reducing carbon emissions through technological advancements, operational improvements and infrastructural changes in the aviation sector, but rather to complement them.

Originally, CORSIA's baseline was set to be an average of 2019 and 2020 emissions, but the COVID-19 crisis caused a significant drop in air transport demand in 2020. To prevent an excessive economic burden on the already weakened airline industry and to align with the spirit of the framework, the ICAO Council decided to use only 2019 emissions as the baseline for the period of 2021-2023. At its 41st Assembly, ICAO set 85% of 2019 emissions as CORSIA's baseline from 2024 to 2035, which the industry supported [26].

Offsetting, a means of compensating emissions by financing reductions elsewhere, is a pivotal component in global, regional and national emissions reduction policies. CORSIA lists eligible emissions units, guided by environmental criteria to ensure the units deliver the required CO_2 reductions. The scheme is anticipated to stabilize international aviation's net CO_2 emissions at around 600 million tonnes, whereas without CORSIA, the International Air Transport Association (IATA) estimates emissions could reach almost 900 million tonnes by 2035 [26].

CORSIA's implementation occurs in phases, with offsetting requirements gradually expanding to encompass all international flights by 2027 [26]. However, certain exemptions exist for specific countries and categories of flights until then.

The international standards and recommended practices for CORSIA have been adopted to ensure uniform regulations within the industry. Uniformity is crucial to avoid market distortions and to maintain CORSIA's environmental integrity.

Furthermore, during the 41st ICAO Assembly, a significant milestone was reached with the adoption of a long-term global aspirational goal (LTAG) for international aviation: achieving net-zero carbon emissions by 2050, aligned with the UNFCCC (United Nations Framework Convention on Climate Change) Paris Agreement. This historic agreement underscores ICAO's leadership in addressing the intersection of international aviation and climate change. The resolution, A41-21, outlines this ambitious goal [27].

The LTAG doesn't impose specific emission reduction targets on individual states. Instead, it acknowledges the unique circumstances and capabilities of each state. Factors such as developmental level, maturity of aviation markets, sustainable growth, just transition and national air transport priorities will guide each state's contribution to this goal within their own timeframe. Each state commits to achieving this aim sustainably, considering social, economic and environmental factors in line with its national context [27].

Enhanced monitoring and reporting systems for CO emissions will facilitate regulatory compliance and environmental stewardship.

The implementation of measures addressing aircraft emissions can be categorized based on existing legislation or regulatory systems as well as those necessitating the development of monitoring systems or new regulations.

Some measures, such as introducing a cruise-NO_x charge and incorporating aircraft NO_x emissions into the EU ETS, fall within existing regulatory frameworks. Additionally, new standards for LTO NO_x emissions and the introduction of an LTO-nvPM standard align with existing legislative mechanisms. Conversely, several measures require the establishment of monitoring systems or further regulatory developments, like monitoring aromatics content and cruise nvPM emissions, along with the introduction of charges or emissions inclusion in the EU ETS [23].

Furthermore, measures requiring the development of a new type of standard, like the aircraft cruise-NO_x standard or engine cruise-NO_x standard, demand additional regulatory frameworks. As for those reliant on scientific research, initiatives focused on holistic optimization of the climate impact and indicators capturing the total climate impact of flights are crucial for a comprehensive approach. Notably, the climate impact of contrails and induced cirrus cloudiness appears less sensitive to changes in background concentrations compared to NO_x emissions, with contrails and cirrus typically exerting a positive radiative forcing effect (warming). Strategies to reduce nvPM emissions, affecting contrails, without increasing CO_2 emissions, predominantly involve fuel changes, urging further exploration into measures enhancing fuel quality [23].

Although measures reliant on scientific research are imperative, those anchored in indicators capturing the comprehensive climate impact of flights stand out as the most effective, offering a holistic view encompassing all trade-offs and synergies among various emissions.

In order to reach the goal of achieving net-zero carbon emissions by 2050, there are several ongoing projects which involve enhancing the Air Traffic Management (ATM) system, refining infrastructure and optimizing operations to establish a sustainable and efficient aviation system [28]. This, along with the rapid integration of innovative aircraft technologies and the expanded use of Sustainable Aviation Fuels (SAFs), will play a pivotal role in reaching the long-term aspiration of achieving net-zero carbon emissions by 2050.

These initiatives will be presented in the next section.

2.3. Ongoing projects and research initiatives

The aviation industry has shown a consistent drive for efficiency improvement. Prior to the COVID-19 pandemic, there was a steady rise in the passenger load factor, reaching a record average of over 82% in 2019. Operational efficiencies have notably reduced fuel burn per passenger kilometer by 55% since 1990 [28]. However, certain infrastructure improvements have not progressed as swiftly as initially anticipated.

Aircraft operations have focused on weight reduction, aerodynamic enhancements to in-service aircraft and the implementation of systems that enhance efficiency during flight. Infrastructure improvements have largely been centered on structural changes in ATM operations and energy-saving initiatives at airports, such as limitations on auxiliary power units and reduced taxi times.

Operational enhancements include various measures, like [28]:

- Retrofitting winglets to save fuel and reduce emissions.
- Adopting lightweight cabin equipment.
- Implementing electric or assisted taxiing, which significantly cuts fuel consumption.
- Exterior paint improvements can also contribute to aerodynamic efficiency.

Infrastructure advancements in airports involve fixed electrical ground power, collaborative decision-making and strategies to manage surface congestion. Air traffic management improvements encompass performance-based navigation, required navigation performance, space-based navigation and continuous descent and climb, among various other initiatives that aim to streamline flight operations and reduce fuel consumption and emissions [28].

Ongoing research into alternative fuels like biofuels and hydrogen has the potential to further decrease CO emissions. The shift towards alternative propulsion technologies like electric and hybrid-electric aircraft holds the potential to virtually eliminate NO_x emissions.

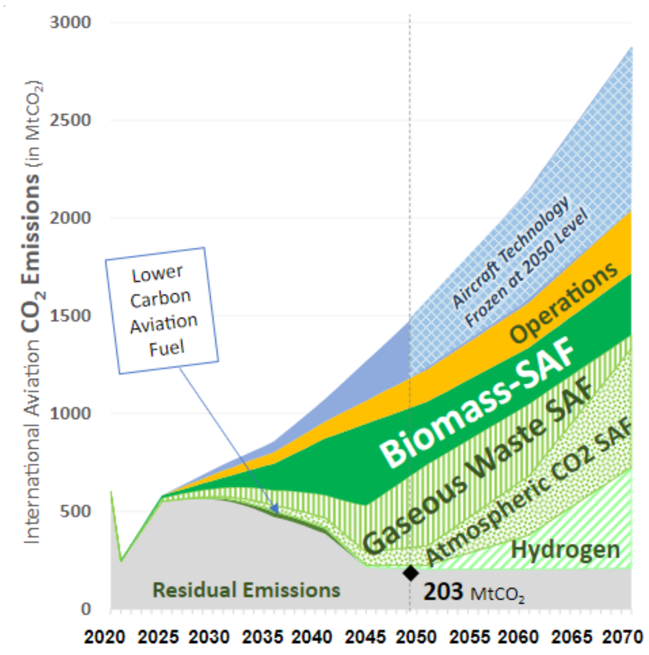


Figure 2.10: International aviation CO_2 emissions prediction through different initiatives [29]

The development of electric and hybrid-electric aircraft is underway, offering the promise of revolutionizing aviation emissions by eliminating CO and NO_x emissions altogether. As we can see in the image below, Airbus, one of the main aircraft manufacturers, has boosted several initiatives in the last years linked to electric and hybrid-electric aircraft, but as of today there is still not a viable way to design a fully electric aircraft.

This report outlines and sets medium-term technology goals for NO_x or nvPM emissions, acknowledging the escalating challenges of achieving further emission reductions, especially as technological advancements reach plateaus. Furthermore, the likelihood of alternative aircraft technologies, like hybrid/electric propulsion, significantly impacting the fleet before 2037 appears remote.

As for hydrogen-powered aircraft, the current scenario is similar. As hydrogen storage solutions are really likely to be bigger than the current jet fuel storage tanks, it is hard to find a solution that would maintain the currently existing commercial aviation standard. Moreover, the aviation industry is heavily regulated, which means all requirements modified with the introduction of these type of technologies would need to be met, for example explosion proofness requirements.

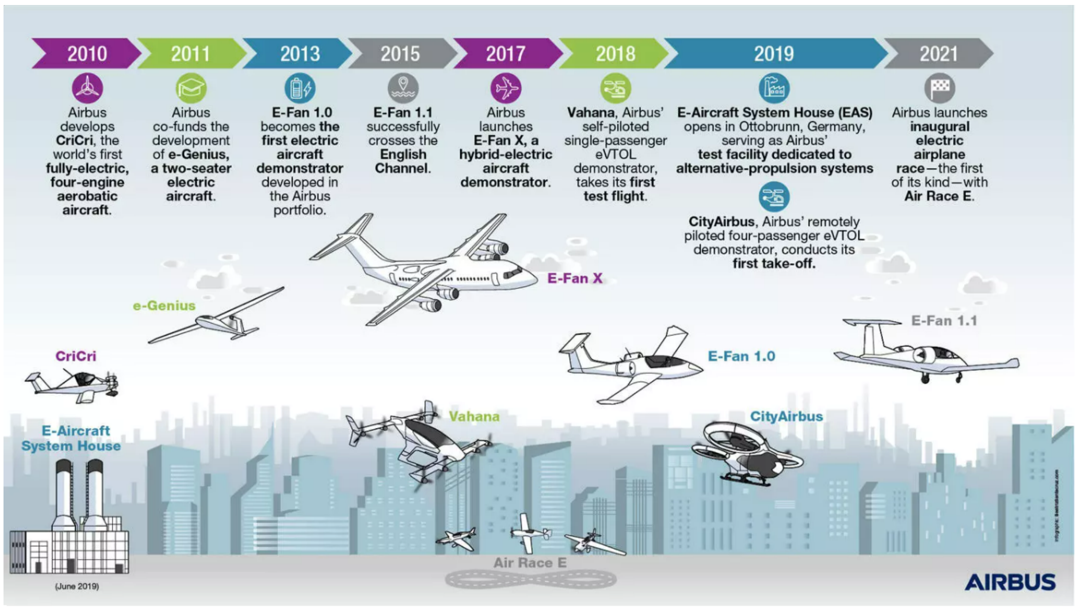

2010
Airbus develops CriCri, the world's first fully-electric, four-engine aerobatic aircraft.
2011
Airbus co-funds the development of e-Genius, a two-seater electric aircraft.
2013
E-Fan 1.0 becomes the first electric aircraft demonstrator developed in the Airbus portfolio.
2015
E-Fan 1.1 successfully crosses the English Channel.
2017
Airbus launches E-Fan X, a hybrid-electric aircraft demonstrator.
2018
Vahana, Airbus' self-piloted single-passenger eVTOL demonstrator, takes its first test flight.
2019
E-Aircraft System House (EAS) opens in Ottobrunn, Germany, serving as Airbus' test facility dedicated to alternative-propulsion systems
CityAirbus, Airbus' remotely piloted four-passenger eVTOL demonstrator, conducts its first take-off.
2021
Airbus launches inaugural electric airplane race—the first of its kind—with Air Race E.
E-Fan X
e-Genius
CriCri
E-Fan 1.1
E-Fan 1.0
E-Aircraft System House
Vahana
CityAirbus
Air Race E
(June 2019)
AIRBUS

Figure 2.11: Airbus initiatives for electric and hybrid-electric aircraft design in the last years [30]

Nevertheless, ongoing research into sustainable aviation fuels continues to drive innovation in emissions reduction strategies. Numerous projects are dedicated to the development and scalability of SAFs, aiming to replace conventional jet fuels and significantly reduce both CO and NO_x emissions.

Sustainable aviation fuel is a liquid fuel utilized in commercial aviation, boasting the capacity to slash CO_2 emissions by up to 80%. It can be sourced from various feedstock, including waste oils, fats, green and municipal waste, and non-food crops. SAF can also be synthetically produced by capturing carbon directly from the air. Its 'sustainability' lies in the fact that the raw feedstock doesn't compete with food crops, water supplies or contribute to forest degradation [31].

Unlike fossil fuels that increase overall CO_2 levels by releasing previously trapped carbon, SAF recycles CO_2 absorbed by the biomass used in its feedstock during its lifecycle. Seven certified biofuel production pathways can produce SAF that operates on par with Jet A1 fuel [31]. These SAFs are designed as drop-in solutions, easily blendable into existing airport fuel infrastructure and fully compatible with modern aircraft.

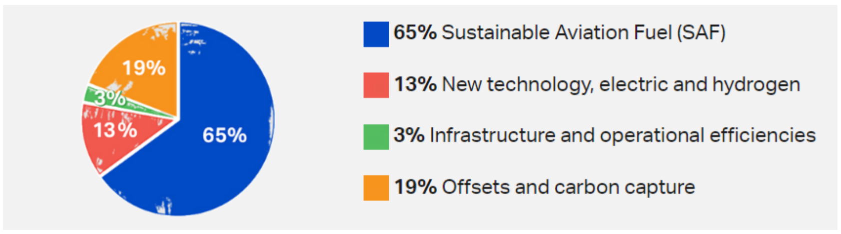


Figure 2.12: Different ways and their contribution to reaching net zero carbon emissions by 2050 [31]

It's estimated that SAF could contribute around 65% of the necessary emissions reduction for aviation to achieve net-zero status by 2050. To meet this demand, substantial production increases are crucial, particularly in the 2030s, as global policy support becomes more prevalent, SAF competes favorably with fossil kerosene and credible offsets become scarcer [31].

Aviation's fuel consumption growth has trailed behind the increase in demand due to advancements in engine fuel efficiency, notably through the adoption of modern high-bypass engine technology. These engines rely on elevated engine pressure ratios and higher temperature combustors to enhance efficiency, resulting in significant reductions in carbon monoxide and unburned hydrocarbons emissions. However, they tend to elevate emissions of nitrogen oxides. Consequently, total NO_x emissions from aircraft are escalating faster than fuel consumption [7].

Ongoing research endeavors seek to improve combustion efficiency and develop environmentally friendly engine designs. Efforts to improve idle efficiency and reduce NO_x involve balancing liner cooling flows, fuel/air ratios, dwell times in the combustor and pressure losses [7]. Managing these factors is crucial to minimize specific fuel consumption (SFC) losses while maintaining low emissions. Achieving an optimal balance is necessary, considering the impact on emissions and the engine's durability. Achieving fuel efficiency while curbing emissions demands a delicate balance between various engine parameters.

Improved fuel efficiency and maintenance practices help minimize the conditions that lead to fuel-rich combustion and CO production. The adoption of lean-burn combustion and catalytic converters in engines has been effective in reducing NO_x emissions [32] [33]. The development of next-generation engines with advanced combustion designs aims to further mitigate NO_x emissions. For example, as it can be seen in the image below, as the Pressure Ratio (PR) increases, NO_x emissions tend to decrease.

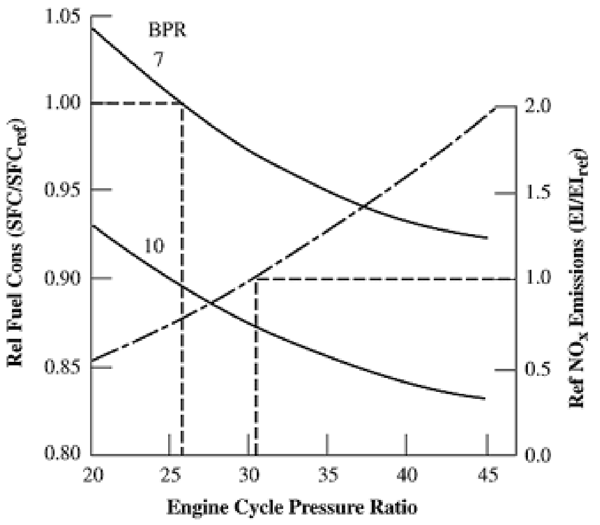


Figure 2.13: Effect of engine pressure ratio on NO_x emissions and relative specific fuel consumption [7]

Aviation-related pollutant emissions, particularly CO and NO_x, have undergone extensive research and mitigation efforts. Advancements in engine technology, the exploration of alternative fuels and the pursuit of innovative propulsion systems hold great promise for reducing these emissions. Sustainable aviation practices and forward-thinking technologies are paramount in mitigating the environmental impact of aviation in the coming decades.

Until this point, the intricate relationship between combustion processes in aircraft engines and ensuing emissions has been explored, while emphasis was also placed on the evolving regulatory landscape governing aviation emissions and future initiatives aiming to advance innovative technologies and sustainable practices to significantly reduce emissions from aircraft engines. This paves the way for the subsequent chapter, which delves into the application of data analysis methodologies like predictive emissions monitoring systems. The following chapter aims to comprehensively explore the role of these analytical tools in predicting pollutant emissions from gas turbine engines, contributing to ongoing efforts aimed at addressing environmental concerns in aviation.

3
Data-based engine emission prediction

This chapter concentrates on the current status of data analytics within aeronautics, specifically emphasizing prediction systems designed for engine emissions. Furthermore, it explores the analytical methods to be employed throughout this Thesis.

3.1. Data analytics in aeronautics

Data analysis has emerged as a pivotal tool in today's information-driven world, finding applications in diverse fields. This Thesis explores its profound significance in aeronautics, where the meticulous analysis of data has the potential to optimize aircraft performance, enhance safety and mitigate environmental impacts.

The groundwork of data analysis, encompassing data preprocessing, exploratory data analysis and statistical techniques, is fundamental for extracting meaningful insights from aeronautical data.

Data analysis plays a pivotal role in several aviation fields:

- Enhances safety through the analysis of Flight Data Recorders (FDRs) and Cockpit Voice Recorders (CVRs).
- Allows for optimization of flight routes and fuel efficiency through the analysis of weather data, flight data and historical flight patterns.
- Has applications in maintenance prediction and condition monitoring of aircraft components using sensor data and predictive maintenance algorithms, with the most important fields being Structural Health Monitoring (SHM) and Engine Health Monitoring (EHM). Predictive analysis aids in the early detection of anomalies and potential failures, ensuring safety and minimizing downtime.
- Data analysis techniques can be used to predict engine performance parameters such as thrust, fuel consumption and efficiency.
- Allows for real-time monitoring and adjustment of engine parameters during flight for optimal operation.

Furthermore, Artificial Intelligence (AI) represents cutting-edge technology aiming to replicate and enhance various human intelligence capabilities for tasks like decision-making, pattern recognition, visual perception and natural language comprehension. In the aerospace industry, numerous potential applications have been identified, showcasing AI's potential to enhance system capabilities.

However, safety remains a paramount focus, and certification entities will not permit AI systems if they compromise safety or increase associated risks. Current standards are inadequate for certifying most AI-based systems due to the complexity of AI algorithms, which shift development from traditional requirement-based models to data-driven approaches, making traceability challenging.

Both disciplines encompass applications facilitating the prediction of aircraft engine emissions. This Thesis aims to forecast gas turbine emissions, a task typically approached through either of two prevailing methods: Continuous Emission Monitoring Systems (CEMS) or Predictive Emission Monitoring Systems (PEMS).

3.2. Data analytics for prediction of engine emissions

As it has been mentioned, in the monitoring of CO and NO_x pollutants emitted during combustion operations in a power plant, two monitoring solutions have been devised: CEMS and PEMS [34].

- CEMS involves the installation of emission monitoring equipment (such as sensor sets) on-site, offering real-time emission data directly from sensors. The accuracy of CEMS measurements relies on proper maintenance and calibration adhering to standard procedures.
- Conversely, PEMS operates as an expert system utilizing process variables (e.g., ambient temperature, turbine inlet temperature) as inputs and employing predictive models trained on historical data to estimate emission components.

The main differences between both methods are the following [35]:

1. Capital cost: PEMS generally cost significantly less, around half or less than CEMS installations. The cost can be further reduced if a CEMS is already present on-site.
2. Calibration gases: PEMS do not require Environmental Protection Agency (EPA) protocol calibration gases, reducing operational complexities compared to CEMS, which necessitate ongoing purchases of calibration gases and additional storage requirements.
3. Preventative maintenance: PEMS have minimal ongoing maintenance requirements, usually limited to periodic cleaning of the PEMS computer. CEMS, on the other hand, demand daily, monthly, quarterly, semi-annual and annual maintenance, along with spare parts inventories, accounting for a significant percentage of the system's cost.
4. Data availability: PEMS ensure higher data availability, approaching 100%, by utilizing various inputs for a robust database. Even during a PEMS failure, data from Distributed Control System (DCS) inputs remain accessible for predictions. Conversely, CEMS maintain around 95% uptime at best, with potential data loss during analyzer or critical component failures.
5. Excess emissions troubleshooting: PEMS offer insights into excess emissions sources by identifying abnormal combustion input parameters for immediate action and diagnostic trails. CEMS lack this capability to pinpoint causes or facilitate corrective actions for excess emissions.
6. Obsolescence and service costs: PEMS analyzers are less prone to obsolescence and can be continually upgraded for regulatory changes or software upgrades through cost-effective ongoing service contracts. In contrast, CEMS incur higher annual service contract expenses.

Overall, PEMS demonstrate cost-efficiency, lower maintenance and higher data availability compared to CEMS, making them a potentially favorable option for emissions monitoring systems.

From this point on, and considering it will be the method to be implemented through this work, the focus will be directed towards PEMS.

3.2.1. Predictive Emission Monitoring Systems (PEMS)

In the aerospace industry, PEMS function as advanced computational models designed to estimate and monitor pollutant emissions generated by aircraft engines. PEMS are utilized for the continuous monitoring of emissions at stationary sources, serving as an alternative and backup for CEMS. They are primarily employed to predict NO_x emissions from combustion processes in both practical applications and the literature.

PEMS establish a mapping between a set of characteristic process parameters of an emission source (e.g., air pressure, turbine after temperature) and the corresponding flue gas emission. If the combustion process variables are continuously monitored and recorded, emission concentrations after combustion can be predicted. PEMS are plant-specific emission monitoring systems with varying methodologies and designs, ranging from relational models to Neural Network (NN) models [34]. PEMS enable the aviation industry to meet stringent environmental regulations by proactively managing and reducing emissions.

PEMS in the aerospace sector employ sophisticated algorithms and machine learning techniques trained on historical emissions data obtained from engine testing, flight simulations and actual in-flight measurements. These models analyze and process the input data to predict emission components, such as nitrogen oxides, carbon monoxide, unburned hydrocarbons and particulate matter.

The functionality of PEMS involves continuous monitoring and analysis of multiple parameters associated with engine performance and combustion processes. By integrating real-time data from sensors and engine control systems, PEMS can make accurate estimations of emissions without the need for direct measurement devices onboard the aircraft.

The predictive capabilities of PEMS contribute significantly to emissions management strategies in the aviation industry. They enable proactive decision-making by providing insights into emission trends, allowing for the optimization of engine performance, and supporting regulatory compliance efforts.

Computational models are pivotal in PEMS, primarily relying on historical process data for their functioning. These models, notably Neural Networks, have been extensively utilized for predicting emissions in various industrial settings.

For instance, Korpela et al. used linear regression and nonlinear NNs to estimate NO_x emissions in natural-gas-fired boilers [36]. Dynamic NNs were employed to create soft sensors for monitoring NO_x and O_2 emissions in industrial boilers [37]. Lv et al. applied a least squares support-vector-machine-based ensemble learning paradigm to forecast NO_x emissions in coal-fired boilers using operational data [38].

Different machine learning techniques were explored for emission prediction. Ćirić et al. employed various neural network architectures to estimate CO_2 emissions in a Serbian context [39]. Additionally, Dragomir and Oprea introduced a multiagent system for monitoring urban air pollution from power plants [40]. Idzwan et al. used support vector machines to predict NO_x emissions in a Malaysian power plant [41]. Furthermore, Liukkonen and Hiltunen presented an innovative emission monitoring platform based on self-organizing maps [42].

Regarding the current state of the art of aircraft engine emissions prediction, numerous works have targeted this field. For instance, a study conducted by Filippone et al. utilized flight performance models and ADS-B (Automatic Dependent Surveillance–Broadcast) data for predicting emissions along specific routes, covering carbon dioxide, carbon monoxide, nitrogen oxides and water vapor. This research incorporated diverse aircraft types and operational altitudes surpassing 3000 feet. The study aimed to bridge a crucial void in aviation emission inventories by utilizing the integration of real-time flight data [43].

Saboohi and Ommi studied semi-analytical prediction methodologies for gas turbine emissions during the conceptual design phase, thus targeting early design stages. To achieve this, a Chemical Reactor Network (CRN) was formulated, utilizing available input data for detailed combustion modeling. The study examines three distinct chemical mechanisms relevant to Jet-A aviation fuel. Results indicated a commendable alignment between CRN outputs and emissions from actual engine test rigs. Remarkably, the augmented CRN demonstrated high efficiency in predicting engine emissions, offering rapid execution (mere seconds) and minimal CPU requirements, akin to those of a personal computer [44].

Kayaalp et al. conducted a study which focused on modeling combustion efficiency and exhaust emission indexes for a turboprop engine during the LTO cycle. Unlike conventional approaches, this research employed Long Short-Term Memory (LSTM), a deep learning method, to analyze varied parameters such as Air-Fuel Ratio (AFR), engine speed, fuel flow rate and exhaust emission data. By utilizing LSTM, the study aimed to predict and understand the intricate relationship between these factors [45].

A study of high relevance for this Thesis was conducted by Kaya et al., employing the identical dataset that will be further studied in this work, intended for predictive emission monitoring systems. This dataset was specifically designed for predictive modeling of CO and NO_x emissions. Through a novel machine learning approach rooted in the Extreme Learning Machine (ELM) framework, the dataset underwent comprehensive analysis, resulting in valuable insights into emission predictions [34].

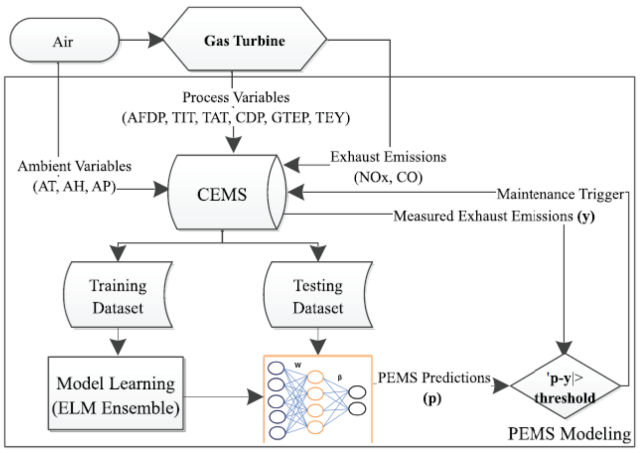


Figure 3.1: Example of PEMS pipeline [34]

Overall, PEMS in the aerospace industry serve as advanced predictive tools that leverage data-driven models to estimate and monitor emissions from aircraft engines, thereby aiding in environmental impact reduction and ensuring adherence to emission standards.

This Thesis aims to demonstrate the pivotal role of data analysis in aeronautics, focusing on forecasting engine emissions using Predictive Emission Modeling Systems (PEMS). The evolving landscape of data analysis techniques and their integration into the aviation industry underscores their significance in shaping aeronautical technologies and environmental sustainability.

Prior to delving into the exploratory data analysis of the flue gas emission dataset, it is crucial to underscore the foundational mathematical framework that will be used throughout this work, essential to support this analysis and the future modeling of engine emission prediction. Consequently, the forthcoming section will comprehensively elucidate the mathematical methods and models intended for utilization in subsequent chapters, aiding in a deeper understanding of the analytical process that will be followed.

3.2.2. Mathematical background

The initial phase in addressing a data analysis problem involves identifying the nature of the problem at hand, such as whether it pertains to supervised or unsupervised learning, regression, classification or other relevant categories.

Supervised and unsupervised learning represent distinct approaches in machine learning. In supervised learning, the algorithm learns from labeled data, wherein the inputs are paired with the corresponding correct outputs. This type of learning aims to predict or infer outcomes based on input-output mappings derived from known data examples. On the other hand, unsupervised learning involves algorithms learning from unlabeled data, seeking to identify hidden patterns or structures within the data itself without explicit output labels.

Regarding regression and classification, they are two fundamental tasks in predictive modeling. Regression involves predicting continuous numerical values, aiming to establish a relationship between input variables and a continuous outcome. This method is used to forecast trends or estimate quantities, such as predicting prices or determining scores based on various factors.

In contrast, classification involves categorizing data into distinct classes or categories. It deals with predicting discrete or categorical outcomes where inputs are assigned to specific classes or labels. For instance, classifying emails as spam or not spam, identifying whether an image contains a cat or a dog, etc.

In the realm of machine learning, solving problems involves a systematic approach comprising distinct phases:

1. The journey begins with Exploratory Data Analysis (EDA), where a comprehensive understanding of the dataset is established through statistical summaries and visualizations.
2. Following this, feature engineering comes into play, focusing on refining and transforming data features to enhance the predictive capability of models.
3. Subsequently, the modeling phase selects suitable algorithms, trains the model on appropriate data subsets and fine-tunes parameters to achieve optimal performance.
4. Finally, the experimental results phase evaluates the model's effectiveness and generalization on unseen data, crucial for validating the model's reliability in solving the intended problem.

This structured process, spanning from data exploration to model evaluation, forms the core framework of addressing machine learning challenges.

During the initial stage of Exploratory Data Analysis, it is usual to start by examining the provided data and attempting to observe basic characteristics. This typically involves checking for the presence of missing values, assessing whether uniform data quantities exist across multiple datasets (if applicable), identifying the target variable and determining the type of problem being addressed. This foundational exploration serves as a starting point to gain insights into the dataset's structure and key attributes before delving deeper into analysis and modeling processes.

Even though this will be further explained in the next chapter, during the initial assessment of the dataset for this Thesis, it can be observed that it represents time series data spread across five years, displaying missing data at the end of each of those years. Unconventionally, the decision was made to fill these missing values at this stage to ensure having complete time series for comprehensive analysis. While such data handling is typically conducted during the feature engineering phase, it was deemed beneficial in this instance to facilitate a more comprehensive understanding of the system.

A time series is a sequence of data points indexed in chronological order. It consists of observations or measurements collected at regular intervals over time. Time series analysis involves studying and interpreting patterns, trends and behaviors within the data to make predictions or understand underlying characteristics, such as seasonality, trends and cyclic patterns. This analysis is commonly used in various fields like economics, finance, weather forecasting and many others to derive insights and inform decision-making processes.

Handling missing data within datasets is a critical part of data preprocessing, ensuring the accuracy and reliability of subsequent analyses or model building. Various methods have been devised to address missing values, each with its unique approach. Some commonly used techniques are the following [46]:

- Forward Fill: it involves substituting missing values with the most recent non-missing value in the dataset. When encountering a missing value, this method simply carries forward the last known value along the sequence until the next non-missing value is reached. This approach is particularly useful in time series data where the assumption is that the most recent observation is a reasonable estimation for the missing value.
- Backward Fill: also known as backward propagation, is the opposite of forward filling. It involves replacing missing values with the next known value in the dataset. When a missing value is encountered, this method fills it with the next available observed value, assuming that future observations might resemble recent data.
- Linear Interpolation: Linear interpolation estimates missing values by creating a linear relationship between adjacent data points. When a data point is missing, this method uses the values before and after the missing point to calculate an estimated value that lies along the straight line connecting the

two known data points. The estimation assumes a linear relationship between the adjacent points and interpolates accordingly.

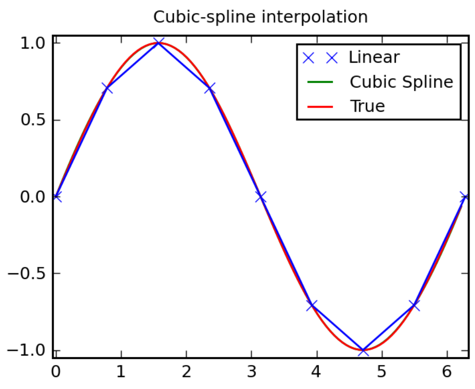


Figure 3.2: Linear Interpolation example [47]

- Seasonal Mean: this technique involves calculating the mean value separately for distinct seasons or specific time intervals within the dataset. When a value is missing for a particular period, it is replaced by the mean value corresponding to the specific season or time interval to which it belongs. This method captures seasonal patterns and replaces missing values with seasonally relevant averages.
- KNN Mean: based on the K-Nearest Neighbors algorithm, it imputes missing values by identifying the K most similar data points to the one with the missing value. The missing value is then replaced by the average value of these nearest neighbors, effectively using the information from the most similar data instances to estimate the missing value.

In a missing value imputation function such as Seasonal Mean and KNN mean, the pivotal parameter would be the value n. This value usually denotes the number of prior observations (lags) taken into account when computing the imputed values.

For Seasonal Mean, n could indicate the count of preceding observations within the identical season or cycle. For example, if n is defined as 12 in monthly time series data, it would compute the average of values from the corresponding month over the previous 12 months to impute the missing value.

For KNN Mean the parameter n typically signifies the count of nearest neighbors taken into consideration for averaging. A greater value of n involves a larger number of neighbors in the weighted average computation used to fill in the missing value.

The optimal value for n relies on the characteristics of the data and the particular problem at hand. A larger n value can offer more consistent estimates but might excessively smooth the data. Conversely, a smaller n could better capture fluctuations but might exhibit increased sensitivity to outliers or noise.

Each of these methods provides a distinct approach to handle missing data, catering to different dataset structures, patterns and characteristics, allowing for reliable replacements of missing values depending on the specific context and dataset properties.

Once these methods have been implemented, it is essential to determine which one best suits the problem at hand. This assessment can be carried out in two ways: by examining the resulting dataset and/or using metrics that evaluate the performance of specific models after filling in missing values within the dataset. These metrics are employed not only in cases of filling missing values but also in evaluating model performance, typically in the final stages of problem-solving.

In machine learning, several key metrics are employed to evaluate the performance of models. These metrics gauge the accuracy, precision and generalization capabilities of a model. Some prominent metrics include [48]:

- Mean Absolute Error (MAE): it measures the average absolute differences between predicted values and actual values in a dataset. It provides an indication of how close the predictions are to the observed values, irrespective of the direction of errors.

$$MAE = \frac{1}{N}\sum_{j=1}^{N} |y_j - \hat{y}_j| \tag{3.1}$$

- Loss functions: they quantify the inconsistency between predicted and actual values during model training. Various types of loss functions exist, such as Mean Squared Error (MSE), Cross-Entropy Loss or Hinge Loss. These functions guide the model's learning process by penalizing errors and adjusting model parameters to minimize the discrepancy between predictions and actual outcomes.

$$MSE = \frac{1}{N}\sum_{j=1}^{N} (y_j - \hat{y}_j)^2 \tag{3.2}$$

- Accuracy: it represents the proportion of correctly predicted instances out of the total instances in the dataset. It provides a basic measure of how well the model predicts the correct outcome.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made} \tag{3.3}$$

- Precision and Recall: precision measures the ratio of correctly predicted positive observations to the total predicted positive observations. Recall, also known as sensitivity, calculates the ratio of correctly predicted positive observations to the actual positive observations in the dataset. These metrics are particularly relevant in binary classification problems.

$$Precision = \frac{True\ positives}{True\ positives + False\ positives} \tag{3.4}$$

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives} \tag{3.5}$$

- F1-Score: it is the harmonic mean of precision and recall. It provides a balanced measure that considers both precision and recall, especially useful when there is an uneven class distribution in the dataset.

$$F1 = 2 \times \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{3.6}$$

- R-Squared (R^2): it measures the proportion of the variance in the dependent variable that is predictable from the independent variables in a regression model. It indicates how well the model fits the data, with values closer to 1 indicating a better fit.

These metrics serve as crucial tools to assess the performance and effectiveness of machine learning models across various problem domains and tasks.

It's also essential to highlight that when visualizing graphs such as time series, depending on the data being handled, filtering the available data can be crucial.

In machine learning problems involving signal filtering, various methods exist to process signals effectively. Among these techniques are LOWESS (Locally Weighted Scatterplot Smoothing), Savitzky-Golay filtering and Butterworth filtering.

LOWESS is a non-parametric regression method used for smoothing noisy data. It works by fitting multiple local polynomial regressions to different subsets of the data, weighing nearby points more heavily than distant ones. This approach provides a smoothed curve that captures the underlying trend within the signal while reducing noise [49].

It's beneficial because it uncovers hidden patterns or trends that might not be obvious from the raw data. By adjusting a smoothing parameter, it is possible to control the degree of flexibility in the fitted curve, balancing between capturing smaller fluctuations and highlighting broader trends.

LOWESS is widely applied in various fields, providing insights into relationships between variables without imposing rigid assumptions. Its flexibility makes it valuable for understanding complex, non-linear relationships, enabling better visualization and comprehension of data patterns for informed decision-making in diverse domains.

For LOWESS smoothing, the following steps are followed [49]:

1. Observations can be found as x and y vectors:

$$x = [x_1 ... x_N] \tag{3.7}$$

$$y = [y_1 ... y_N] \tag{3.8}$$

2. First, the distance between x$_i$ and the following observation x$_j$ is calculated:

$$d_{ij} = [y_{i0} ... y_{iN}] \tag{3.9}$$

3. The tricube weighting function is employed to augment observations that are closer in proximity, based on these distances:

$$w_i = (1 - |d_{ij}|^3)^3 \tag{3.10}$$

4. Through this method, a conventional weighted least squares system is resolved:

$$\hat{\beta} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W}_y \tag{3.11}$$

   X represents a specified first-order model, a linear model containing an intercept (constant value of 1) and a slope. Nevertheless, higher-order polynomials can be employed. W is essentially a diagonal matrix created from the weights:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \tag{3.12}$$

$$\mathbf{W} = \begin{bmatrix} w_1 & & \\ & \ddots & \\ & & w_N \end{bmatrix} \tag{3.13}$$

5. Upon solving the system, it yields $\hat{\beta}$ - β_0 as the intercept and β_1 as the slope. The smoothed observation $\hat{y_{sm}}$ in LOWESS is predicted or fitted from the system's row corresponding to i:

$$\hat{y_{sm}} = \mathbf{X}_i \hat{\beta} \tag{3.14}$$

6. Algorithmically, each observation i is iterated through sequentially. The **W** is updated with the new weights, followed by resolving the aforementioned least-squares system to update $\hat{\beta}$ and subsequently predict each $\hat{y_{sm}}$.

Savitzky-Golay filtering is a method used for smoothing and noise reduction in signal processing. It operates by fitting successive polynomial regression models to adjacent subsets of data. This filtering technique effectively removes high-frequency noise while preserving the original shape and features of the signal [50].

This method is advantageous because it preserves the shape of the data while reducing noise, making it particularly useful for retaining important features like peaks or valleys in the original signal. By adjusting parameters such as the window size (the number of neighboring points used for polynomial fitting) and the polynomial degree, users can control the trade-off between smoothing and retaining fine details in the data [50].

The Savitzky-Golay filter is widely applied in various fields such as signal processing, spectroscopy and biomedical engineering, offering an effective way to denoise and smooth graphs without significantly distorting essential features. Its ability to balance noise reduction with the preservation of important information makes it a valuable tool for data analysis and visualization.

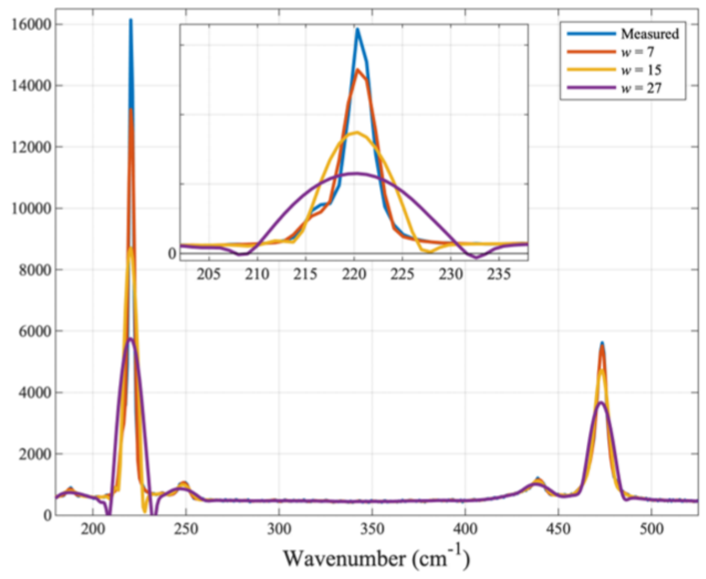


Figure 3.3: Savitzky-Golay filtering example [50]

Butterworth filtering is a type of signal processing technique commonly used in digital signal processing to achieve a specific frequency response. It belongs to the category of Infinite Impulse Response (IIR) filters and is characterized by a maximally flat frequency response within the passband while gradually attenuating frequencies outside this range [51].

The key principle of Butterworth filters lies in their design to minimize ripples or variations in the frequency response within the passband. This is accomplished by optimizing the filter's transfer function in such a way that the magnitude response is as flat as possible, resulting in a smooth roll-off from the passband to the stopband without any ripples or peaks [51].

This filter design is determined by its order, which defines the rate of attenuation beyond the cutoff frequency. Higher-order Butterworth filters provide steeper roll-off but may introduce phase distortion, especially in the transition region between the passband and stopband [51].

Utilizing mathematical equations involving the Butterworth polynomial and transfer function coefficients, this filter can be implemented across various applications, including audio signal processing, telecommunications and biomedical signal analysis, to extract or manipulate specific frequency components while minimizing unwanted noise or interference [51].

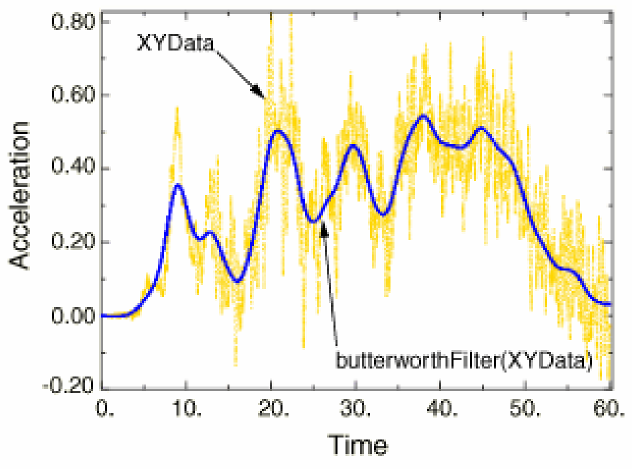


Figure 3.4: Butterworth filtering example [52]

All filtering methods offer distinct approaches to filter signals in machine learning problems, enabling the extraction of meaningful information while mitigating the impact of noise in the data.

Furthermore, utilizing multiple visualizations in machine learning is crucial as it aids in comprehending complex relationships, identifying patterns and assessing data distributions, ultimately enhancing the understanding of underlying structures within the dataset. Visualizations serve as powerful tools to explore data from different angles, uncover anomalies and facilitate better decision-making in model building and analysis. The following plots will be used during the dataset analysis:

- Lmplot is a function in Python's Seaborn library used to create scatter plots with regression lines fitting the data. This type of plot is particularly useful for visualizing relationships between two variables and displaying linear regression models along with their confidence intervals. Lmplot provides a quick way to assess the strength and direction of the relationship between variables and observe if a linear trend exists between them.

- Pairplot, also available in Seaborn, generates a grid of pairwise plots for a given dataset, showcasing the relationships between all pairs of variables. It displays scatterplots for numeric variables and histograms for the diagonal where the same variable is plotted against itself. Pairplot is beneficial in quickly visualizing multivariate relationships, enabling the identification of correlations or patterns among multiple variables simultaneously.

- A Boxplot, or Box-and-Whisker plot, is a graphical representation that displays the distribution of a dataset along with its statistical summary. It showcases the median, quartiles and potential outliers within the data. The box represents the interquartile range (IQR), while the whiskers extend to show the range of the data, excluding outliers. Boxplots are effective in comparing distributions and identifying potential anomalies or differences among groups or categories within the data.

- A Lag plot is a graphical technique used to identify patterns or autocorrelation in time series data. It is used in time series and sequential data analysis, and shows the relationship between a data value at a given time and its value at a previous time (lag).
  In a lag plot the X-axis typically represents the value of the series at time t, and the Y-axis shows the value of the series at time t+1, t+2, t+3, and so on, depending on the lag. Each point on the plot represents a pair of values from consecutive time points, and if the points are randomly scattered and don't follow a specific pattern, it suggests a weak serial correlation, indicating randomness in the data. If the points exhibit a pattern, like a diagonal line or a distinct shape, it indicates serial correlation and a potential pattern in the data.
  Lag plots are instrumental in visually detecting relationships or dependencies between observations at different time intervals, aiding in understanding the temporal structure and potential autocorrelation within the time series data.

These visualization techniques offer valuable insights into the data, allowing for a better understanding of relationships, distributions and variations, which in turn aids in making informed decisions during the machine learning process.

An immensely significant concept mentioned earlier is the notion of an outlier. Outliers refer to data points that significantly differ from the majority of observations in a dataset. These values are notably distant from other data points and can skew statistical analyses, leading to misleading interpretations or models if not appropriately addressed. Detecting outliers is crucial in various fields such as data analysis, machine learning and statistics due to several reasons:

- Impact on statistical measures: outliers can substantially affect statistical measures like mean, variance and standard deviation, leading to inaccurate estimations of central tendencies and variability.

- Model performance: in machine learning, outliers can negatively impact model performance. They might cause overfitting, where the model excessively fits the training data including outliers, leading to poor generalization on new data. Conversely, outliers might cause underfitting by affecting the model's ability to capture the underlying patterns in the data.

- Data integrity and insights: outliers might represent errors or anomalies in the data collection process. Identifying and addressing outliers help maintain data integrity and ensure that insights drawn from the data are reliable.

- Robustness of analyses: outliers can distort relationships between variables, impacting correlations, regressions and other analytical techniques. Detecting and handling outliers appropriately improves the robustness and accuracy of analyses.

To mitigate the influence of outliers, various methods are employed, such as trimming (removing extreme values), winsorizing (replacing extreme values with less extreme ones) or using robust statistical techniques that are less sensitive to outliers. Identifying outliers is essential to ensure the reliability and accuracy of analyses, preventing misleading conclusions or poorly performing models.

Identifying correlations among variables or within a single variable is also critical in data analysis and machine learning for several reasons. Understanding these relationships helps in discerning how changes in one variable might affect another, revealing potential dependencies, patterns or associations that can significantly impact predictive models or analyses. The identification of correlations assists in feature selection, allowing the inclusion of only relevant and influential variables in model building, which can enhance predictive accuracy and reduce overfitting.

Correlation matrices play a pivotal role in unveiling these relationships. A correlation matrix is a table representing the pairwise correlations between variables in a dataset. Each cell in the matrix indicates the correlation coefficient, a numerical measure representing the strength and direction of the relationship between two variables. A correlation coefficient of +1 denotes a perfect positive linear relationship, -1 indicates a perfect negative linear relationship and 0 implies no linear relationship between the variables.

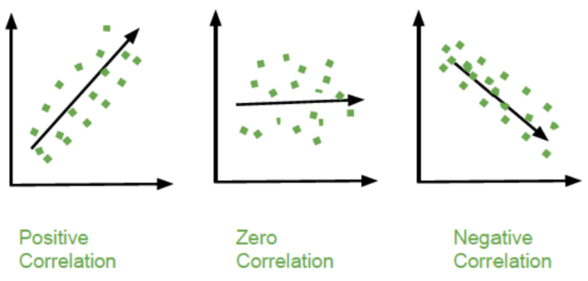


Figure 3.5: Correlation types [53]

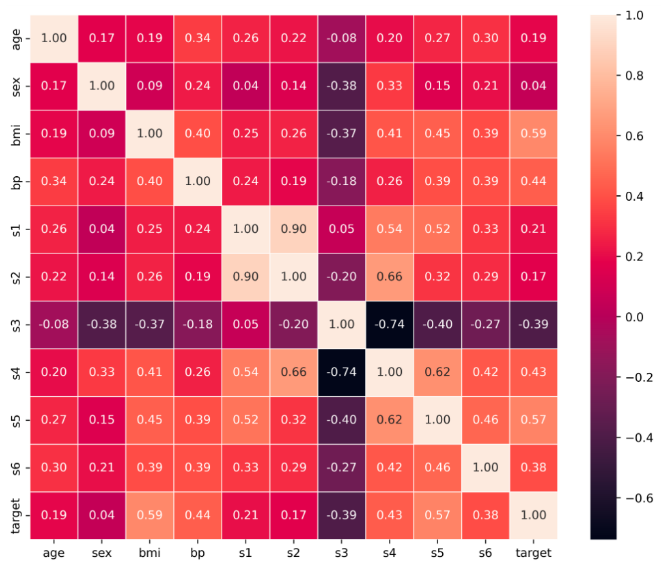


Figure 3.6: Correlation matrix example [53]

Linear and nonlinear correlation metrics distinguish the type of relationship between variables. Linear correlation metrics, such as Pearson's correlation coefficient, measure the strength and direction of a linear relationship between variables. It evaluates the linear association between two continuous variables and assumes a linear pattern between them [53].

$$r = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \sum(y_i - \overline{y})^2}} \tag{3.15}$$

However, in cases where the relationship between variables doesn't follow a linear pattern, nonlinear correlation metrics are employed. Among the popular nonlinear correlation metrics are Kendall's Tau and Spearman's Rank correlation coefficients [54] [55]:

- Kendall's Tau assesses the strength and direction of the ordinal association between two variables, regardless of the data distribution, making it suitable for variables with rankings or ordinal data.

  $$\tau = \frac{C - D}{C + D} = \frac{C - D}{\frac{n}{2}(n-1)} \tag{3.16}$$

  Here, C and D are counts of concordant and discordant pairs, respectively. Concordant pairs are those in which the order of both variables is consistent, while discordant pairs have inconsistent orders between the variables.
  Kendall's Tau ranges from -1 to 1, where: if $\tau = 1$, it indicates perfect agreement between the rankings of the two variables; if $\tau = -1$, it signifies perfect disagreement between the rankings of the two variables; and if $\tau = 0$, there is no association between the rankings of the two variables.

- Spearman's Rank correlation, akin to Kendall's Tau, evaluates monotonic relationships between variables, focusing on the consistency of the relative order of values rather than their specific numerical difference. The formula to compute Spearman's Rank correlation coefficient involves the calculation of the covariance of the ranks of the variables, as well as their standard deviations. Given d_i as the difference between the ranks of paired observations and n as the number of paired observations, the formula is expressed as:

  $$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{3.17}$$

  As for Kendall's Tau, Spearman's Rank correlation coefficient ranges from -1 to 1.

While Pearson's correlation measures the linear relationship between variables based on their covariance and standard deviations, Kendall's Tau and Spearman's Rank correlation focus on the order or ranks of observations, making them particularly useful when linear correlation assumptions do not hold or when dealing with categorical or ordinal data.

Both Kendall's Tau and Spearman's correlation coefficients are robust against outliers and deviations from linear patterns, making them suitable for nonlinear relationships or when dealing with non-normally distributed data. Notably, the Kendall correlation involves a computational complexity of $O(n^2)$, unlike the $O(n \cdot \log(n))$ complexity associated with Spearman correlation, where n represents the sample size.

Overall, the choice of correlation metric depends on the nature of the data and the specific relationships one aims to explore, highlighting the necessity of understanding the nuances and characteristics of different correlation measures for robust analysis and modeling in machine learning and data science.

On the other hand, a histogram is a graphical representation of the distribution of numerical data. It consists of a series of bars where each bar represents the frequency or count of values falling within specific intervals, known as bins, along the horizontal axis. The vertical axis typically displays the frequency or relative frequency of occurrences within each bin.

There are various types of histograms that depict different patterns of data distribution [56]:

- Symmetric (or bell-shaped) histogram: this type forms a symmetrical shape resembling a bell curve, indicating a normal distribution where the data is evenly distributed around the mean. It's also known as a Gaussian or normal distribution.

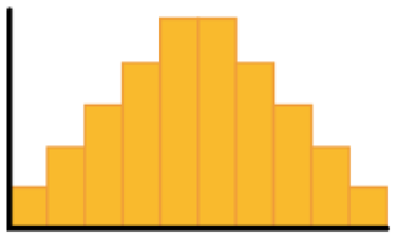


Figure 3.7: Symmetric histogram example [56]

- Skew-left (negatively skewed) histogram: a skew-left histogram displays a longer tail on the left side, indicating that the majority of data points are concentrated on the right side, with few outliers stretching the distribution to the left.

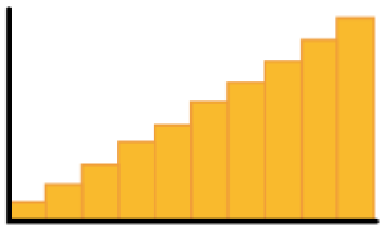


Figure 3.8: Skew-left histogram example [56]

- Skew-right (positively skewed) histogram: conversely, a skew-right histogram exhibits a longer tail on the right side, implying that most data points are clustered on the left side, while a few outliers extend the distribution to the right.

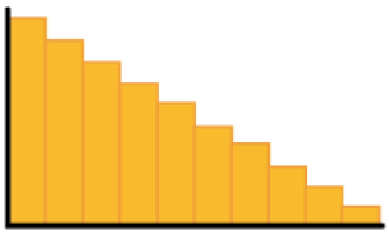


Figure 3.9: Skew-right histogram example [56]

- Uniform histogram: in a uniform histogram, all bins or intervals have approximately the same frequency, indicating an even distribution of data across the entire range without any prominent peaks or tails.

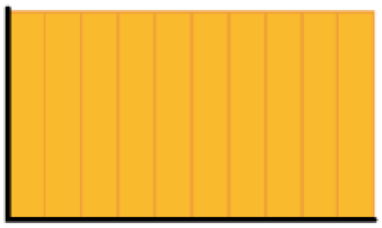


Figure 3.10: Uniform histogram example [56]

- Bimodal histogram: this type of histogram displays two distinct peaks or modes, indicating that the data has two different centers or patterns within the distribution.

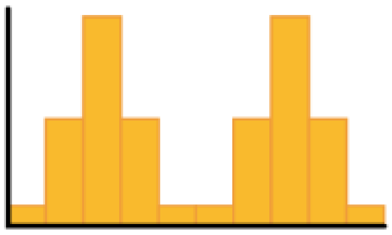


Figure 3.11: Bimodal histogram example [56]

Histograms serve as valuable tools in data analysis, providing insights into the shape, center, spread and possible skewness of a dataset, facilitating a better understanding of its characteristics and aiding in making informed decisions during data exploration and analysis.

Subsequently, an interesting technique for analyzing the dataset involves its decomposition, offering valuable insights into its structural components and patterns.

Time series decomposition entails a technique that dissects a time series into multiple components, each representing distinct underlying patterns: trend, seasonality and noise [57].

- Seasonality: describes the periodic signal in the time series.
- Trend: describes whether the time series is decreasing, constant, or increasing over time.
- Noise: describes what remains behind the separation of seasonality and trend from the time series. In other words, it's the variability in the data that cannot be explained by the model.

There are two types of decomposition: additive and multiplicative. Additive decomposition sees a time series as the sum of its components (trend, seasonality and residual), while multiplicative decomposition views it as the product of these components. Additive is used when variations in the series are independent of its level, while multiplicative suits data where fluctuations relate to the series' level. Additive is for constant fluctuations and multiplicative is for variable fluctuations with the series' level.

$$y(t)_{additive} = Trend + Seasonality + Noise \tag{3.18}$$

$$y(t)_{multiplicative} = Trend \times Seasonality \times Noise \tag{3.19}$$

Decomposition serves as a crucial tool in time series analysis, aiding in understanding trends, seasonality and irregular patterns for informed modeling and forecasting decisions. While it provides a structured framework, real-world data often deviates from idealized additive or multiplicative models, presenting complexities that may challenge accurate decomposition. Nonetheless, it remains a fundamental approach for analyzing and forecasting time series data.

For the upcoming explanations, it's vital to introduce a new concept: p-value.

The p-value is a statistical concept used to determine the strength of evidence against a null hypothesis in a scientific study or experiment. It helps researchers assess whether their findings are statistically significant or simply the result of random chance [58].

In a hypothetical scenario where you function as a scientist evaluating a new drug's efficacy, the null hypothesis posits that the drug yields no effect and any variances noted in the study are random chance occurrences. The p-value serves as a metric to quantify the probability that observed outcomes stem solely from chance.

Here's how it works [58]:

- Calculating the p-value: following the completion of the experiment, data analysis is conducted using statistical techniques. The p-value, ranging from 0 to 1, holds significance; smaller values, such as 0.05 or lower, indicate that the observed outcomes are improbable to arise solely from chance.
- Interpreting the p-value: a p-value below 0.05 (often employed as a benchmark) indicates substantial evidence opposing the null hypothesis. In the drug scenario, this implies that the drug possesses a tangible and substantial impact. Conversely, a p-value surpassing 0.05 suggests that outcomes are more likely due to random chance, thereby lacking sufficient evidence to dismiss the null hypothesis.

Fundamentally, a p-value serves as an indicator of whether observed results hold significance or could have occurred merely by chance. A diminished p-value (below 0.05) implies that the findings are more likely authentic rather than a coincidence, whereas an elevated p-value suggests that the results might lack persuasiveness.

It's crucial to note that while the p-value is integral, it's only a component of the scientific method. It does not quantify the magnitude or practical relevance of the observed effect under study and it's not the sole determinant when assessing the significance of research findings. The interpretation of p-values should always be within the broader framework of the entire study and with a comprehensive understanding of the specific research domain.

Although graphs serve as valuable tools for observation and assessment, having a quantitative method to ascertain the stationarity of a given series holds significant value.

A stationary time series maintains consistent statistical properties such as mean, variance and autocorrelation throughout its duration. This characteristic holds several implications [59]:

- Simpler modeling: stationary time series make it easier to build statistical and predictive models, such as ARIMA (autoregressive integrated moving average) models, which assume stationarity.

- More precise estimates: parameter estimates in stationary series tend to be more accurate, improving predictive capabilities.
- Valid assumptions: many statistical methods assume stationarity, making these methods valid for such data.
- Interpretable analysis: stationary series are easier to understand, and meaningful relationships are more apparent.

Determining if a given time series is stationary or not can be done using statistical tests called Unit Root tests. There are multiple variations of them, where the tests check if a time series is non-stationary and possess a unit root.

There are multiple implementations of Unit Root tests like [59]:

- Augmented Dickey Fuller test (ADF Test).
- Kwiatkowski-Phillips-Schmidt-Shin – KPSS test (trend stationary).

The widely utilized method is the ADF test, in which the null hypothesis stipulates the time series containing a unit root and being non-stationary. Therefore, if the obtained p-value from the ADF test falls below the significance level (0.05), rejecting the null hypothesis implies that the time series is stationary. Conversely, a p-value above 0.05 suggests the time series can be considered as non-stationary. The KPSS test, conversely, evaluates trend stationarity. In contrast to the ADF test, its null hypothesis and p-value interpretation are reversed. Therefore, a p-value below 0.05 indicates a stationary trend [59].

The subsequent step entails analyzing whether detrending the time series yields significant effects, aiding in comprehending the dataset's characteristics and patterns.

Detrending a time series involves removing or modeling the underlying trend in the data. It is used to make data more stationary, reveal short-term patterns, improve modeling, enhance prediction accuracy and simplify interpretation.

Detrending can be done using methods like differencing, regression modeling, smoothing or decomposition (with the last two methods already being explained above). The specific technique chosen depends on the data and analysis goals.

Another important technique usually employed in EDA consists in examining the seasonality of a time series. This involves identifying recurring patterns or cycles in the data.

To investigate this aspect, the autocorrelation plot method is usually employed. This method aids in visualizing patterns by presenting the autocorrelation of a time series across various lag points.

Autocorrelation, in the context of time series analysis, refers to the correlation of a series with its own lagged values. It measures the relationship between observations in a time series separated by a specific time interval, known as the lag.

In simpler terms, autocorrelation assesses how each value in a time series relates to its preceding values at different time lags. A positive autocorrelation at a specific lag indicates that observations at that lagged position have a similar pattern or tendency to the current observation. Conversely, negative autocorrelation suggests an inverse relationship between the current observation and the one at a given lag.

Autocorrelation is fundamental in understanding the underlying structure of a time series. It helps identify patterns, trends and seasonality within the data. Autocorrelation plots or correlograms visually represent these relationships across various lags, aiding in determining the optimal lag for forecasting or modeling purposes and in detecting potential dependencies or trends within the time series data.

Peaks or spikes in the plot indicate potential seasonality within the data. For instance, in a monthly time series where patterns recur yearly, notable spikes should appear at intervals such as the 12th, 24th, 36th and so on.

Autocorrelation (ACF) and Partial Autocorrelation (PACF) are statistical measures used in time series analysis to understand the relationship between data points at different time lags.

ACF measures the correlation between a data point and its lagged values (previous time points) up to a specified number of lags (e.g., 50). A high ACF at a particular lag suggests a strong correlation with that past value [60].

PACF, on the other hand, measures the correlation between a data point and its lagged values while controlling for the influence of intervening lags. It helps identify the direct influence of past time points on the current value [60].

In time series analysis, ACF and PACF are useful for [60]:

- Identifying seasonality: ACF often exhibits periodic patterns at specific lags, indicating seasonality in the data.
- Determining model orders: PACF helps identify the order of autoregressive and moving average components in models like ARIMA.
- Model diagnostics: ACF and PACF plots are used to check if a chosen model is a good fit for the data. Residuals should have no significant correlations in these plots.
- Forecasting: ACF and PACF provide insights into how past values affect future values, aiding in time series forecasting.

Lastly, in EDA, it holds significance to assess the predictability or forecastability of the time series data.

Forecastability refers to the ability to predict future values in a time series. It can be assessed through various techniques, and one of them is the entropy method.

In this analysis, the entropy of the time series will be computed using a method based on the natural logarithm of the variance (square of the standard deviation) divided by two. This method considers entropy as a measure of uncertainty or variability within the dataset. Higher variability corresponds to increased entropy, serving as an indicator of forecastability, representing the predictability or stability of the data.

$$Entropy = \frac{\ln(2\pi e\sigma^2)}{2} \tag{3.20}$$

In this sense, the lower the entropy value, the higher the predictability of the series. The more repetitive the patterns in the series are, the lower the entropy, and hence, the higher the predictability.

It is important to note that this is a simple and fast approximation to calculate entropy as a measure of forecastability, as more exact methods would require too much computational cost.

In the context of feature engineering, this phase usually involves the handling of outliers and the transformation of data. Such actions facilitate the enhancement of predictive capabilities within future models intended for deriving experimental results.

Among the various methods already mentioned for outliers treatment, the Interquantile Range (IQR) method stands out as one of the most widely utilized.

In statistics, the distribution of a dataset is divided into quartiles, namely Q1 (the first quartile) and Q3 (the third quartile). The IQR is the range between these quartiles and represents the middle 50% of the data.

It is calculated by subtracting Q1 from Q3.

The process involves several steps:

1. Calculating quartiles: first, the dataset is sorted in ascending order. Q1 is the value below which 25% of the data falls, while Q3 is the value below which 75% of the data falls.
2. Computing the IQR: the IQR is determined by subtracting Q1 from Q3. This range represents the spread of the middle 50% of the data.
3. Establishing bounds for outliers: lower and upper bounds are set to identify potential outliers. These bounds are typically defined as values a certain distance (often 1.5 times the IQR) away from Q1 and Q3.
4. Identifying outliers: any data point that falls below the lower bound or above the upper bound is considered a potential outlier.
5. Treatment of outliers: outliers can be handled in various ways depending on the nature of the data and the context. Treatment options may include removal, transformation or imputation.

The IQR method is particularly useful because it is less sensitive to extreme values compared to other methods such as mean and standard deviation-based techniques. It provides a robust way to detect and manage outliers, allowing for a more accurate analysis of the dataset and improving the reliability of subsequent statistical analyses or machine learning models.

During feature engineering, another crucial task involves assessing the statistical significance of features to discern their relevance. Simultaneously, identifying and potentially discarding features that contribute insignificant or irrelevantly for modeling purposes, often termed as noise, is a common practice.

Canonical Correlation Analysis (CCA) is a statistical method used to explore the relationships between two sets of variables and to identify the strongest associations or correlations between them. It aims to find linear combinations of variables from each set that are highly correlated. It involves these key steps [61]:

1. Data preparation: organizing the dataset into two sets of variables, for example, X and Y, where X contains the features to analyze and Y represents the target or related features. The canonical variables are represented as linear combinations:

$$U = X \cdot W_x \tag{3.21}$$

$$V = Y \cdot W_y \tag{3.22}$$

   Here, U and V are the canonical variables, and W$_x$ and W$_y$ are the weight matrices for X and Y, respectively.

2. Calculating correlation: CCA seeks to find linear combinations (canonical variables) of X and Y that have maximum correlation. It uses mathematical techniques to compute correlation matrices and extract the canonical variables.
   The objective is to maximize the correlation between U and V, which is achieved by finding W$_x$ and W$_y$. The correlation between U and V is given by the canonical correlation coefficient ρ, which is calculated as follows:

$$\rho = \sqrt{\lambda_i} \tag{3.23}$$

   Where λ_i represents the canonical correlation coefficient, which measures the strength of the relationship between the canonical variables of X and Y, or eigenvalues obtained from the generalized eigenvalue problem:

$$\sum_{xx}^{-1/2} \sum_{xy} \sum_{yy}^{-1/2} = \sum_{xx}^{-1/2} \sum_{xy} \sum_{yy}^{-1/2} \Lambda \tag{3.24}$$

Here, $\sum_{xx}$ and $\sum_{yy}$ are the covariance matrices of X and Y, while $\sum_{xy}$ is the cross-covariance matrix between X and Y.
The weight matrices W_x and W_y are obtained from the eigenvectors corresponding to the largest eigenvalues of the generalized eigenvalue problem.

3. Feature selection: utilizing these formulas, CCA derives linear combinations of variables from X and Y that are highly correlated, facilitating the identification of significant relationships between the two sets of variables for effective feature selection and modeling.

Canonical Correlation Analysis holds significance as it enables the exploration of relationships between two sets of variables, unveiling the most robust associations between them. This method aids in understanding how different sets of features interact, assisting in feature selection by highlighting influential variables while disregarding less impactful ones, ultimately refining predictive models for enhanced accuracy.

The last step related to feature engineering is data normalization.

Data normalization is a critical aspect of feature engineering that addresses the issue of disparate scales or dimensions within the dataset's features. As features often vary widely in their scales and dimensions, normalization aims to bring them onto a consistent scale to facilitate effective analysis.

The mean value of each feature across the dataset is calculated. Then, this mean value is subtracted from each data point within that feature. This process centers the data around zero.

Following mean subtraction, each data point is divided by the standard deviation of the feature. This step scales the data, ensuring that each feature has a standard deviation of one. It transforms the data to have a mean of zero and a standard deviation of one.

It's worth emphasizing that during normalization, the mean and standard deviation should be computed solely using the training dataset. This is crucial to prevent any data leakage or bias in the modeling process. By exclusively using the training data, the models are trained on normalized values, ensuring that the validation and test sets remain unseen and unbiased. This approach helps maintain the integrity of the model evaluation process by preserving the separation between training and testing data.

Splitting a dataset into three distinct subsets (train, validation and test) is a fundamental procedure in machine learning for effective model development and evaluation.

- Training set: it comprises the largest portion of the dataset. It is used to train the machine learning model. During training, the model learns patterns and relationships within the data.
- Validation set: it is employed to fine-tune the model's hyperparameters and to assess its performance. By adjusting parameters based on validation set performance, the model can improve its predictive accuracy.
- Test set: this set serves as an independent dataset unseen by the model during training and validation. It is used to evaluate the final performance and generalization capability of the model. This assessment helps determine how well the model can predict outcomes on new, unseen data.

The process of splitting the dataset into these subsets ensures a structured approach to model development, preventing overfitting. Moreover, it helps in avoiding biases that might arise from using the same data for both training and evaluation purposes. This division enables a comprehensive evaluation of the model's performance, ensuring its robustness and reliability when applied to real-world scenarios.

Prior to delving into the explanation of the various models to be utilized in this Thesis, it has been considered crucial to provide an explanation of what deep learning really is.

Deep learning, as a subset of machine learning, emphasizes learning successive layers of increasingly meaningful representations from data. These layered representations are learned via neural networks that transform input data into successively different and informative representations.

Each layer within a neural network is defined by a set of weights, which essentially represent the mathematical operations performed on the input data. These weights are initially random and are adjusted iteratively during training to help the network learn to map inputs to desired outputs accurately. The network aims to minimize a loss function, a measurement that evaluates how well the network's predictions align with the actual targets [62].

During the training process, an optimizer algorithm, often based on backpropagation, refines these weights systematically. Backpropagation involves propagating the error from the network's output back through the layers, adjusting the weights along the way to minimize the difference between the predicted outputs and the actual targets. This adjustment of weights is a gradual process that occurs over numerous iterations, commonly referred to as the training loop [62].

Throughout this iterative training loop, the network's weights are continually updated to minimize the loss function, effectively steering the model towards making predictions that closely match the desired targets. This optimization process enables the network to learn from the data and improve its predictive accuracy, ultimately resulting in a trained network capable of generating outputs that are highly aligned with the intended targets [62].

Despite its complexity, the core principle of deep learning lies in these simple mechanisms scaled up, creating models that exhibit impressive capabilities when processing vast amounts of data.

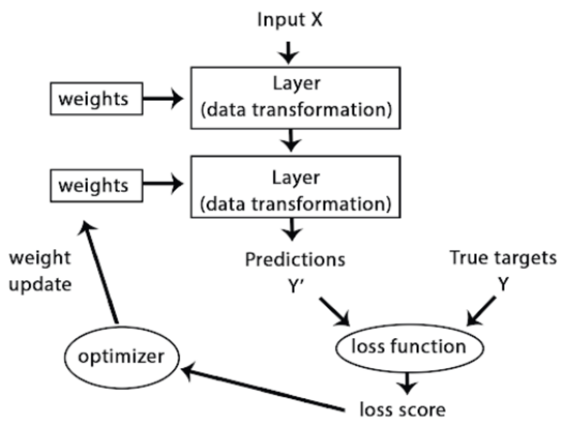


Figure 3.12: Deep learning neural network scheme [62]

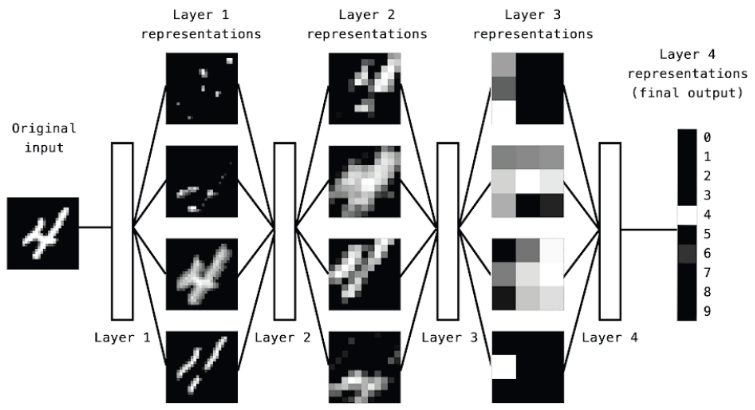


Figure 3.13: Example of deep learning model [62]

In the context of machine learning for regression and time series analysis, a model represents a mathematical or computational framework that learns patterns and relationships within the provided data to make predictions or produce desired outputs. It encompasses a set of algorithms, functions or layers organized to process input data, extracting meaningful features and generating predictions or outputs.

In this specific case, a data generator is utilized to prepare input-output pairs for training deep learning models. The generator yields tuples containing two elements: samples and targets. Samples represent a batch of input data, while targets correspond to the array of target values or predictions associated with the input samples.

The data generator accepts various arguments [62]:

- data: the original array of normalized floating-point data.
- lookback: it determines the number of time steps backward the input data spans.
- delay: specifies the number of future time steps for the target values.
- $\min_{index}$ and $\max_{index}$: indices in the data array that define the segments of data for training, validation and testing.
- shuffle: dictates whether the samples are shuffled or drawn in chronological order.
- batch size: indicates the number of samples per batch used for training.
- step: specifies the interval, in time steps, at which data is sampled.

The models employed in this scenario, such as the 1D-CNN and GRU, operate with tensorized data. Tensors are multidimensional arrays used to represent data in a format suitable for these models. Timeseries data or sequence data, for example, is usually represented by 3D tensors of shape (samples, timesteps, features).

For instance, in a time series regression problem, the shape of the tensor resulting from the data generator for samples might be (batch size, lookback, features), where batch size represents the number of samples per batch, lookback denotes the number of time steps back in the input data and features refer to the number of features or variables considered at each time step.

The tuples yielded by the data generator, comprising input data samples and corresponding target values, are structured in a way that conforms to the input requirements of the models. This organized format facilitates the models' ability to learn from the data's temporal relationships and perform regression tasks effectively. Additionally, by preserving chronological order and splitting the data into consecutive windows, the validation and test results are evaluated realistically, ensuring a robust evaluation based on data collected after training the models.

It is also important to note that in time series problems, the data is not randomly shuffled, but rather kept ordered chronologically. This is for two reasons: it ensures that it is still possible to split the data into consecutive sample windows and it ensures that the validation/test results are more realistic and evaluated on data collected after training the model.

Further exploration into the models themselves involves an evaluation of two specific types: the 1D CNN and GRU, as previously mentioned.

In the context of convolutional neural networks, convolution is a mathematical operation that combines two functions to produce a third function representing how one of the functions modifies the other. It is applied by sliding a function (referred to as a kernel or filter) over another function (referred to as input) and calculating the integral of the product of the two functions at each position. This operation is widely used in various fields such as signal processing, image processing, machine learning and general data processing.

Convolution is employed to perform operations like smoothing, feature detection, extraction of relevant information and data analysis, enabling the identification of specific patterns or features within input data.

In the context of image processing, convolution involves applying a filter over small regions of an image to extract local patterns. For instance, employing an edge-detection filter entails sliding it across the image, multiplying filter values with corresponding region values in each step. This produces a feature map where each pixel represents a weighted combination of neighboring pixel values from the original image. These filters learn specific features and as multiple convolutional layers are applied in a neural network, deeper layers can learn more complex and abstract features [62].

For instance, considering an image as a pixel matrix, convolution operates by applying a filter (kernel) over small sections of the image, performing multiplication and summation to generate a new feature map. These feature maps represent specific patterns in the image, such as edges, textures or shapes. Through multiple convolutional layers, the neural network learns hierarchies of features by combining and refining these representations to identify more intricate patterns in the original image [62].

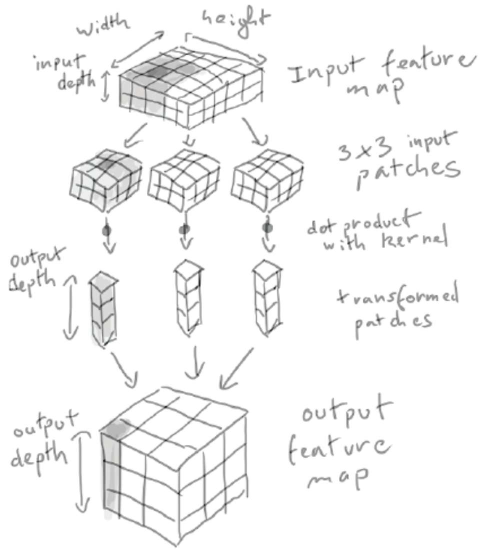


Figure 3.14: How convolution works [62]

Convolution layers operate by examining locally situated patterns within spatial data through the application of consistent geometric transformations across various spatial locations, referred to as patches, within an input tensor. This process generates representations that possess translation invariance, enhancing the efficiency and modular nature of convolution layers in handling data. Notably, this concept extends to spaces of varying dimensionality such as 1D, which is usually employed for sequence processing [62].

A Convolutional Neural Network (CNN) is a type of neural network used for processing structured data, including sequences and multidimensional arrays. CNNs are proficient in extracting intricate patterns and relationships within data. Comprised of several layers, CNNs feature distinct components like convolutional, pooling and fully connected layers. CNNs are highly adaptable and find utility in various domains, including computer vision and natural language processing.

Convolutional neural networks are comprised of layered structures integrating convolution and max-pooling layers. Max-pooling layers facilitate the spatial downsampling of data, an essential step in maintaining manageable feature map sizes amidst increasing feature complexity. This downsampling also enables subsequent convolutional layers to encompass a broader spatial context within the inputs [62].

To conclude CNNs, a common practice involves applying either a Flatten operation or a global pooling layer, transforming spatially structured feature maps into vectors. Subsequently, Dense layers are applied to execute classification or regression tasks.

Recurrent Neural Networks (RNNs) function by sequentially processing input sequences, handling one timestep at a time while retaining a continual state. This state is typically represented as a vector or a set of vectors, signifying a specific point within a geometric space of states. RNNs are recommended over 1D CNNs specifically for sequences where temporal patterns of interest aren't invariant to temporal shifts. For instance, in time series data where recent past observations hold more significance than distant historical data points, RNNs prove to be a more suitable choice [62].

The most commonly used types of RNNs are LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit), with a preference for GRU as a simpler and more cost-effective alternative to LSTM. In comparison to LSTM, GRU offers a more streamlined and resource-efficient structure. This characteristic makes GRU preferable in many cases for its ability to capture long-term dependencies in sequences of data while reducing computational complexity. This benefit translates to improved efficiency and faster training speeds for the recurrent neural network.

GRUs address typical RNN issues like long-term dependencies and problems with vanishing or exploding gradients. These units utilize gates to regulate the flow of information throughout sequences, allowing the network to learn long-range dependencies efficiently. Similar to LSTM units, GRUs manage information over time, but with a simpler structure having fewer gates, resulting in fewer parameters and often lower computational costs. They are particularly beneficial in applications requiring more efficient sequence models, such as natural language processing or sequential data analysis.

As it has been already mentioned, deep learning involves models structured in layers. In GRU and 1D CNNs, several key layers can be employed, such as: Dense, MaxPooling1D or GlobalMaxPooling. Each serves a specific purpose within the network architecture.

- Dense layer: this layer is the basic neural network layer where each neuron is connected to every neuron in the subsequent layer, allowing complex relationships to be learned between inputs and outputs [63].

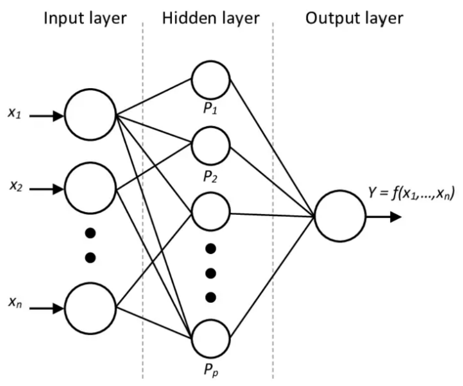


Figure 3.15: Example of network with one dense layer [63]

- MaxPooling1D layer: this layer downsamples the input representation by taking the maximum value over a specific window. It reduces the computational load and focuses on the most important features in a sequence [64].

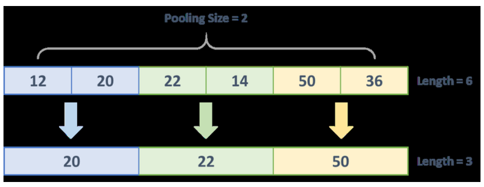


Figure 3.16: Example of MaxPooling1D [65]

- GlobalMaxPooling layer: similar to MaxPooling1D, but instead of a specific window, it takes the maximum value across the entire input sequence, capturing the most significant feature [66].

Now, moving on to fundamental concepts [67] [68]:

- Learning rate: it is a hyperparameter determining the step size at which the model adapts during training. A higher learning rate can speed up learning but may overshoot the optimal solution, while a lower rate might converge slowly.

- Optimizer: this is an algorithm that adjusts the weights of the neural network based on the loss function to minimize errors during training.
  RMS (Root Mean Square) prop, an adaptive optimizer in deep learning, addresses varying gradient problems by adjusting learning rates for individual weights. It improves convergence speed by utilizing squared gradients to control step sizes, aiming to stabilize training. However, defining an optimal learning rate manually remains a challenge across diverse applications. Its formulation is the following:

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2 \tag{3.25}$$

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w) \tag{3.26}$$

  Gradient descent, a fundamental optimization method in machine learning, utilizes gradients to iteratively update model parameters toward a local minimum. It mimics a ball rolling downhill in a bowl, seeking the steepest path to lower the cost function. While effective, it can be computationally expensive with large datasets and struggles with nonconvex functions in determining optimal steps along the gradient.

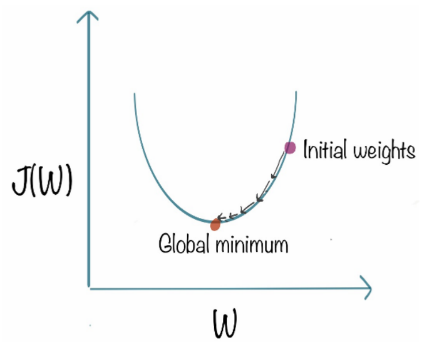


Figure 3.17: Visual representation of gradient descent [68]

  Another popular optimizer is Adam (Adaptive Moment Estimation), which combines the advantages of RMSprop and momentum optimization. It adapts the learning rates of each parameter, providing better convergence and efficiency. Adam is also one of the preferred optimizers when working with regression problems. Its formulation is the following:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\left[\frac{\delta L}{\delta w_t}\right] v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\delta L}{\delta w_t}\right]^2 \tag{3.27}$$

- Loss function: the loss function measures the difference between predicted and actual values. MAE is often preferred for its robustness against outliers. It computes the average of the absolute differences between predictions and true values.

Within different layers, various parameters can be tuned [63] [64] [69]:

- Filters: in convolutional layers, filters refer to the number of learned features. Each filter extracts different features from the input data.

- Units or dense units: indicate the number of neurons or nodes in a layer.
- Pool size: specifies the window size over which pooling operations are applied.
- Activation function: activation functions in neural networks are like decision-making tools for each neuron. They decide whether a neuron should be "activated" or not by determining the output of that neuron based on the input it receives. Think of them as filters that add non-linearities to what the neuron sees, allowing neural networks to learn complex patterns and relationships in data. These functions can transform the input data in various ways, like squishing numbers between certain ranges or only letting positive values pass through.
  The tanh activation function is a type of activation function commonly used in neural networks. It squashes input values to be within the range of -1 to 1, aiding in normalizing the output of a neural network layer. It is one of the most commonly used when working with time series problems.

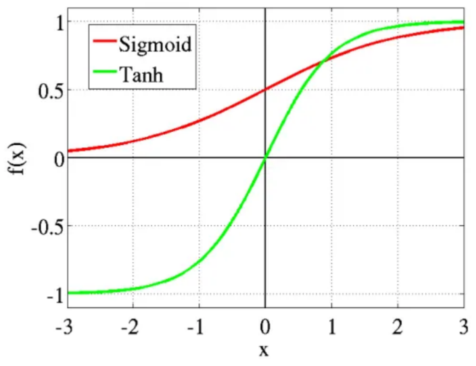


Figure 3.18: Tanh vs. logistic sigmoid functions [69]

This systematic arrangement of layers and parameters allows for the construction and optimization of deep learning models tailored to specific tasks and datasets.

Optimizing hyperparameters in deep learning is pivotal for several fundamental reasons. Hyperparameters, unlike learned parameters, significantly influence a neural network model's performance and its ability to generalize to unseen data. Properly tuning these hyperparameters can determine the difference between a well-performing model and one that fails to achieve satisfactory results.

The impact of hyperparameters is substantial, affecting network architecture, training speed, convergence and the model's generalizability to new, unseen data. Choices such as learning rate, optimizer selection, the number of layers and neurons and activation functions, among others, all play crucial roles that need careful adjustment for each specific dataset and problem.

Hyperparameter optimization involves seeking the optimal combination that maximizes the model's performance while minimizing risks of overfitting or underfitting. This process employs techniques such as grid search, random search, Bayesian optimization or other automated strategies to explore the hyperparameter space.

Last, it is important to describe a parameter already mentioned before: overfitting. Overfitting occurs when a machine learning model learns the training data too well, capturing noise or random fluctuations in the data rather than the underlying patterns. As a result, the model performs well on the training set but poorly on unseen or test data, indicating a lack of generalization.

To mitigate overfitting, various techniques can be employed [62] [70]:

- Dropout: it is a regularization technique used during training in neural networks. It randomly drops (deactivates) a fraction of neurons in a layer during each training step. This prevents individual neurons from relying too much on specific features, thus promoting more robust learning across the network. It helps prevent overfitting by creating a more generalized representation of the data.

- Recurrent dropout: specifically applied in recurrent neural networks, recurrent dropout extends the concept of dropout to recurrent connections. It drops connections between recurrent units, helping prevent overfitting in sequential data by introducing noise and reducing reliance on specific temporal patterns.
- Regularization: it is a technique used to penalize complex models by adding a regularization term to the loss function during training.
  L2 regularization (also known as weight decay) adds the squared magnitude of weights to the loss function, encouraging smaller weights and preventing any single weight from becoming excessively large. This helps in preventing the model from fitting noise in the training data and enhances its generalization to unseen data.
- Early stopping: involves monitoring the model's performance on a separate validation dataset during training. Training stops when the model's performance on the validation set starts deteriorating, even if the performance on the training set continues to improve. This prevents the model from becoming overly specialized to the training data, as it tends to start overfitting when trained for too many epochs.

These techniques collectively aim to counteract overfitting by introducing constraints or modifications during the training phase, encouraging the model to learn relevant patterns in the data while preventing it from memorizing noise or peculiarities specific to the training dataset.

Concluding this segment, a robust framework for understanding the mathematical methods and models essential in assessing engine emissions has been established. As this section draws to a close, the subsequent chapter will transition into the Exploratory Data Analysis, leveraging the groundwork laid out here. This progression marks the beginning of a comprehensive exploration of the dataset's intricacies, facilitating a deeper understanding of the data, to inform subsequent modeling and predictive analyses.

4 Exploratory Data Analysis (EDA)

4.1. Problem statement

In this Thesis, PEMS will be utilized to forecast CO and NO_x emissions from a gas turbine. A Python code will be employed via the Google Colab platform to solve this problem.

The study involves working with a gas turbine emission dataset spanning five years (from January 1st, 2011 to December 31st, 2015) [71]. The dataset consists of hourly average sensor readings of eleven variables, comprising 36733 instances. Among these variables, nine serve as input measures (independent variables), while the remaining two act as target variables. The input measurements can be grouped into ambient variables and gas turbine process parameters. The data collection takes place within an operational range spanning from partial load (75%) to full load (100%) [34].

Considering there are 1826 days between 2011 and 2015 (inclusive), theoretically yielding 43824 instances, it becomes apparent that there are missing measurements. Potential reasons for this discrepancy could be attributed to sensor failures during operation, resulting in discontinued data collection for certain hours. Additionally, it's plausible that the data collected during those hours might have been excluded from the set. Another plausible explanation is that the gas turbine wasn't operational for the entirety of those years, leading to gaps in the recorded measurements.

It is also important to note that the gas turbine which data has been collected is located in the Turkey's northwestern region according to literature [71]. This will be important as some features will be influenced by the ambient conditions (ambient variables, to say the least).

The figure below depicts the sensor placements and the sources of turbine parameters on the gas turbine.

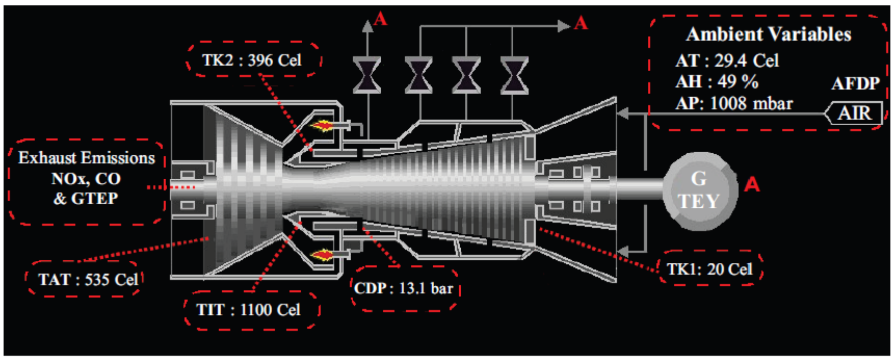


Figure 4.1: Sensor locations/parameter sources (dashed red rectangles used for parameters and dashed white arrows for sensor locations) [34]

In order to start addressing the problem subject of this Thesis, it is essential to conduct an initial Exploratory Data Analysis to gain a comprehensive understanding of the dataset in use. In the following section, the layers of the dataset will be systematically analyzed, visualizing metrics, histograms or correlation matrices. This approach aims to comprehend how each feature influences the problem, the behavior of output data or the associated physics behind these aspects.

However, it's crucial to note that not all generated graphs will be included in the main text of this work to prevent overwhelming saturation. To manage this, an appendix has been established where all figures will be provided (Appendix II). Only the most pertinent graphs for understanding the analysis and facilitating comprehension of the whole text will be incorporated herein. As for the Python code used to implement this analysis, it can be found on Appendix I.

4.2. Exploratory Data Analysis (EDA)

The initial observation reveals that there are eleven data columns present in each dataset, with identical variables found across all of them.

AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NO_x
4.5878	1018.7	83.675	3.5758	23.979	1086.2	549.83	134.67	11.898	0.32663	81.952
4.2932	1018.3	84.235	3.5709	23.951	1086.1	550.05	134.67	11.892	0.44784	82.377
3.9045	1018.4	84.858	3.5828	23.990	1086.5	550.19	135.10	12.042	0.45144	83.776
3.7436	1018.3	85.434	3.5808	23.911	1086.5	550.17	135.03	11.990	0.23107	82.505
3.7516	1017.8	85.182	3.5781	23.917	1085.9	550.00	134.67	11.910	0.26747	82.028

Table 4.1: First five lines of data for the complete dataset

These variables can be grouped in three:

- Ambient variables:
  - Ambient Temperature (AT), measured in °C.
  - Ambient Pressure (AP), measured in mbar.
  - Ambient Humidity (AH), measured in %.
- Gas turbine process parameters:
  - Air Filter Difference Pressure (AFDP), measured in mbar.
  - Gas Turbine Exhaust Pressure (GTEP), measured in mbar.
  - Turbine Inlet Temperature (TIT), measured in °C.
  - Turbine After Temperature (TAT), measured in °C.
  - Turbine Energy Yield (TEY), measured in MWH.
  - Compressor Discharge Pressure (CDP), measured in mbar.
- Emissions:
  - Carbon monoxide (CO), measured in mg/m^3.
  - Nitrogen oxides (NO_x), measured in mg/m^3.

Given the objective of predicting emissions using a set of gas turbine variables, it can be affirmed that ambient variables and process parameters will function as input variables for this model. Conversely, emissions will serve as the target variables.

With the presence of input and output variables, the aim is to approximate the mapping function. This enables to predict the output when provided with new input variables. Consequently, this scenario falls within the realm of a supervised problem. Moreover, since all variables, including the target or output, are numerical, this entails that a regression problem is being addressed.

Furthermore, the initial observation of the information associated to each of the variables reveals that all attributes are numerical (of type float) and there are no null values present in any of the datasets.

The dataset comprises hourly average sensor measurements, possibly accounting for the varying number of measurements across different years. These measurements are obtained within an operational range spanning from partial load (75%) to full load (100%). While specific times of data collection were not recorded, the dataset is organized chronologically by measurement date [71]. Consequently, the data represent distinct temporal series that can be visualized for a more comprehensive analysis of their features.

To enhance the analysis of the gas turbine's operation, the dataset will be indexed based on the hourly measurements. This indexing aims to improve data traceability and facilitate the plotting of various time series for a more comprehensive analysis.

Subsequently, an exploration of various methods to fill the missing data will be conducted. To determine the most suitable approach for handling missing values, their performance will be assessed using a basic GRU neural network model.

These steps hold significant importance since it will become evident later that certain variables exhibit a notable dependency on the time of the year under investigation. For instance, ambient temperature demonstrates a strong correlation with whether the gas turbine operates during summer or winter, displaying a recurring pattern across different years.

For this scenario, the same n value has been selected for both methods to maintain impartiality regarding which method might be superior. An n value of 2920 has been chosen, corresponding to a 4-month period. This choice aims to achieve consistency in the time series across different years, facilitating repeatability, which should be a common occurrence, at least for certain features.

It's noteworthy that certain methods, like backward fill, were not tested due to the presence of missing data spanning over one month at the end of the dataset. In this case, backward fill wouldn't function appropriately since a final value is required to propagate the missing values forward.

As it has been mentioned before, in order to not saturate this report, only the most relevant figures will be plotted. For this particular case, graphs and tables will only refer to the ambient temperature variable. The graphs and tables corresponding to the rest of the features can be found in Appendix II.

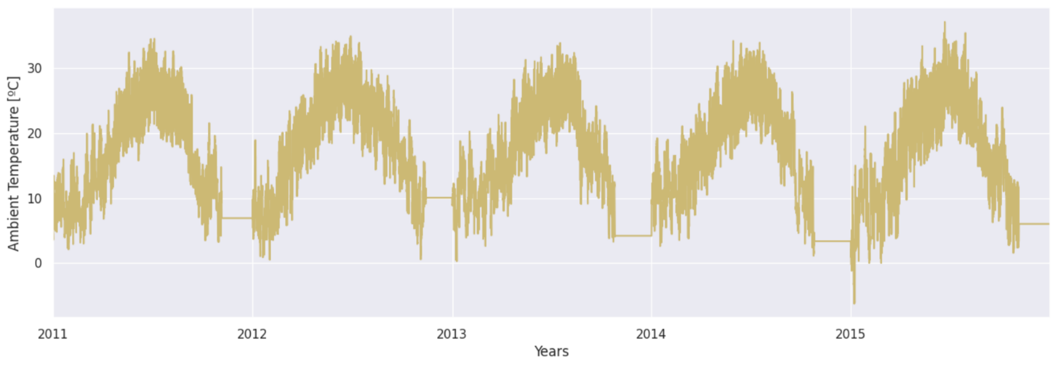


Figure 4.2: Ambient Temperature graph obtained with forward fill method for filling missing values

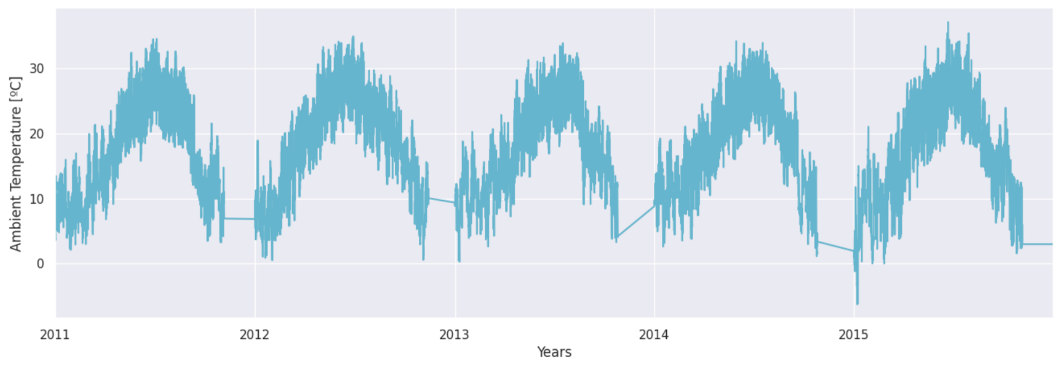


Figure 4.3: Ambient temperature graph obtained with linear interpolation method for filling missing values

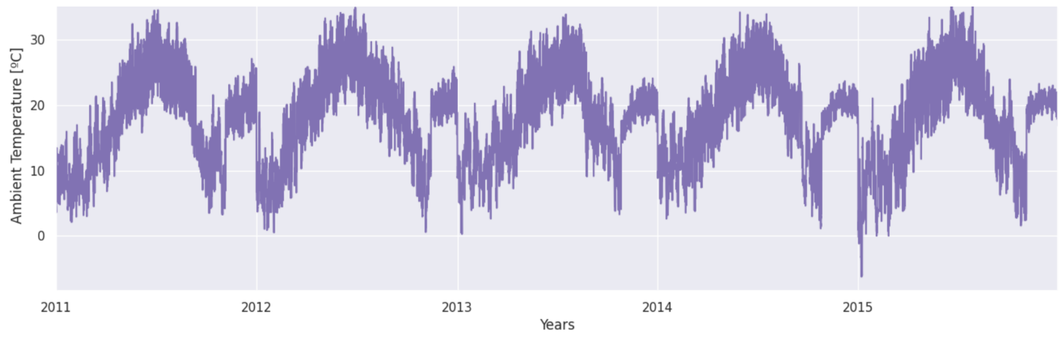


Figure 4.4: Ambient temperature graph obtained with seasonal mean (n=2920) method for filling missing values

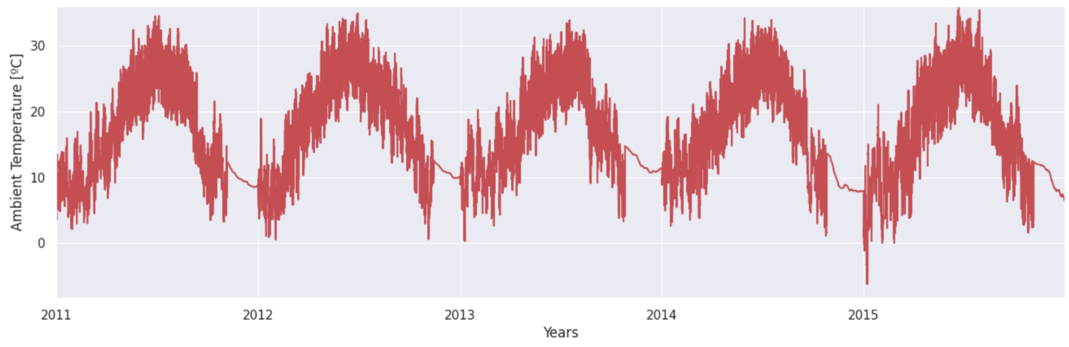


Figure 4.5: Ambient temperature graph obtained with KNN mean (n=2920) method for filling missing values

	MSE	MAE
Forward Fill	73.67	7.36
Linear Interpolation	80.25	7.64
Seasonal Mean	49.01	5.66
KNN Mean	57.85	6.52

Table 4.2: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for ambient temperature feature

Upon visualizing the outcomes obtained from various methods, the determination is that, for the time series under examination, the most fitting approach would be Seasonal Mean. Across the 11 variables analyzed, it showcases the lowest Mean Absolute Error (MAE) and Mean Squared Error (MSE) in 8 out of 11 instances (although the lowest MAE does not invariably translate to the lowest MSE).

Based on the visualization of the resultant time series, it can be observed that the Seasonal Mean method provides the most closely aligned time series resembling what would be expected in reality. There is an exception noted with ambient temperature, where the results obtained could be realistic in certain scenarios (such as an approximate ambient temperature of 20°C in winter, which might seem unusual but not as implausible as obtaining 60°C). It's evident that dividing the result by 2 would address this issue, and this adjustment will be implemented in the subsequent section to obtain the final feature.

However, the examination of these variations was conducted using a straightforward method. Consequently, by delving into the analysis of the model's hyperparameters, even more enhanced outcomes could be attained. Additionally, it's crucial to highlight that the refinement of results could also be achieved through the treatment of outliers, but this approach was deemed enough for the initial phase of this analysis.

The Seasonal Mean method will now be applied to all variables to obtain the complete dataset for further study in this thesis. Additionally, to facilitate visualization and comparison of these graphs, they will be smoothed using Locally Weighted Scatterplot Smoothing (LOWESS).

For this particular case, it is indeed considered relevant to plot the corresponding figures for all the variables, as they will underpin many of the future sections.

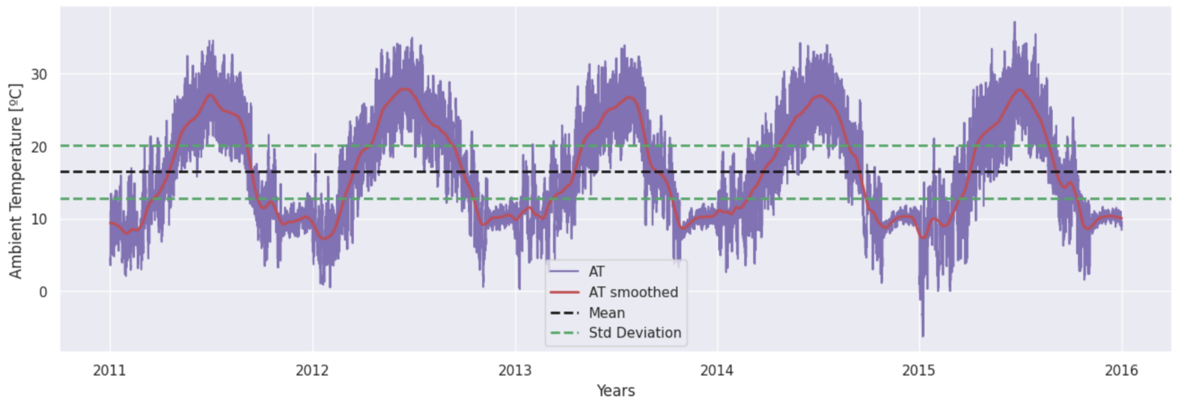


Figure 4.6: Ambient temperature with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation)

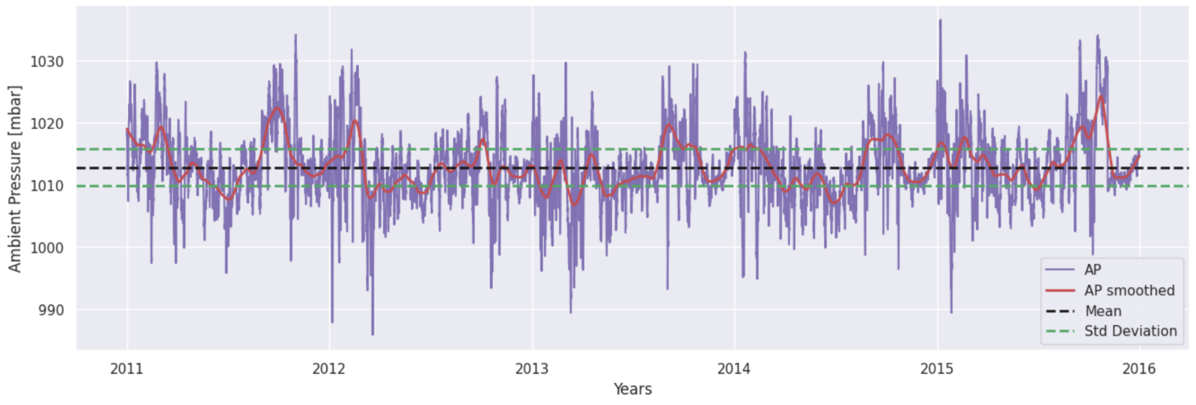


Figure 4.7: Ambient pressure with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation)

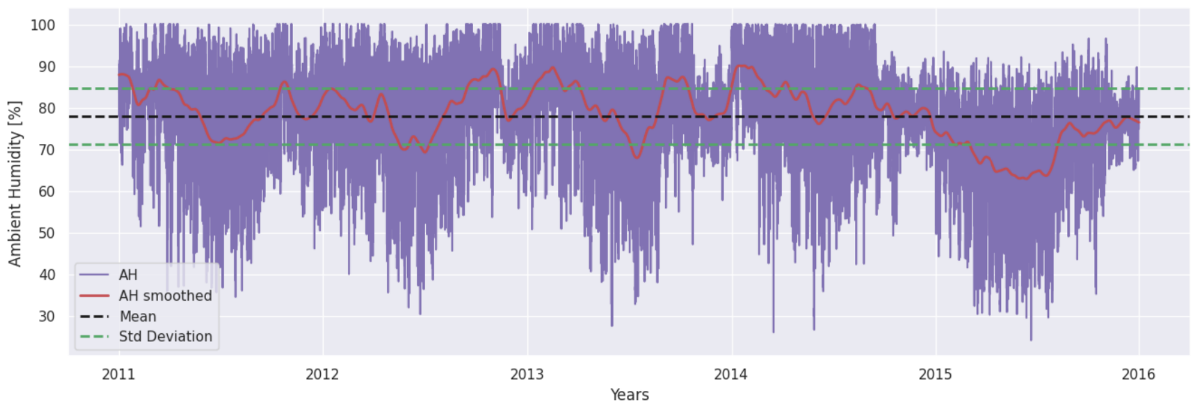


Figure 4.8: Ambient humidity with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation)

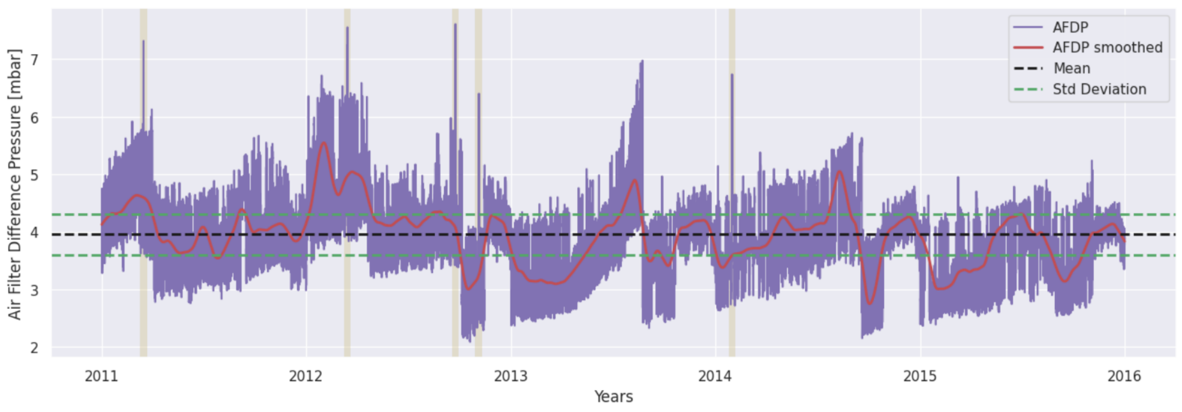


Figure 4.9: Air filter difference pressure with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation) with abnormal values highlighted

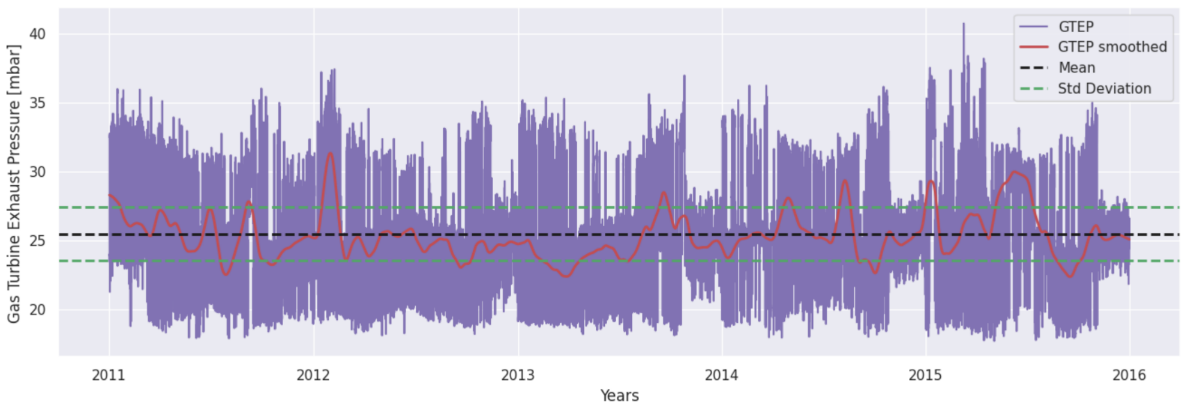


Figure 4.10: Gas turbine exhaust pressure with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation)

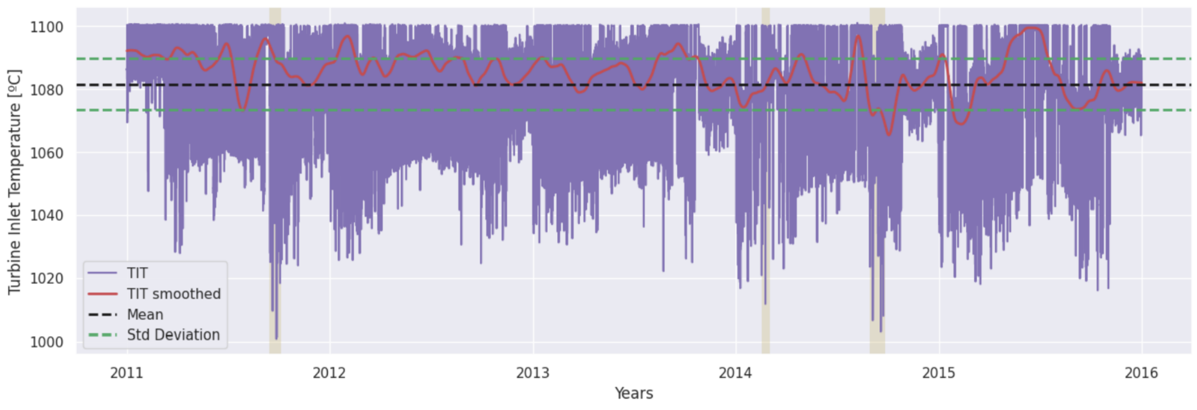


Figure 4.11: Turbine inlet temperature with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation) with abnormal values highlighted

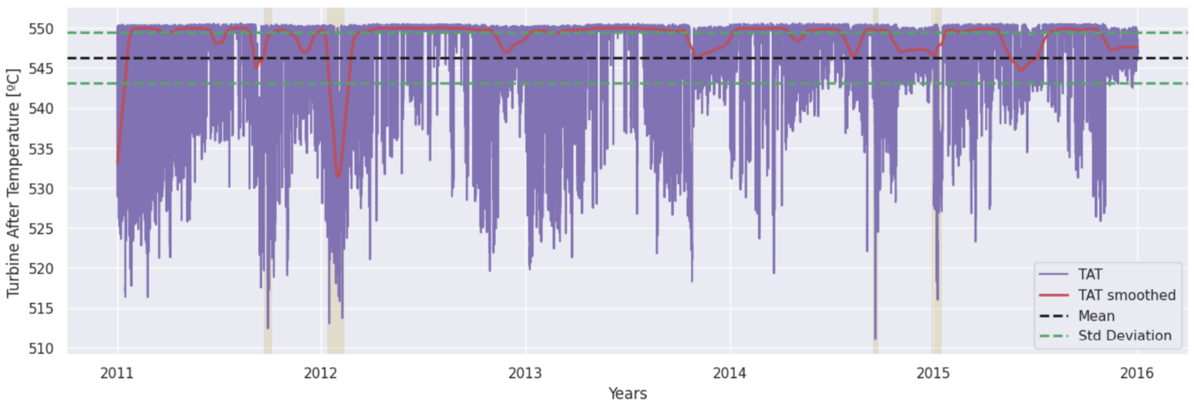


Figure 4.12: Turbine after temperature with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation) with abnormal values highlighted

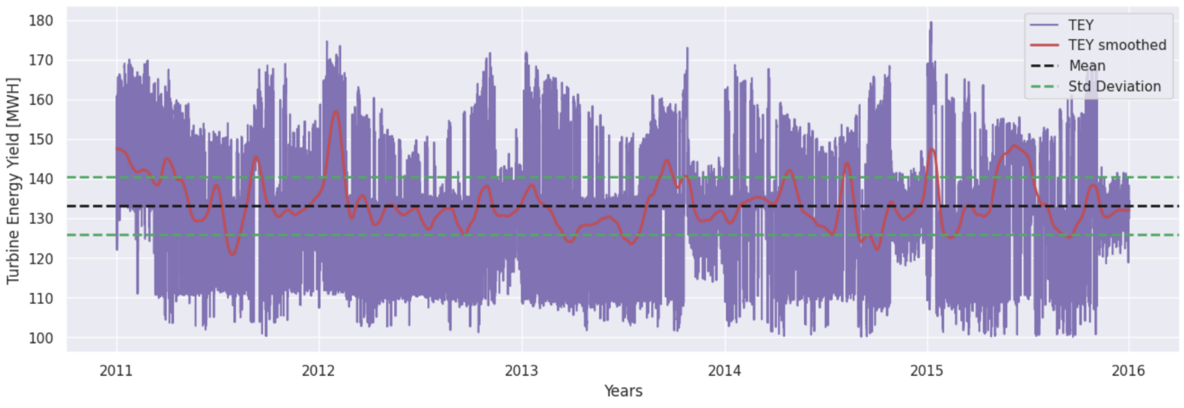


Figure 4.13: Turbine energy yield with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation)

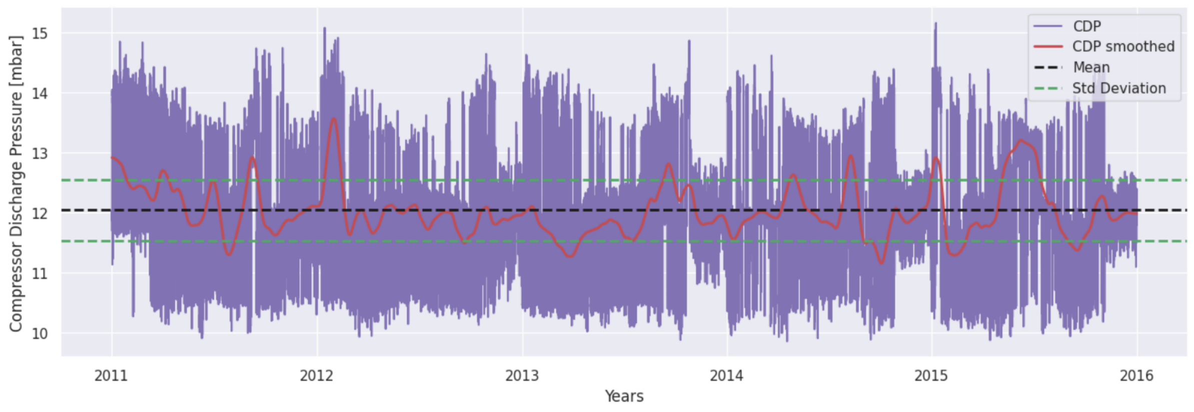


Figure 4.14: Compressor discharge pressure with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation)

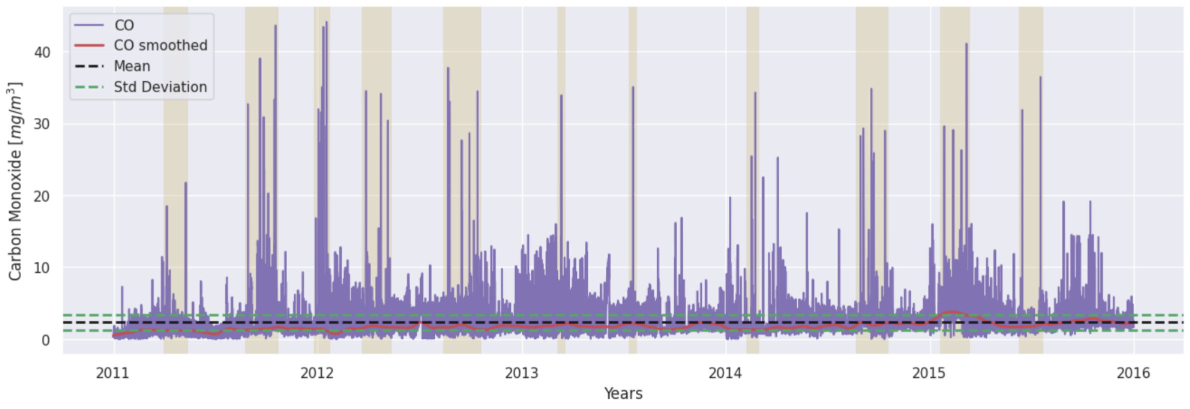


Figure 4.15: Carbon monoxide emissions with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation) with abnormal values highlighted

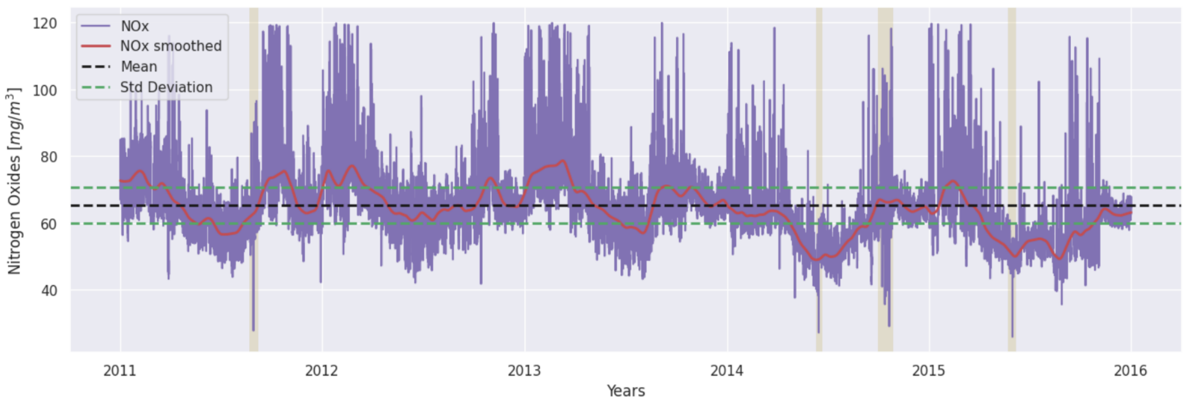


Figure 4.16: Nitrogen oxides emissions with Seasonal Mean (n=2920) for filling missing values and LOWESS smoothing (showing mean and standard deviation) with abnormal values highlighted

Based on these graphs, initial assumptions can be formulated, which will subsequently be elucidated and expanded upon in further detail.

The repetitive pattern previously mentioned for certain variables is already discernible, notably in ambient temperature, which exhibits the most conspicuous repetition.

In the case of other graphs, the proximity of values results in a significant level of noise upon slight changes, making it challenging to discern the actual trend followed by the variable. To address this issue, smoothed graphs have also been plotted to offer a clearer depiction of the underlying tendencies.

Finally, even though this marks the conclusion of the Exploratory Data Analysis, certain graphs exhibit indications of outliers. For instance, a significant majority of carbon monoxide emissions values fall within the range of 0 to 10 mg/m^3. However, there are numerous values that extend well beyond this interval. This observation suggests the potential existence of numerous outliers for this particular variable, necessitating further examination and consideration for appropriate treatment.

Peaks, potentially indicating outliers, are also noticeable in air filter difference pressure, turbine inlet temperature, turbine after temperature or nitrogen oxides emissions. Certain peaks have been highlighted in the graphs for visual emphasis, although not all have been highlighted to avoid overwhelming the graph with excessive noise.

As previously indicated, ambient variables such as temperature, pressure and humidity are expected to correlate with the typical values in the region where the gas turbine operates. For instance, during winter months, temperatures should be lower compared to summer months, while pressure may correspond to the local weather patterns characteristic of different periods throughout the year in that region.

Before proceeding with this Data Analysis, it's essential to acknowledge that certain variables will exhibit similar patterns due to their direct interrelation governed by thermodynamic laws.

For instance, upon reviewing the preceding graphs, it is likely that air filter difference pressure and gas turbine exhaust pressure are correlated. Additionally, gas turbine exhaust pressure, turbine energy yield and compressor discharge pressure appear to be interconnected variables based on their observed patterns.

Prior to delving further into this aspect, it has been deemed essential for this analysis to initially generate histograms illustrating the periodicity of the various variables at hand.

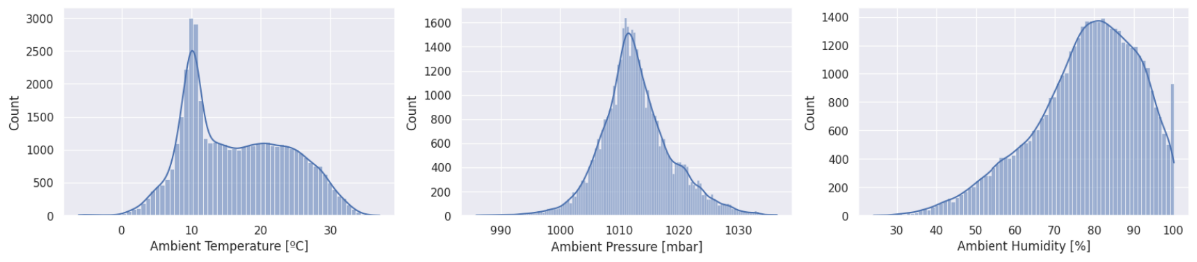


Figure 4.17: Histograms for ambient temperature, ambient pressure and ambient humidity, respectively

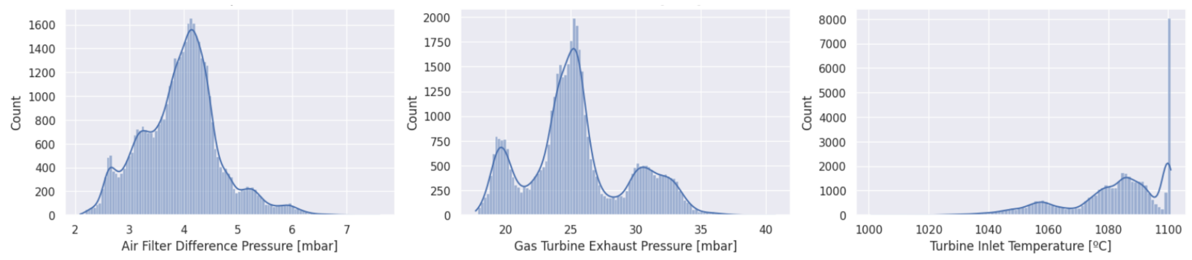


Figure 4.18: Histograms for air filter difference pressure, gas turbine exhaust pressure and turbine inlet temperature, respectively

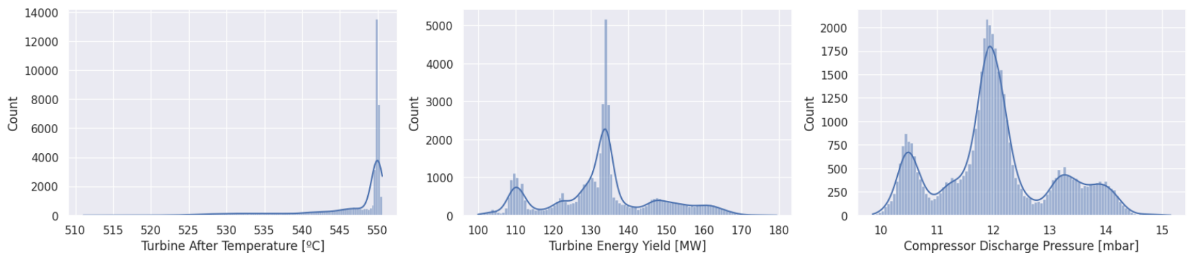


Figure 4.19: Histograms for turbine after temperature, turbine energy yield and compressor discharge pressure, respectively

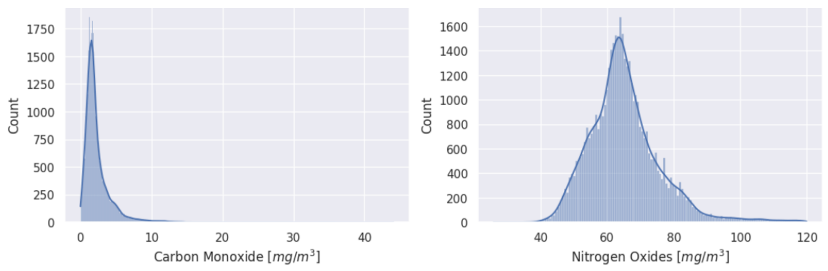


Figure 4.20: Histograms for carbon monoxide emissions and nitrogen oxides emissions, respectively

Given that the data originates from a gas turbine situated in Turkey's northwestern region [71], variables associated with temperature, pressure, and humidity are profoundly influenced by the climate prevailing in that specific area.

As an initial observation, examining these three graphs suggests that the data likely corresponds to a location near or similar to Istanbul. Ambient pressures exhibit minor fluctuations around the reference ambient pressure (1013 mbar or 1 atm), while ambient temperatures align with the typical data linked to this region of the globe, ranging between 5 and 30°C depending on the time of year [72].

It can be inferred that certain features exhibit values that significantly outnumber the counts of other recorded values, indicating a higher frequency of repetition. In addition to the previously mentioned features like ambient temperature and ambient pressure, higher occurrences of specific values in ambient humidity are attributable to the proximity of the gas turbine's location to the sea or its location in a very humid or rainy region. Moreover, the most frequently repeated values in turbine inlet temperature and turbine after temperature, surpassing other values in recurrence, likely stem from the gas turbine's operation adhering to a well-defined thermodynamic cycle. This cycle is typically optimized to ensure the gas turbine generates maximum thrust while minimizing fuel consumption, achieving a balanced solution.

These graphs reveal discernible patterns indicating that certain features adhere to well-known distributions:

- Ambient temperature demonstrates characteristics akin to a binomial distribution, albeit exhibiting slight deformations at certain points. It displays a relatively uniform distribution of values, mirroring the behavior observed in ambient pressure and nitrogen oxides emissions variables.
- Ambient humidity follows a Poisson distribution, skewed left, similar to the distribution observed in turbine after temperature.
- Carbon monoxide emissions conform to a skewed right distribution.
- The remaining features depict multimodal distributions.

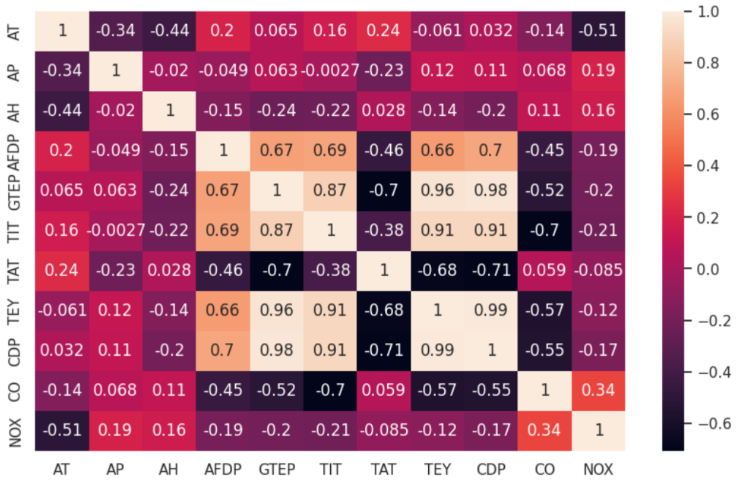
Another noteworthy observation derived from the histograms, particularly when focusing solely on the target variables, is that nitrogen oxides emissions exhibit considerably higher values compared to carbon monoxide emissions.

Certain values within the dataset also appear to be misattributed to specific features. For instance:

- Gas turbine exhaust pressure should resemble ambient pressure, yet this similarity is not observed.
- Compressor discharge pressure values are expected to be higher than the recorded values in the dataset. Upon closer inspection, it appears that the feature might represent the compressor Pressure Ratio (PR), which typically assumes a dimensionless nature, rather than compressor discharge pressure.

At this stage, the intention is not to analyze the outliers. However, it is worth noting that certain histograms display an elongated distribution, stretching far from the area where the most frequently occurring values are situated. This suggests that these variables are likely to contain outliers.

For enhanced analysis and understanding of relationships between features and target variables, it is often advantageous to visualize the correlation matrix among the variables.



	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NOX
AT	1	-0.34	-0.44	0.2	0.065	0.16	0.24	-0.061	0.032	-0.14	-0.51
AP	-0.34	1	-0.02	-0.049	0.063	-0.0027	-0.23	0.12	0.11	0.068	0.19
AH	-0.44	-0.02	1	-0.15	-0.24	-0.22	0.028	-0.14	-0.2	0.11	0.16
AFDP	0.2	-0.049	-0.15	1	0.67	0.69	-0.46	0.66	0.7	-0.45	-0.19
GTEP	0.065	0.063	-0.24	0.67	1	0.87	-0.7	0.96	0.98	-0.52	-0.2
TIT	0.16	-0.0027	-0.22	0.69	0.87	1	-0.38	0.91	0.91	-0.7	-0.21
TAT	0.24	-0.23	0.028	-0.46	-0.7	-0.38	1	-0.68	-0.71	0.059	-0.085
TEY	-0.061	0.12	-0.14	0.66	0.96	0.91	-0.68	1	0.99	-0.57	-0.12
CDP	0.032	0.11	-0.2	0.7	0.98	0.91	-0.71	0.99	1	-0.55	-0.17
CO	-0.14	0.068	0.11	-0.45	-0.52	-0.7	0.059	-0.57	-0.55	1	0.34
NOX	-0.51	0.19	0.16	-0.19	-0.2	-0.21	-0.085	-0.12	-0.17	0.34	1

Figure 4.21: Pearson correlation matrix

The initial examination of the matrix reveals a trend that might have been apparent from the outset: ambient variables and gas turbine process variables generally exhibit weak correlations. Conversely, gas turbine process parameters showcase strong interrelationships among themselves, wherein turbine after temperature displays the weakest correlation among these parameters.

Moreover, it's evident that carbon monoxide emissions are more significantly impacted by gas turbine process variables compared to nitrogen oxides emissions.

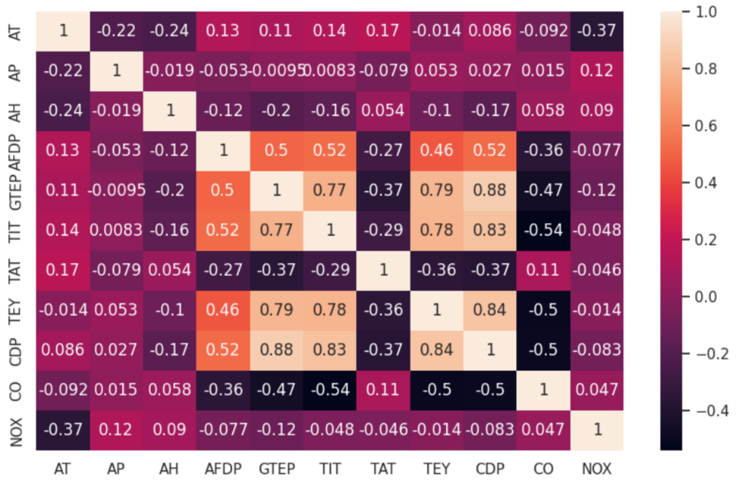
Upon closer examination of the target variables, it becomes apparent that lower turbine inlet temperature correlates with increased production of carbon monoxide emissions. This relationship aligns logically, considering that incomplete combustion at lower temperatures tends to generate more CO, as indicated by the negative correlation coefficient of approximately -0.7.

Regarding nitrogen oxides emissions, the most prominent association identified is with ambient temperature, exhibiting a correlation coefficient of approximately -0.51. This suggests that operating at higher temperatures serves as an effective method to reduce NO_x emissions.

It's notably evident that compressor discharge pressure, gas turbine exhaust pressure, turbine energy yield and turbine inlet temperature exhibit strong interrelationships among themselves.

Air filter difference pressure and turbine after temperature also demonstrate notable relationships with those variables, albeit not as robust as the previously mentioned associations.

The decision was made to contrast the linear correlation matrix acquired via the Pearson coefficient with the non-linear correlation matrix obtained using the Kendall coefficient. Kendall was preferred over Spearman due to its tendency to yield more reliable outcomes, particularly in scenarios featuring outliers within the dataset (as previously observed and set for analysis later on).



	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NOX
AT	1	-0.22	-0.24	0.13	0.11	0.14	0.17	-0.014	0.086	-0.092	-0.37
AP	-0.22	1	-0.019	-0.053	-0.0095	0.0083	-0.079	0.053	0.027	0.015	0.12
AH	-0.24	-0.019	1	-0.12	-0.2	-0.16	0.054	-0.1	-0.17	0.058	0.09
AFDP	0.13	-0.053	-0.12	1	0.5	0.52	-0.27	0.46	0.52	-0.36	-0.077
GTEP	0.11	-0.0095	-0.2	0.5	1	0.77	-0.37	0.79	0.88	-0.47	-0.12
TIT	0.14	0.0083	-0.16	0.52	0.77	1	-0.29	0.78	0.83	-0.54	-0.048
TAT	0.17	-0.079	0.054	-0.27	-0.37	-0.29	1	-0.36	-0.37	0.11	-0.046
TEY	-0.014	0.053	-0.1	0.46	0.79	0.78	-0.36	1	0.84	-0.5	-0.014
CDP	0.086	0.027	-0.17	0.52	0.88	0.83	-0.37	0.84	1	-0.5	-0.083
CO	-0.092	0.015	0.058	-0.36	-0.47	-0.54	0.11	-0.5	-0.5	1	0.047
NOX	-0.37	0.12	0.09	-0.077	-0.12	-0.048	-0.046	-0.014	-0.083	0.047	1

Figure 4.22: Kendall correlation matrix

Upon comparing both matrices, it is apparent that the relationship between variables generally appears weaker in the non-linear analysis compared to the linear assessment. However, the discrepancy is relatively minor. Therefore, it is reasonable to assume that the results derived from the linear matrix can be considered valid and applicable throughout the experiment.

At this juncture, it becomes crucial to consider the physical context of the problem as it offers significant insights into the relationships between various variables, serving as a complement to the Data Analysis.

Multiple interactions within the gas turbine process parameters need consideration, as these parameters adhere to fundamental thermodynamic laws.

In a gas turbine system, a rise in turbine inlet temperature implies a greater amount of thermal energy accessible for conversion into work within the turbine. Consequently, this leads to an elevation in the power output. This correlation plays a pivotal role in the energy performance of the turbine, as a higher inlet temperature typically augments the turbine's efficiency and power generation capacity. Conversely, a lower inlet temperature can negatively impact its performance.

Consequently, turbine after temperature should elevate in tandem with turbine energy yield.

Raising compressor discharge pressure (representing compressor pressure ratio) signifies that the compressor must exert more effort to attain a heightened pressure level. This action could potentially affect the overall efficiency of the gas turbine system. Consequently, it could also contribute to an increase in turbine energy yield. However, practical limitations may exist regarding the extent to which compressor discharge pressure can be elevated.

Moreover, it's typically observed that gas turbine exhaust pressure tends to enhance turbine energy yield by extracting additional work from exhaust gases.

In the context of a gas turbine, when the compressor elevates air pressure (compressor discharge pressure), it typically implies that the turbine extracts more energy from the exhaust gases (resulting in higher gas turbine exhaust pressure). Consequently, these factors are interconnected: increased compression often correlates with higher exhaust pressure.

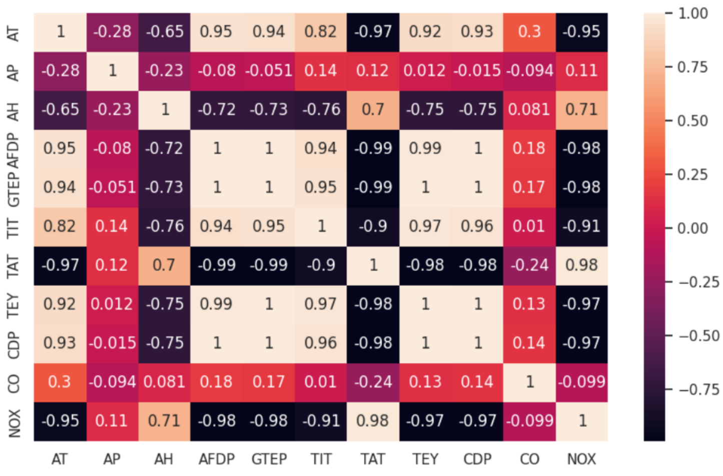
Ambient temperature and ambient pressure have a direct influence on turbine energy yield. Elevated ambient temperature can potentially diminish turbine energy yield due to the presence of less dense air and reduced oxygen available for combustion. Conversely, increased ambient pressure generally augments turbine energy yield by supplying a greater volume of air for combustion.

It's crucial to recognize that these relationships represent ideal scenarios. In reality, the operation of a gas turbine is influenced by numerous variables that extend beyond the scope of this particular study.

Another significant aspect to consider regarding ambient variables is the application of the ideal gas law, which states that P·V=n·R·T. From this law, it can be deduced that $\rho = P/RT$. Although this law is purely theoretical and represents an ideal scenario, it implies the existence of a mathematical expression linking ambient pressure and ambient temperature. This point will be further developed in the next chapter of this work.

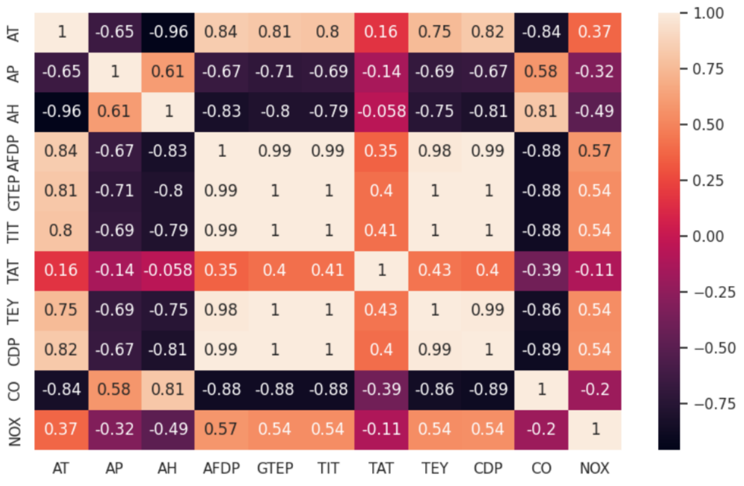
To delve deeper into this topic, the next step involves visualizing the correlation matrix for the initial year (2011) during two distinct seasons of the year: summer and winter. This analysis will focus on January 1st and August 1st as representative days for these seasons.

This approach will facilitate the identification of correlations that might be evident only during particular seasons of the year.



	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NOX
AT	1	-0.28	-0.65	0.95	0.94	0.82	-0.97	0.92	0.93	0.3	-0.95
AP	-0.28	1	-0.23	-0.08	-0.051	0.14	0.12	0.012	-0.015	-0.094	0.11
AH	-0.65	-0.23	1	-0.72	-0.73	-0.76	0.7	-0.75	-0.75	0.081	0.71
AFDP	0.95	-0.08	-0.72	1	1	0.94	-0.99	0.99	1	0.18	-0.98
GTEP	0.94	-0.051	-0.73	1	1	0.95	-0.99	1	1	0.17	-0.98
TIT	0.82	0.14	-0.76	0.94	0.95	1	-0.9	0.97	0.96	0.01	-0.91
TAT	-0.97	0.12	0.7	-0.99	-0.99	-0.9	1	-0.98	-0.98	-0.24	0.98
TEY	0.92	0.012	-0.75	0.99	1	0.97	-0.98	1	1	0.13	-0.97
CDP	0.93	-0.015	-0.75	1	1	0.96	-0.98	1	1	0.14	-0.97
CO	0.3	-0.094	0.081	0.18	0.17	0.01	-0.24	0.13	0.14	1	-0.099
NOX	-0.95	0.11	0.71	-0.98	-0.98	-0.91	0.98	-0.97	-0.97	-0.099	1

Figure 4.23: Pearson correlation matrix for winter period (January 1st, 2011)



	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NOX
AT	1	-0.65	-0.96	0.84	0.81	0.8	0.16	0.75	0.82	-0.84	0.37
AP	-0.65	1	0.61	-0.67	-0.71	-0.69	-0.14	-0.69	-0.67	0.58	-0.32
AH	-0.96	0.61	1	-0.83	-0.8	-0.79	-0.058	-0.75	-0.81	0.81	-0.49
AFDP	0.84	-0.67	-0.83	1	0.99	0.99	0.35	0.98	0.99	-0.88	0.57
GTEP	0.81	-0.71	-0.8	0.99	1	1	0.4	1	1	-0.88	0.54
TIT	0.8	-0.69	-0.79	0.99	1	1	0.41	1	1	-0.88	0.54
TAT	0.16	-0.14	-0.058	0.35	0.4	0.41	1	0.43	0.4	-0.39	-0.11
TEY	0.75	-0.69	-0.75	0.98	1	1	0.43	1	0.99	-0.86	0.54
CDP	0.82	-0.67	-0.81	0.99	1	1	0.4	0.99	1	-0.89	0.54
CO	-0.84	0.58	0.81	-0.88	-0.88	-0.88	-0.39	-0.86	-0.89	1	-0.2
NOX	0.37	-0.32	-0.49	0.57	0.54	0.54	-0.11	0.54	0.54	-0.2	1

Figure 4.24: Pearson correlation matrix for summer period (August 1st, 2011)

From these matrices, notable observations emerge.

During summer, there is a stronger correlation among ambient variables compared to the winter period. Additionally, it's intriguing to note that while ambient pressure and humidity exhibit an inverse relationship during winter, they demonstrate a direct relationship during summer.

Overall, ambient temperature displays weaker correlations with gas turbine process parameters during summer compared to winter, except for three instances: notably high correlations are observed with turbine after temperature (-0.97 compared to 0.16) and nitrogen oxides emissions (-0.95 compared to 0.37) during winter. Additionally, there is a correlation factor of 0.3 for carbon monoxide emissions during winter, contrasting with a -0.84 factor during summer.

Ambient pressure, however, generally exhibits stronger correlations with gas turbine process parameters during summer. Concerning ambient humidity, the correlation values remain similar between both seasons, except for specific variables: turbine after temperature (0.7 during winter and -0.058 during summer), carbon monoxide emissions (0.081 during winter and 0.81 during summer) and nitrogen oxides emissions (0.71 during winter and -0.49 during summer).

Among gas turbine process parameters, turbine after temperature exhibits notably stronger correlations with other parameters during winter compared to summer. Notably, during winter, it displays an inverse relationship with other parameters, whereas during summer, it demonstrates a direct relationship with all other parameters.

In winter, nitrogen oxides emissions showcase stronger correlations with these parameters, whereas for carbon monoxide emissions, the opposite trend is observed.

Regarding the correlation between both types of emissions, their relationship is twice as strong during summer compared to winter, although the correlation value in either case is not particularly high.

Now, supplementary graphs supporting all the hypotheses aforementioned will be presented below.

Thus, it is possible to visualize some of the hypotheses made earlier, particularly emphasizing the dependency among all gas turbine process parameters, with specific emphasis on turbine energy yield, compressor discharge pressure and gas turbine exhaust pressure.

For improved visualization of dependencies between target variables and other features, lmplots will be employed.

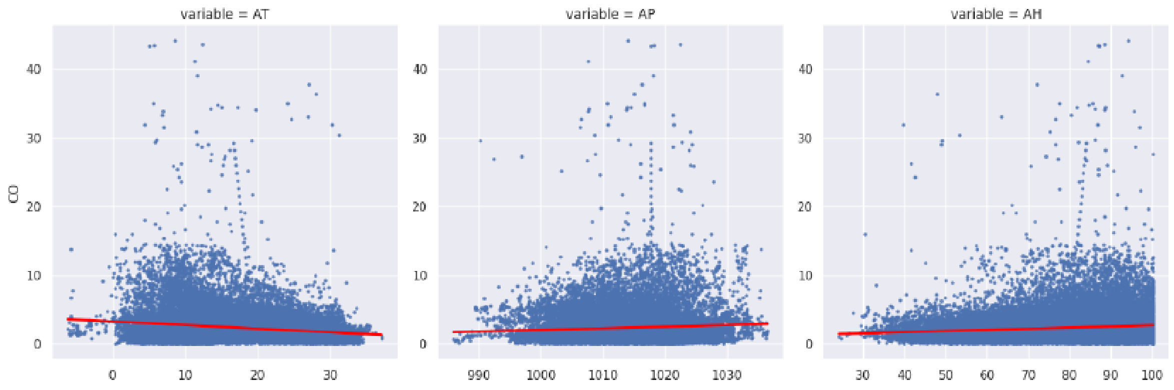


Figure 4.25: Lmplot for carbon monoxide emissions (I)

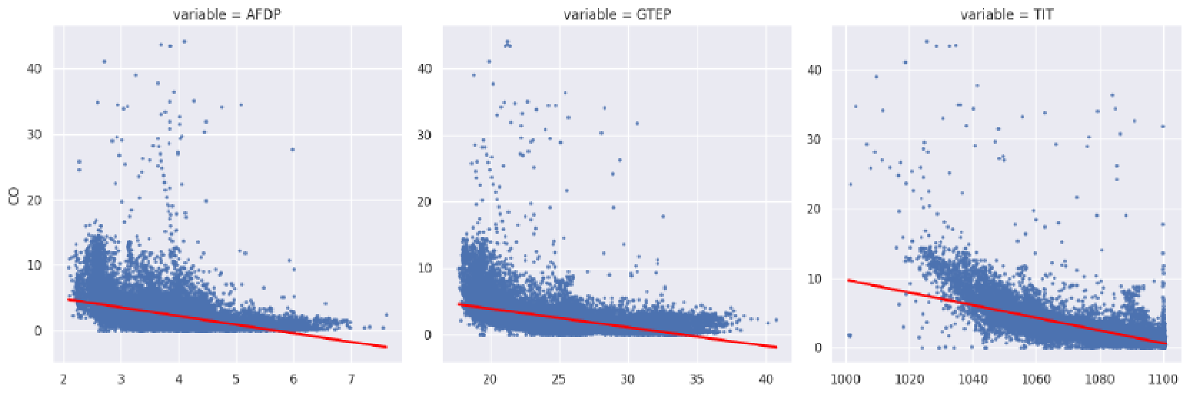


Figure 4.26: Lmplot for carbon monoxide emissions (II)

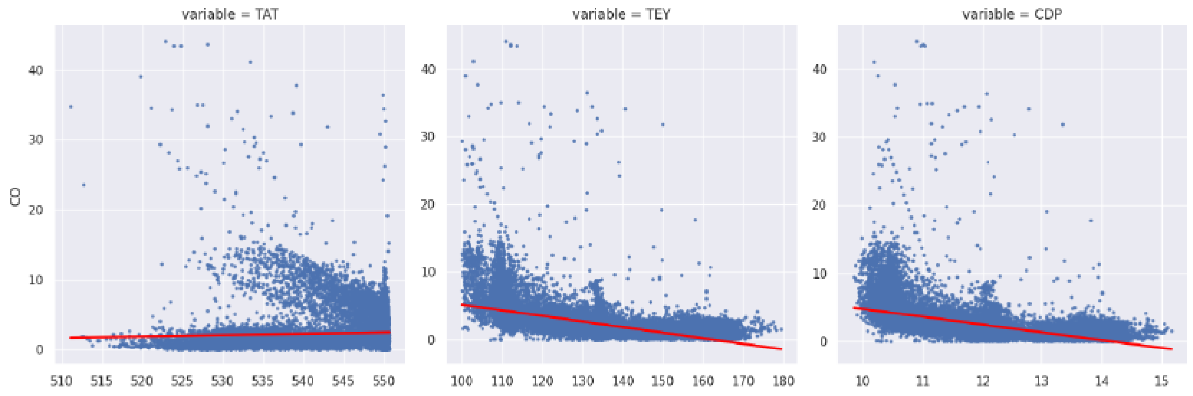


Figure 4.27: Lmplot for carbon monoxide emissions (III)

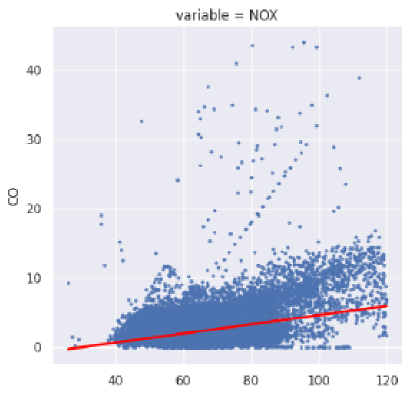


Figure 4.28: Lmplot for carbon monoxide emissions (IV)

As previously mentioned, the dependency of carbon monoxide emissions primarily relates to gas turbine process parameters rather than ambient variables. Additionally, as previously observed, the most significant relationship exists with turbine inlet temperature.

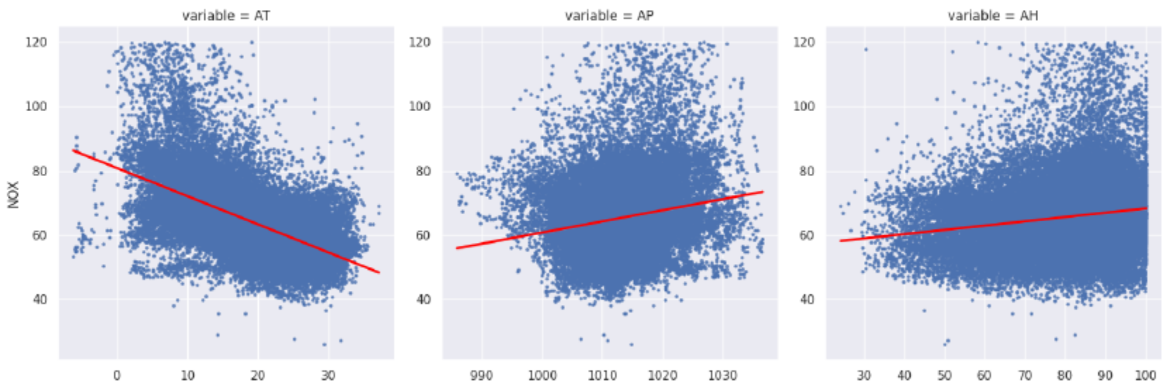


Figure 4.29: Lmplot for nitrogen oxides emissions (I)

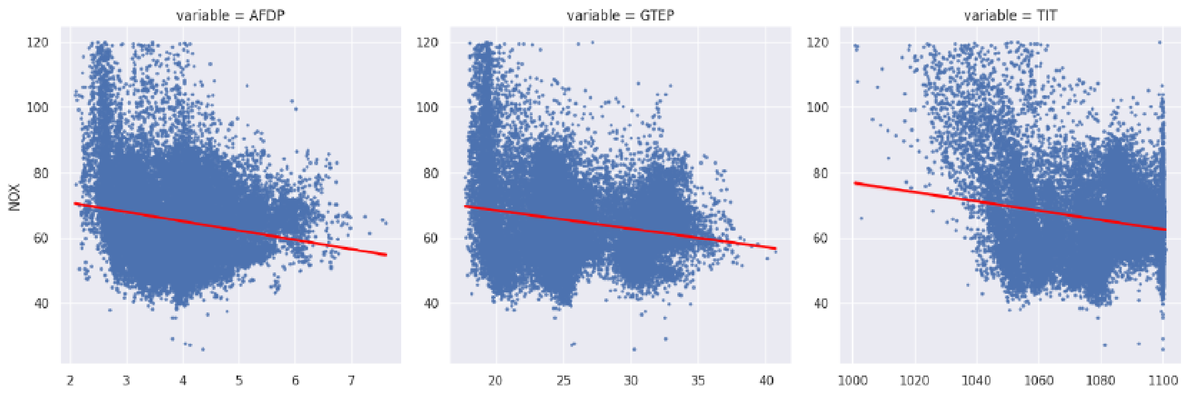


Figure 4.30: Lmplot for nitrogen oxides emissions (II)

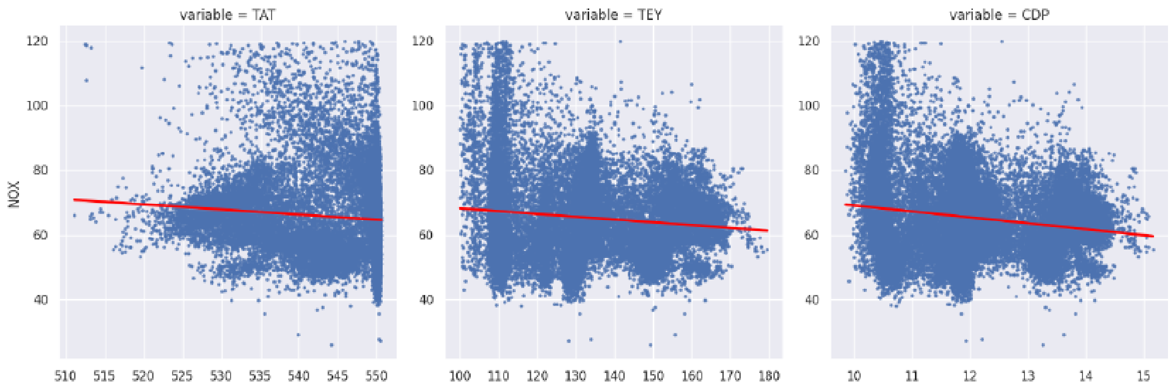


Figure 4.31: Lmplot for nitrogen oxides emissions (III)

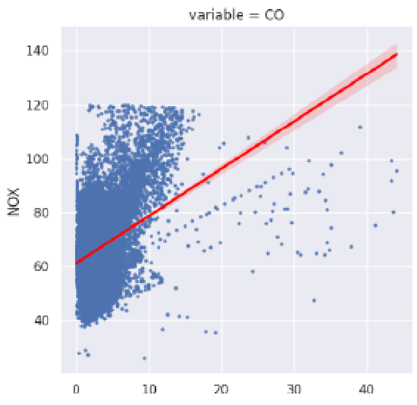


Figure 4.32: Lmplot for nitrogen oxides emissions (IV)

In the case of nitrogen oxides emissions, their correlation leans more towards ambient variables, although they are also influenced by gas turbine process parameters. As previously observed, the most prominent relationship exists with ambient temperature. It is noteworthy that carbon monoxide emissions appear to exert a substantial influence on nitrogen oxides emissions as well.

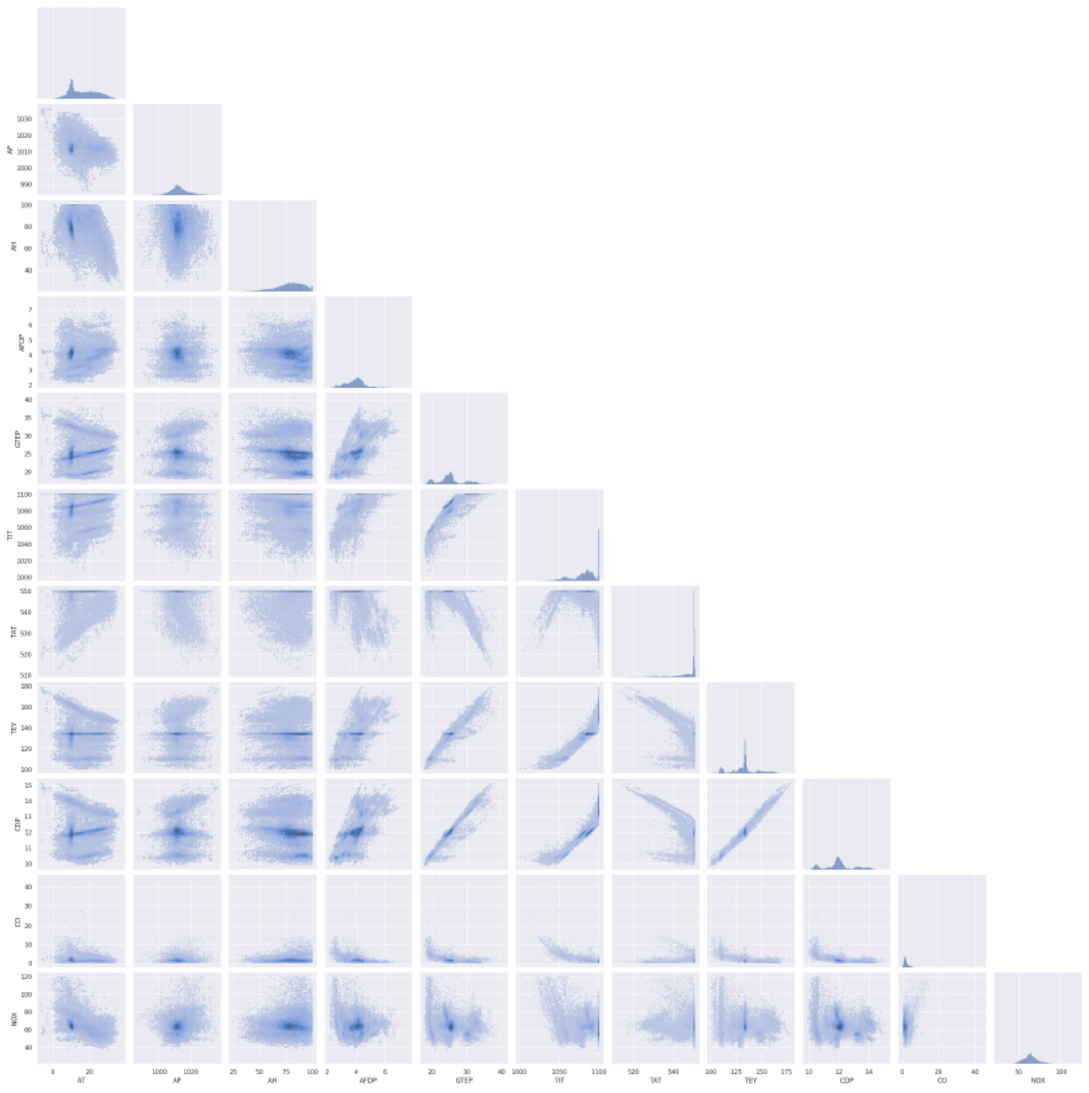


Figure 4.33: Pairplot for the different variables

To validate the previously made assumptions, seasonal plots will be generated to confirm which variables exhibit a recurring pattern across different years. To achieve this, a new smoothing technique, Savitzky-Golay filtering, will be utilized.

A consolidated approach involving the plotting of multiple time series on the same graph will be implemented to verify the applicability of the earlier hypotheses concerning relationships between variables.

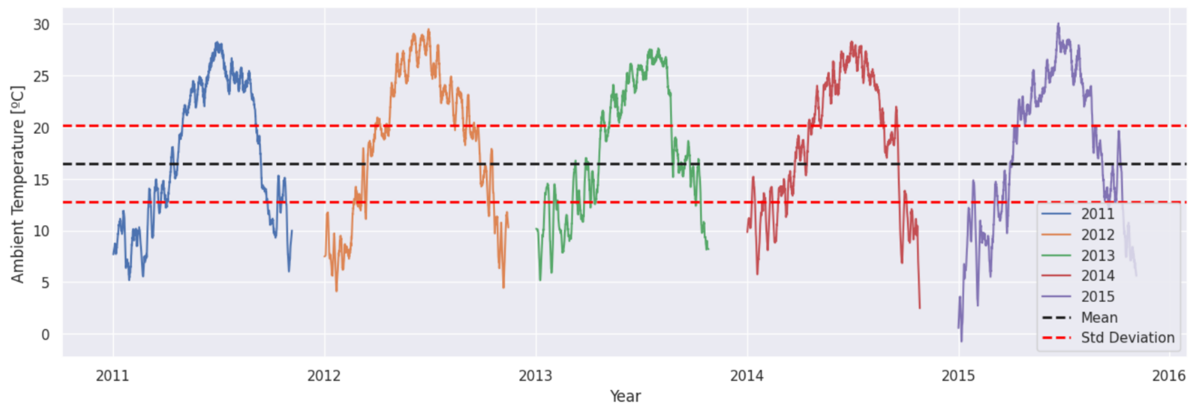


Figure 4.34: Ambient temperature seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

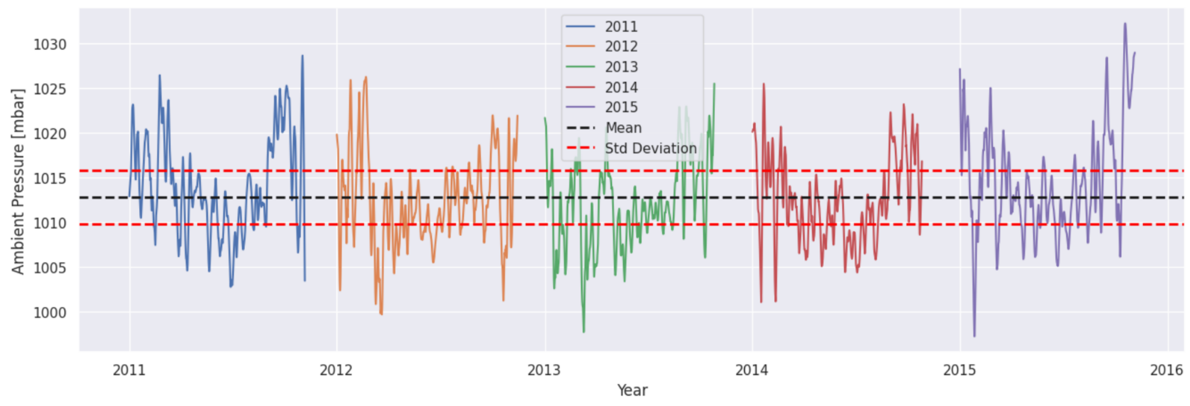


Figure 4.35: Ambient pressure seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

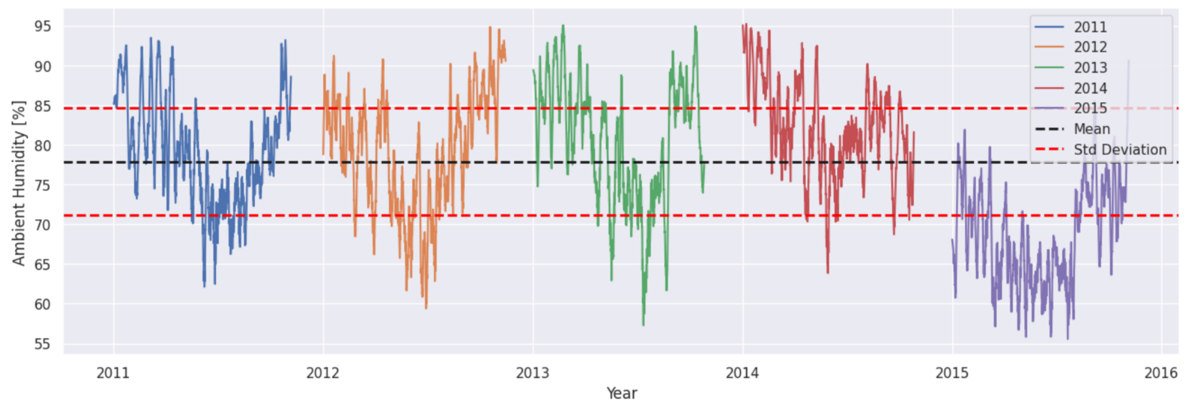


Figure 4.36: Ambient humidity seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

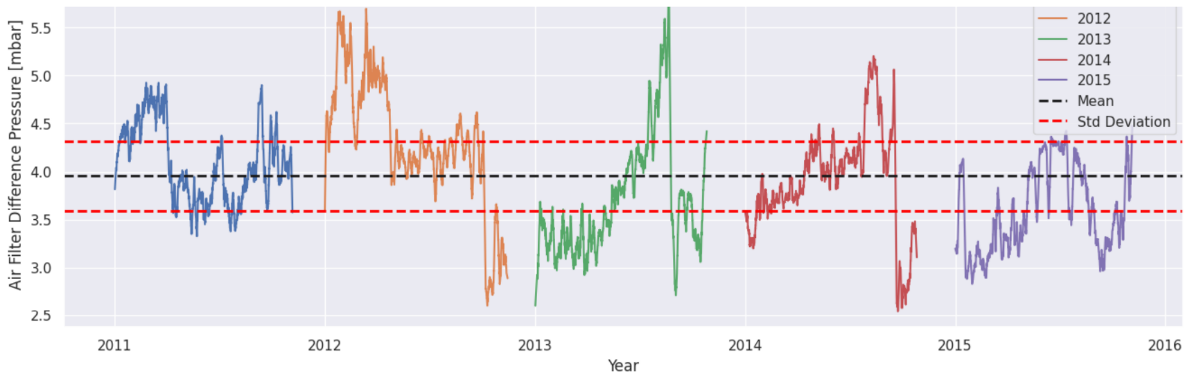


Figure 4.37: Air filter difference pressure seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

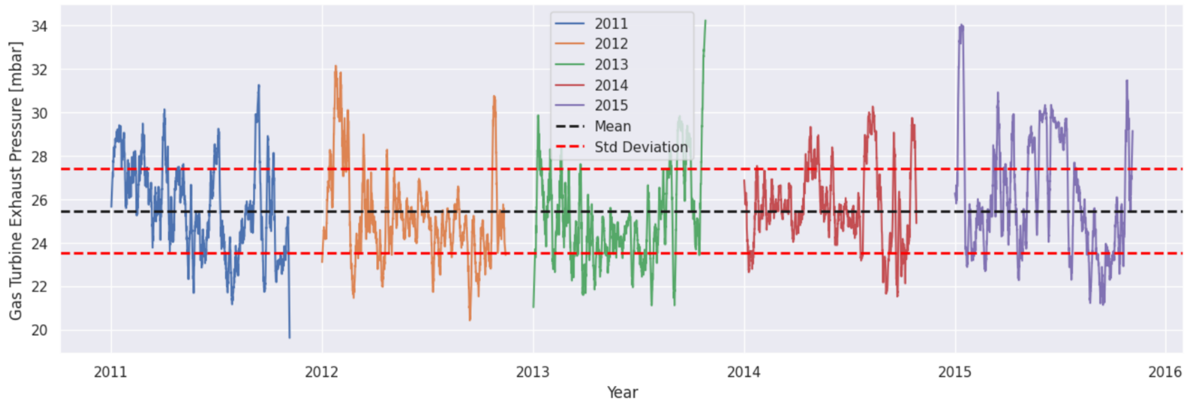


Figure 4.38: Gas turbine exhaust pressure seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

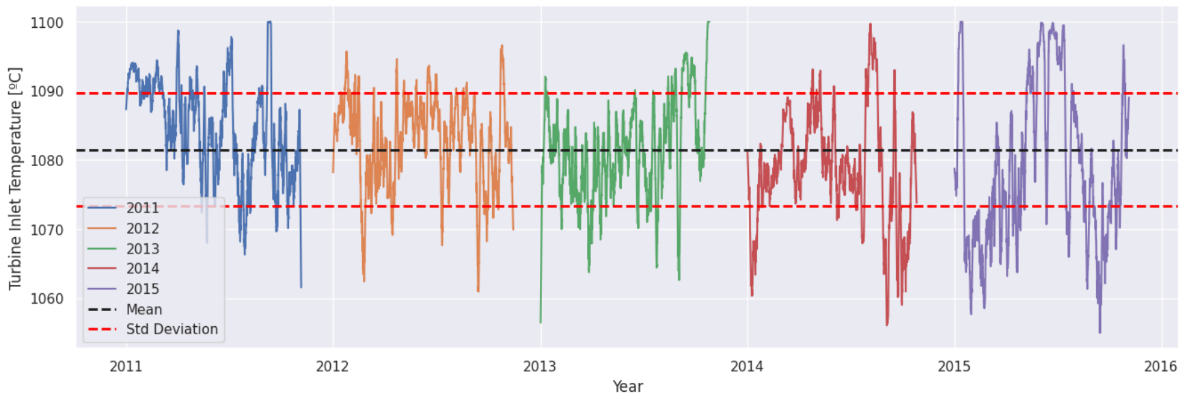


Figure 4.39: Turbine inlet temperature seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

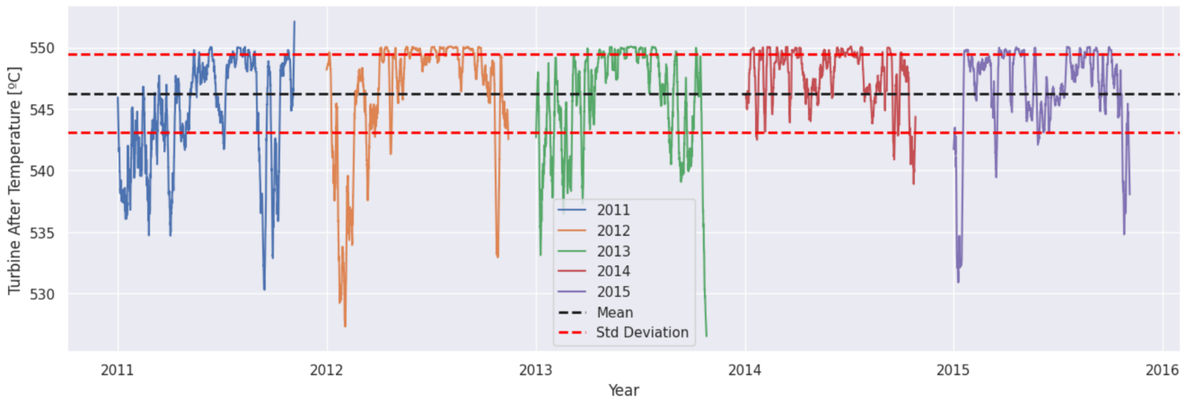


Figure 4.40: Turbine after temperature seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

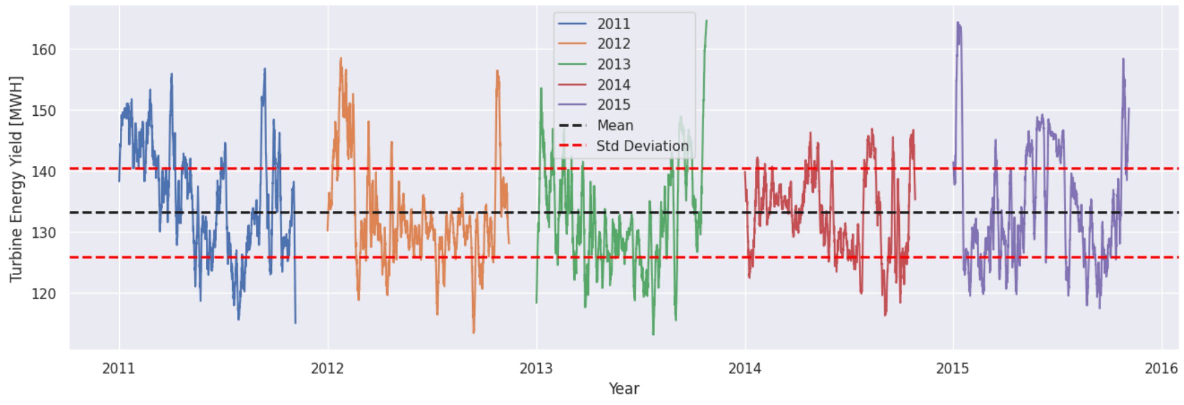


Figure 4.41: Turbine energy yield seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

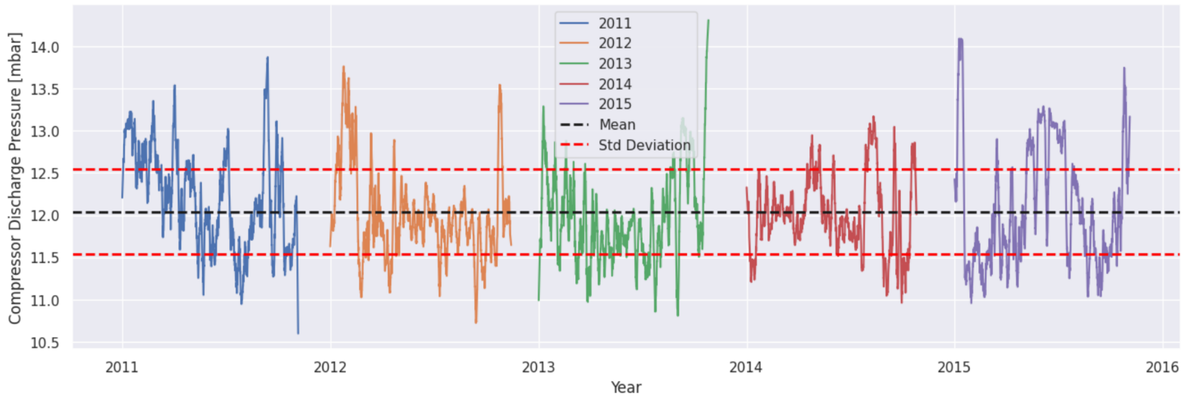


Figure 4.42: Compressor discharge pressure seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

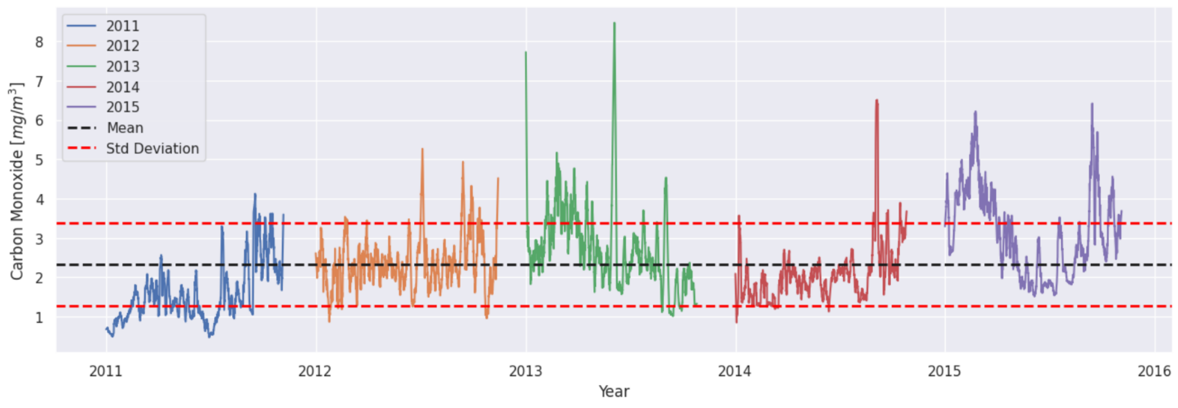


Figure 4.43: Carbon monoxide emissions seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

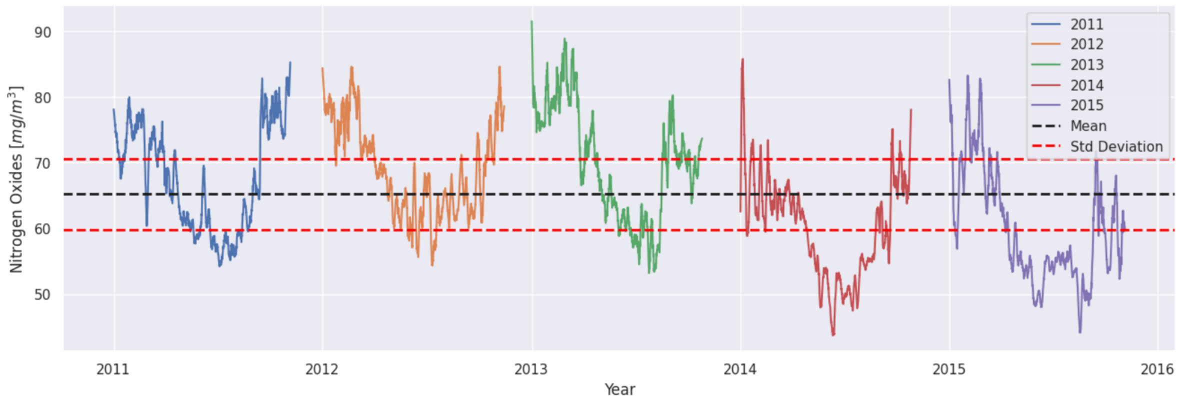


Figure 4.44: Nitrogen oxides emissions seasonal plot with Savitzky-Golay filtering (showing mean and standard deviation)

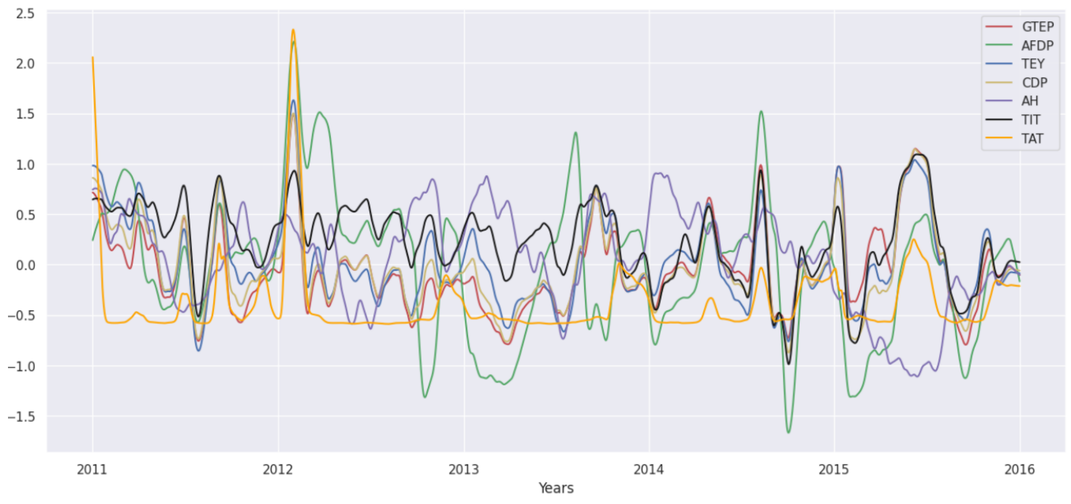


Figure 4.45: Time series comparison after normalization

As anticipated, these graphs furnish us with valuable insights and information.

Ambient temperature demonstrates a recurring pattern that repeats annually, aligning with the typical temperatures anticipated in each month within the Turkish region [72].

Ambient pressure is also expected to exhibit a recurring pattern. However, visualizing this pattern is challenging due to the closely proximate values.

Ambient humidity appears to be significantly influenced by other ambient variables. However, a discernible pattern is observed solely for the initial three years. Previous plots have indicated notably lower humidity values specifically in 2015 compared to the other years.

Gas turbine exhaust pressure, being significantly influenced by ambient variables, exhibits a recurring pattern that repeats annually, although certain values stand out from the norm, potentially indicating outliers.

Turbine inlet and after temperatures adhere to a similar pattern observed in ambient pressure; however, the closely proximate values between them make it challenging to definitively confirm the existence of distinct patterns.

Finally, it appears that nitrogen oxides emissions exhibit a consistent trend across the recorded years, albeit with potential outliers present. The trend notably falters during 2014 and 2015, where more distant values are observed, potentially indicating the presence of outliers during those periods.

An intriguing observation is the remarkable similarity in the trajectory of turbine energy yield, compressor discharge pressure and gas turbine exhaust pressure across the various years, a pattern more distinctly visible in the final graph. However, there is a notable divergence in the trajectories of ambient humidity, air filter difference pressure and turbine inlet temperature, although they exhibit some semblance of similarity among themselves.

The behavior of turbine after temperature is notably intriguing, as it demonstrates an inverse pattern compared to the other variables, albeit with periods where the divergence is significantly more pronounced.

Ambient temperature and ambient pressure have not been included in the plot, as it is evident that they do not exhibit a similar trend as the other variables.

The next step in this Data Analysis involves conducting a time series decomposition for the various variables. For this particular case, only additive decomposition can be used, as multiplicative doesn't work correctly for data with negative values (which is the case for some of the present features).

As before, in order to not saturate this work, only the graphs corresponding to ambient temperature will be presented.

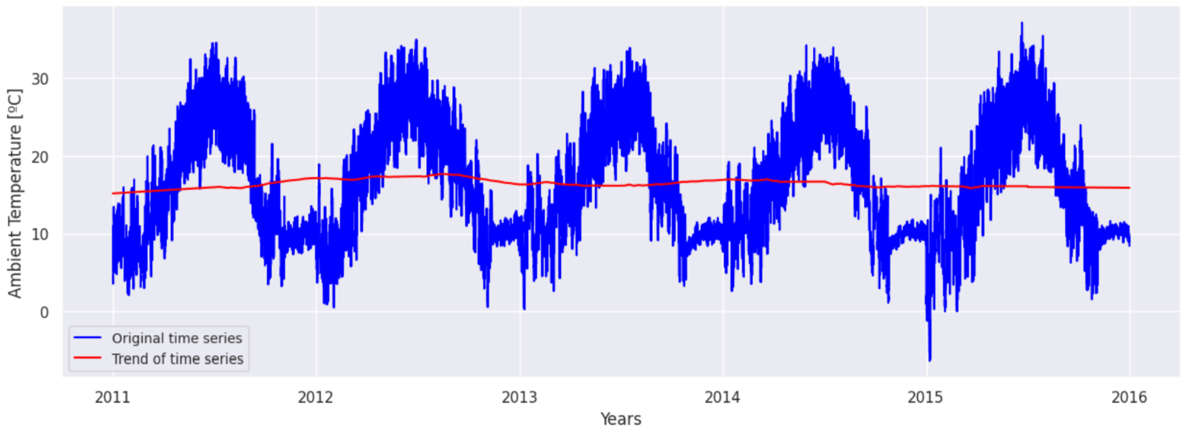


Figure 4.46: Original time series and trend for ambient temperature feature with additive decomposition

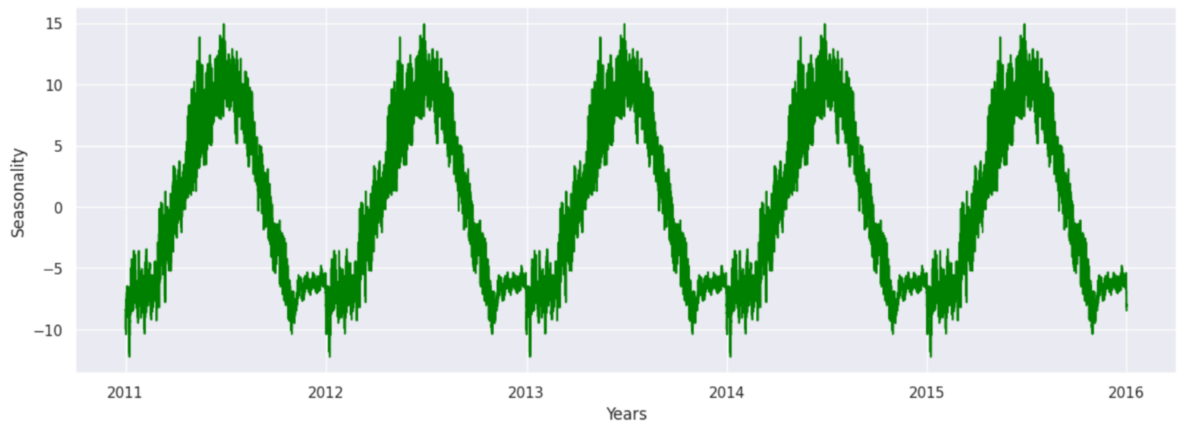


Figure 4.47: Seasonality plot for ambient temperature feature with additive decomposition

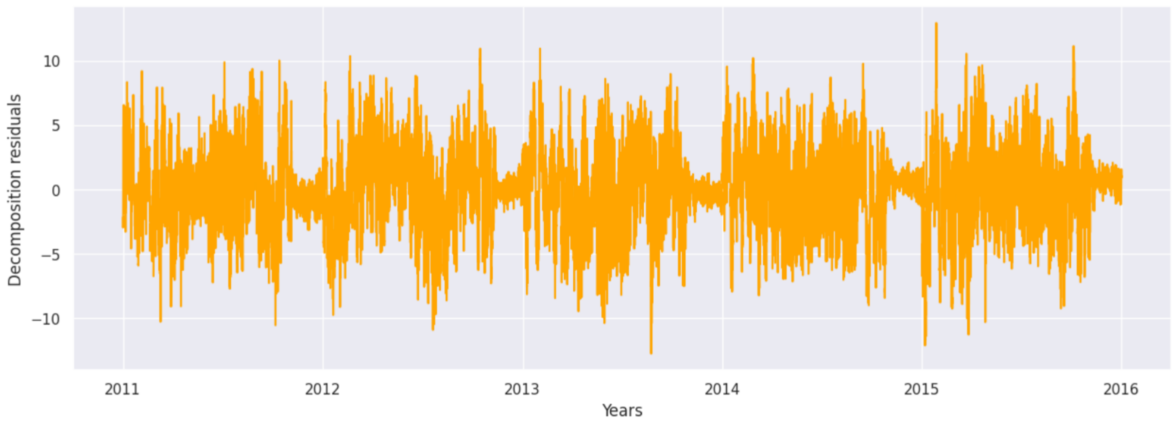


Figure 4.48: Decomposition residuals (noise) for ambient temperature feature with additive decomposition

Several observations align with previously noted patterns and trends evident in the data.

- Ambient temperature is almost perfectly periodic between the different years, with residuals being quite low.
- Trend is almost constant for ambient temperature, ambient pressure, gas turbine exhaust pressure, turbine inlet temperature, turbine energy yield, compressor discharge pressure and carbon monoxide emissions.
- Ambient humidity loses its trend during 2014 and 2015, and the residual is higher than for the previous variables.
- Air filter difference pressure has a varying trend but a very low residual. The residual value for compressor discharge pressure is also very low.
- Turbine inlet temperature and turbine energy yield have a much higher residual value than turbine after temperature and gas turbine exhaust pressure.
- Residuals for carbon monoxide and nitrogen oxides emissions are very high.

As anticipated from earlier hypotheses, the only variable distinctly exhibiting a periodic trend is ambient temperature. Variables with elevated residual values are likely to encompass a higher concentration of outliers.

Subsequently, assessing the stationarity of the time series proves beneficial, for which the implementation of ADF and KPSS tests will be considered:

	ADF	KPSS
AT	$1.30 \cdot 10^{-5}$	0.087
AP	$1.87 \cdot 10^{-26}$	0.070
AH	$5.61 \cdot 10^{-26}$	0.010
AFDP	$4.67 \cdot 10^{-15}$	0.010
GTEP	$2.03 \cdot 10^{-28}$	0.010
TIT	$4.73 \cdot 10^{-29}$	0.010
TAT	$9.31 \cdot 10^{-25}$	0.010
TEY	$7.08 \cdot 10^{-28}$	0.012
CDP	$2.44 \cdot 10^{-28}$	0.021
CO	$2.29 \cdot 10^{-28}$	0.010
NO_x	$1.14 \cdot 10^{-14}$	0.010

Table 4.3: Results for ADF and KPSS tests for the different features

Examining the outcomes, it becomes apparent that the trends deemed stationary include ambient humidity, air filter difference pressure, gas turbine exhaust pressure, turbine inlet temperature, turbine after temperature, turbine energy yield, carbon monoxide emissions and nitrogen oxides emissions.

None of the time series can be considered stationary as per this test (which makes sense with the previous visualized graphs).

To extend this analysis, the next step involves attempting to detrend the time series. In this scenario, the approach involves subtracting the trend component acquired from the earlier time series decomposition.

For this specific case, the only figures that will be presented hereafter are the ones for ambient pressure, carbon monoxide emissions and nitrogen oxides emissions.

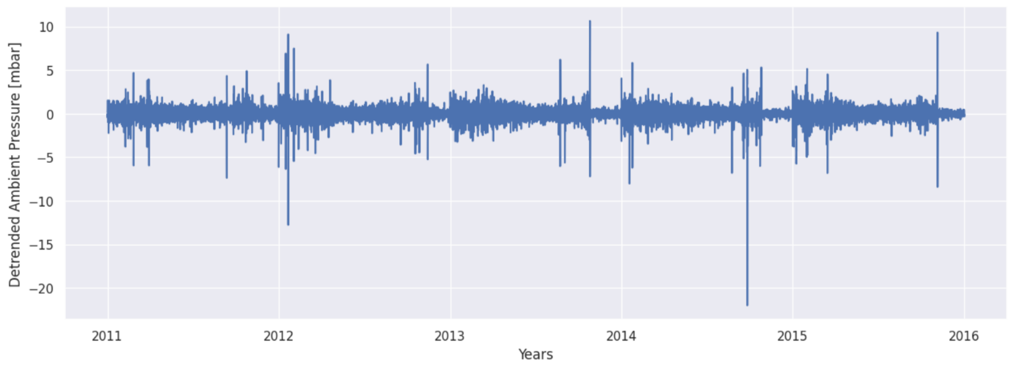


Figure 4.49: Detrended plot for ambient pressure feature

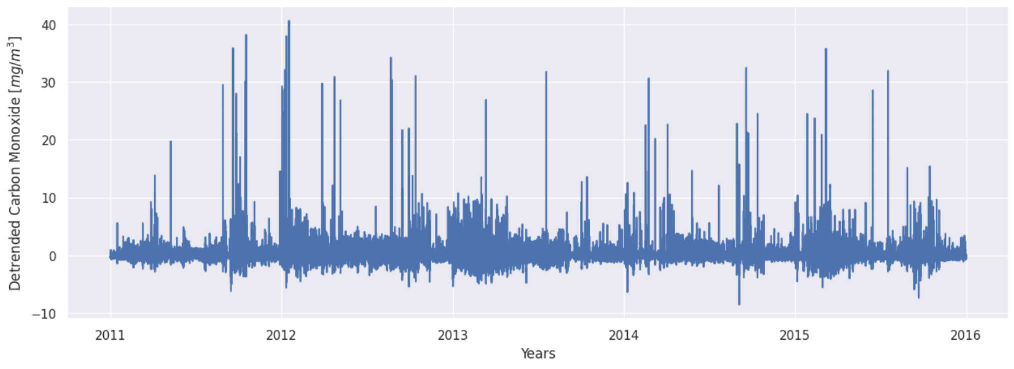


Figure 4.50: Detrended plot for carbon monoxide emissions feature

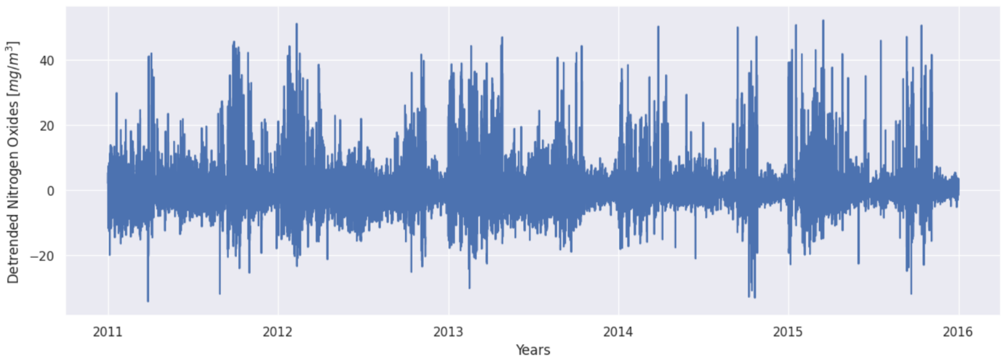


Figure 4.51: Detrended plot for nitrogen oxides emissions feature

The prominent observation from these graphs remains consistent with our previous understanding: certain features notably exhibit a substantial number of outliers.

This is particularly evident in instances of ambient pressure, ambient humidity, air filter difference pressure, turbine inlet temperature and turbine after temperature. However, it is particularly notable in the case of carbon monoxide emissions and nitrogen oxides emissions.

To explore this matter further, the next step involves analyzing the seasonality within the time series data. In this case, in order to observe two different behaviours, the graphs plotted will be the ones for ambient temperature and ambient pressure.

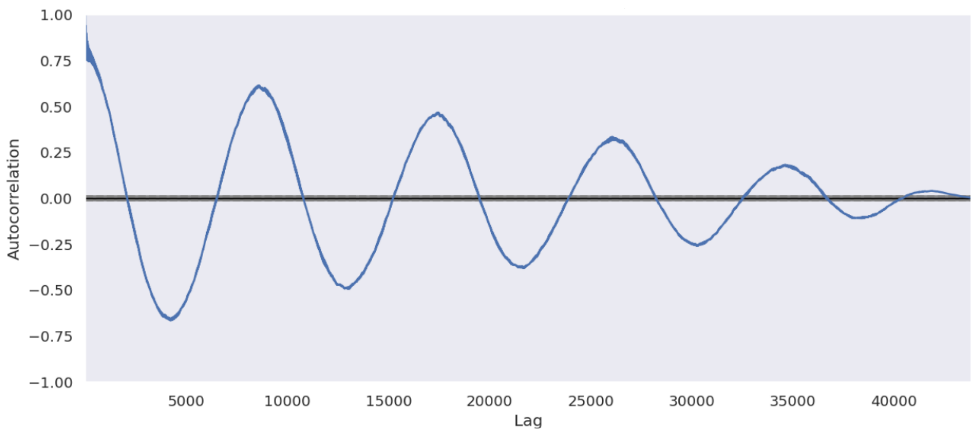


Figure 4.52: Seasonality plot for ambient temperature feature

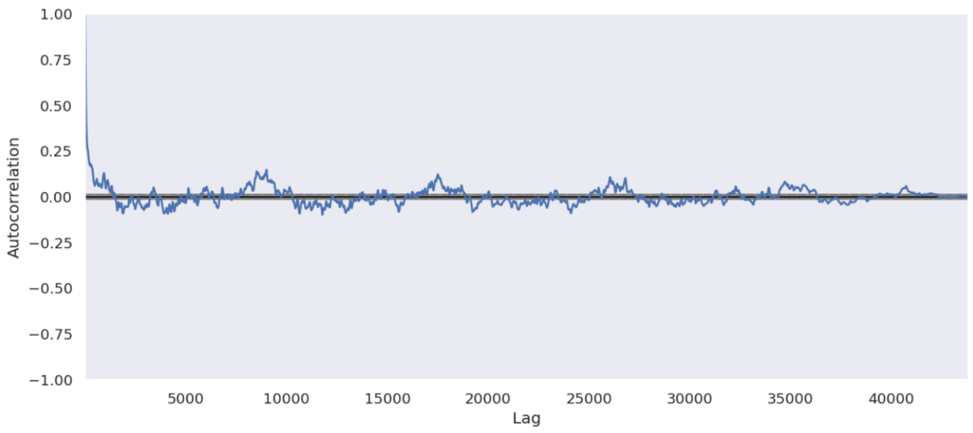


Figure 4.53: Seasonality plot for ambient pressure feature

Regarding the prior observations, ambient temperature demonstrates a distinctive pattern. Its seasonality is identifiable by a sinusoidal wave recurring every year (around 8760 lags), indicating a yearly seasonal pattern in the feature.

Similar patterns are observable in some other features, although their patterns appear less distinct, possibly due to the presence of substantial noise, likely stemming from outliers within the data.

Air filter difference pressure, for example, does not demonstrate a clear seasonal behavior, likely influenced by the presence of outliers affecting its pattern.

Turbine inlet temperature and gas turbine exhaust pressure, due to their closely distributed values, are challenging to visually assess in terms of seasonal patterns. However, this lack of visibility doesn't significantly impact the analysis, as we've previously identified trends and smoother representations of these variables.

The level of seasonality observed in nitrogen oxides emissions appears to be more pronounced compared to carbon monoxide emissions, potentially influenced by the presence of outliers in both datasets.

The upcoming process involves plotting Autocorrelation and Partial Autocorrelation graphs for all the features under study. For this analysis, 8760 lags were employed, signifying an examination of the initial complete year of data.

For this case, only the graphs corresponding to ambient temperature will be plotted.

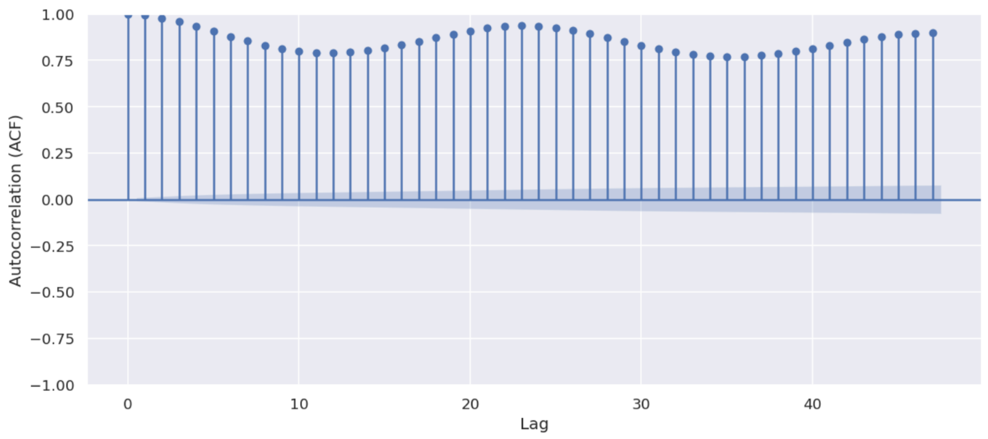


Figure 4.54: ACF plot for ambient temperature feature

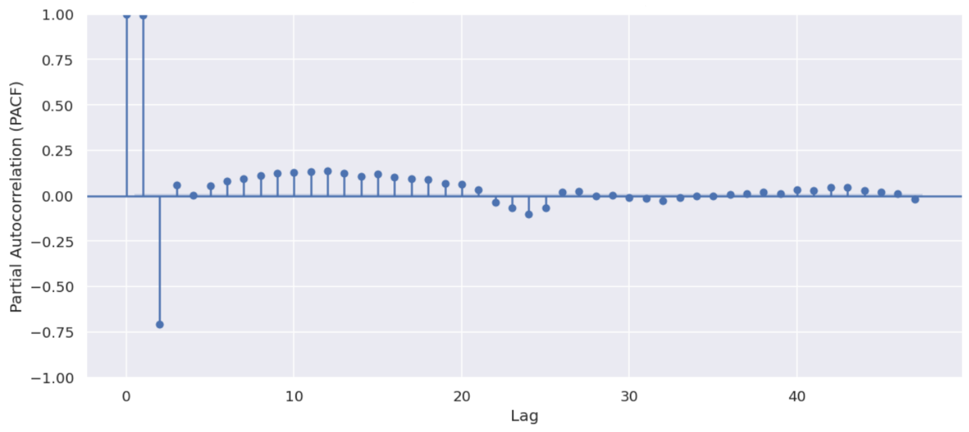


Figure 4.55: PACF plot for ambient temperature feature

The observed pattern shows that most lags either approximate 1 or are significantly above the confidence interval, indicating statistical significance for all variables.

In the case of ambient humidity, gas turbine exhaust pressure, turbine inlet temperature, turbine energy yield, compressor discharge pressure and carbon monoxide emissions, it's evident that several lags fall below 0.25, indicating significantly lower statistical significance compared to the other lags in these features. Nevertheless, though statistically signficant, all features' partial autocorrelation after the first few lags is very low.

Lag plots are intimately related with autocorrelation. They are useful for identifying autocorrelation in a time series, aiding in selecting appropriate models for time series analysis and forecasting. They can complement ACF and PACF analyses.

Only the lag plot graphs corresponding to ambient variables will be plotted.

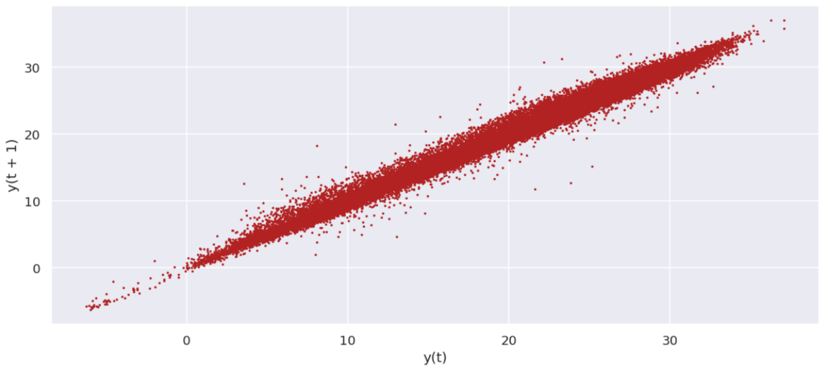


Figure 4.56: Lag plot for ambient temperature feature

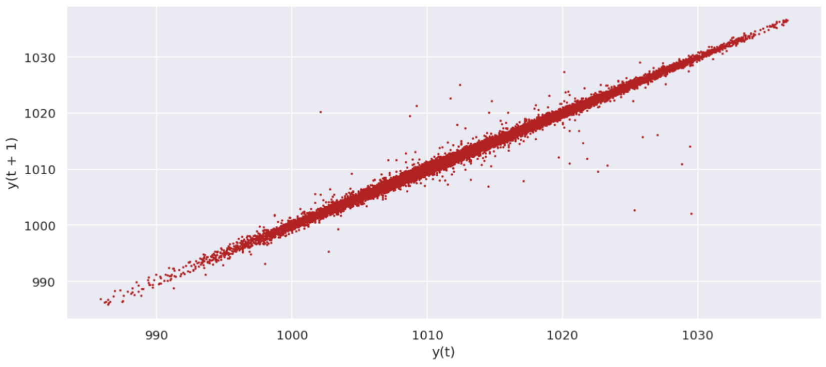


Figure 4.57: Lag plot for ambient pressure feature

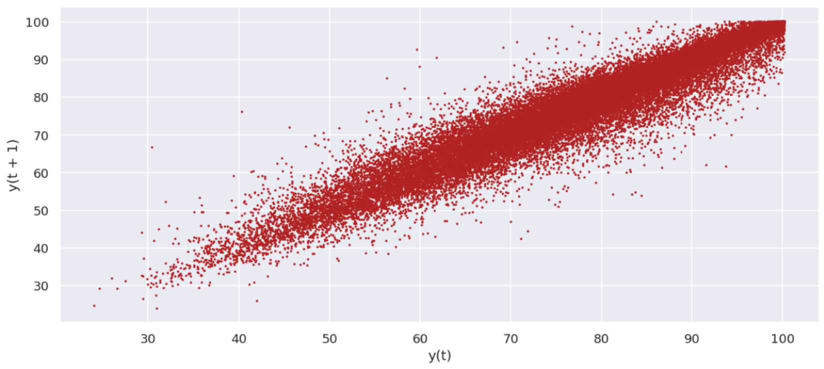


Figure 4.58: Lag plot for ambient humidity feature

As observed, the only features demonstrating autocorrelation with themselves are the ambient variables, particularly highlighting ambient pressure.

In conclusion for this section, the focus will shift towards studying the forecastability of the time series data.

	Entropy
AT	3.42
AP	3.21
AH	4.02
AFDP	1.09
GTEP	2.78
TIT	4.21
TAT	3.27
TEY	4.10
CDP	1.43
CO	2.16
NO_x	3.80

Table 4.4: Forecastability entropy for the different features

The computed entropy values generally do not demonstrate high levels, with the lowest values attributed to air filter difference pressure and compressor discharge pressure. These will thus be the features with the higher predictability (forecastability).

At this juncture, it is possible to examine potential outliers within the dataset. To initiate this process, exploring the data's metrics will be the first step.

	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NO_x
Count	43824	43824	43824	43824	43824	43824	43824	43824	43824	43824	43824
Mean	16.46	1012.85	77.89	3.95	25.47	1081.51	546.27	133.18	12.04	2.33	65.24
Std. deviation	7.40	5.99	13.48	0.72	3.91	16.35	6.36	14.57	1.02	2.10	10.80
Minimum	-6.23	985.85	24.09	2.09	17.70	1000.80	511.04	100.02	9.85	0.00	25.91
25%	10.22	1009.30	69.92	3.48	23.39	1074.00	545.13	126.18	11.53	1.23	58.53
50%	15.42	1012.30	79.60	4.00	25.09	1085.00	549.80	133.66	11.96	1.77	64.09
75%	22.51	1016.10	88.09	4.35	27.17	1093.80	550.02	138.91	12.45	2.72	70.25
Maximum	37.10	1036.60	100.20	7.61	40.72	1100.90	550.61	179.50	15.16	44.10	119.91

Table 4.5: Statistical metrics for the complete dataset

Based on the statistical data derived from the gas turbine parameters and the aforementioned outcomes, certain assumptions can now be formulated:

- The minimum ambient temperature reached on 2015, even though can be due to a extreme climate situation, can be considered as an outlier, as it distances too much from the normal temperatures expected in Turkey.
- The amount of NO_x emissions oscillate much between different years.
- The minimum amount of NO_x emissions are really different between years.
- The minimum ambient humidity oscillates between different years.
- The standard deviation of turbine inlet temperature during 2015 is greater than for the rest of the years. The standard deviation for turbine after temperature varies greatly between years, as well as the one for turbine energy yield. This can mean that those values are not promediated as indicated at the beginning

of this work, but rather calculated, due to the fact that sensor data were promediated in hourly intervals, for which a small standard deviation would be expected.

In order to finish visualizing the outliers and closing the Exploratory Data Analysis related to this work, box plots for the different features will be presented.

Only the box plots corresponding to ambient temperature and nitrogen oxides emissions will be presented. The rest can be found in Appendix II.

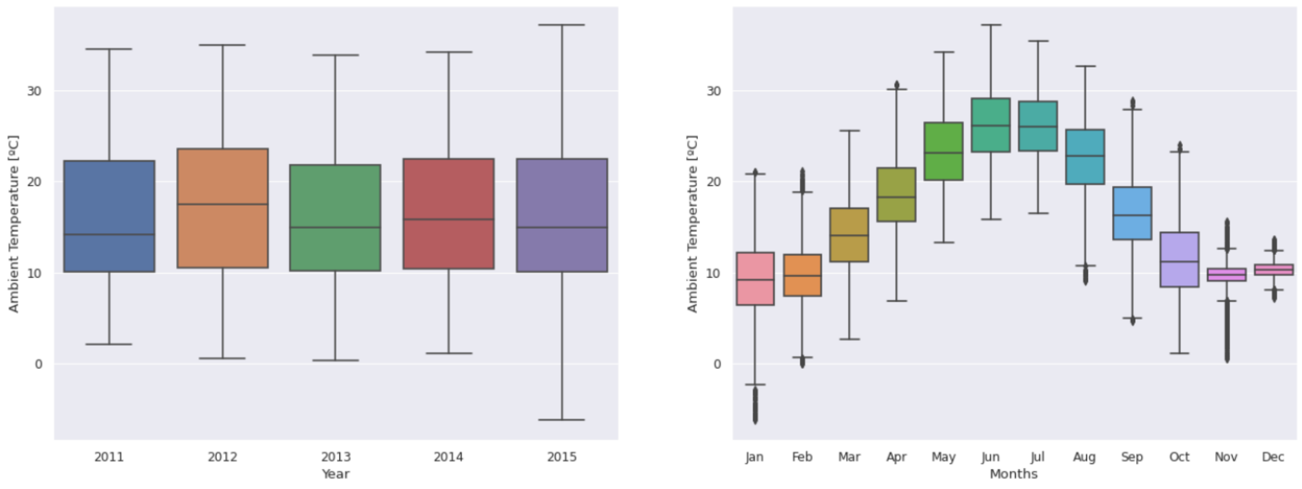


Figure 4.59: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for ambient temperature feature

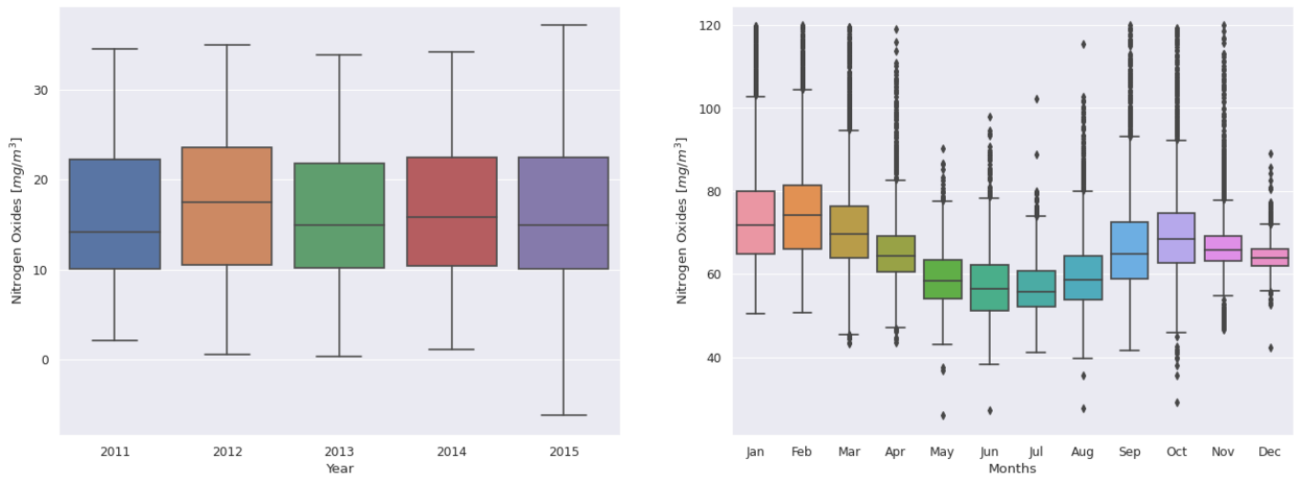


Figure 4.60: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for nitrogen oxides emissions feature

Another informative aspect for this preliminary outlier analysis is to visualize the relative (percentage) and absolute change in values across the entire period (2011-2015) for all the variables. This method will facilitate the identification of outliers within the dataset.

For this case, only the graph for carbon monoxide emissions will be presented herein.

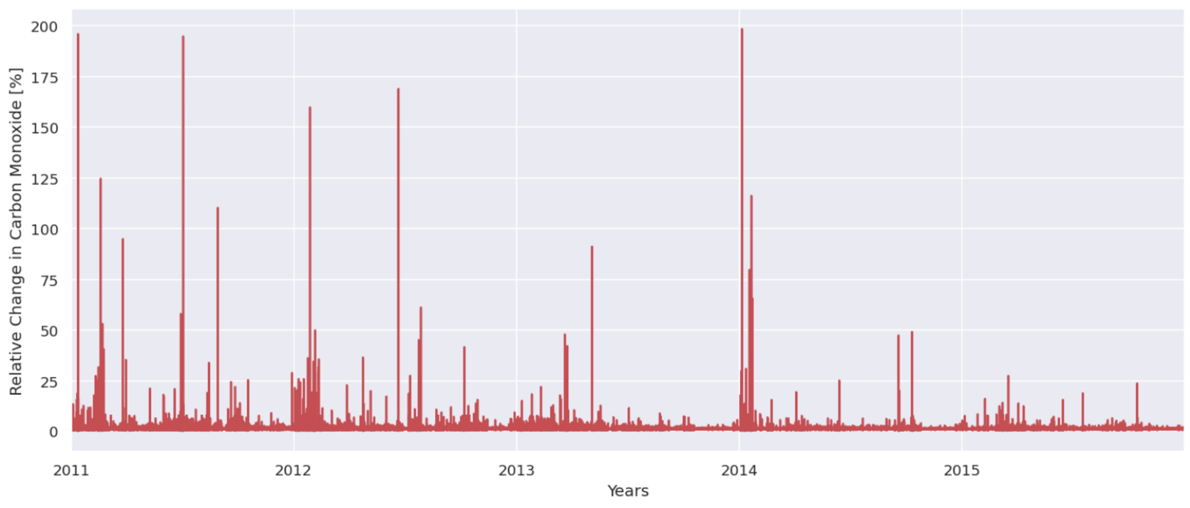


Figure 4.61: Relative change for carbon monoxide emissions feature over the years

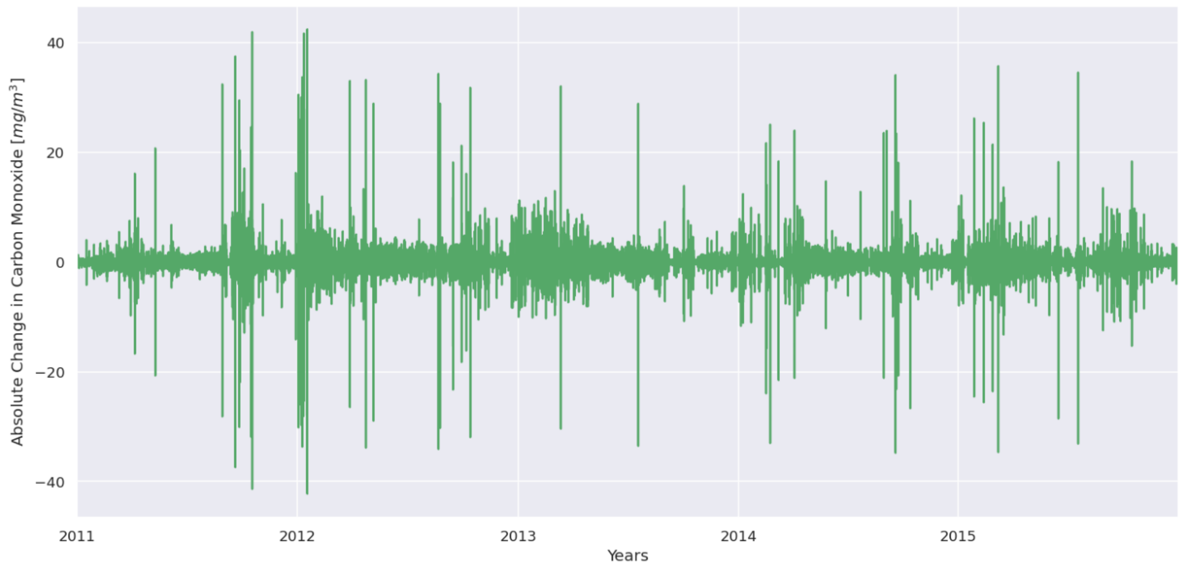


Figure 4.62: Absolute change for carbon monoxide emissions feature over the years

These graphs confirm the observations derived from statistical data and provide additional insights.

The first one that stands out is that 2015 is the year that presents the highest number of outliers for all features.

The data reveals that the initial and final months of the year tend to accumulate the highest number of outliers, with November being quite particular for all features.

The boxplots indicate a considerable presence of outliers within the measurements of ambient humidity, air filter difference pressure, carbon monoxide emissions and nitrogen oxides emissions. Additionally, notable values are visible in turbine inlet temperature or ambient pressure. However, it is crucial to consider that certain extreme values might be entirely valid from a physical standpoint, such as those observed for ambient pressure.

Diverse fuel types can cause variations in temperatures before and after the turbine, despite generally using the same fuel across all engines. Elevated levels of CO during transient gas turbine operations, such as startup or warm-up periods, are common, although these phases are typically brief.

The absolute and relative change graphs indicate potential outlier treatment necessary for carbon monoxide emissions and nitrogen oxides emissions. Additionally, outliers are noticeable in ambient temperature, particularly in early 2015. Regarding ambient pressure, although "spikes" are visible in the graphs, upon closer inspection of the values, the changes appear minor.

In the upcoming chapter, determinations will be made regarding the outliers to be retained and those to be eliminated from the dataset.

5

CO and NO_x emissions prediction

5.1. Experimental procedure

5.1.1. Feature engineering

The feature engineering section of this Thesis serves as a pivotal aspect in the construction of predictive models, emphasizing the transformation and creation of meaningful variables from raw data.

This section aims to detail the intricate process of selecting, deriving and refining features that contribute significantly to the model's predictive power while addressing issues such as handling missing values or scaling variables. The application of domain knowledge, statistical methodologies and algorithmic techniques is thoroughly explored to enhance the quality and effectiveness of the predictive model's input features, thereby improving its overall performance and robustness in real-world applications.

As previously stated in the preceding chapter, a mathematical relationship between ambient pressure and ambient temperature has been identified. Consequently, the initial step involves plotting the graph for the ratio of ambient pressure to ambient temperature (P/T), which is expected to exhibit a relatively consistent pattern.

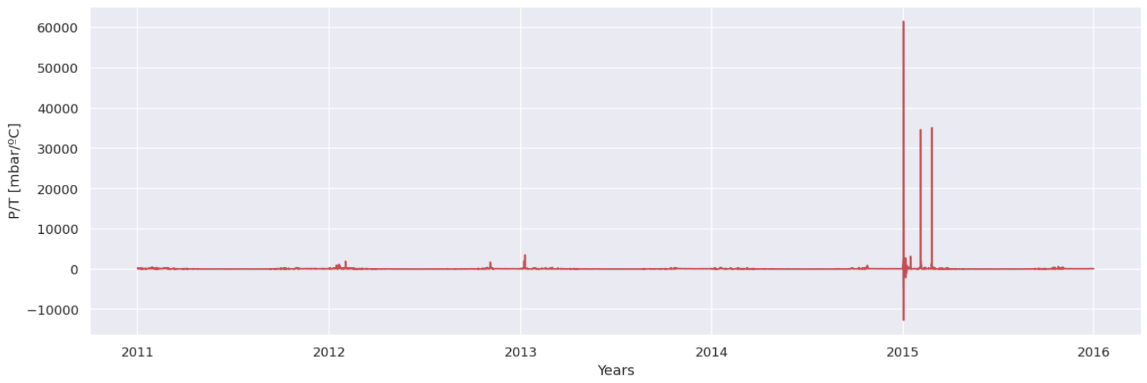


Figure 5.1: Ratio of ambient pressure to ambient temperature (P/T)

The observed trend indicates a predominantly linear pattern, suggesting constancy; however, notably extreme values are evident, particularly noted at the beginning of 2015.

Upon examining the preceding chapter's final graphs related to ambient temperature, a resemblance between the graph for ambient temperature relative change and the current graph is apparent. 2015 emerges as the year marked by notably extreme ambient temperature values, notably during January and February, exhibiting an accumulation of outliers during these months.

Throughout this text, emphasis has been placed on the significance of outliers in data analysis, significantly influencing modeling and subsequent results. Consequently, addressing outliers before commencing modeling is a common practice.

In this context, the approach involving the interquartile range method will be utilized to manage outliers, although numerous alternative methods exist, such as z-score or clustering methods.

In the process, a grand total of 8335 instances were altered and substituted with the mean value of every analyzed feature. This accounts for roughly 19% of the initial dataset. It's crucial to highlight that the mean value for each feature was calculated without considering outlier values.

To assess whether the dataset's visual representation has improved post-cleaning, we will revisit some of the previous graphs and examine the updated metrics for a clearer evaluation.

As on the previous chapter, only the box plots corresponding to ambient temperature and nitrogen oxides emissions will be presented. The rest can be found in Appendix II.

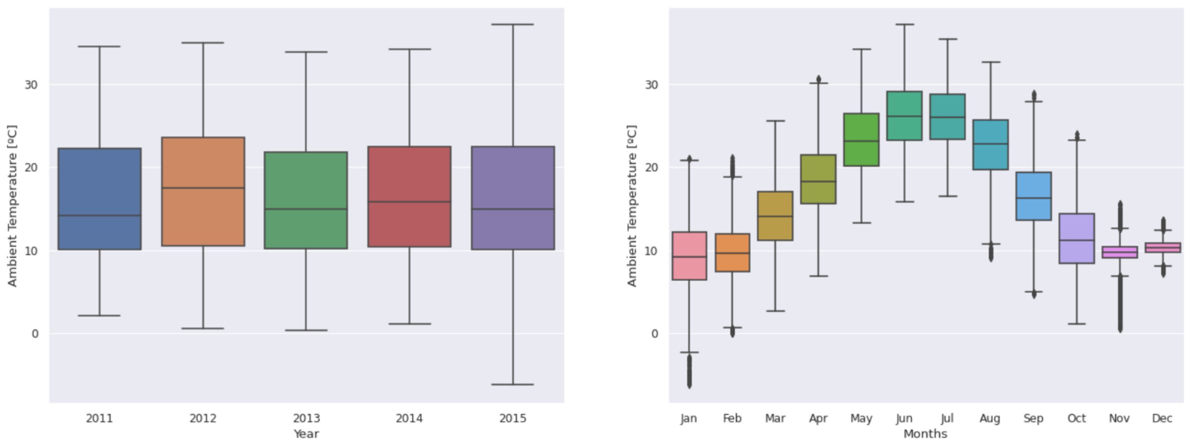


Figure 5.2: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for ambient temperature feature after outliers removal

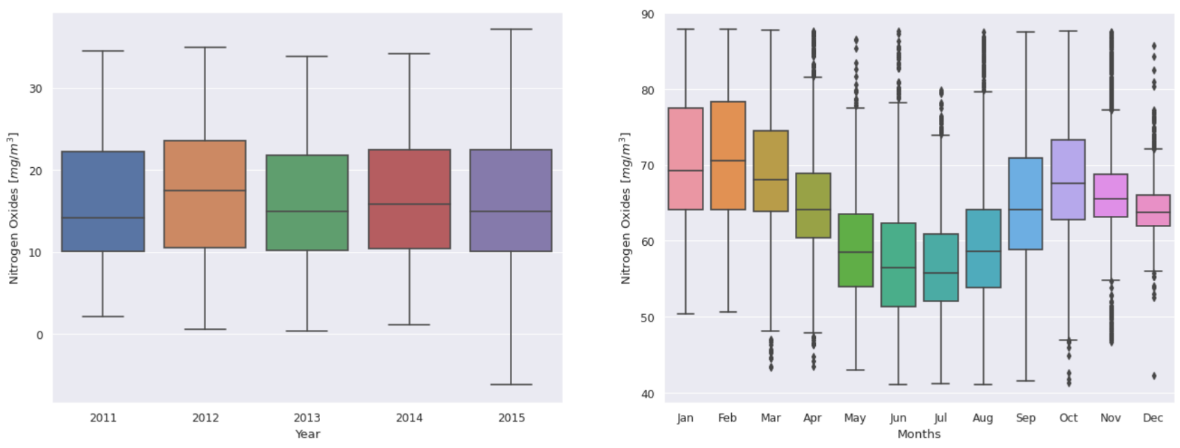


Figure 5.3: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for nitrogen oxides emissions feature after outliers removal

	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NO_x
Count	43824	43824	43824	43824	43824	43824	43824	43824	43824	43824	43824
Mean	16.46	1012.66	78.39	3.91	25.09	1082.73	548.36	131.42	11.91	1.89	64.16
Std. deviation	7.40	5.09	12.72	0.65	3.47	14.55	2.74	11.58	0.84	0.98	8.70
Minimum	-6.23	999.10	42.67	2.19	17.74	1044.40	537.80	107.11	10.16	0.00	41.02
25%	10.22	1009.50	70.58	3.48	23.39	1075.72	548.36	127.20	11.57	1.23	58.56
50%	15.42	1012.40	79.60	3.96	25.09	1085.00	549.80	132.83	11.91	1.77	64.10
75%	22.51	1015.60	88.09	4.32	26.33	1093.80	550.02	134.92	12.22	2.29	69.16
Maximum	37.10	1026.30	100.20	5.66	32.84	1100.90	550.61	157.99	13.82	4.95	87.82

Table 5.1: Statistical metrics for the complete dataset after outliers removal

Overall, improvements are observed compared to the previous chapter, notably indicated by smaller standard deviation values across all features (except for ambient temperature, which has the same standard deviation value in both cases).

Considering that the standard deviation values remain notably high and that the feature values exhibit considerable dispersion, the decision was made to implement Butterworth filtering on the various features. This step aims to assist the future models in achieving improved predictive performance, as certain models typically struggle to capture such significant dispersion of values.

The figures below depict the original values alongside their filtering, showcasing graphs specifically for carbon monoxide and nitrogen oxides emissions in this section. Additional graphs for the remaining features can be found in Appendix II.

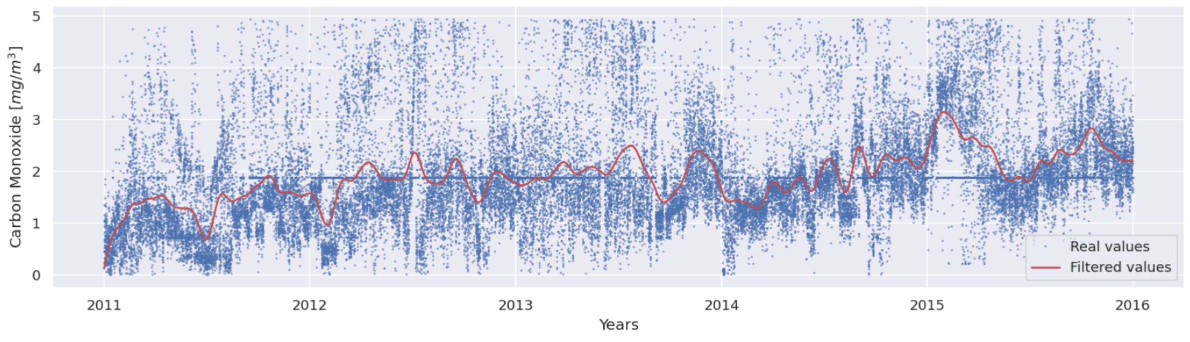


Figure 5.4: Carbon monoxide emissions feature and butterworth filtering after outliers removal

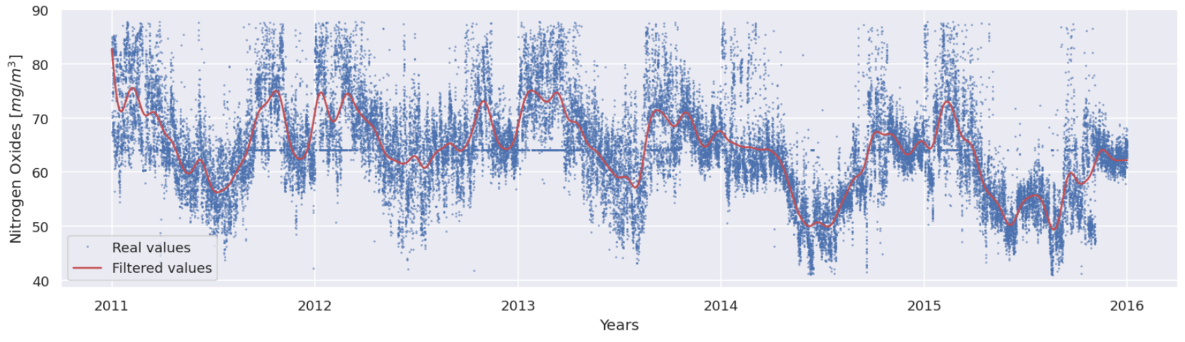


Figure 5.5: Nitrogen oxides emissions feature and butterworth filtering after outliers removal

Prior to commencing the modeling process, an essential step involves determining the statistical significance of variables and identifying any potentially less influential ones. To accomplish this task, Canonical Correlation Analysis (CCA) will be employed.

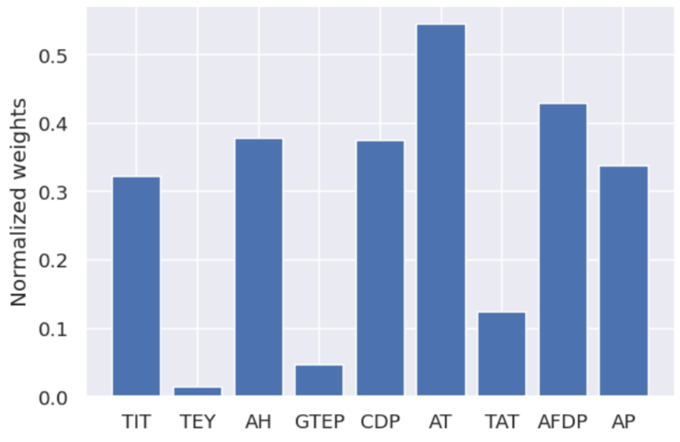


Figure 5.6: Canonical Correlation Analysis for carbon monoxide emissions

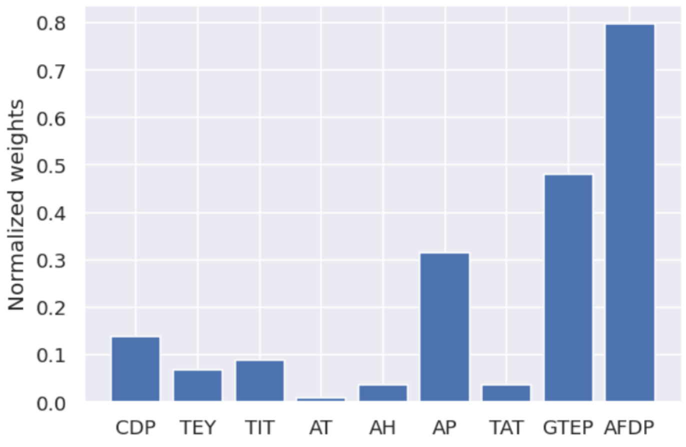


Figure 5.7: Canonical Correlation Analysis for nitrogen oxides emissions

The most notable observation is the significantly higher normalized weights of the features for carbon monoxide emissions compared to nitrogen oxides emissions. Additionally, it is evident that the features indicated in the graphs as having the highest normalized weights do not encompass those with the strongest Pearson correlations.

The features with the highest normalized weights for each case are:

- Ambient temperature, ambient pressure, ambient humidity, air filter difference pressure, turbine inlet temperature and compressor discharge pressure for carbon monoxide emissions.
- Ambient pressure, air filter difference pressure and gas turbine exhaust pressure for nitrogen oxides emissions.

Furthermore, the number of features selected to represent the 95% of the cumulated importance (thus, the ones to be employed for feature selection) are:

- Ambient temperature, ambient humidity, gas turbine exhaust pressure, turbine inlet temperature, turbine energy yield and compressor discharge pressure to represent carbon monoxide emissions.
- Ambient temperature, ambient humidity, turbine inlet temperature, turbine energy yield and compressor discharge pressure to represent nitrogen oxides emissions.

Having identified the appropriate features for each target variable, it becomes imperative to consider the scaling of these features. This is essential due to the variations in scale and dimensions across different features.

As emphasized in preceding chapters, the normalization process involves subtracting the mean and dividing by the standard deviation of the training set. Although the delineation into train, validation and test sets is beyond the scope of this section, an overview will be provided to facilitate understanding of the normalization procedure. It is expected that data pertaining to 2011 and 2012 will serve for training, 2013 for validation, and 2014 as well as 2015 for testing.

Upon completion of this process, there will be a distribution of 17544 samples for training, 8760 samples for validation and 17520 samples for testing.

Integrating this normalization technique with the earlier feature selection of the most relevant variables will culminate in the creation of the final dataset intended for modeling purposes.

5.1.2. Modeling

Given the presence of two distinct target variables (carbon monoxide emissions and nitrogen oxides emissions), this section will be bifurcated accordingly. The initial focus will center on the prediction of carbon monoxide emissions, followed by the subsequent exploration pertaining to the prediction of nitrogen oxides emissions.

The data generator utilized in both cases remains consistent, differing only in the data associated with the specific target variable being modeled. It generates a tuple (samples, targets), where samples represent a batch of input data and targets correspond to the array of the targeted emissions. The input parameters for this generator are as follows:

- lookback: observations extend back over a period of 10 days.
- delay: target predictions are set for 24 hours in the future.
- $\min_{index}$ and $\max_{index}$: these indices are determined based on the number of samples allocated for the train, validation and test sets, as detailed in the previous section.
- shuffle: given the dependency on periodic features, maintaining the order of data is crucial; hence, the data will not be shuffled.
- batch size: set at 128.
- step: observations will be sampled at intervals of one point per 12 hours.

This sampling strategy aligns with the tensorized data requirement of the employed models (1D CNN and GRU).

Preceding the explanation of the utilized models, it is imperative to emphasize that the selection of variables and parameters involved a thorough iterative process rather than random selection. Initially, a foundational model was established, followed by iterative parameter adjustments. Continuous evaluation for overfitting was pivotal due to the dataset's limited size and the prominent presence of overfitting in early trials.

Hence, constructing a model with minimized overfitting, yielding relatively good results and possessing manageable computational costs was essential. Some models can take hours to run, especially when assessing numerous hyperparameters, underscoring the importance of efficiency in model performance evaluation.

To construct the generator and models utilized in this Thesis, the process outlined by François Chollet (creator of the Keras deep learning library) in ref. [62] has been employed.

This entailed the need to parse the data, which involved transforming the data into a format that is comprehensible and suitable for utilization by the data generator.

To enable the comparison of model performance across various scenarios, identical models were employed for both the prediction of carbon monoxide emissions and nitrogen oxides emissions.

The GRU model utilized a single GRU layer, considering that simpler models with fewer layers tend to be more resilient against overfitting and require less computational cost, being faster to converge to a solution.

To address overfitting concerns, L2 regularization, dropout and recurrent dropout were incorporated. Additionally, early stopping was implemented for this purpose. Given the time series nature of the data, the model employed the Adam optimizer and MAE loss function.

A final Dense layer has also been implemented. In deep learning problems, as already explained, the necessity of a final Dense layer lies in its ability to perform the ultimate classification or regression task. This Dense layer is utilized to transform the features extracted by preceding layers into interpretable predictions.

Regarding hyperparameter optimization, various configurations of GRU units and learning rate values were explored to enhance training outcomes and develop an improved model for testing. Upon finalization of the training phase, the following values were selected:

- 32 GRU units for CO emissions prediction and 64 for NO_x emissions prediction.
- 0.0001 learning rate for both cases.

In the figures below, representations of the various models' structures are depicted. Special attention should be given to the previously mentioned fact in earlier chapters: as input values for the different layers change dimensions, the models' layers change their input/output tensor dimensions accordingly.

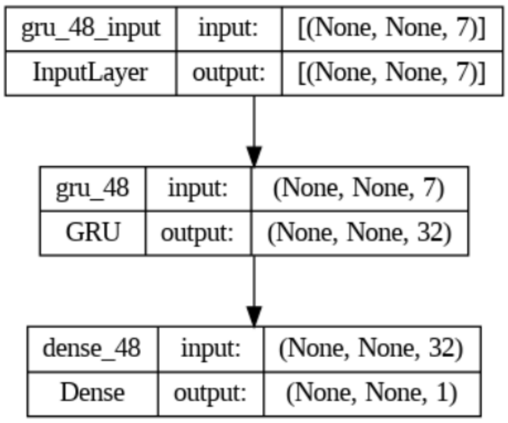

gru_48_input | input: | [(None, None, 7)]
InputLayer | output: | [(None, None, 7)]
gru_48 | input: | (None, None, 7)
GRU | output: | (None, None, 32)
dense_48 | input: | (None, None, 32)
Dense | output: | (None, None, 1)

Figure 5.8: GRU model plot for carbon monoxide emissions prediction (with dimensions)

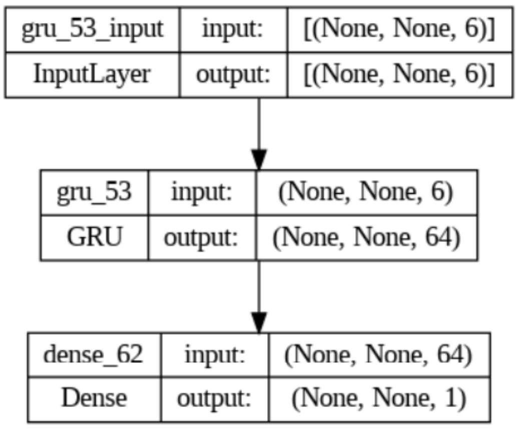
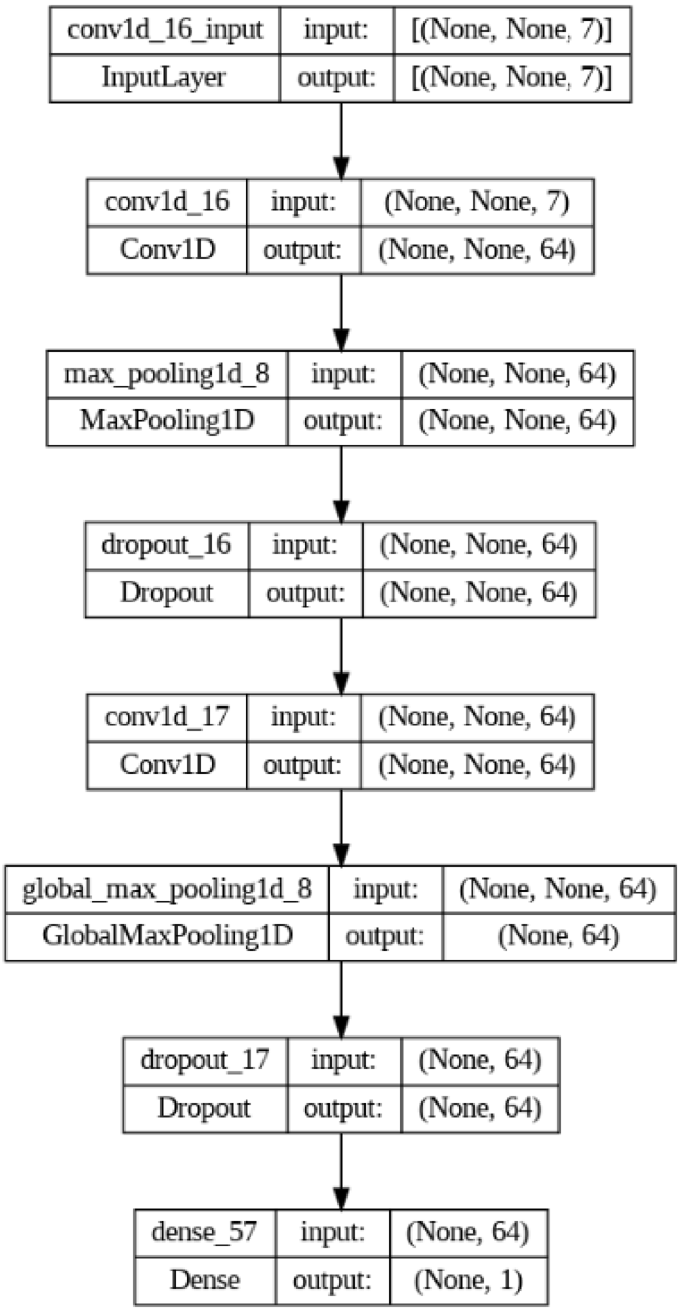

gru_53_input | input: | [(None, None, 6)]
InputLayer | output: | [(None, None, 6)]
gru_53 | input: | (None, None, 6)
GRU | output: | (None, None, 64)
dense_62 | input: | (None, None, 64)
Dense | output: | (None, None, 1)

Figure 5.9: GRU model plot for nitrogen oxides emissions prediction (with dimensions)

The 1D CNN model, conversely, tackles overfitting concerns by using L2 regularization, dropout and early stopping. Similar to the previous scenario, the model utilized the Adam optimizer and MAE loss function, along with the tanh activation function. Additionally, MaxPooling1D and GlobalMaxPooling1D layers were utilized. As in the previous case, a final Dense layer has also been employed.

Regarding hyperparameter optimization, various configurations of CNN filters, pool size and learning rate values were explored. Upon finalization of the training phase, the following values were selected:

- 64 CNN filters for CO emissions prediction and 32 for NO_x emissions prediction.
- 3 pool size for MaxPooling1D and GlobalMaxPooling1D for both cases.
- 5×10^{-5} learning rate for both cases.



conv1d_16_input | input: | [(None, None, 7)]
InputLayer | output: | [(None, None, 7)]
conv1d_16 | input: | (None, None, 7)
Conv1D | output: | (None, None, 64)
max_pooling1d_8 | input: | (None, None, 64)
MaxPooling1D | output: | (None, None, 64)
dropout_16 | input: | (None, None, 64)
Dropout | output: | (None, None, 64)
conv1d_17 | input: | (None, None, 64)
Conv1D | output: | (None, None, 64)
global_max_pooling1d_8 | input: | (None, None, 64)
GlobalMaxPooling1D | output: | (None, 64)
dropout_17 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_57 | input: | (None, 64)
Dense | output: | (None, 1)

Figure 5.10: 1D CNN model plot for carbon monoxide emissions prediction (with dimensions)

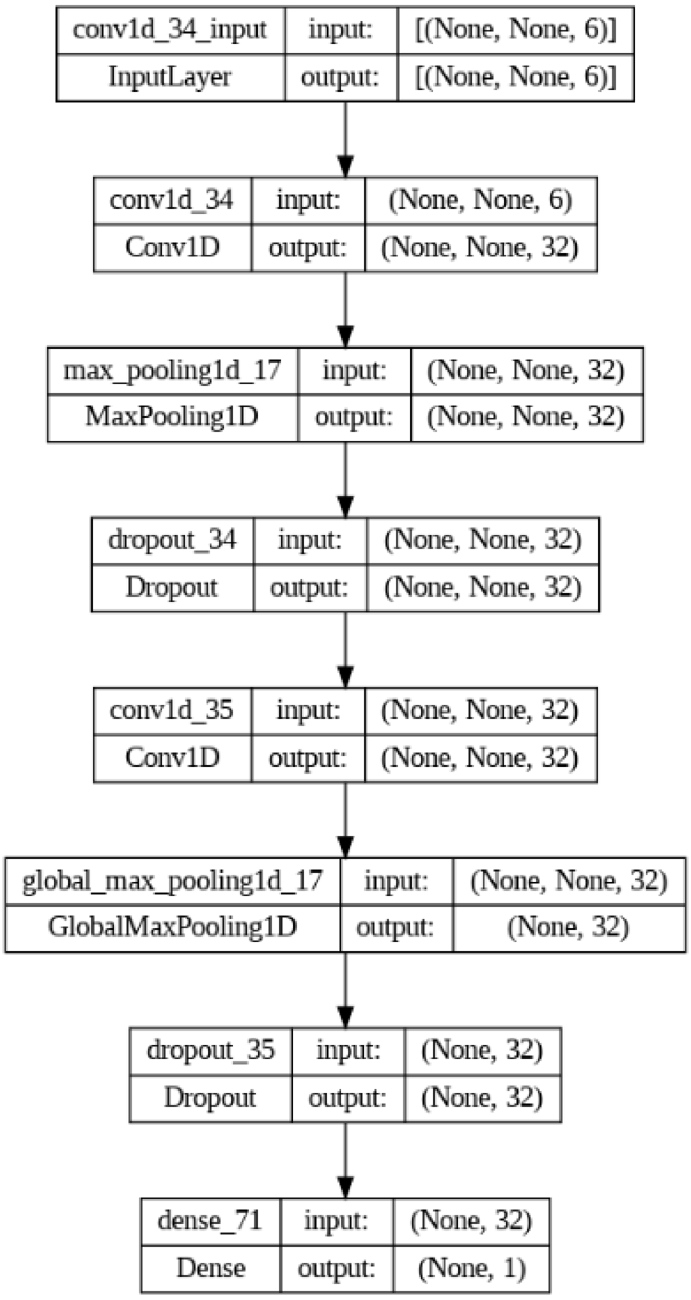

conv1d_34_input | input: | [(None, None, 6)]
InputLayer | output: | [(None, None, 6)]
conv1d_34 | input: | (None, None, 6)
Conv1D | output: | (None, None, 32)
max_pooling1d_17 | input: | (None, None, 32)
MaxPooling1D | output: | (None, None, 32)
dropout_34 | input: | (None, None, 32)
Dropout | output: | (None, None, 32)
conv1d_35 | input: | (None, None, 32)
Conv1D | output: | (None, None, 32)
global_max_pooling1d_17 | input: | (None, None, 32)
GlobalMaxPooling1D | output: | (None, 32)
dropout_35 | input: | (None, 32)
Dropout | output: | (None, 32)
dense_71 | input: | (None, 32)
Dense | output: | (None, 1)

Figure 5.11: 1D CNN model plot for nitrogen oxides emissions prediction (with dimensions)

Once the utilized models along with their defined parameters have been explored, the upcoming section will focus on visualizing and analyzing the results obtained for each case.

5.2. Experimental results

Upon implementing the models elucidated in the preceding section, the graphs depicting MAE against epochs (epochs represent the number of complete passes through the entire dataset during the training process) are as follows:

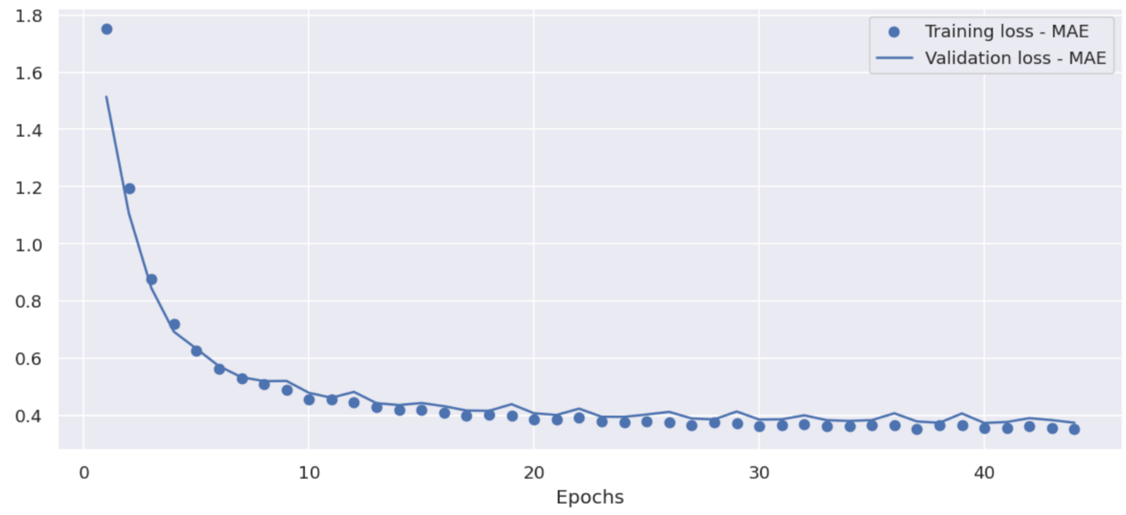


Figure 5.12: MAE vs. epochs during training and validation for GRU model for CO emissions prediction

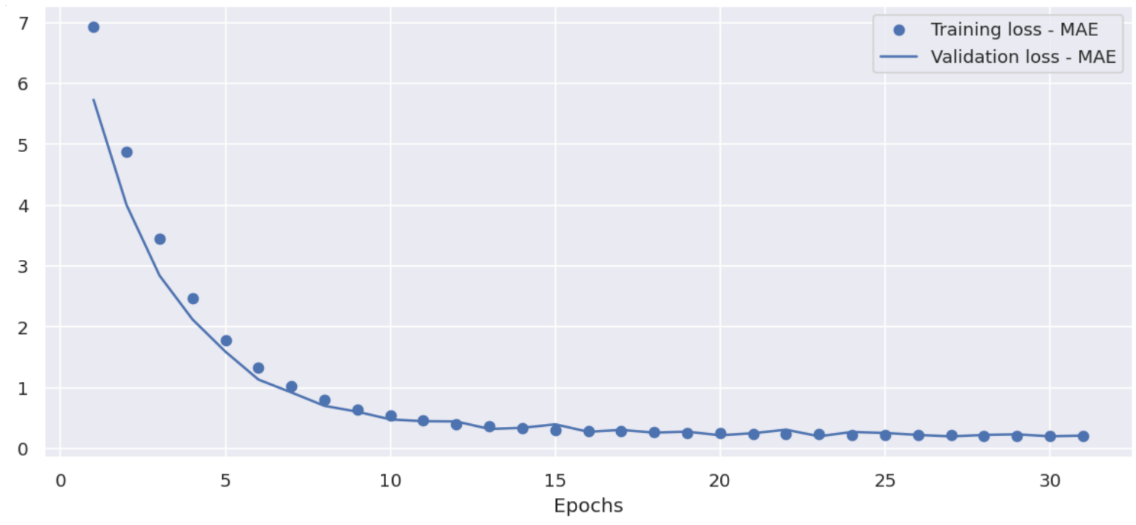


Figure 5.13: MAE vs. epochs during training and validation for 1D CNN model for CO emissions prediction

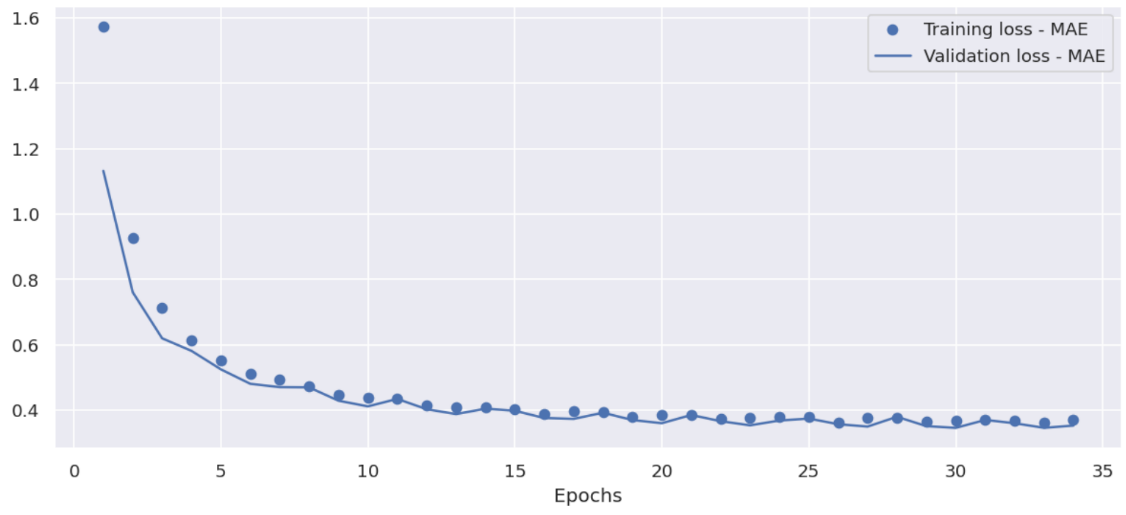


Figure 5.14: MAE vs. epochs during training and validation for GRU model for NO_x emissions prediction

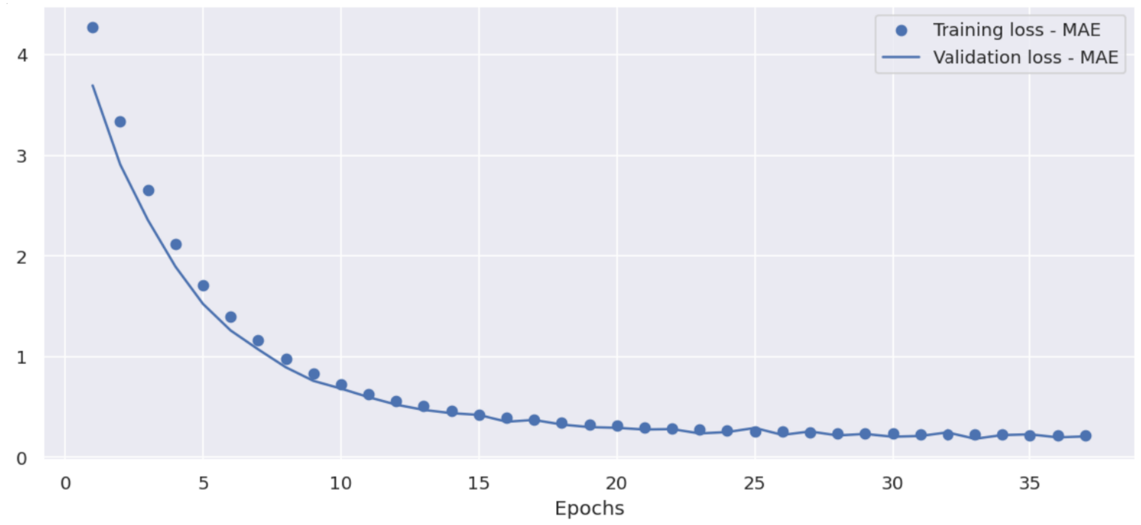


Figure 5.15: MAE vs. epochs during training and validation for 1D CNN model for NO_x emissions prediction

As depicted in the figures above, overfitting becomes evident after a few epochs in most cases, indicated by the pivotal point where the training loss becomes lower than the validation loss. The most favorable performance is exhibited by the 1D CNN model for predicting nitrogen oxides emissions, also having the most stable validation loss.

Regarding the numerical values obtained during the training, validation and testing evaluation phases, they are as follows:

	Validation MAE	Test MAE
GRU model for CO prediction	0.3715	0.4444
1D CNN model for CO prediction	0.1982	0.2052
GRU model for NO_x prediction	0.3435	0.3838
1D CNN model for NO_x prediction	0.2126	0.2126

Table 5.2: Validation and test loss with the different models

Test performance is usually expected to display a higher MAE compared to validation performance. This occurs because the model hasn't encountered the test data during training or parameter adjustment. The validation set is employed for fine-tuning and preventing overfitting by adjusting parameters. The unseen test set evaluates how well the model generalizes to new, unseen data. If test performance surpasses validation performance (test MAE is lower than validation MAE), it could indicate issues such as biased test data or accidental model overfitting, affecting its ability to generalize to new data. Lower test performance, in comparison to validation, helps gauge the model's capability to generalize to new data.

In this specific instance, it's observed that this condition is met, except for the 1D CNN model used for NO_x prediction, where both test and validation MAE are identical. This indicates that all models across all cases are functioning as expected.

Finally, upon plotting the resulting predictions obtained from each model (following the denormalization of obtained data), the subsequent outcomes are obtained when juxtaposed with the initial target predictions:

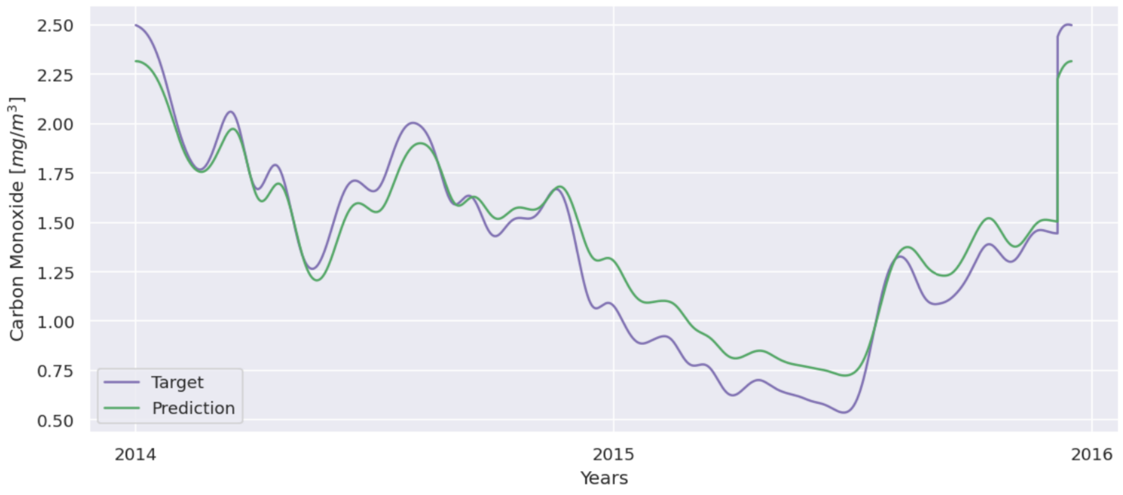


Figure 5.16: Target and predicted values after CO emissions prediction with GRU model

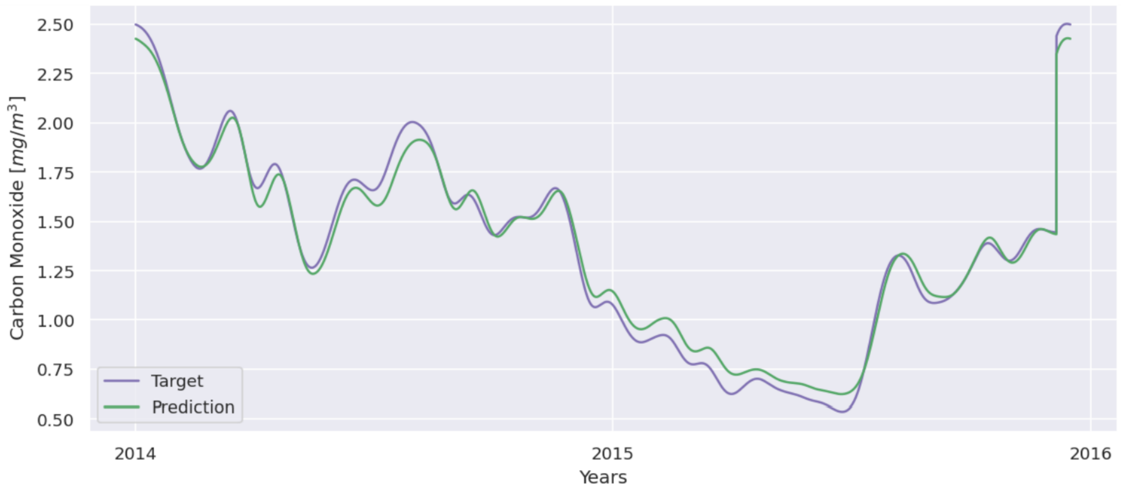


Figure 5.17: Target and predicted values after CO emissions prediction with 1D CNN model

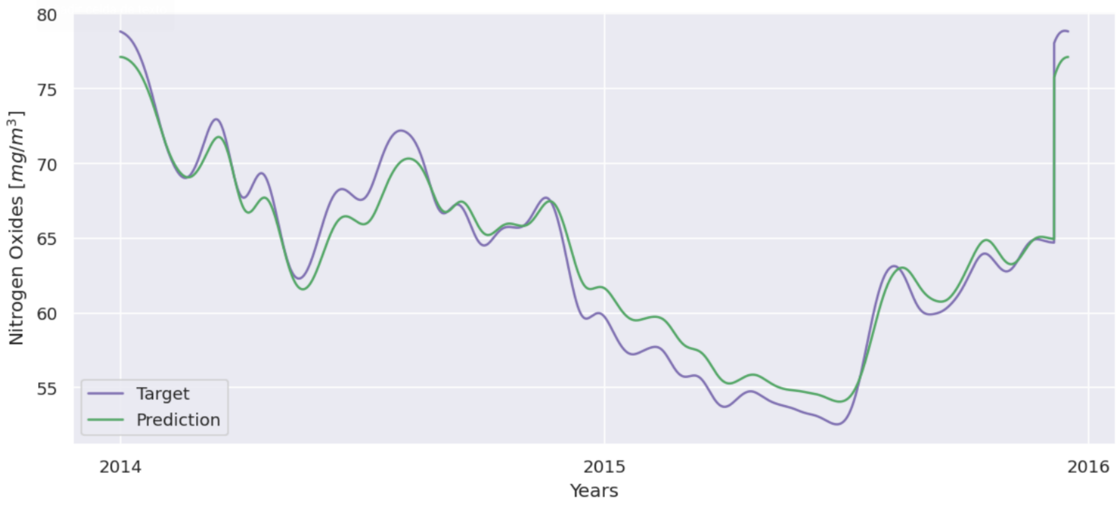


Figure 5.18: Target and predicted values after NO_x emissions prediction with GRU model

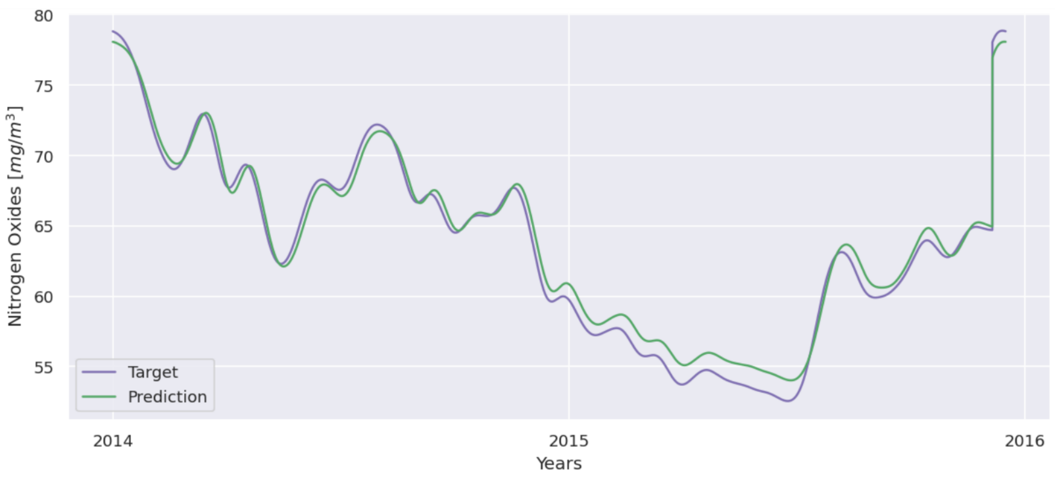


Figure 5.19: Target and predicted values after NO$_x$ emissions prediction with 1D CNN model

It's important to note that the shape of the resulting graphs is not the same as that of the original time series. This is due to the data transformation set by the generator. For instance, through parameters like lookback and delay, the generator impacts how the input dataset is structured for the model, simplifying result prediction, reducing computational cost, but sacrificing result interpretability.

The primary benefit of generators lies in their ability to process data on demand, generating data batches instead of loading the entire dataset into memory simultaneously. This is particularly useful when working with large datasets. Generators allow batch training, wherein the model is trained with data fragments at a time, potentially enhancing training stability and speed. For instance, in real-world applications where data is constantly changing or arriving sequentially (such as sensor data, stock data, etc.), generators enable processing data as it arrives, contributing to the model's adaptability to new data instances.

Nonetheless, the only relevant aspect remains the interpretability of the data and its representation. Based on the metrics, target values and predictions, it's confirmed that the models effectively adapt to predicting test set data, signifying their capability to adjust to forecasting unseen, new data instances.

Consequently, it is possible to conclude that the primary objective of this Thesis, which was to predict pollutant levels which were already known, is achieved.

When comparing the prediction of each of the target variables with each of the models employed, the results are the following:

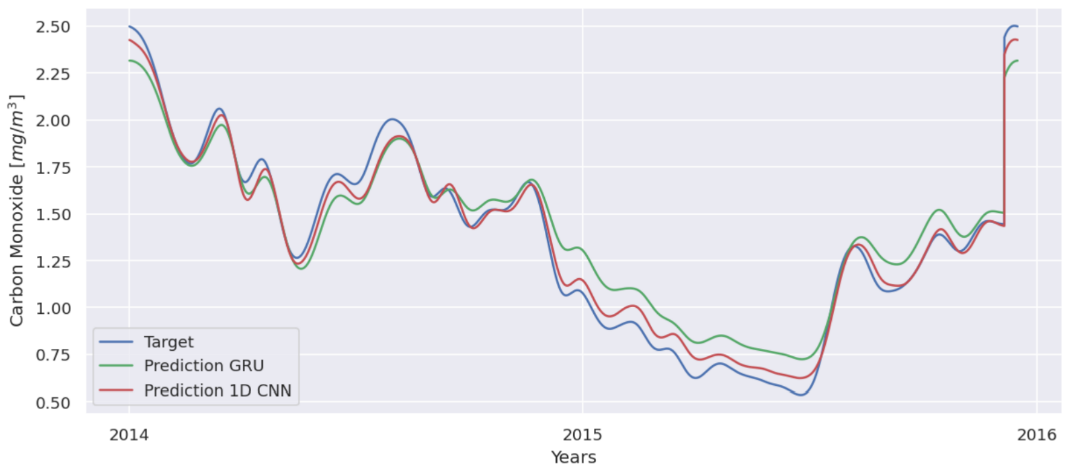


Figure 5.20: Target and predicted values comparison between GRU and 1D CNN models after CO emissions prediction

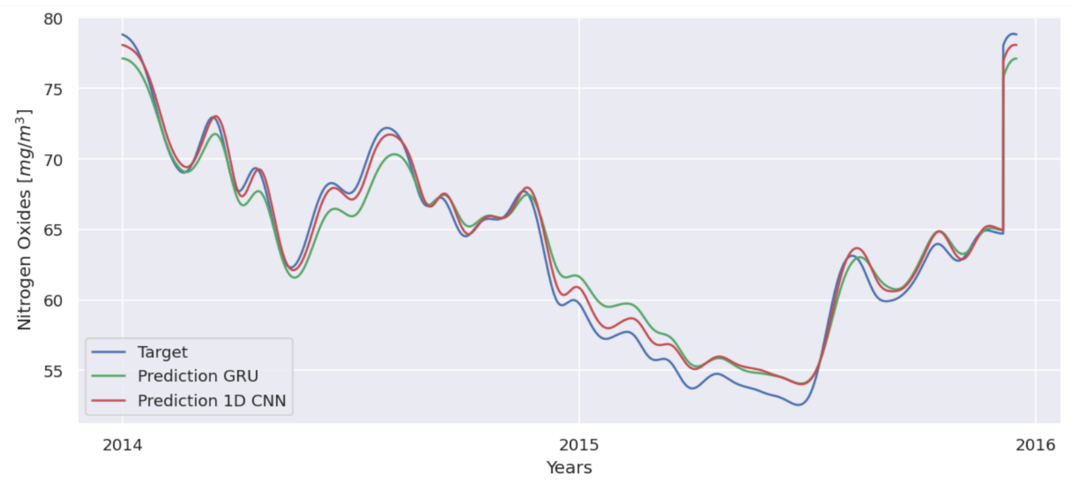


Figure 5.21: Target and predicted values comparison between GRU and 1D CNN models after NO_x emissions prediction

Based on the visualization of previous graphs and the analysis of MAE values, it becomes evident that the GRU model excels in predicting NO_x emissions instead of CO emissions, whereas the 1D CNN model performs better in predicting CO emissions.

Furthermore, the 1D CNN model demonstrates lower validation and test MAE values compared to the GRU model in both cases, indicating superior performance.

The graphs illustrate that the GRU model doesn't adjust as effectively to abrupt changes in the targets, showing less adaptability to significant increases or decreases in the target plot.

Consequently, the optimal model for this problem is determined to be the 1D CNN.

6 Conclusions

To finish this work, it is worth exposing the main conclusions that have been drawn from the studies carried out.

The prediction of aircraft emissions holds critical significance in compliance with increasingly stringent regulatory frameworks outlined in the initial chapters of this Thesis. Emissions, largely influenced by the combustion process, are subject to stringent regulations driving exploration into avenues to reduce aviation pollution.

Within this context, the implementation of emission prediction methods, specifically Continuous Emission Monitoring Systems (CEMS) and Predictive Emission Monitoring Systems (PEMS), presents viable measures. This Thesis primarily focuses on PEMS, which, despite their challenges, offer substantial advantages over CEMS, forming the core framework for the methodologies employed.

A comprehensive mathematical background was provided, elucidating the methodologies and models incorporated into an own Python-based code for practical implementation. This laid the foundation for an in-depth Exploratory Data Analysis (EDA), revealing the complexity of real-world datasets, especially with the inclusion of environmental, process variables and pollutant data collected over a five-year span from a gas turbine.

The comprehensive analysis of the dataset derived from a gas turbine located in the northwestern region of Turkey revealed intricate relationships between ambient variables and gas turbine process parameters. Ambient temperature, pressure and humidity exhibited characteristics mirroring the typical climate patterns of the region. Recurring values within specific features and the presence of outliers, especially within emissions variables, indicated potential implications for data reliability, necessitating strategies for outlier treatment.

Correlation analyses unveiled weak associations between ambient variables and gas turbine process parameters, while the latter had shown strong interrelationships. For example, lower turbine inlet temperatures corresponded to increased carbon monoxide emissions, a consequence of incomplete combustion at lower temperatures. Conversely, higher ambient temperatures correlated with reduced nitrogen oxides emissions, offering an effective mean for their mitigation. Seasonal variations highlighted variable correlations during distinct periods, emphasizing the dynamic nature of these relationships.

The dataset's periodic trends, varying residuals across features and statistical significance in lag correlations provided valuable insights into data characteristics. Understanding these complexities has been pivotal for refining outlier treatment approaches and ensuring the accuracy and reliability of subsequent analyses in gas turbine operation and emissions management.

The feature engineering process within this Thesis played a crucial role in constructing robust predictive models, entailing the transformation and creation of significant variables from raw data. An in-depth exploration was conducted, employing domain knowledge, statistical methodologies and algorithmic techniques to select, derive and refine features that substantially contributed to the predictive power of the models. Addressing issues such as outliers treatment and scaling variables was fundamental in enhancing the quality and effectiveness of the models' input features, thereby improving their overall performance and applicability in real-world scenarios.

The initial steps involved the application of the interquartile range method fot outliers handling, resulting in the modification of 19% of the dataset. The resultant cleaner representation of the data, assessed through box plots and metric evaluations, displayed improved standard deviation values across most features. Recognizing substantial dispersion within the features, a Butterworth filtering method was implemented to manage the data's dispersion. This step aimed to enhance predictive performance, as some models often struggle with interpreting significant dispersion among feature values.

Canonical Correlation Analysis (CCA) was then used to determine the most influential variables for predicting carbon monoxide and nitrogen oxides emissions. Subsequently, feature selection focused on retaining variables encompassing 95% of the cumulative importance. Scaling these chosen features became imperative due to variations in scale and dimensions, and a normalization process was applied by subtracting the mean and dividing by the standard deviation of the training set.

In the feature selection process based on the 95% cumulative importance, the chosen variables for predicting carbon monoxide emissions were ambient temperature, ambient humidity, gas turbine exhaust pressure, turbine inlet temperature, turbine energy yield and compressor discharge pressure. For predicting nitrogen oxides emissions, the selected variables encompassed ambient temperature, ambient humidity, turbine inlet temperature, turbine energy yield and compressor discharge pressure.

The resulting dataset, composed of the most relevant variables and normalized for modeling purposes, underwent prediction modeling for carbon monoxide and nitrogen oxides emissions. Employing GRU and 1D CNN models involved careful parameter selection and iterative adjustments to mitigate overfitting issues and ensure computational efficiency. Evaluation of model performance against validation and test datasets revealed notable results, highlighting the 1D CNN model's superior performance in predicting both pollutants due to its lower validation and test MAE values compared to the GRU model.

Results analysis highlighted that while the 1D CNN model outperformed GRU overall, intriguingly, GRU excelled in predicting NO_x emissions instead of CO emissions, whereas 1D CNN exhibited superior performance for CO prediction. This finding diverges from conventional recommendations favoring GRU over 1D CNN in sequences where temporal patterns are not invariant to shifts. For instance, in time series data where recent observations hold more significance than distant historical data points.

Ultimately, the 1D CNN model emerged as the optimal choice for predicting pollutant emissions due to its adaptability to forecast unseen data instances and its ability to effectively handle abrupt changes in the target variables, surpassing the GRU model's performance.

In summary, this Thesis navigates through a multifaceted exploration of aircraft emissions prediction, showcasing the significance of methodologies, data transformations and model performances in predicting emissions from gas turbines. The findings shed light on the intricacies of emissions prediction, indicating promising directions for future research and practical applications in the aviation industry's quest for pollution reduction.

All in all, it can be concluded that the objectives of this Thesis have been fulfilled.

6.1. Way forward

Looking at the results obtained in the different chapters, it is easy to identify some ways in which this project could be improved and continued, which are the ones that follow:

- To enhance the mitigation of overfitting and achieve greater stability in the training and validation loss graphs, the utilization of data augmentation could have been considered.

Data augmentation involves creating new training data by applying various transformations or modifications to existing data. These alterations include rotations, flips, noise addition, etc., diversifying the dataset without changing its fundamental characteristics. By providing the model with more varied instances of the same data, data augmentation helps prevent overfitting. Augmentation mitigates this by exposing the model to a wider range of data variations, improving its robustness and generalization to unseen instances. Ultimately, data augmentation aids in enhancing the model's performance and mitigating overfitting by presenting diverse perspectives within the same dataset.

- As the carbon monoxide and nitrogen oxides emission issues are not decoupled, stemming from the same gas turbine and significantly influenced by similar features, there was potential for a joint prediction of both values rather than separating their predictions into two independent problems, through multi-prediction models.
- As observed from the performance of the GRU model, which did not meet to perform as well as intended, it might be beneficial to explore alternative models such as LSTM. This exploration could help determine whether their performance is superior or not through comparative analysis.
- As in any machine or deep learning problem, computational constraints significantly influence solution implementation. One potential avenue for enhancing this project involves assessing a broader range of hyperparameters or incorporating more complex networks into the models. However, it's crucial to note that more complex networks might exacerbate the issue of overfitting. Nevertheless, this approach could potentially yield even more reliable results.
- To conduct further verification and conclusively demonstrate the model's capability to predict future emission data, it would be interesting to extend predictions beyond the years for which data is currently available.
  Extending a work beyond available data involves extrapolation, a theoretical process fundamental in forecasting trends or patterns beyond the observed timeframe. This method relies on assuming that the established patterns or relationships within the existing data will continue to hold true into the future. It necessitates employing statistical models or mathematical algorithms that project the current trajectory onto forthcoming periods. However, extrapolation comes with inherent risks, as it assumes stability in underlying factors, potentially disregarding unforeseen shifts or disruptions. Utilizing extrapolation requires a cautious approach, acknowledging its limitations and the need for continuous validation to ensure the credibility of extended predictions.

All these options are interesting in themselves, but combined would result in a more complete project, covering an even wider field.

Bibliography

[1] Swiss Federal Laboratories for Materials Science and Technology. *Jet engines to become cleaner in future.* Phys.org. February 26, 2016.

[2] W.R. Graham, C.A. Hall, M. Vera Morales. *The potential of future aircraft technology for noise and pollutant emissions reduction.* Transport Policy 2014; 34, 36-51. doi: 10.1016/j.tranpol.2014.02.017

[3] Albert A. Presto, Ngoc T. Nguyen, Manish Ranjan, Aaron J. Reeder, Eric M. Lipsky, Christopher J. Hennigan, Marissa A. Miracolo, Daniel D. Riemer, Allen L. Robinson. *Fine particle and organic vapor emissions from staged tests of an in-use aircraft engine.* Atmospheric Environment 2011; 45 (21), 3603-3612. doi: 10.1016/j.atmosenv.2011.03.061

[4] Bruce E. Anderson, Gao Chen, Donald R. Blake. *Hydrocarbon emissions from a modern commercial airliner.* Atmospheric Environment 2006; 40 (19), 3601-3612. doi: 10.1016/j.atmosenv.2005.09.072

[5] Lewis, Jerry S., et al. *Aircraft technology and its relation to emissions. Aviation and the Global Atmosphere.* ISBN: 9780521663007. June 1999.

[6] David S. Lee, David W. Fahey, Piers M. Forster, Peter J. Newton, Ron C.N. Wit, Ling L. Lim, Bethan Owen, Robert Sausen. *Aviation and global climate change in the 21st century.* Atmospheric Environment 2009; 43 (22-23), 3520-3537. doi: 10.1016/j.atmosenv.2009.04.024

[7] Intergovernmental Panel on Climate Change. *Aviation and the Global Atmosphere.*

[8] Mauro Masiol, Roy M. Harrison. *Aircraft engine exhaust emissions and other airport-related contributions to ambient air pollution: A review.* Atmospheric Environment 2014; 95, 409-455. doi: 10.1016/j.atmosenv.2014.05.070

[9] ICAO (International Civil Aviation Organization). *Annex 16 - Environmental Protection.* Volume II - Aircraft Engine Emissions. July 2023.

[10] ICAO (International Civil Aviation Organization). *Airport Air Quality Manual.* 2011.

[11] Brandon Graver. *POLISHING MY CRYSTAL BALL: AIRLINE TRAFFIC IN 2050.* ICCT (International Council on Clean Transportation). January 31, 2022.

[12] Tim C. Lieuwen, Vigor Yang, et al. *Gas Turbine Emissions.* Cambridge University Press. September 2013.

[13] Jermanto S. Kurniawan and S. Khardi. *Comparison of methodologies estimating emissions of aircraft pollutants, environmental impact assessment around airports.* Environmental Impact Assessment Review 2011; 31 (3), 240-252. doi: 10.1016/j.eiar.2010.09.001

[14] Bethan Owen, Julien G. Anet, Nicolas Bertier, Simon Christie, Michele Cremaschi, et al. *Review: Particulate matter emissions from aircraft.* Atmosphere 2012; 13 (8), 1230. doi: 10.3390/atmos13081230

[15] J.H. Seinfeld and S.N. Pandis. *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change.* Third Edition. Hoboken, New Jersey: John Wiley & Sons. 2016.

[16] Evangelia Samoli, Giota Touloumi, Joel Schwartz, Hugh Ross Anderson, Christian Schindler, Bertil Forsberg, Maria Angela Vigotti, Judith Vonk, Mitja Košnik, Jiri Skorkovsky and Klea Katsouyanni. *Short-Term Effects of Carbon Monoxide on Mortality: An Analysis within the APHEA Project.* Environmental Health Perspectives 2007; 115 (11), 1578-1583. doi: 10.1289/ehp.103

[17] Sutkus Jr., D.J., Baughcum, S.L., DuBois, D.P. *Scheduled Civil Aircraft Emission Inventories for 1999: Database Development and Analysis.* NASA/CR 2001-211216. October 2001.

[18] ICCT (International Council on Clean Transportation). *PROPOSED EPA CO2 STANDARD LAGS NEW AIRCRAFT FUEL EFFICIENCY BY MORE THAN A DECADE.* July 22, 2020.

[19] Joda Wormhoudt, Scott C. Herndon, Paul E. Yelvington, Richard C. Miake-Lye and Changlie Wey. *Nitrogen Oxide (NO/NO2/HONO) Emissions Measurements in Aircraft Exhausts.* Journal of Propulsion and Power 2007; 23 (5). doi: 10.2514/1.23461

[20] Ezra C. Wood, Scott C. Herndon, Michael T. Timko, Paul E. Yelvington, and Richard C. Miake-Lye. *Speciation and Chemical Evolution of Nitrogen Oxides in Aircraft Exhaust near Airports.* Environmental Science & Technology 2008; 42 (6), 1884-1891. doi: 10.1021/es072050a

[21] Hannah Ritchie. *Climate change and flying: what share of global CO2 emissions come from aviation?.* Our World in Data. October 2020.

[22] International Energy Agency. *Tracking Aviation.*

[23] EASA. *Updated analysis of the non-CO2 climate impacts of aviation and potential policy measures pursuant to EU Emissions Trading System Directive Article 30(4).* November 25, 2020.

[24] ICAO (International Civil Aviation Organization). *Environmental Protection - Local Air Quality.*

[25] ICAO (International Civil Aviation Organization). *Environmental Protection - Climate Change.*

[26] International Air Transport Association (IATA). *Offsetting CO2 Emissions with CORSIA.*

[27] ICAO (International Civil Aviation Organization). *Environmental Protection - Long term global aspirational goal (LTAG) for international aviation.*

[28] International Air Transport Association (IATA). *Net zero 2050: operational and infrastructure improvements.* June 2023.

[29] ICAO (International Civil Aviation Organization). *Sustainable Aviation Fuel (SAF).*

[30] Airbus. *Hybrid and electric flight.*

[31] International Air Transport Association (IATA). *Developing Sustainable Aviation Fuel (SAF).*

[32] Rolls-Royce. *Rolls-Royce Lean-Burn Combustion test engine runs for first time.*

[33] Cat Hofacker. *Battling those other emissions: nitrogen oxides.* American Institute of Aeronautics and Astronautics (AIAA). February/March 2021.

[34] Heysem Kaya, Pinar Tüfekci and Erdin Uzun. *Predicting CO and NOx emissions from gas turbines: novel data and a benchmark PEMS.* Turkish Journal of Electrical Engineering and Computer Sciences 2019; 27 (6), 4783-4796. doi: 10.3906/elk-1807-87

[35] CMC Solutions. *PEMS vs. CEMS. A side by side comparison.*

[36] Timo Korpela, Pekka Kumpulainen, Yrjö Majanne and Anna Häyrinen. *Model based NOx emission monitoring in natural gas fired hot water boilers.* IFAC-PapersOnLine 2015; 48 (30), 385-390. doi: 10.1016/j.ifacol.2015.12.409

[37] M. Shakil, M. Elshafei, M.A. Habib and F. Maleki. *Soft sensor for NO_x and O_2 using dynamic neural networks.* Computers & Electrical Engineering 2009; 35 (4), 578-586. doi: 10.1016/j.compeleceng.2008.08.007

[38] You Lv, Jizhen Liu, Tingting Yang and Deliang Zeng. *A novel least squares support vector machine ensemble model for NO_x emission prediction of a coal-fired boiler.* Energy 2013; 55, 319-329. doi: 10.1016/j.energy.2013.02.062

[39] Ćirić Ivan T., Ćojbašić Žarko M., Nikolić Vlastimir D., Živković Predrag M. and Tomić Mladen A.. *Air quality estimation by computational intelligence methodologies.* Thermal Science 2012; 16, S493-S504. doi: 10.2298/TSCI120503186C

[40] Elia Georgiana Dragomir and Mihaela Oprea. *A Multi-Agent System for Power Plants Air Pollution Monitoring*. IFAC Proceedings Volumes 2013; 46 (6), 89-94. doi: 10.3182/20130522-3-RO-4035.00017

[41] Saiful Idzwan Abdul Hamid, Chen Chai Phing and Tiong Sieh Kiong. *Prediction of NO_x using support vector machine for gas turbine emission at Putrajaya power station*. Journal of Advanced Science and Engineering Research 2014; 4 (1), 37-46.

[42] M. Liukkonen and T. Hiltunen. *Monitoring and analysis of air emissions based on condition models derived from process history*. Cogent Engineering 2016; 3 (1), 1174182. doi: 10.1080/23311916.2016.1174182

[43] A. Filippone, B. Parkes, N. Bojdo and T. Kelly. *Prediction of aircraft engine emissions using ADS-B flight data*. The Aeronautical Journal 2021; 125 (1288), 988-1012. doi: 10.1017/aer.2021.2

[44] Z. Saboohi and F. Ommi. *Emission prediction in conceptual design of the aircraft engines using augmented CRN*. The Aeronautical Journal 2017; 121 (1241), 1005-1028. doi: 10.1017/aer.2017.40

[45] K. Kayaalp, S. Metlek, S. Ekici and Y. Şöhret. *Developing a model for prediction of the combustion performance and emissions of a turboprop engine using the long short-term memory method*. Fuel 2021; 302, 121202. doi: 10.1016/j.fuel.2021.121202

[46] Ahmed Abulkhair. *Data Imputation Demystified | Time Series Data*. Medium. June 2013.

[47] The Scipy community. *Interpolation (scipy.interpolate)*.

[48] Aditya Mishra. *Metrics to Evaluate your Machine Learning Algorithm*. Towards Data Science. February 24, 2018.

[49] James Brennan. *Confidence intervals for LOWESS models in python*. March 23, 2020.

[50] Neal B. Gallagher. *Savitzky-Golay Smoothing and Differentiation Filter*. Eigenvector.

[51] ScienceDirect. *Butterworth Filter*.

[52] *Applying Butterworth filtering to an X–Y data object*.

[53] Geeks for Geeks. *Exploring Correlation in Python*. March 16, 2023.

[54] DATAtab. *Kendall's Tau*.

[55] Aryan Gupta. *Spearman's Rank Correlation: The Definitive Guide To Understand*. Simplilearn. February 2, 2023.

[56] LabXchange. *How to Interpret Histograms*. July 6, 2021.

[57] Jason Brownlee. *How to Decompose Time Series Data into Trend and Seasonality*. Machine Learning Mastery. December 10, 2020.

[58] Brian Beers. *P-Value: What It Is, How to Calculate It, and Why It Matters*. Investopedia. March 28, 2023.

[59] Vijay Kumar G. *Statistical Tests to Check Stationarity in Time Series*. Analytics Vidhya. September 13, 2023.

[60] Ilyas Ahmed. *What are ACF and PACF Plots in Time Series Analysis?* Medium. May 28, 2023.

[61] Pennsylvania State University. *Lesson 13: Canonical Correlation Analysis*.

[62] François Chollet. *Deep Learning with Python*. Version 6. Manning Publications. 2017.

[63] Yugesh Verma. *A Complete Understanding of Dense Layers in Neural Networks*. Analytics India Magazine. September 19, 2021.

[64] Keras. *MaxPooling1D layer*.

[65] Mohamed Ragab, Said Jadid Abdulkadir, NorShakirah Aziz, Qasem Al-Tashi, Yousif Alyousifi, Hitham Alhussian, and Alawi Alqushaibi. *A Novel One-Dimensional CNN with Exponential Adaptive Gradients for Air Pollution Index Prediction*. Sustainability 2020; 12 (23), 10090. doi: 10.3390/su122310090

[66] Keras. *GlobalMaxPooling1D layer*.

[67] Jason Brownlee. *Understand the Impact of Learning Rate on Neural Network Performance*. Machine Learning Mastery. September 12, 2020.

[68] Ayush Gupta. *A Comprehensive Guide on Optimizers in Deep Learning*. Analytics Vidhya. September 13, 2023.

[69] Sagar Sharma. *Activation Functions in Neural Networks*. Medium. September 6, 2017.

[70] Keras. *EarlyStopping*.

[71] UCI Machine Learning Repository. *Gas Turbine CO and NOx Emission Data Set*. 2019. doi: 10.24432/C5WC95

[72] HOLIDAY-WEATHER.COM. *ISTANBUL ANNUAL WEATHER AVERAGES, TURKEY*.

Appendix

I. Python code

```
# Library import
%matplotlib inline
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
from datetime import timedelta
from scipy.interpolate import interp1d
from scipy import signal
from statsmodels.nonparametric.smoothers_lowess import lowess
from statsmodels.tsa.seasonal import seasonal_decompose
import statistics
from statsmodels.tsa.stattools import adfuller, kpss
from pandas.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense
from sklearn.metrics import mean_squared_error, mean_absolute_error
from scipy.signal import butter, filtfilt
from sklearn.cross_decomposition import CCA
from keras import layers
from keras.regularizers import l2
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from keras.utils import plot_model
sns.set_theme()

# Suppress warnings
import warnings
warnings.filterwarnings('ignore')
```

```
# Read csvs
gt_2011 = pd.read_csv("gt_2011.csv")
gt_2012 = pd.read_csv("gt_2012.csv")
gt_2013 = pd.read_csv("gt_2013.csv")
gt_2014 = pd.read_csv("gt_2014.csv")
gt_2015 = pd.read_csv("gt_2015.csv")

# Join all of the data in a single Dataframe
gt = pd.concat([gt_2011,gt_2012,gt_2013,gt_2014,gt_2015], ignore_index=True)

# Erase limitations on vectors and columns visualization
pd.options.display.max_columns = None
np.set_printoptions(linewidth=1000)
```

```
# See what attributes we have in the different datasets
gt_2011.head()
```

```
gt_2012.head()
```

```
gt_2013.head()
```

```
gt_2014.head()
```

```
gt_2015.head()
```

```python
gt.head()
```

```python
# Information of the different attributes we have in the different Datasets
gt_2011.info()
gt_2012.info()
gt_2013.info()
gt_2014.info()
gt_2015.info()
gt.info()
```

```python
# Check for null values
np.round(gt.isnull().mean() * 100,1)
```

```python
# First, the index will be converted to datetime in hourly intervals
def add_date(df, date_a, date_b):
  delta = timedelta(minutes=60)
  timestamps = [date_a]
  date_x = date_a
  while date_x < date_b:
    date_x += timedelta(minutes=60)
    timestamps.append(date_x)
  Date = [d.strftime('%Y-%m-%d %H:%M:%S') for d in timestamps]
  df['datetime'] = Date
  df['datetime'] = pd.to_datetime(df['datetime'])

add_date(gt_2011, datetime.datetime(2011, 1, 1, 0),
         datetime.datetime(2011, 11, 5, 18))
add_date(gt_2012, datetime.datetime(2012, 1, 1, 0),
         datetime.datetime(2012, 11, 13, 19))
add_date(gt_2013, datetime.datetime(2013, 1, 1, 0),
         datetime.datetime(2013, 10, 25, 23))
add_date(gt_2014, datetime.datetime(2014, 1, 1, 0),
         datetime.datetime(2014, 10, 26, 5))
add_date(gt_2015, datetime.datetime(2015, 1, 1, 0),
         datetime.datetime(2015, 11, 4, 15))

gt_2011_ts = gt_2011.set_index('datetime')
gt_2012_ts = gt_2012.set_index('datetime')
gt_2013_ts = gt_2013.set_index('datetime')
gt_2014_ts = gt_2014.set_index('datetime')
gt_2015_ts = gt_2015.set_index('datetime')

gt_ts = pd.concat([gt_2011_ts,gt_2012_ts,gt_2013_ts,gt_2014_ts,gt_2015_ts])
```

```python
# Filling missing values - Cell used for testing different methods
end_time = datetime.datetime(2015, 12, 31, 23).strftime('%Y-%m-%d %H:%M:%S')

all_times = pd.date_range(start=gt_ts.index.min(), end=end_time, freq='H')
df = gt_ts.reindex(all_times)

def model_test(df, value):
  train = df['2011-01-01':'2012-12-31']
  X_train = train.index
  y_train = train[value]
  val = df['2013-01-01':'2013-12-31']
  X_val = val.index
  y_val = val[value]
  test = df['2014-01-01':'2015-12-31']
  X_test = test.index
  y_test = test[value]
  X_train = X_train.astype(int) // 10**9
  X_val = X_val.astype(int) // 10**9
  X_test = X_test.astype(int) // 10**9
  X_train = np.array(X_train)
  X_val = np.array(X_val)
  X_test = np.array(X_test)
  X_train = X_train.reshape(-1, 1, 1)
  X_val = X_val.reshape(-1, 1, 1)
  X_test = X_test.reshape(-1, 1, 1)
  model = Sequential()
  model.add(GRU(64, activation='tanh', input_shape=(None, 1)))
  model.add(Dense(1, activation='linear'))
  model.compile(optimizer='adam', loss='mean_squared_error',
                metrics=['mean_absolute_error'])
  model.fit(X_train, y_train, epochs=20, batch_size=32,
            validation_data=(X_val, y_val))
  loss, mean_absolute_error = model.evaluate(X_test, y_test)

  print(f'Loss in test data: {loss}')
  print(f'MAE in test data: {mean_absolute_error}')


def fill_test(df, value, xlabel, ylabel):
  ## 1. Forward Fill
  plt.figure(figsize=(16,5))
  df['ffill'] = df[value].ffill()
  df['ffill'].plot(color='y')
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.title('Forward Fill')
  plt.show()

  ## 2. Linear Interpolation
  plt.figure(figsize=(16,5))
  df['rownum'] = np.arange(df.shape[0])
  df_nona = df.dropna(subset = [value])
  f = interp1d(df_nona['rownum'], df_nona[value],
               bounds_error=False, fill_value=3)
  df['linear_fill'] = f(df['rownum'])
  df['linear_fill'].plot(color='c')
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.title('Linear Interpolation')
  plt.show()

  ## 3. Seasonal Mean
  def seasonal_mean(ts, n):
    out = np.copy(ts)
    for i, val in enumerate(ts):
        if np.isnan(val):
            ts_seas = ts[i-1::-n] # previous seasons only
            if np.isnan(np.nanmean(ts_seas)):
              ts_seas = np.concatenate([ts[i-1::-n], ts[i::n]]) # prev and forw
            out[i] = np.nanmean(ts_seas)
    return out

  plt.figure(figsize=(16,5))
  df['seasonal_mean'] = seasonal_mean(df[value], n=2920) # 2920     4 months
  df['seasonal_mean'].plot(color='m')
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.title('Seasonal Mean')
  plt.show()

  ## 4. Mean of 'n' Nearest Past Neighbors (KNN)
  def knn_mean(ts, n):
    out = np.copy(ts)
    for i, val in enumerate(ts):
        if np.isnan(val):
            n_by_2 = np.ceil(n/2)
            lower = np.max([0, int(i-n_by_2)])
            upper = np.min([len(ts)+1, int(i+n_by_2)])
            ts_near = np.concatenate([ts[lower:i], ts[i:upper]])
            out[i] = np.nanmean(ts_near)
    return out

  plt.figure(figsize=(16,5))
  df['knn_mean'] = knn_mean(df[value].values, 2920) # 2920     4 months
  df['knn_mean'].plot(color='r')
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.title('KNN')
  plt.show()

  return df
```

```

def method_test(df, value, xlabel, ylabel):
  fill_test(df, value, xlabel, ylabel)
  model_test(df, value='ffill')
  model_test(df, value='linear_fill')
  model_test(df, value='seasonal_mean')
  model_test(df, value='knn_mean')

method_test(df, value='AT', xlabel='Years',
            ylabel='Ambient Temperature [ C ]')
method_test(df, value='AP', xlabel='Years',
            ylabel='Ambient Pressure [mbar]')
method_test(df, value='AH', xlabel='Years',
            ylabel='Ambient Humidity [%]')
method_test(df, value='AFDP', xlabel='Years',
            ylabel='Air Filter Difference Pressure [mbar]')
method_test(df, value='GTEP', xlabel='Years',
            ylabel='Gas Turbine Exhaust Pressure [mbar]')
method_test(df, value='TIT', xlabel='Years',
            ylabel='Turbine Inlet Temperature [ C ]')
method_test(df, value='TAT', xlabel='Years',
            ylabel='Turbine After Temperature [ C ]')
method_test(df, value='TEY', xlabel='Years',
            ylabel='Turbine Energy Yield [MWH]')
method_test(df, value='CDP', xlabel='Years',
            ylabel='Compressor Discharge Pressure [mbar]')
method_test(df, value='CO', xlabel='Years',
            ylabel='Carbon Monoxide [$mg/m^3$]')
method_test(df, value='NOX', xlabel='Years',
            ylabel='Nitrogen Oxides [$mg/m^3$]')
```

```
# Plot year intervals (full time series) filling missing values -
#          Seasonal Mean + Smoothed temporal series - LOWESS 2%
end_time = datetime.datetime(2015, 12, 31, 23).strftime('%Y-%m-%d %H:%M:%S')
all_times = pd.date_range(start=gt_ts.index.min(), end=end_time, freq='H')
gt_ts = gt_ts.reindex(all_times)

def seasonal(df, value):
  def seasonal_mean(ts, n):
    out = np.copy(ts)
    for i, val in enumerate(ts):
        if np.isnan(val):
            ts_seas = ts[i-1::-n] # previous seasons only
            if np.isnan(np.nanmean(ts_seas)):
                ts_seas = np.concatenate([ts[i-1::-n], ts[i::n]]) # prev & forw
            out[i] = np.nanmean(ts_seas)
    return out

  df[value] = seasonal_mean(df[value].values, 2920) # 2920     4 months
  return df

def seasonal_AT(df, value):
  def seasonal_mean_AT(ts, n):
    out = np.copy(ts)
    for i, val in enumerate(ts):
        if np.isnan(val):
            ts_seas = ts[i-1::-n]  # previous seasons only
            if np.isnan(np.nanmean(ts_seas)):
                ts_seas = np.concatenate([ts[i-1::-n], ts[i::n]]) # prev & forw
            out[i] = np.nanmean(ts_seas) / 2
    return out

  df[value] = seasonal_mean_AT(df[value].values, 2920) # 2920     4 months
  return df

def lowess_plot(df, value, xlabel, ylabel, legend1, legend2):
  df_lowess = pd.DataFrame(lowess(df[value], np.arange(len(df[value])),
                                  frac=0.02)[:, 1], index=df.index,
                           columns=[value])
  plt.figure(figsize=(16,5))
  plt.plot(df.index, df[value], color='m')
```

```python
  plt.plot(df.index, df_lowess, color='r', linewidth=2)
  mean = np.mean(df[value])
  std_dev = statistics.stdev(df[value])
  ymax = mean + (std_dev/2)
  ymin = mean - (std_dev/2)
  plt.axhline(y=mean, color='k', linestyle='--', linewidth=2)
  plt.axhline(y=ymax, color='g', linestyle='--', linewidth=2)
  plt.axhline(y=ymin, color='g', linestyle='--', linewidth=2)
  plt.legend([legend1, legend2, 'Mean', 'Std Deviation'])
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

def lowess_AFDP(df, value, xlabel, ylabel, legend1, legend2):
  df_lowess = pd.DataFrame(lowess(df[value], np.arange(len(df[value])),
                                  frac=0.02)[:, 1], index=df.index,
                           columns=[value])
  plt.figure(figsize=(16,5))
  plt.plot(df.index, df[value], color='m')
  plt.plot(df.index, df_lowess, color='r', linewidth=2)
  mean = np.mean(df[value])
  std_dev = statistics.stdev(df[value])
  ymax = mean + (std_dev/2)
  ymin = mean - (std_dev/2)
  plt.axhline(y=mean, color='k', linestyle='--', linewidth=2)
  plt.axhline(y=ymax, color='g', linestyle='--', linewidth=2)
  plt.axhline(y=ymin, color='g', linestyle='--', linewidth=2)
  plt.axvspan(gt_ts.index[1650], gt_ts.index[1950], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[10375], gt_ts.index[10675], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[15000], gt_ts.index[15300], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[16000], gt_ts.index[16300], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[26850], gt_ts.index[27150], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.legend([legend1, legend2, 'Mean', 'Std Deviation'])
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

def lowess_TIT(df, value, xlabel, ylabel, legend1, legend2):
  df_lowess = pd.DataFrame(lowess(df[value], np.arange(len(df[value])),
                                  frac=0.02)[:, 1], index=df.index,
                           columns=[value])
  plt.figure(figsize=(16,5))
  plt.plot(df.index, df[value], color='m')
  plt.plot(df.index, df_lowess, color='r', linewidth=2)
  mean = np.mean(df[value])
  std_dev = statistics.stdev(df[value])
  ymax = mean + (std_dev/2)
  ymin = mean - (std_dev/2)
  plt.axhline(y=mean, color='k', linestyle='--', linewidth=2)
  plt.axhline(y=ymax, color='g', linestyle='--', linewidth=2)
  plt.axhline(y=ymin, color='g', linestyle='--', linewidth=2)
  plt.axvspan(gt_ts.index[6150], gt_ts.index[6650], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[27400], gt_ts.index[27750], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[32100], gt_ts.index[32750], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.legend([legend1, legend2, 'Mean', 'Std Deviation'])
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

def lowess_TAT(df, value, xlabel, ylabel, legend1, legend2):
  df_lowess = pd.DataFrame(lowess(df[value], np.arange(len(df[value])),
                                  frac=0.02)[:, 1], index=df.index,
                           columns=[value])
  plt.figure(figsize=(16,5))
  plt.plot(df.index, df[value], color='m')
  plt.plot(df.index, df_lowess, color='r', linewidth=2)
  mean = np.mean(df[value])
  std_dev = statistics.stdev(df[value])
  ymax = mean + (std_dev/2)
  ymin = mean - (std_dev/2)
  plt.axhline(y=mean, color='k', linestyle='--', linewidth=2)
  plt.axhline(y=ymax, color='g', linestyle='--', linewidth=2)
  plt.axhline(y=ymin, color='g', linestyle='--', linewidth=2)
  plt.axvspan(gt_ts.index[6300], gt_ts.index[6650], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[9000], gt_ts.index[9750], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[32450], gt_ts.index[32700], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[35000], gt_ts.index[35400], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.legend([legend1, legend2, 'Mean', 'Std Deviation'])
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

def lowess_CO(df, value, xlabel, ylabel, legend1, legend2):
  df_lowess = pd.DataFrame(lowess(df[value], np.arange(len(df[value])),
                                    frac=0.02)[:, 1], index=df.index,
                             columns=[value])
  plt.figure(figsize=(16,5))
  plt.plot(df.index, df[value], color='m')
  plt.plot(df.index, df_lowess, color='r', linewidth=2)
  mean = np.mean(df[value])
  std_dev = statistics.stdev(df[value])
  ymax = mean + (std_dev/2)
  ymin = mean - (std_dev/2)
  plt.axhline(y=mean, color='k', linestyle='--', linewidth=2)
  plt.axhline(y=ymax, color='g', linestyle='--', linewidth=2)
  plt.axhline(y=ymin, color='g', linestyle='--', linewidth=2)
  plt.axvspan(gt_ts.index[2150], gt_ts.index[3200], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[5650], gt_ts.index[7100], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[8600], gt_ts.index[9300], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[10650], gt_ts.index[11950], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[14150], gt_ts.index[15800], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[19050], gt_ts.index[19400], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[22150], gt_ts.index[22500], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[27200], gt_ts.index[27750], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[31900], gt_ts.index[33300], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[35500], gt_ts.index[36800], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[38900], gt_ts.index[39950], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.legend([legend1, legend2, 'Mean', 'Std Deviation'])
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

def lowess_NOx(df, value, xlabel, ylabel, legend1, legend2):
  df_lowess = pd.DataFrame(lowess(df[value], np.arange(len(df[value])),
                                    frac=0.02)[:, 1], index=df.index,
                             columns=[value])
  plt.figure(figsize=(16,5))
  plt.plot(df.index, df[value], color='m')
  plt.plot(df.index, df_lowess, color='r', linewidth=2)
  mean = np.mean(df[value])
  std_dev = statistics.stdev(df[value])
  ymax = mean + (std_dev/2)
  ymin = mean - (std_dev/2)
  plt.axhline(y=mean, color='k', linestyle='--', linewidth=2)
  plt.axhline(y=ymax, color='g', linestyle='--', linewidth=2)
  plt.axhline(y=ymin, color='g', linestyle='--', linewidth=2)
  plt.axvspan(gt_ts.index[5600], gt_ts.index[6000], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[30150], gt_ts.index[30450], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[32850], gt_ts.index[33500], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.axvspan(gt_ts.index[38500], gt_ts.index[38850], facecolor='y',
              edgecolor='none', alpha=.3)
  plt.legend([legend1, legend2, 'Mean', 'Std Deviation'])
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

# Plot and highlight max or min values
seasonal_AT(gt_ts, value='AT')
lowess_plot(gt_ts, 'AT', xlabel='Years', ylabel='Ambient Temperature [ C ]',
            legend1='AT', legend2='AT smoothed')

seasonal(gt_ts, value='AP')
lowess_plot(gt_ts, 'AP', xlabel='Years', ylabel='Ambient Pressure [mbar]',
            legend1='AP', legend2='AP smoothed')

seasonal(gt_ts, value='AH')
lowess_plot(gt_ts, 'AH', xlabel='Years', ylabel='Ambient Humidity [%]',
            legend1='AH', legend2='AH smoothed')

seasonal(gt_ts, value='AFDP')
lowess_AFDP(gt_ts, 'AFDP', xlabel='Years',
            ylabel='Air Filter Difference Pressure [mbar]',
            legend1='AFDP', legend2='AFDP smoothed')

seasonal(gt_ts, value='GTEP')
lowess_plot(gt_ts, 'GTEP', xlabel='Years',
            ylabel='Gas Turbine Exhaust Pressure [mbar]',
            legend1='GTEP', legend2='GTEP smoothed')

seasonal(gt_ts, value='TIT')
lowess_TIT(gt_ts, 'TIT', xlabel='Years',
           ylabel='Turbine Inlet Temperature [ C ]',
           legend1='TIT', legend2='TIT smoothed')

seasonal(gt_ts, value='TAT')
lowess_TAT(gt_ts, 'TAT', xlabel='Years',
           ylabel='Turbine After Temperature [ C ]',
           legend1='TAT', legend2='TAT smoothed')

seasonal(gt_ts, value='TEY')
lowess_plot(gt_ts, 'TEY', xlabel='Years', ylabel='Turbine Energy Yield [MWH]',
            legend1='TEY', legend2='TEY smoothed')

seasonal(gt_ts, value='CDP')
lowess_plot(gt_ts, 'CDP', xlabel='Years',
            ylabel='Compressor Discharge Pressure [mbar]',
            legend1='CDP', legend2='CDP smoothed')

seasonal(gt_ts, value='CO')
lowess_CO(gt_ts, 'CO', xlabel='Years', ylabel='Carbon Monoxide [$mg/m^3$]',
          legend1='CO', legend2='CO smoothed')

seasonal(gt_ts, value='NOX')
lowess_NOx(gt_ts, 'NOX', xlabel='Years', ylabel='Nitrogen Oxides [$mg/m^3$]',
           legend1='NOx', legend2='NOx smoothed')
```

```
# Data visualization - Histograms and tendency
fig, axes = plt.subplots(4,3, figsize=(20,17))

sns.histplot(data=gt_ts, x="AT", kde=True,ax=axes[0,0])
axes[0,0].set_xlabel("Ambient Temperature [ C ]")
axes[0,0].set_ylabel("Count")

```

```
sns.histplot(data=gt_ts, x="AP", kde=True,ax=axes[0,1])
axes[0,1].set_xlabel("Ambient Pressure [mbar]")
axes[0,1].set_ylabel("Count")

sns.histplot(data=gt_ts, x="AH", kde=True,ax=axes[0,2])
axes[0,2].set_xlabel("Ambient Humidity [%]")
axes[0,2].set_ylabel("Count")

sns.histplot(data=gt_ts, x="AFDP", kde=True,ax=axes[1,0])
axes[1,0].set_xlabel("Air Filter Difference Pressure [mbar]")
axes[1,0].set_ylabel("Count")

sns.histplot(data=gt_ts, x="GTEP", kde=True,ax=axes[1,1])
axes[1,1].set_xlabel("Gas Turbine Exhaust Pressure [mbar]")
axes[1,1].set_ylabel("Count")

sns.histplot(data=gt_ts, x="TIT", kde=True,ax=axes[1,2])
axes[1,2].set_xlabel("Turbine Inlet Temperature [ C ]")
axes[1,2].set_ylabel("Count")

sns.histplot(data=gt_ts, x="TAT", kde=True,ax=axes[2,0])
axes[2,0].set_xlabel("Turbine After Temperature [ C ]")
axes[2,0].set_ylabel("Count")

sns.histplot(data=gt_ts, x="TEY", kde=True,ax=axes[2,1])
axes[2,1].set_xlabel("Turbine Energy Yield [MW]")
axes[2,1].set_ylabel("Count")

sns.histplot(data=gt_ts, x="CDP", kde=True,ax=axes[2,2])
axes[2,2].set_xlabel("Compressor Discharge Pressure [mbar]")
axes[2,2].set_ylabel("Count")

sns.histplot(data=gt_ts, x="CO", kde=True,ax=axes[3,0])
axes[3,0].set_xlabel("Carbon Monoxide [$mg/m^3$]")
axes[3,0].set_ylabel("Count")

sns.histplot(data=gt_ts, x="NOX", kde=True,ax=axes[3,1])
axes[3,1].set_xlabel("Nitrogen Oxides [$mg/m^3$]")
axes[3,1].set_ylabel("Count")
```

```
# Correlation matrix - Pearson
corr_df = gt_ts.corr(method='pearson')

plt.figure(figsize=(10,6))
sns.heatmap(corr_df, annot=True)
plt.show()
```

```
# Correlation matrix - Kendall
corr_df = gt_ts.corr(method='kendall')

plt.figure(figsize=(10,6))
sns.heatmap(corr_df, annot=True)
plt.show()
```

```
# Correlation matrix during Winter - Pearson
corr_wint = gt_ts.loc['2011-01-01'].corr(method='pearson')

plt.figure(figsize=(10,6))
sns.heatmap(corr_wint, annot=True)
plt.show()
```

```
# Correlation matrix during Summer - Pearson
corr_summ = gt_ts.loc['2011-08-01'].corr(method='pearson')

plt.figure(figsize=(10,6))
sns.heatmap(corr_summ, annot=True)
plt.show()
```

```
# Data visualization - Pairplot
sns.pairplot(gt_ts, corner=True, kind='hist');
```

```
# Data visualization - Lmplot for CO
ax = sns.lmplot(
  x='value',
  y='CO',
  data=gt.melt(id_vars='CO', value_vars=gt_ts),
  col='variable',
  col_wrap=3,
  sharex=False,
  sharey=False,
  line_kws={'color': 'red'},
  scatter_kws={'s': 5})
ax.tight_layout()
```

```
# Data visualization - Lmplot for NOx
ax = sns.lmplot(
  x='value',
  y='NOX',
  data=gt.melt(id_vars='NOX', value_vars=gt_ts),
  col='variable',
  col_wrap=3,
  sharex=False,
  sharey=False,
  line_kws={'color': 'red'},
  scatter_kws={'s': 5})
ax.tight_layout();
```

```
# Seasonal plot - Savitzky-Golay filter
def seas_plot(df, value):
  y = signal.savgol_filter(df[value], window_length=100, polyorder=1)
  plt.plot(df.index, y)
  return

def joint_plot_s(value, xlabel, ylabel):
  plt.figure(figsize=(16,5))
  seas_plot(gt_2011_ts, value)
  seas_plot(gt_2012_ts, value)
  seas_plot(gt_2013_ts, value)
  seas_plot(gt_2014_ts, value)
  seas_plot(gt_2015_ts, value)
  mean = np.mean(gt_ts[value])
  std_dev = statistics.stdev(gt_ts[value])
  ymax = mean + (std_dev/2)
  ymin = mean - (std_dev/2)
  plt.axhline(y=mean, color='k', linestyle='--', linewidth=2)
  plt.axhline(y=ymax, color='red', linestyle='--', linewidth=2)
  plt.axhline(y=ymin, color='red', linestyle='--', linewidth=2)
  plt.legend(['2011', '2012', '2013', '2014', '2015', 'Mean', 'Std Deviation'])
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

joint_plot_s('AT', xlabel='Year',
             ylabel='Ambient Temperature [ C ]')
joint_plot_s('AP', xlabel='Year',
             ylabel='Ambient Pressure [mbar]')
joint_plot_s('AH', xlabel='Year',
             ylabel='Ambient Humidity [%]')
joint_plot_s('AFDP', xlabel='Year',
             ylabel='Air Filter Difference Pressure [mbar]')
joint_plot_s('GTEP', xlabel='Year',
             ylabel='Gas Turbine Exhaust Pressure [mbar]')
joint_plot_s('TIT', xlabel='Year',
             ylabel='Turbine Inlet Temperature [ C ]')
joint_plot_s('TAT', xlabel='Year',
             ylabel='Turbine After Temperature [ C ]')
joint_plot_s('TEY', xlabel='Year',
             ylabel='Turbine Energy Yield [MWH]')
joint_plot_s('CDP', xlabel='Year',
             ylabel='Compressor Discharge Pressure [mbar]')
joint_plot_s('CO', xlabel='Year',
             ylabel='Carbon Monoxide [$mg/m^3$]')
joint_plot_s('NOX', xlabel='Year',
            ylabel='Nitrogen Oxides [$mg/m^3$]')
```

```
# Joint time series with normalization and comparison -
#          They start at the same point
scaler = StandardScaler()
df_normalized = scaler.fit_transform(gt_ts)
df_norm = pd.DataFrame(df_normalized, index=gt_ts.index,
                       columns=['AT', 'AP', 'AH', 'AFDP', 'GTEP', 'TIT', 'TAT',
                                'TEY', 'CDP', 'CO', 'NOX'])

loess_GTEP = pd.DataFrame(lowess(df_norm['GTEP'],
                                 np.arange(len(df_norm['GTEP'])),
                                 frac=0.02)[:, 1], index=df_norm.index,
                          columns=['GTEP'])
loess_AFDP = pd.DataFrame(lowess(df_norm['AFDP'],
                                 np.arange(len(df_norm['AFDP'])),
                                 frac=0.02)[:, 1], index=df_norm.index,
                          columns=['AFDP'])
loess_TEY = pd.DataFrame(lowess(df_norm['TEY'],
                                np.arange(len(df_norm['TEY'])),
                                frac=0.02)[:, 1], index=df_norm.index,
                         columns=['TEY'])
loess_CDP = pd.DataFrame(lowess(df_norm['CDP'],
                                np.arange(len(df_norm['CDP'])),
                                frac=0.02)[:, 1], index=df_norm.index,
                         columns=['CDP'])
loess_AH = pd.DataFrame(lowess(df_norm['AH'],
                               np.arange(len(df_norm['AH'])),
                               frac=0.02)[:, 1], index=df_norm.index,
                        columns=['AH'])
loess_TIT = pd.DataFrame(lowess(df_norm['TIT'],
                                np.arange(len(df_norm['TIT'])),
                                frac=0.02)[:, 1], index=df_norm.index,
                         columns=['TIT'])
loess_TAT = pd.DataFrame(lowess(df_norm['TAT'],
                                np.arange(len(df_norm['TAT'])),
                                frac=0.02)[:, 1], index=df_norm.index,
                         columns=['TAT'])

plt.figure(figsize=(16,7))
plt.plot(df_norm.index, loess_GTEP, color='r', label='GTEP')
plt.plot(df_norm.index, loess_AFDP, color='g', label='AFDP')
plt.plot(df_norm.index, loess_TEY, color='b', label='TEY')
plt.plot(df_norm.index, loess_CDP, color='y', label='CDP')
plt.plot(df_norm.index, loess_AH, color='m', label='AH')
plt.plot(df_norm.index, loess_TIT, color='k', label='TIT')
plt.plot(df_norm.index, -loess_TAT, color='orange', label='TAT')
plt.gca().set(xlabel='Years')
plt.legend()
plt.show()
```

```
# Additive Decomposition
end_time = datetime.datetime(2015, 12, 31, 23).strftime('%Y-%m-%d %H:%M:%S')

def plot_comp(df, value, results, xlabel, ylabel):
  series = df[value]
  trend_estimate = results.trend
  periodic_estimate = results.seasonal
  residual = results.resid

  plt.figure(figsize=(15,5))
  plt.plot(series, label='Original time series', color='blue')
  plt.plot(trend_estimate, label='Trend of time series', color='red')
  plt.legend(fontsize=10)
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

  plt.figure(figsize=(15,5))
  plt.plot(periodic_estimate, color='green')
  plt.gca().set(xlabel=xlabel, ylabel='Seasonality')
  plt.show()
```

```python

  plt.figure(figsize=(15,5))
  plt.plot(residual, color='orange')
  plt.gca().set(xlabel=xlabel, ylabel='Decomposition residuals')
  plt.show()

  # Actual Values = Sum of (Seasonal + Trend + Resid)
  df_reconstructed = pd.concat([results.seasonal, results.trend, results.resid,
                                results.observed], axis=1)
  df_reconstructed.columns = ['seas', 'trend', 'resid', 'actual_values']
  print(df_reconstructed.head())

def add_decom_plot(df, value, ylabel, end_time=end_time):
  results = seasonal_decompose(df[value], model='additive',
                               extrapolate_trend='freq', period=8764) # 1 year
  plot_comp(df, value, results, xlabel='Years', ylabel=ylabel)

add_decom_plot(gt_ts, value='AT',
               ylabel='Ambient Temperature [ C ]')
add_decom_plot(gt_ts, value='AP',
               ylabel='Ambient Pressure [mbar]')
add_decom_plot(gt_ts, value='AH',
               ylabel='Ambient Humidity [%]')
add_decom_plot(gt_ts, value='AFDP',
               ylabel='Air Filter Difference Pressure [mbar]')
add_decom_plot(gt_ts, value='GTEP',
               ylabel='Gas Turbine Exhaust Pressure [mbar]')
add_decom_plot(gt_ts, value='TIT',
               ylabel='Turbine Inlet Temperature [ C ]')
add_decom_plot(gt_ts, value='TAT',
               ylabel='Turbine After Temperature [ C ]')
add_decom_plot(gt_ts, value='TEY',
               ylabel='Turbine Energy Yield [MWH]')
add_decom_plot(gt_ts, value='CDP',
               ylabel='Compressor Discharge Pressure [mbar]')
add_decom_plot(gt_ts, value='CO',
               ylabel='Carbon Monoxide [$mg/m^3$]')
add_decom_plot(gt_ts, value='NOX',
               ylabel='Nitrogen Oxides [$mg/m^3$]')
```

```python
# ADF Test - Time series stationarity
def ADF(input):
  result = adfuller(input, autolag='AIC')
  print(f'ADF p-value: {result[1]}')

# KPSS Test - Trend stationarity
def KPSS(input):
  result = kpss(input, regression='c')
  print('KPSS p-value: %f' % result[1])

print('Values for Ambient Temperature:')
ADF(gt_ts.AT.values)
KPSS(gt_ts.AT.values)
print('Values for Ambient Pressure:')
ADF(gt_ts.AP.values)
KPSS(gt_ts.AP.values)
print('Values for Ambient Humidity:')
ADF(gt_ts.AH.values)
KPSS(gt_ts.AH.values)
print('Values for Air Filter Difference Pressure:')
ADF(gt_ts.AFDP.values)
KPSS(gt_ts.AFDP.values)
print('Values for Gas Turbine Exhaust Pressure:')
ADF(gt_ts.GTEP.values)
KPSS(gt_ts.GTEP.values)
print('Values for Turbine Inlet Temperature:')
ADF(gt_ts.TIT.values)
KPSS(gt_ts.TIT.values)
print('Values for Turbine After Temperature:')
ADF(gt_ts.TAT.values)
KPSS(gt_ts.TAT.values)
```

```
print('Values for Turbine Energy Yield:')
ADF(gt_ts.TEY.values)
KPSS(gt_ts.TEY.values)
print('Values for Compressor Discharge Pressure:')
ADF(gt_ts.CDP.values)
KPSS(gt_ts.CDP.values)
print('Values for Carbon Monoxide:')
ADF(gt_ts.CO.values)
KPSS(gt_ts.CO.values)
print('Values for Nitrogen Oxides:')
ADF(gt_ts.NOX.values)
KPSS(gt_ts.NOX.values)
```

```
# Detrend the series - Substracting the Trend Component
def detrend(df, input, value, xlabel, ylabel):
  result_add = seasonal_decompose(df[value], model='additive',
                                  extrapolate_trend='freq')
  detrended = input - result_add.trend
  plt.figure(figsize=(15,5))
  plt.plot(detrended)
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()


detrend(gt_ts, input=gt_ts.AT.values, value='AT', xlabel='Years',
        ylabel='Detrended Ambient Temperature [ C ]')
detrend(gt_ts, input=gt_ts.AP.values, value='AP', xlabel='Years',
        ylabel='Detrended Ambient Pressure [mbar]')
detrend(gt_ts, input=gt_ts.AH.values, value='AH', xlabel='Years',
        ylabel='Detrended Ambient Humidity [%]')
detrend(gt_ts, input=gt_ts.AFDP.values, value='AFDP', xlabel='Years',
        ylabel='Detrended Air Filter Difference Pressure [mbar]')
detrend(gt_ts, input=gt_ts.GTEP.values, value='GTEP', xlabel='Years',
        ylabel='Detrended Gas Turbine Exhaust Pressure [mbar]')
detrend(gt_ts, input=gt_ts.TIT.values, value='TIT', xlabel='Years',
        ylabel='Detrended Turbine Inlet Temperature [ C ]')
detrend(gt_ts, input=gt_ts.TAT.values, value='TAT', xlabel='Years',
        ylabel='Detrended Turbine After Temperature [ C ]')
detrend(gt_ts, input=gt_ts.TEY.values, value='TEY', xlabel='Years',
        ylabel='Detrended Turbine Energy Yield [MWH]')
detrend(gt_ts, input=gt_ts.CDP.values, value='CDP', xlabel='Years',
        ylabel='Detrended Compressor Discharge Pressure [mbar]')
detrend(gt_ts, input=gt_ts.CO.values, value='CO', xlabel='Years',
        ylabel='Detrended Carbon Monoxide [$mg/m^3$]')
detrend(gt_ts, input=gt_ts.NOX.values, value='NOX', xlabel='Years',
        ylabel='Detrended Nitrogen Oxides [$mg/m^3$]')
```

```
# Test for seasonality
def seas_test(input, title):
  plt.rcParams.update({'figure.figsize':(12,5), 'figure.dpi':120})
  autocorrelation_plot(input.tolist())
  plt.title(title)
  plt.show()

seas_test(gt_ts.AT, title='Autocorrelation for Ambient Temperature')
seas_test(gt_ts.AP, title='Autocorrelation for Ambient Pressure')
seas_test(gt_ts.AH, title='Autocorrelation for Ambient Humidity')
seas_test(gt_ts.AFDP,
          title='Autocorrelation for Air Filter Difference Pressure')
seas_test(gt_ts.GTEP, title='Autocorrelation for Gas Turbine Exhaust Pressure')
seas_test(gt_ts.TIT, title='Autocorrelation for Turbine Inlet Temperature')
seas_test(gt_ts.TAT, title='Autocorrelation for Turbine After Temperature')
seas_test(gt_ts.TEY, title='Autocorrelation for Turbine Energy Yield')
seas_test(gt_ts.CDP, title='Autocorrelation for Compressor Discharge Pressure')
seas_test(gt_ts.CO, title='Autocorrelation for Carbon Monoxide')
seas_test(gt_ts.NOX, title='Autocorrelation for Nitrogen Oxides')
```

```
# Autocorrelation ACF and Partial Autocorrelation PACF
def acf_pacf(input, title1, title2):
  plot_acf(input)
  plt.title(title1)
  plt.gca().set(xlabel='Lag')
  plt.gca().set(ylabel='Autocorrelation (ACF)')
  plt.show()
  plot_pacf(input)
  plt.title(title2)
  plt.gca().set(xlabel='Lag')
  plt.gca().set(ylabel='Partial Autocorrelation (PACF)')
  plt.show()

acf_pacf(gt_ts.AT, title1='ACF for Ambient Temperature',
         title2='PACF for Ambient Temperature')
acf_pacf(gt_ts.AP, title1='ACF for Ambient Pressure',
         title2='PACF for Ambient Pressure')
acf_pacf(gt_ts.AH, title1='ACF for Ambient Humidity',
         title2='PACF for Ambient Humidity')
acf_pacf(gt_ts.AFDP, title1='ACF for Air Filter Difference Pressure',
         title2='PACF for Air Filter Difference Pressure')
acf_pacf(gt_ts.GTEP, title1='ACF for Gas Turbine Exhaust Pressure',
         title2='PACF for Gas Turbine Exhaust Pressure')
acf_pacf(gt_ts.TIT, title1='ACF for Turbine Inlet Temperature',
         title2='PACF for Turbine Inlet Temperature')
acf_pacf(gt_ts.TAT, title1='ACF for Turbine After Temperature',
         title2='PACF for Turbine After Temperature')
acf_pacf(gt_ts.TEY, title1='ACF for Turbine Energy Yield',
         title2='PACF for Turbine Energy Yield')
acf_pacf(gt_ts.CDP, title1='ACF for Compressor Discharge Pressure',
         title2='PACF for Compressor Discharge Pressure')
acf_pacf(gt_ts.CO, title1='ACF for Carbon Monoxide',
         title2='PACF for Carbon Monoxide')
acf_pacf(gt_ts.NOX, title1='ACF for Nitrogen Oxides',
         title2='PACF for Nitrogen Oxides')
```

```
# Lag plot with lag=1
def lag_plt(df, value, title):
  pd.plotting.lag_plot(df[value], lag=1, s=1, c='firebrick')
  plt.title(title)
  plt.show()

lag_plt(gt_ts, value='AT', title='Lag Plot for Ambient Temperature')
lag_plt(gt_ts, value='AP', title='Lag Plot for Ambient Pressure')
lag_plt(gt_ts, value='AH', title='Lag Plot for Ambient Humidity')
lag_plt(gt_ts, value='AFDP',
        title='Lag Plot for Air Filter Difference Pressure')
lag_plt(gt_ts, value='GTEP', title='Lag Plot for Gas Turbine Exhaust Pressure')
lag_plt(gt_ts, value='TIT', title='Lag Plot for Turbine Inlet Temperature')
lag_plt(gt_ts, value='TAT', title='Lag Plot for Turbine After Temperature')
lag_plt(gt_ts, value='TEY', title='Lag Plot for Turbine Energy Yield')
lag_plt(gt_ts, value='CDP', title='Lag Plot for Compressor Discharge Pressure')
lag_plt(gt_ts, value='CO', title='Lag Plot for Carbon Monoxide')
lag_plt(gt_ts, value='NOX', title='Lag Plot for Nitrogen Oxides')
```

```
# Forecastability - Entropy
def forecas_entr(df, value):
  std_dev = np.std(df[value])
  # Calculate entropy as half the natural logarithm (base e) of the variance
  entropy = 0.5 * np.log(2 * np.pi * np.e * std_dev**2)
  return entropy

entropy_value = forecas_entr(gt_ts, value='AT')
print(f"Forecastability Entropy for AT: {entropy_value}")
entropy_value = forecas_entr(gt_ts, value='AP')
print(f"Forecastability Entropy for AP: {entropy_value}")
entropy_value = forecas_entr(gt_ts, value='AH')
print(f"Forecastability Entropy for AH: {entropy_value}")
entropy_value = forecas_entr(gt_ts, value='AFDP')
print(f"Forecastability Entropy for AFDP: {entropy_value}")
entropy_value = forecas_entr(gt_ts, value='GTEP')
print(f"Forecastability Entropy for GTEP: {entropy_value}")
entropy_value = forecas_entr(gt_ts, value='TIT')
print(f"Forecastability Entropy for TIT: {entropy_value}")
```

```
entropy_value = forecas_entr(gt_ts, value='TAT')
print(f"Forecastability Entropy for TAT: {entropy_value}")
entropy_value = forecas_entr(gt_ts, value='TEY')
print(f"Forecastability Entropy for TEY: {entropy_value}")
entropy_value = forecas_entr(gt_ts, value='CDP')
print(f"Forecastability Entropy for CDP: {entropy_value}")
entropy_value = forecas_entr(gt_ts, value='CO')
print(f"Forecastability Entropy for CO: {entropy_value}")
entropy_value = forecas_entr(gt_ts, value='NOX')
print(f"Forecastability Entropy for NOX: {entropy_value}")
```

```
# Metrics for the different Datasets
gt_2011.describe()
```

```
gt_2012.describe()
```

```
gt_2013.describe()
```

```
gt_2014.describe()
```

```
gt_2015.describe()
```

```
gt_ts.describe()
```

```
# Boxplot of Month-wise (Seasonal) and Year-wise (trend) Distribution for all
#                                                                       years
def box_seas(df, value, xlabel1, xlabel2, ylabel):
  df['year'] = [d.year for d in df.index]
  df['month'] = [d.strftime('%b') for d in df.index]
  years = df['year'].unique()
  fig, axes = plt.subplots(1, 2, figsize=(20,7), dpi= 80)
  sns.boxplot(x='year', y='AT', data=df, ax=axes[0])
  sns.boxplot(x='month', y=value, data=df.loc[~df.year.isin([2010, 2016]), :])
  axes[0].set_xlabel(xlabel1)
  axes[0].set_ylabel(ylabel)
  axes[1].set_xlabel(xlabel2)
  axes[1].set_ylabel(ylabel)
  plt.show()
  df.drop(['year'], axis=1, inplace=True)
  df.drop(['month'], axis=1, inplace=True)
  return

box_seas(gt_ts, value='AT', xlabel1='Year', xlabel2='Months',
         ylabel='Ambient Temperature [ C ]')
box_seas(gt_ts, value='AP', xlabel1='Year', xlabel2='Months',
         ylabel='Ambient Pressure [mbar]')
box_seas(gt_ts, value='AH', xlabel1='Year', xlabel2='Months',
         ylabel='Ambient Humidity [%]')
box_seas(gt_ts, value='AFDP', xlabel1='Year', xlabel2='Months',
         ylabel='Air Filter Difference Pressure [mbar]')
box_seas(gt_ts, value='GTEP', xlabel1='Year', xlabel2='Months',
         ylabel='Gas Turbine Exhaust Pressure [mbar]')
box_seas(gt_ts, value='TIT', xlabel1='Year', xlabel2='Months',
         ylabel='Turbine Inlet Temperature [ C ]')
box_seas(gt_ts, value='TAT', xlabel1='Year', xlabel2='Months',
         ylabel='Turbine After Temperature [ C ]')
box_seas(gt_ts, value='TEY', xlabel1='Year', xlabel2='Months',
         ylabel='Turbine Energy Yield [MWH]')
box_seas(gt_ts, value='CDP', xlabel1='Year', xlabel2='Months',
         ylabel='Compressor Discharge Pressure [mbar]')
box_seas(gt_ts, value='CO', xlabel1='Year', xlabel2='Months',
         ylabel='Carbon Monoxide [$mg/m^3$]')
box_seas(gt_ts, value='NOX', xlabel1='Year', xlabel2='Months',
         ylabel='Nitrogen Oxides [$mg/m^3$]')
```

```
# Relative change
def per_change(df, value, xlabel, ylabel):
  plt.figure(figsize=(15,6))
  df[value].div(df[value].shift()).plot(color='r')
```

```python
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

per_change(gt_ts, value='AT', xlabel='Years',
           ylabel='Relative Change in Ambient Temperature [%]')
per_change(gt_ts, value='AP', xlabel='Years',
           ylabel='Relative Change in Ambient Pressure [%]')
per_change(gt_ts, value='AH', xlabel='Years',
           ylabel='Relative Change in Ambient Humidity [%]')
per_change(gt_ts, value='AFDP', xlabel='Years',
           ylabel='Relative Change in Air Filter Difference Pressure [%]')
per_change(gt_ts, value='GTEP', xlabel='Years',
           ylabel='Relative Change in Gas Turbine Exhaust Pressure [%]')
per_change(gt_ts, value='TIT', xlabel='Years',
           ylabel='Relative Change in Turbine Inlet Temperature [%]')
per_change(gt_ts, value='TAT', xlabel='Years',
           ylabel='Relative Change in Turbine After Temperature [%]')
per_change(gt_ts, value='TEY', xlabel='Years',
           ylabel='Relative Change in Turbine Energy Yield [%]')
per_change(gt_ts, value='CDP', xlabel='Years',
           ylabel='Relative Change in Compressor Discharge Pressure [%]')
per_change(gt_ts, value='CO', xlabel='Years',
           ylabel='Relative Change in Carbon Monoxide [%]')
per_change(gt_ts, value='NOX', xlabel='Years',
           ylabel='Relative Change in Nitrogen Oxides [%]')
```

```python
# Absolute change in successive rows
def abs_change(df, value, xlabel, ylabel):
  plt.figure(figsize=(15,7))
  df[value].diff().plot(color='g')
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.show()

abs_change(gt_ts, value='AT', xlabel='Years',
           ylabel='Absolute Change in Ambient Temperature [ C ]')
abs_change(gt_ts, value='AP', xlabel='Years',
           ylabel='Absolute Change in Ambient Pressure [mbar]')
abs_change(gt_ts, value='AH', xlabel='Years',
           ylabel='Absolute Change in Ambient Humidity [%]')
abs_change(gt_ts, value='AFDP', xlabel='Years',
           ylabel='Absolute Change in Air Filter Difference Pressure [mbar]')
abs_change(gt_ts, value='GTEP', xlabel='Years',
           ylabel='Absolute Change in Gas Turbine Exhaust Pressure [mbar]')
abs_change(gt_ts, value='TIT', xlabel='Years',
           ylabel='Absolute Change in Turbine Inlet Temperature [ C ]')
abs_change(gt_ts, value='TAT', xlabel='Years',
           ylabel='Absolute Change in Turbine After Temperature [ C ]')
abs_change(gt_ts, value='TEY', xlabel='Years',
           ylabel='Absolute Change in Turbine Energy Yield [MWH]')
abs_change(gt_ts, value='CDP', xlabel='Years',
           ylabel='Absolute Change in Compressor Discharge Pressure [mbar]')
abs_change(gt_ts, value='CO', xlabel='Years',
           ylabel='Absolute Change in Carbon Monoxide [$mg/m^3$]')
abs_change(gt_ts, value='NOX', xlabel='Years',
           ylabel='Absolute Change in Nitrogen Oxides [$mg/m^3$]')
```

```python
# Ambient variables - P/T = cte
constant = gt_ts['AP']/gt_ts['AT']
const = constant.to_frame(name='const')

plt.figure(figsize=(16,5))
plt.plot(const.index, const, color='r')
plt.gca().set(xlabel='Years', ylabel='P/T [mbar/ C ]')
plt.show()
```

```python
# Outliers treatment - Interquartile range method
columns = ['AT', 'AP', 'AH', 'AFDP', 'GTEP', 'TIT', 'TAT', 'TEY', 'CDP', 'CO',
           'NOX']

Q1 = gt_ts[columns].quantile(0.25)
```

```python
Q3 = gt_ts[columns].quantile(0.75)
IQR = Q3 - Q1

lim_inf = Q1 - 1.5 * IQR
lim_sup = Q3 + 1.5 * IQR

outliers = (gt_ts[columns] < lim_inf) | (gt_ts[columns] > lim_sup)
gt_ts_copy = gt_ts.copy()

for col in columns:
    col_mean = gt_ts_copy.loc[~outliers[col], col].mean()
    gt_ts_copy.loc[outliers[col], col] = col_mean

gt_ts_wo_outliers = gt_ts_copy.copy()
data_copy = gt_ts_wo_outliers.copy()

data_copy.info()
```

```python
# Boxplot of Month-wise (Seasonal) and Year-wise (trend) Distribution for all
#                             years without outliers
def box_seas(df, value, xlabel1, xlabel2, ylabel):
  df['year'] = [d.year for d in df.index]
  df['month'] = [d.strftime('%b') for d in df.index]
  years = df['year'].unique()
  fig, axes = plt.subplots(1, 2, figsize=(20,7), dpi= 80)
  sns.boxplot(x='year', y='AT', data=df, ax=axes[0])
  sns.boxplot(x='month', y=value, data=df.loc[~df.year.isin([2010, 2016]), :])
  axes[0].set_xlabel(xlabel1)
  axes[0].set_ylabel(ylabel)
  axes[1].set_xlabel(xlabel2)
  axes[1].set_ylabel(ylabel)
  plt.show()
  df.drop(['year'], axis=1, inplace=True)
  df.drop(['month'], axis=1, inplace=True)
  return

box_seas(gt_ts_wo_outliers, value='AT', xlabel1='Year', xlabel2='Months',
         ylabel='Ambient Temperature [ C ]')
box_seas(gt_ts_wo_outliers, value='AP', xlabel1='Year', xlabel2='Months',
         ylabel='Ambient Pressure [mbar]')
box_seas(gt_ts_wo_outliers, value='AH', xlabel1='Year', xlabel2='Months',
         ylabel='Ambient Humidity [%]')
box_seas(gt_ts_wo_outliers, value='AFDP', xlabel1='Year', xlabel2='Months',
         ylabel='Air Filter Difference Pressure [mbar]')
box_seas(gt_ts_wo_outliers, value='GTEP', xlabel1='Year', xlabel2='Months',
         ylabel='Gas Turbine Exhaust Pressure [mbar]')
box_seas(gt_ts_wo_outliers, value='TIT', xlabel1='Year', xlabel2='Months',
         ylabel='Turbine Inlet Temperature [ C ]')
box_seas(gt_ts_wo_outliers, value='TAT', xlabel1='Year', xlabel2='Months',
         ylabel='Turbine After Temperature [ C ]')
box_seas(gt_ts_wo_outliers, value='TEY', xlabel1='Year', xlabel2='Months',
         ylabel='Turbine Energy Yield [MWH]')
box_seas(gt_ts_wo_outliers, value='CDP', xlabel1='Year', xlabel2='Months',
         ylabel='Compressor Discharge Pressure [mbar]')
box_seas(gt_ts_wo_outliers, value='CO', xlabel1='Year', xlabel2='Months',
         ylabel='Carbon Monoxide [$mg/m^3$]')
box_seas(gt_ts_wo_outliers, value='NOX', xlabel1='Year', xlabel2='Months',
         ylabel='Nitrogen Oxides [$mg/m^3$]')
```

```python
gt_ts_wo_outliers.describe()
```

```python
# Butterworth filtering
def butterworth_low(data, cutout_freq, order, samp_rate):
  nyquist_freq = samp_rate / 2.0
  normalized_cutoff = cutout_freq / nyquist_freq
  b, a = butter(order, normalized_cutoff, btype='low', analog=False)
  return filtfilt(b, a, data)

order = 3
cutout_freq = 0.1
```

```python
samp_rate = 100

for column in data_copy.columns:
  gt_ts_wo_outliers[column] = butterworth_low(data_copy[column], cutout_freq,
                                               order, samp_rate)

def butterworth_plot(value, xlabel, ylabel):
  plt.figure(figsize=(16, 4))
  plt.plot(data_copy.index, data_copy[value], label='Real values', marker='o',
           linestyle='', markersize=0.4, color='b')
  plt.plot(data_copy.index, gt_ts_wo_outliers[value], label='Filtered values',
           color='r')
  plt.gca().set(xlabel=xlabel, ylabel=ylabel)
  plt.legend()
  plt.show()

butterworth_plot(value='AT', xlabel='Years', ylabel='Ambient Temperature [ C ]')
butterworth_plot(value='AP', xlabel='Years', ylabel='Ambient Pressure [mbar]')
butterworth_plot(value='AH', xlabel='Years', ylabel='Ambient Humidity [%]')
butterworth_plot(value='AFDP', xlabel='Years',
                 ylabel='Air Filter Difference Pressure [mbar]')
butterworth_plot(value='GTEP', xlabel='Years',
                 ylabel='Gas Turbine Exhaust Pressure [mbar]')
butterworth_plot(value='TIT', xlabel='Years',
                 ylabel='Turbine Inlet Temperature [ C ]')
butterworth_plot(value='TAT', xlabel='Years',
                 ylabel='Turbine After Temperature [ C ]')
butterworth_plot(value='TEY', xlabel='Years',
                 ylabel='Turbine Energy Yield [MWH]')
butterworth_plot(value='CDP', xlabel='Years',
                 ylabel='Compressor Discharge Pressure [mbar]')
butterworth_plot(value='CO', xlabel='Years',
                 ylabel='Carbon Monoxide [$mg/m^3$]')
butterworth_plot(value='NOX', xlabel='Years',
                 ylabel='Nitrogen Oxides [$mg/m^3$]')
```

```python
# Statistically significant variables - Canonical Correlation Analysis
X = gt_ts_wo_outliers.iloc[:, :9]

cca = CCA(n_components=1)

def sig_var(X,y):
  cca.fit(X, y)

  correlation_coeffs = cca.x_weights_
  feature_ranking = np.argsort(np.abs(correlation_coeffs.ravel()))[::-1]
  cumulative_sum = np.cumsum(np.abs(correlation_coeffs[feature_ranking]))
  cumulative_percentage = cumulative_sum / np.sum(np.abs(correlation_coeffs))

  top_features = np.where(cumulative_percentage <= 0.95)[0][-1] + 1
  selected_features = X.iloc[:, feature_ranking[:top_features]]

  column_names = X.columns[feature_ranking]

  plt.figure(figsize=(6, 4))
  plt.bar(column_names, np.abs(correlation_coeffs.ravel()))
  plt.ylabel('Normalized weights')
  plt.show()

  print(f"Number of features selected to represent the 95% of the cumulated"
                       f" importance: {top_features}")
  print("Selected features:")
  print(selected_features.columns.tolist())

sig_var(X, y = gt_ts_wo_outliers.iloc[:, 9])
sig_var(X, y = gt_ts_wo_outliers.iloc[:, 10])
```

```python
# Division in train, validation and test
gt_train = gt_ts_wo_outliers[(gt_ts_wo_outliers.index.year == 2011) |
            (gt_ts_wo_outliers.index.year == 2012)] # Train 2011 and 2012
gt_val = gt_ts_wo_outliers[(gt_ts_wo_outliers.index.year == 2013)]
```

```
                                                        # Validation 2013
gt_test = gt_ts_wo_outliers[(gt_ts_wo_outliers.index.year == 2014) |
            (gt_ts_wo_outliers.index.year == 2015)] # Test 2014 and 2015

print("Number of training samples:", len(gt_train))
print("Number of validation samples:", len(gt_val))
print("Number of test samples:", len(gt_test))
```

```
# Data standarization - Normalization
train_mean = gt_train.mean()
train_std = gt_train.std()

train_df = (gt_train - train_mean) / train_std
val_df = (gt_val - train_mean) / train_std
test_df = (gt_test - train_mean) / train_std

df_data = pd.concat([train_df, val_df, test_df])
df_data.info()
```

```
# Feature selection
df_data_CO = df_data[['AT', 'AH', 'GTEP', 'TIT', 'TEY', 'CDP', 'CO']]
df_data_NOx = df_data[['AT', 'AH', 'TIT', 'TEY', 'CDP', 'NOX']]
```

```
df_data_CO.head()
```

```
df_data_NOx.head()
```

```
# Data parsing for CO
df_reset_CO = df_data_CO.reset_index().rename(columns={'index': 'Date Time'})
df_num_CO = df_reset_CO.drop('Date Time', axis=1)

float_data_CO = np.array(df_num_CO)
float_data_CO
```

```
# Data generator - Tensorize data
def generator(data, lookback, delay, min_index, max_index,
              shuffle, batch_size, step):
  if max_index is None:
    max_index = len(data) - delay - 1
  i = min_index + lookback
  while 1:
    if shuffle:
      rows = np.random.randint(min_index + lookback, max_index, size=batch_size)
    else:
      if i + batch_size >= max_index:
        i = min_index + lookback
      rows = np.arange(i, min(i + batch_size, max_index))
      i += len(rows)
    samples = np.zeros((len(rows), lookback // step, data.shape[-1]))
    targets = np.zeros((len(rows),))
    for j, row in enumerate(rows):
      indices = range(rows[j] - lookback, rows[j], step)
      samples[j] = data[indices]
      targets[j] = data[rows[j] + delay][1]
    yield samples, targets
```

```
# Training, validation and test generators for CO emissions
lookback = 240 # our observations will go back 10 days
step = 12 # our observations will be sampled at one point per 12 hours
delay = 24 # our targets will be 24 hours in the future
batch_size = 128

train_min_index = 0
train_max_index = len(gt_train) - 1 # 2011 and 2012

val_min_index = len(gt_train)
val_max_index = len(gt_train) + len(gt_val) - 1 # 2013

test_min_index = len(gt_train) + len(gt_val)
```

```
test_max_index = len(gt_train) + len(gt_val) + len(gt_test) - 1 # 2014 and 2015

train_gen_CO = generator(float_data_CO,
  lookback=lookback,
  delay=delay,
  min_index=train_min_index,
  max_index=train_max_index, # 2011 and 2012
  shuffle=False,
  step=step,
  batch_size=batch_size)

val_gen_CO = generator(float_data_CO,
  lookback=lookback,
  delay=delay,
  min_index=val_min_index,
  max_index=val_max_index, # 2013
  shuffle=False,
  step=step,
  batch_size=batch_size)

test_gen_CO = generator(float_data_CO,
  lookback=lookback,
  delay=delay,
  min_index=test_min_index,
  max_index=test_max_index,
  shuffle=False,
  step=step,
  batch_size=batch_size)

val_steps_CO = (val_max_index - val_min_index - lookback) // batch_size
print("This is how many steps to draw from ‘val_gen‘ in order to see the whole"
              " validation set:", val_steps_CO)

test_steps_CO = (test_max_index - test_min_index - lookback) // batch_size
print("This is how many steps to draw from ‘test_gen‘ in order to see the whole"
              " test set:", test_steps_CO)
```

```
# GRU model for CO - Hyperparameter optimization
def build_model_GRU_CO(units, learning_rate):
    model = Sequential()
    model.add(GRU(units,
                  kernel_regularizer=l2(0.1),
                  recurrent_regularizer=l2(0.001),
                  dropout=0.3,
                  recurrent_dropout=0.5,
                  return_sequences=True,
                  input_shape=(None, float_data_CO.shape[-1])))
    model.add(Dense(1))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='mae')
    return model

early_stopping = EarlyStopping(monitor='val_loss', patience=4,
                               restore_best_weights=True)

units_values = [32, 64]
learning_rates = [0.00005, 0.0001]

best_loss = float('inf')
best_params = {}

for units in units_values:
  for lr in learning_rates:
      print(f'Training for units={units}, learning_rate={lr}')

      model = build_model_GRU_CO(units=units, learning_rate=lr)

      history = model.fit_generator(train_gen_CO,
                                    steps_per_epoch=500,
                                    epochs=60,
                                    validation_data=val_gen_CO,
                                    validation_steps=val_steps_CO,
                                    callbacks=[early_stopping],
                                    verbose=0)

      val_loss = history.history['val_loss'][-1]
      print(f'Validation Loss: {val_loss}')

      if val_loss < best_loss:
          best_loss = val_loss
          best_params['units'] = units
          best_params['learning_rate'] = lr
          best_model = model

print("Best Parameters:")
print(best_params)
print("Best Validation Loss:", best_loss)
```

```
# Results for test set and GRU model plot for CO
best_model_GRU_CO = build_model_GRU_CO(units=best_params['units'],
                                    learning_rate=best_params['learning_rate'])

history_final_GRU_CO = best_model_GRU_CO.fit_generator(train_gen_CO,
                                                steps_per_epoch=500,
                                                epochs=60,
                                                validation_data=val_gen_CO,
                                                validation_steps=val_steps_CO,
                                                callbacks=[early_stopping])

# Overfitting check

loss_GRU_CO = history_final_GRU_CO.history['loss']
val_loss_GRU_CO = history_final_GRU_CO.history['val_loss']

epochs = range(1, len(loss_GRU_CO) + 1)

plt.figure()

plt.plot(epochs, loss_GRU_CO, 'bo', label='Training loss - MAE')
plt.plot(epochs, val_loss_GRU_CO, 'b', label='Validation loss - MAE')
plt.xlabel('Epochs')
plt.legend()

plt.show()

# Test results

test_mae = best_model_GRU_CO.evaluate_generator(test_gen_CO,
                                                steps=test_steps_CO)
print("MAE on test set:", test_mae)

# Model plot

plot_model(best_model_GRU_CO, show_shapes=True)
```

```
# Predictions for GRU model for CO
predictions_GRU_CO = best_model_GRU_CO.predict_generator(test_gen_CO,
                                                         steps=test_steps_CO)
```

```
# Denormalize results for GRU model for CO
predictions_denormalized_GRU_CO = (predictions_GRU_CO[:,-1,:] *
                                   train_std[9]) + train_mean[9]

real_values = []
for i in range(test_steps_CO):
  batch_x, batch_y = next(test_gen_CO)
  real_values.extend(batch_y)

real_values_denorm_GRU_CO = (np.array(real_values) *
                             train_std[9]) + train_mean[9]
```

```
# Graph of results for GRU model for CO
```

```
last_2_GRU_CO = real_values_denorm_GRU_CO[16896:]
rest_2_GRU_CO = real_values_denorm_GRU_CO[:16896]
real_values_denormalized_GRU_CO = np.concatenate((last_2_GRU_CO, rest_2_GRU_CO))

plt.figure()
plt.plot(gt_test.index[0:17152], real_values_denormalized_GRU_CO,
         label='Target', color='m')
plt.plot(gt_test.index[0:17152], predictions_denormalized_GRU_CO,
         label='Prediction', color='g')
plt.xlabel('Years')
plt.ylabel('Carbon Monoxide [$mg/m^3$]')
plt.gca().xaxis.set_major_locator(plt.matplotlib.dates.YearLocator())
plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))
plt.legend()
plt.show()
```

```
# 1D CNN model for CO - Hyperparameter optimization
def build_model_CNN_CO(filters, pool_size, learning_rate):
    model = Sequential()
    model.add(layers.Conv1D(filters, 5, activation='tanh',
                            kernel_regularizer=l2(0.1),
                            input_shape=(None, float_data_CO.shape[-1])))
    model.add(layers.MaxPooling1D(pool_size))
    model.add(layers.Dropout(rate=0.3))
    model.add(layers.Conv1D(filters, 5, activation='tanh',
                            kernel_regularizer=l2(0.1)))
    model.add(layers.GlobalMaxPooling1D())
    model.add(layers.Dropout(rate=0.3))
    model.add(layers.Dense(1))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='mae')
    return model

early_stopping = EarlyStopping(monitor='val_loss', patience=4,
                               restore_best_weights=True)

filters_values = [32, 64]
pool_size_values = [2, 3]
learning_rates = [0.00005, 0.0001]

best_loss = float('inf')
best_params = {}

for filters in filters_values:
  for pool_size in pool_size_values:
    for lr in learning_rates:
        print(f'Training for filters={filters}, pool_size={pool_size},'
                            f' learning_rate={lr}')

        model = build_model_CNN_CO(filters=filters, pool_size=pool_size,
                                   learning_rate=lr)

        history = model.fit_generator(train_gen_CO,
                                      steps_per_epoch=500,
                                      epochs=60,
                                      validation_data=val_gen_CO,
                                      validation_steps=val_steps_CO,
                                      callbacks=[early_stopping],
                                      verbose=0)

        val_loss = history.history['val_loss'][-1]
        print(f'Validation Loss: {val_loss}')

        if val_loss < best_loss:
            best_loss = val_loss
            best_params['filters'] = filters
            best_params['pool_size'] = pool_size
            best_params['learning_rate'] = lr
            best_model = model

print("Best Parameters:")
print(best_params)
```

```python
print("Best Validation Loss:", best_loss)
```

```python
# Results for test set and 1D CNN model plot for CO
best_model_CNN_CO = build_model_CNN_CO(filters=best_params['filters'],
                                       pool_size=best_params['pool_size'],
                                    learning_rate=best_params['learning_rate'])

history_final_CNN_CO = best_model_CNN_CO.fit_generator(train_gen_CO,
                                                  steps_per_epoch=500,
                                                  epochs=60,
                                                  validation_data=val_gen_CO,
                                                  validation_steps=val_steps_CO,
                                                  callbacks=[early_stopping])

# Overfitting check

loss_CNN_CO = history_final_CNN_CO.history['loss']
val_loss_CNN_CO = history_final_CNN_CO.history['val_loss']

epochs = range(1, len(loss_CNN_CO) + 1)

plt.figure()

plt.plot(epochs, loss_CNN_CO, 'bo', label='Training loss - MAE')
plt.plot(epochs, val_loss_CNN_CO, 'b', label='Validation loss - MAE')
plt.xlabel('Epochs')
plt.legend()

plt.show()

# Test results

test_mae = best_model_CNN_CO.evaluate_generator(test_gen_CO,
                                                steps=test_steps_CO)
print("MAE on test set:", test_mae)

plot_model(best_model_CNN_CO, show_shapes=True)
```

```python
# Predictions for 1D CNN model for CO
predictions_CNN_CO = best_model_CNN_CO.predict_generator(test_gen_CO,
                                                         steps=test_steps_CO)
```

```python
# Denormalize results for 1D CNN model for CO
predictions_denorm_CNN_CO = (predictions_CNN_CO *
                             train_std[9]) + train_mean[9]

real_values = []
for i in range(test_steps_CO):
  batch_x, batch_y = next(test_gen_CO)
  real_values.extend(batch_y)

real_values_denorm_CNN_CO = (np.array(real_values) *
                             train_std[9]) + train_mean[9]
```

```python
# Graph of results for 1D CNN model for CO
last_1_CNN_CO = predictions_denorm_CNN_CO[16768:]
rest_1_CNN_CO = predictions_denorm_CNN_CO[:16768]
predictions_denormalized_CNN_CO = np.concatenate((last_1_CNN_CO, rest_1_CNN_CO))

last_2_CNN_CO = real_values_denorm_CNN_CO[15616:]
rest_2_CNN_CO = real_values_denorm_CNN_CO[:15616]
real_values_denormalized_CNN_CO = np.concatenate((last_2_CNN_CO, rest_2_CNN_CO))

plt.figure()
plt.plot(gt_test.index[0:17152], real_values_denormalized_CNN_CO,
         label='Target', color='m')
plt.plot(gt_test.index[0:17152], predictions_denormalized_CNN_CO,
         label='Prediction', color='g')
plt.xlabel('Years')
plt.ylabel('Carbon Monoxide [$mg/m^3$]')
plt.gca().xaxis.set_major_locator(plt.matplotlib.dates.YearLocator())
plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))
plt.legend()
plt.show()
```

```python
# Comparison of models for CO
plt.figure()
plt.plot(gt_test.index[0:17152], real_values_denormalized_GRU_CO,
         label='Target', color='b')
plt.plot(gt_test.index[0:17152], predictions_denormalized_GRU_CO,
         label='Prediction GRU', color='g')
plt.plot(gt_test.index[0:17152], predictions_denormalized_CNN_CO,
         label='Prediction 1D CNN', color='r')
plt.xlabel('Years')
plt.ylabel('Carbon Monoxide [$mg/m^3$]')
plt.gca().xaxis.set_major_locator(plt.matplotlib.dates.YearLocator())
plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))
plt.legend()
plt.show()
```

```python
# Data parsing for NOx
df_reset_NOx = df_data_NOx.reset_index().rename(columns={'index': 'Date Time'})
df_num_NOx = df_reset_NOx.drop('Date Time', axis=1)

float_data_NOx = np.array(df_num_NOx)
float_data_NOx
```

```python
# Training, validation and test generators for NOx
lookback = 240 # our observations will go back 10 days
step = 12 # our observations will be sampled at one point per 12 hour
delay = 24 # our targets will be 24 hours in the future
batch_size = 128

train_min_index = 0
train_max_index = len(gt_train) - 1 # 2011 and 2012

val_min_index = len(gt_train)
val_max_index = len(gt_train) + len(gt_val) - 1 # 2013

test_min_index = len(gt_train) + len(gt_val)
test_max_index = len(gt_train) + len(gt_val) + len(gt_test) - 1 # 2014 and 2015

train_gen_NOx = generator(float_data_NOx,
  lookback=lookback,
  delay=delay,
  min_index=train_min_index,
  max_index=train_max_index, # 2011 and 2012
  shuffle=False,
  step=step,
  batch_size=batch_size)

val_gen_NOx = generator(float_data_NOx,
  lookback=lookback,
  delay=delay,
  min_index=val_min_index,
  max_index=val_max_index, # 2013
  shuffle=False,
  step=step,
  batch_size=batch_size)

test_gen_NOx = generator(float_data_NOx,
  lookback=lookback,
  delay=delay,
  min_index=test_min_index,
  max_index=test_max_index,
  shuffle=False,
  step=step,
  batch_size=batch_size)

val_steps_NOx = (val_max_index - val_min_index - lookback) // batch_size
```

```
print("This is how many steps to draw from `val_gen` in order to see the whole"
                            " validation set:", val_steps_NOx)

test_steps_NOx = (test_max_index - test_min_index - lookback) // batch_size
print("This is how many steps to draw from `test_gen` in order to see the whole"
                            " test set:", test_steps_NOx)
```

```
# GRU model for NOx - Hyperparameter optimization
def build_model_GRU_NOx(units, learning_rate):
    model = Sequential()
    model.add(GRU(units,
                  kernel_regularizer=l2(0.1),
                  recurrent_regularizer=l2(0.001),
                  dropout=0.3,
                  recurrent_dropout=0.5,
                  return_sequences=True,
                  input_shape=(None, float_data_NOx.shape[-1])))
    model.add(Dense(1))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='mae')
    return model

early_stopping = EarlyStopping(monitor='val_loss', patience=4,
                               restore_best_weights=True)

units_values = [32, 64]
learning_rates = [0.00005, 0.0001]

best_loss = float('inf')
best_params = {}

for units in units_values:
  for lr in learning_rates:
      print(f'Training for units={units}, learning_rate={lr}')

      model = build_model_GRU_NOx(units=units, learning_rate=lr)

      history = model.fit_generator(train_gen_NOx,
                                    steps_per_epoch=500,
                                    epochs=60,
                                    validation_data=val_gen_NOx,
                                    validation_steps=val_steps_NOx,
                                    callbacks=[early_stopping],
                                    verbose=0)

      val_loss = history.history['val_loss'][-1]
      print(f'Validation Loss: {val_loss}')

      if val_loss < best_loss:
          best_loss = val_loss
          best_params['units'] = units
          best_params['learning_rate'] = lr
          best_model = model

print("Best Parameters:")
print(best_params)
print("Best Validation Loss:", best_loss)
```

```
# Results for test set and GRU model plot for NOx
best_model_GRU_NOx = build_model_GRU_NOx(units=best_params['units'],
                                    learning_rate=best_params['learning_rate'])

history_final_GRU_NOx = best_model_GRU_NOx.fit_generator(train_gen_NOx,
                                                    steps_per_epoch=500,
                                                    epochs=60,
                                                    validation_data=val_gen_NOx,
                                                    validation_steps=val_steps_NOx,
                                                    callbacks=[early_stopping])

# Overfitting check

loss_GRU_NOx = history_final_GRU_NOx.history['loss']
```

```python
val_loss_GRU_NOx = history_final_GRU_NOx.history['val_loss']

epochs = range(1, len(loss_GRU_NOx) + 1)

plt.figure()

plt.plot(epochs, loss_GRU_NOx, 'bo', label='Training loss - MAE')
plt.plot(epochs, val_loss_GRU_NOx, 'b', label='Validation loss - MAE')
plt.xlabel('Epochs')
plt.legend()

plt.show()

# Test results

test_mae = best_model_GRU_NOx.evaluate_generator(test_gen_NOx,
                                                 steps=test_steps_NOx)
print("MAE on test set:", test_mae)

plot_model(best_model_GRU_NOx, show_shapes=True)
```

```python
# Predictions for GRU model for NOx
predictions_GRU_NOx = best_model_GRU_NOx.predict_generator(test_gen_NOx,
                                                           steps=test_steps_NOx)
```

```python
# Denormalize results for GRU model for NOx
predictions_denormalized_GRU_NOx = (predictions_GRU_NOx[:,-1,:] *
                                    train_std[10]) + train_mean[10]

real_values = []
for i in range(test_steps_NOx):
  batch_x, batch_y = next(test_gen_NOx)
  real_values.extend(batch_y)

real_values_denorm_GRU_NOx = (np.array(real_values) *
                              train_std[10]) + train_mean[10]
```

```python
# Graph of results for GRU model for NOx
last_2_GRU_NOx = real_values_denorm_GRU_NOx[16896:]
rest_2_GRU_NOx = real_values_denorm_GRU_NOx[:16896]
real_values_denormalized_GRU_NOx = np.concatenate((last_2_GRU_NOx,
                                                   rest_2_GRU_NOx))

plt.figure()
plt.plot(gt_test.index[0:17152], real_values_denormalized_GRU_NOx,
         label='Target', color='m')
plt.plot(gt_test.index[0:17152], predictions_denormalized_GRU_NOx,
         label='Prediction', color='g')
plt.xlabel('Years')
plt.ylabel('Nitrogen Oxides [$mg/m^3$]')
plt.gca().xaxis.set_major_locator(plt.matplotlib.dates.YearLocator())
plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))
plt.legend()
plt.show()
```

```python
# 1D CNN model for NOx - Hyperparameter optimization
def build_model_CNN_NOx(filters, pool_size, learning_rate):
    model = Sequential()
    model.add(layers.Conv1D(filters, 5, activation='tanh',
                            kernel_regularizer=l2(0.1),
                            input_shape=(None, float_data_NOx.shape[-1])))
    model.add(layers.MaxPooling1D(pool_size))
    model.add(layers.Dropout(rate=0.3))
    model.add(layers.Conv1D(filters, 5, activation='tanh',
                            kernel_regularizer=l2(0.1)))
    model.add(layers.GlobalMaxPooling1D())
    model.add(layers.Dropout(rate=0.3))
    model.add(layers.Dense(1))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='mae')
    return model
```

```

early_stopping = EarlyStopping(monitor='val_loss', patience=4,
                               restore_best_weights=True)

filters_values = [32, 64]
pool_size_values = [2, 3]
learning_rates = [0.00005, 0.0001]

best_loss = float('inf')
best_params = {}

for filters in filters_values:
  for pool_size in pool_size_values:
    for lr in learning_rates:
        print(f'Training for filters={filters}, pool_size={pool_size},'
                                  f' learning_rate={lr}')

        model = build_model_CNN_NOx(filters=filters, pool_size=pool_size,
                                  learning_rate=lr)

        history = model.fit_generator(train_gen_NOx,
                                    steps_per_epoch=500,
                                    epochs=60,
                                    validation_data=val_gen_NOx,
                                    validation_steps=val_steps_NOx,
                                    callbacks=[early_stopping],
                                    verbose=0)

        val_loss = history.history['val_loss'][-1]
        print(f'Validation Loss: {val_loss}')

        if val_loss < best_loss:
            best_loss = val_loss
            best_params['filters'] = filters
            best_params['pool_size'] = pool_size
            best_params['learning_rate'] = lr
            best_model = model

print("Best Parameters:")
print(best_params)
print("Best Validation Loss:", best_loss)
```

```
# Results for test set and 1D CNN model plot for NOx
best_model_CNN_NOx = build_model_CNN_NOx(filters=best_params['filters'],
                                    pool_size=best_params['pool_size'],
                                    learning_rate=best_params['learning_rate'])

history_final_CNN_NOx = best_model_CNN_NOx.fit_generator(train_gen_NOx,
                                                steps_per_epoch=500,
                                                epochs=60,
                                                validation_data=val_gen_NOx,
                                                validation_steps=val_steps_NOx,
                                                callbacks=[early_stopping])

# Overfitting check

loss_CNN_NOx = history_final_CNN_NOx.history['loss']
val_loss_CNN_NOx = history_final_CNN_NOx.history['val_loss']

epochs = range(1, len(loss_CNN_NOx) + 1)

plt.figure()

plt.plot(epochs, loss_CNN_NOx, 'bo', label='Training loss - MAE')
plt.plot(epochs, val_loss_CNN_NOx, 'b', label='Validation loss - MAE')
plt.xlabel('Epochs')
plt.legend()

plt.show()

# Test results
```

```python

test_mae = best_model_CNN_NOx.evaluate_generator(test_gen_NOx,
                                                 steps=test_steps_NOx)
print("MAE on test set:", test_mae)

plot_model(best_model_CNN_NOx, show_shapes=True)
```

```python
# Predictions for 1D CNN model for NOx
predictions_CNN_NOx = best_model_CNN_NOx.predict_generator(test_gen_NOx,
                                                           steps=test_steps_NOx)
```

```python
# Denormalize results for 1D CNN model for NOx
predictions_denorm_CNN_NOx = (predictions_CNN_NOx *
                              train_std[10]) + train_mean[10]

real_values = []
for i in range(test_steps_NOx):
  batch_x, batch_y = next(test_gen_NOx)
  real_values.extend(batch_y)

real_values_denorm_CNN_NOx = (np.array(real_values) *
                               train_std[10]) + train_mean[10]
```

```python
# Graph of results for 1D CNN model for NOx
last_1_CNN_NOx = predictions_denorm_CNN_NOx[16384:]
rest_1_CNN_NOx = predictions_denorm_CNN_NOx[:16384]
predictions_denormalized_CNN_NOx = np.concatenate((last_1_CNN_NOx,
                                                   rest_1_CNN_NOx))

last_2_CNN_NOx = real_values_denorm_CNN_NOx[15232:]
rest_2_CNN_NOx = real_values_denorm_CNN_NOx[:15232]
real_values_denormalized_CNN_NOx = np.concatenate((last_2_CNN_NOx,
                                                   rest_2_CNN_NOx))

plt.figure()
plt.plot(gt_test.index[0:17152], real_values_denormalized_CNN_NOx,
         label='Target', color='m')
plt.plot(gt_test.index[0:17152], predictions_denormalized_CNN_NOx,
         label='Prediction', color='g')
plt.xlabel('Years')
plt.ylabel('Nitrogen Oxides [$mg/m^3$]')
plt.gca().xaxis.set_major_locator(plt.matplotlib.dates.YearLocator())
plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))
plt.legend()
plt.show()
```

```python
# Comparison of models for NOx
plt.figure()
plt.plot(gt_test.index[0:17152], real_values_denormalized_GRU_NOx,
         label='Target', color='b')
plt.plot(gt_test.index[0:17152], predictions_denormalized_GRU_NOx,
         label='Prediction GRU', color='g')
plt.plot(gt_test.index[0:17152], predictions_denormalized_CNN_NOx,
         label='Prediction 1D CNN', color='r')
plt.xlabel('Years')
plt.ylabel('Nitrogen Oxides [$mg/m^3$]')
plt.gca().xaxis.set_major_locator(plt.matplotlib.dates.YearLocator())
plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))
plt.legend()
plt.show()
```

II. Additional figures and tables

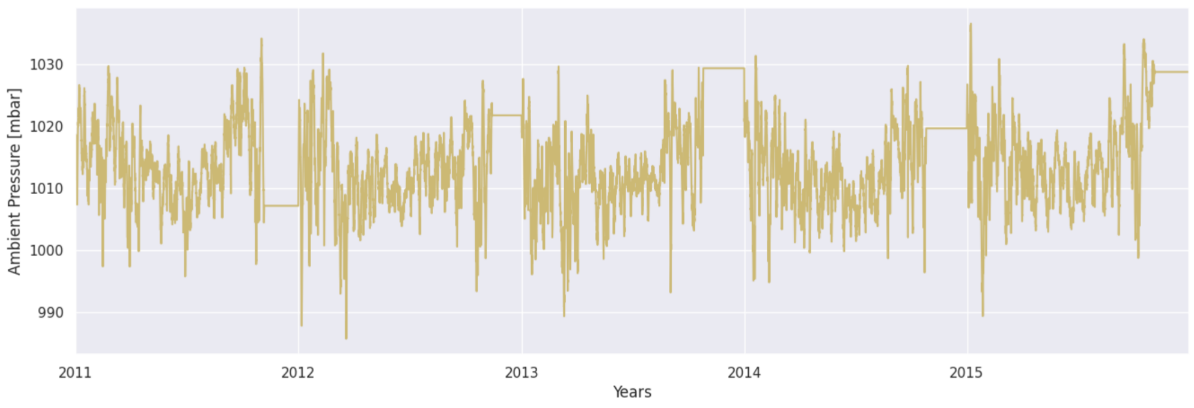


Figure 1: Ambient pressure graph obtained with forward fill method for filling missing values

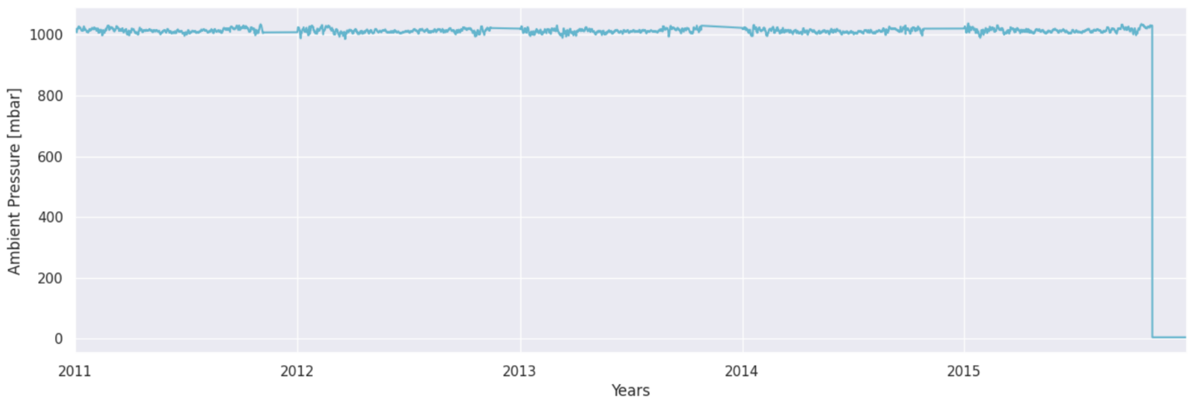


Figure 2: Ambient pressure graph obtained with forward linear interpolation for filling missing values

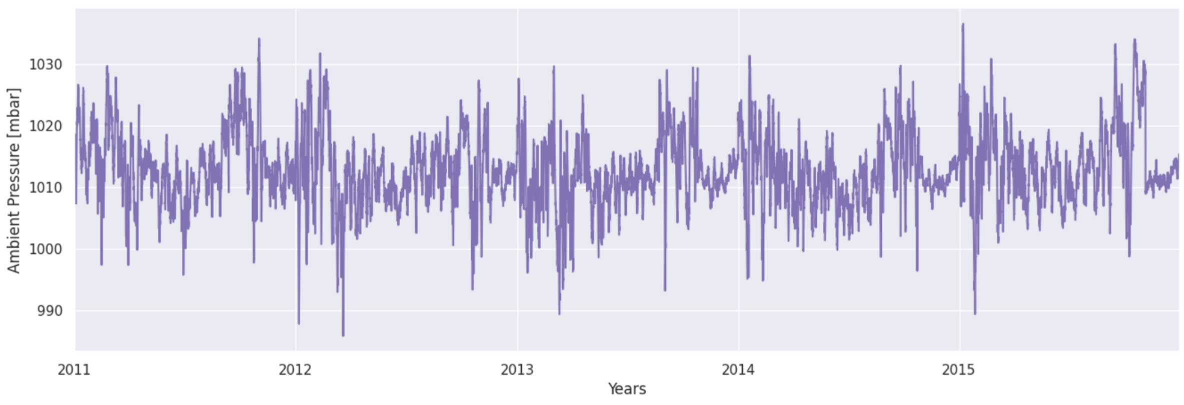


Figure 3: Ambient pressure graph obtained with seasonal mean (n=2920) method for filling missing values

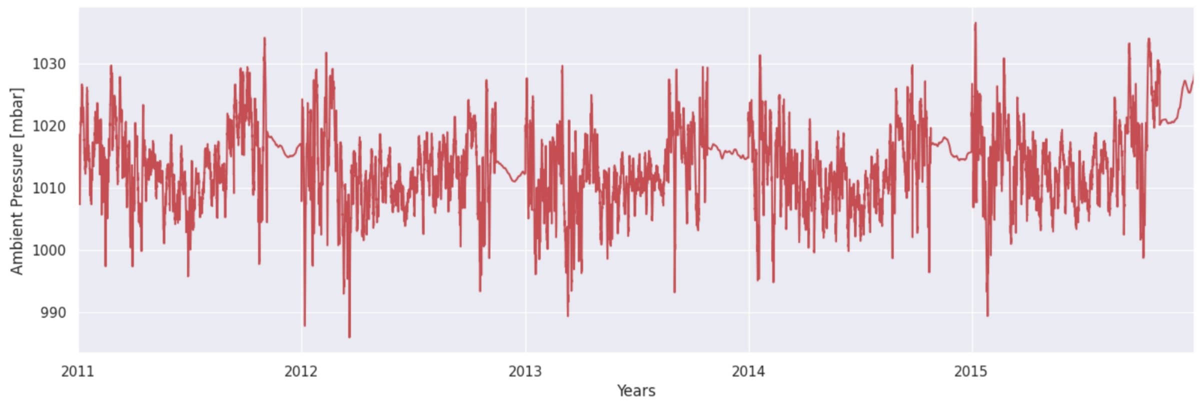


Figure 4: Ambient pressure graph obtained with KNN mean (n=2920) method for filling missing values

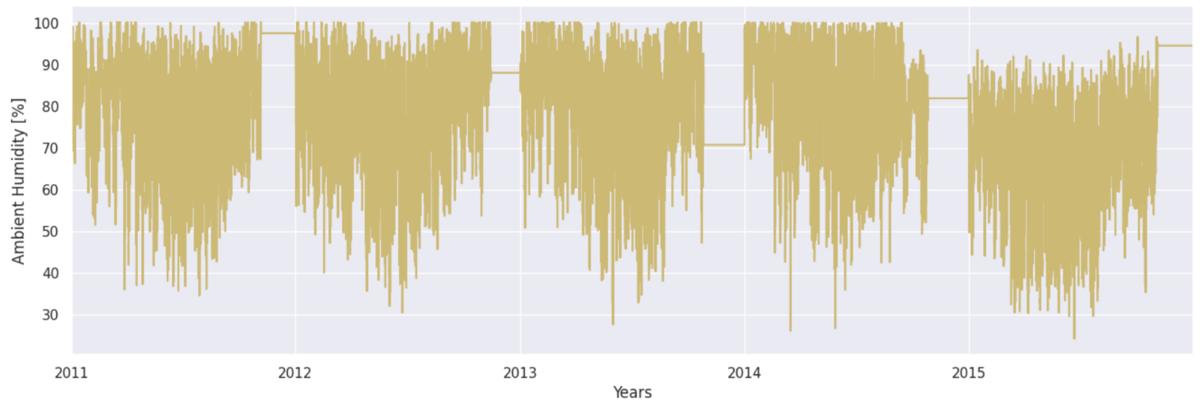


Figure 5: Ambient humidity graph obtained with forward fill method for filling missing values

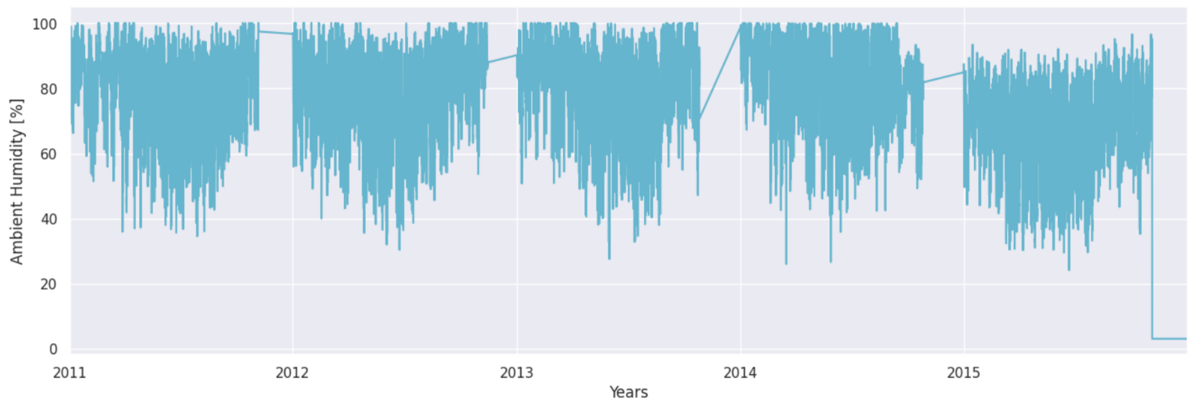


Figure 6: Ambient humidity graph obtained with forward linear interpolation for filling missing values

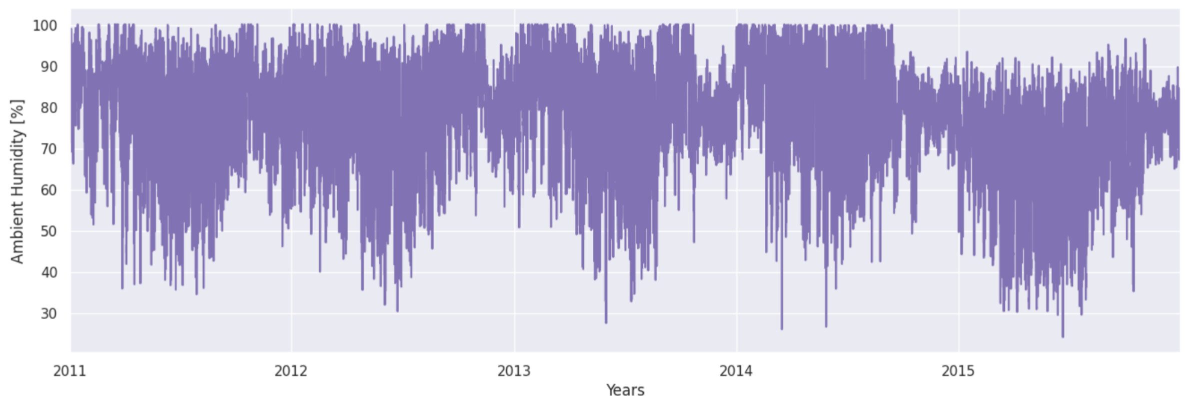


Figure 7: Ambient humidity graph obtained with seasonal mean (n=2920) method for filling missing values

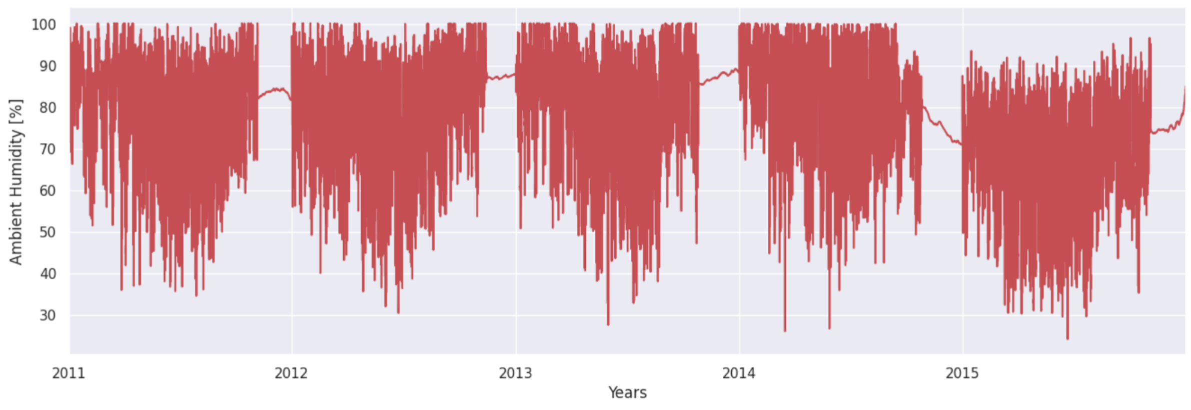


Figure 8: Ambient humidity graph obtained with KNN mean (n=2920) method for filling missing values

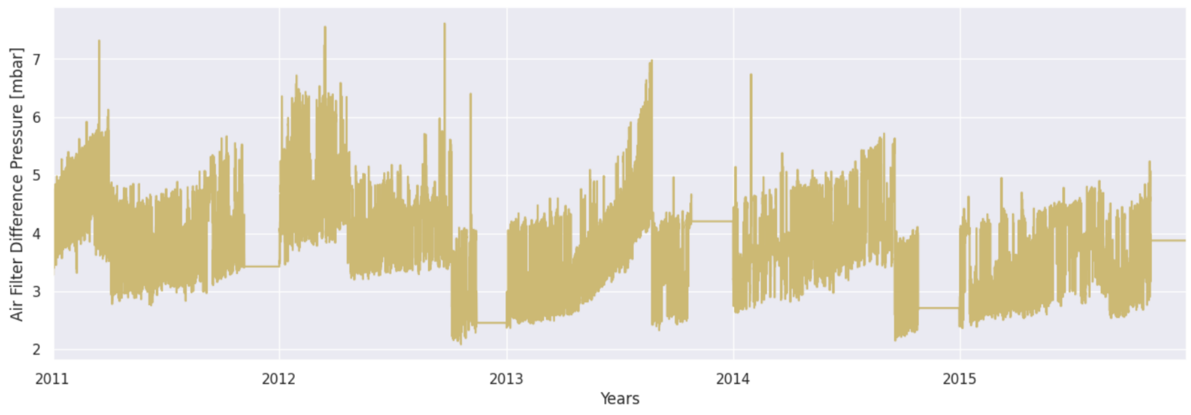


Figure 9: Air filter difference pressure graph obtained with forward fill method for filling missing values

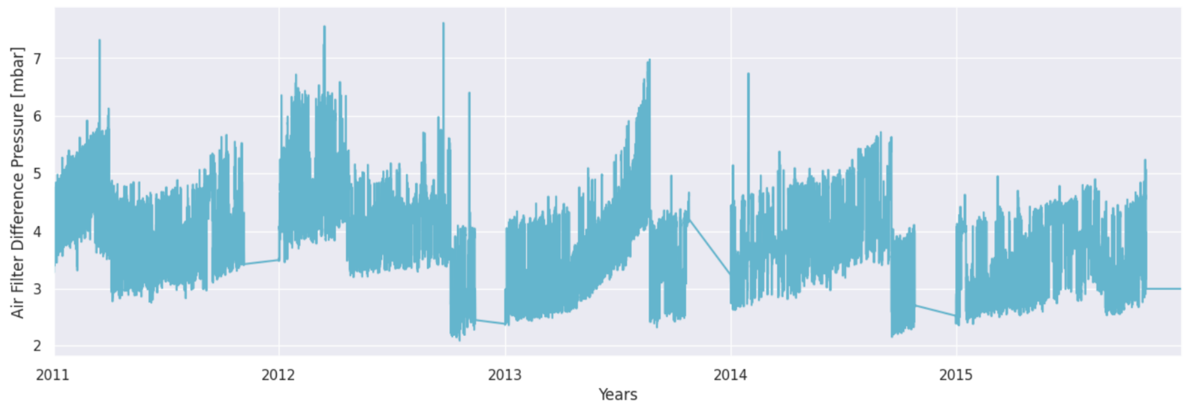


Figure 10: Air filter difference pressure graph obtained with forward linear interpolation for filling missing values

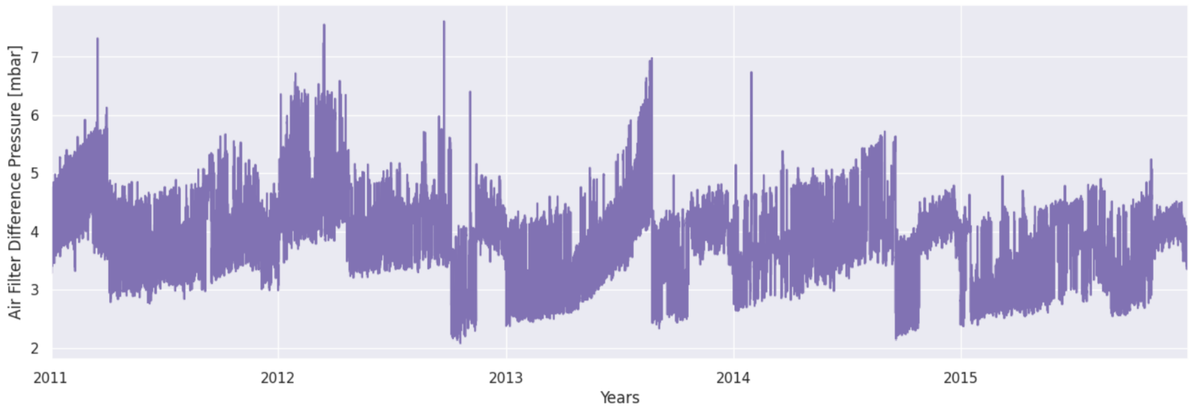


Figure 11: Air filter difference pressure graph obtained with seasonal mean (n=2920) method for filling missing values

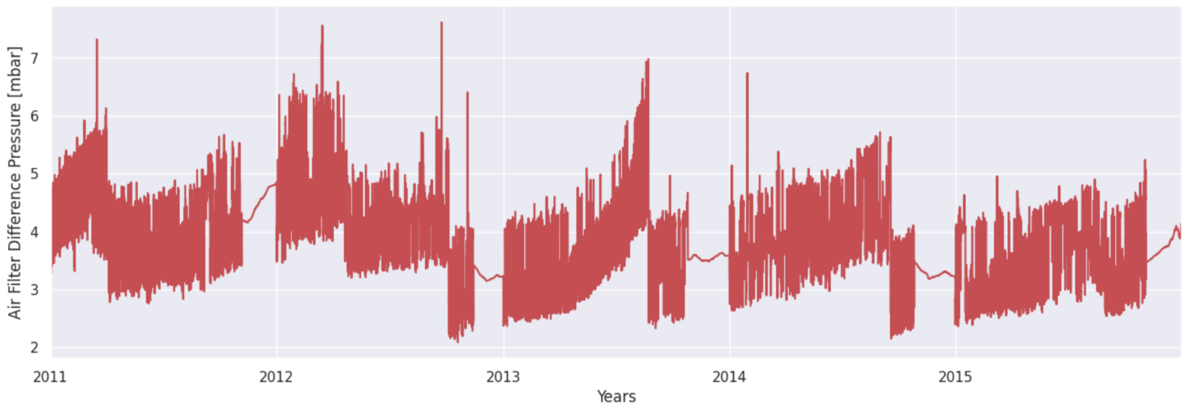


Figure 12: Air filter difference pressure graph obtained with KNN mean (n=2920) method for filling missing values

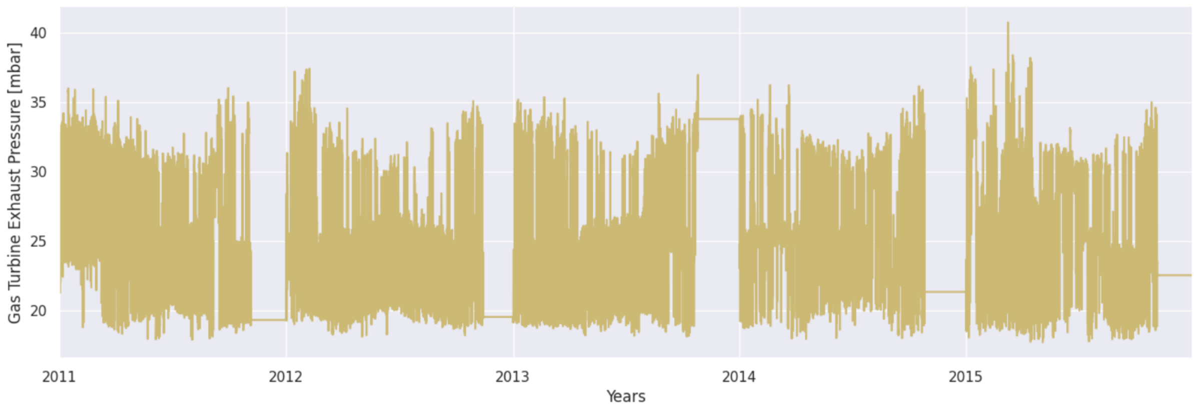


Figure 13: Gas turbine exhaust pressure graph obtained with forward fill method for filling missing values

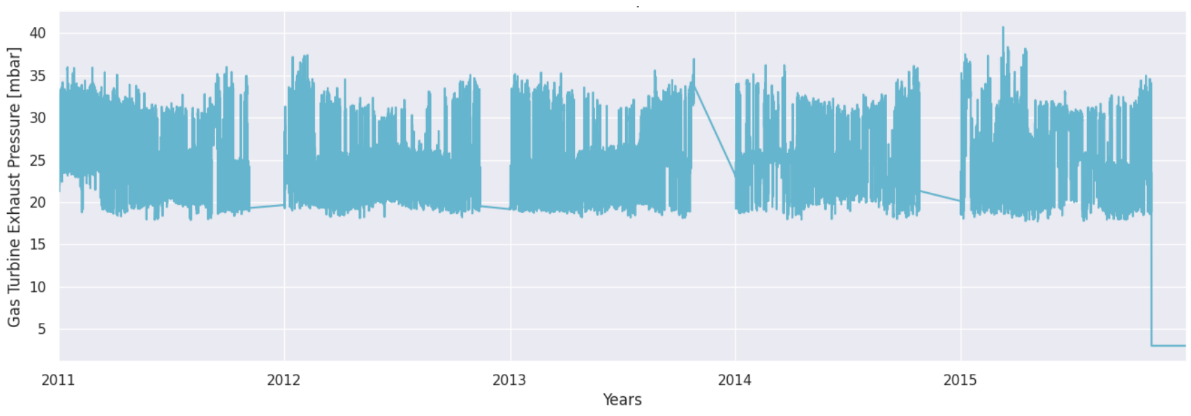


Figure 14: Gas turbine exhaust pressure graph obtained with forward linear interpolation for filling missing values

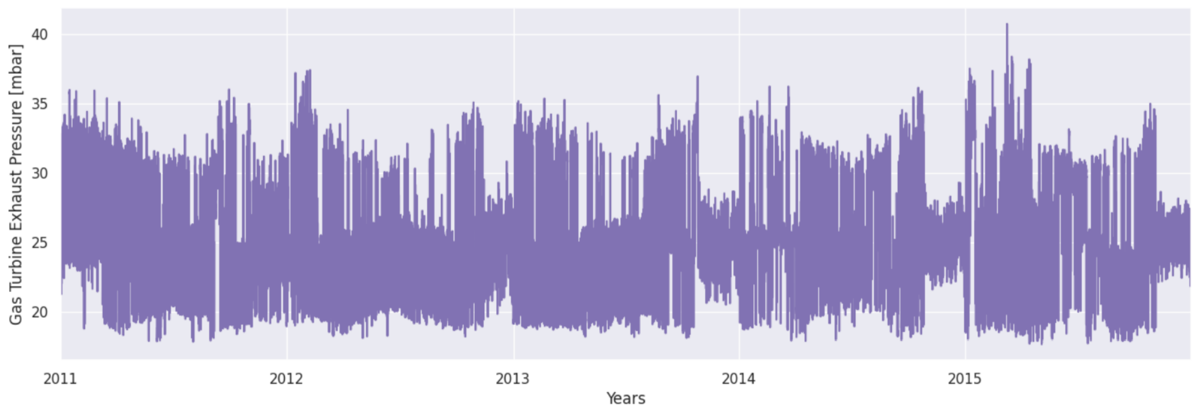


Figure 15: Gas turbine exhaust pressure graph obtained with seasonal mean (n=2920) method for filling missing values

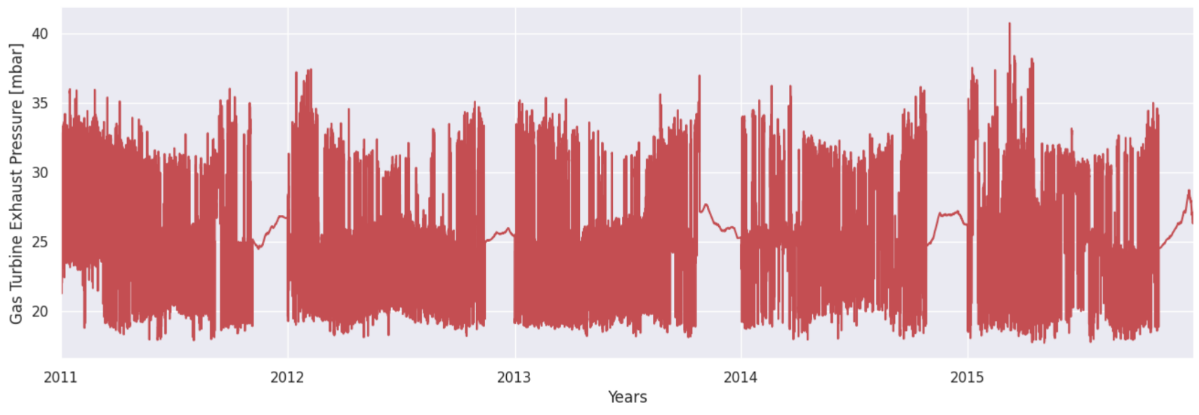


Figure 16: Gas turbine exhaust pressure graph obtained with KNN mean (n=2920) method for filling missing values

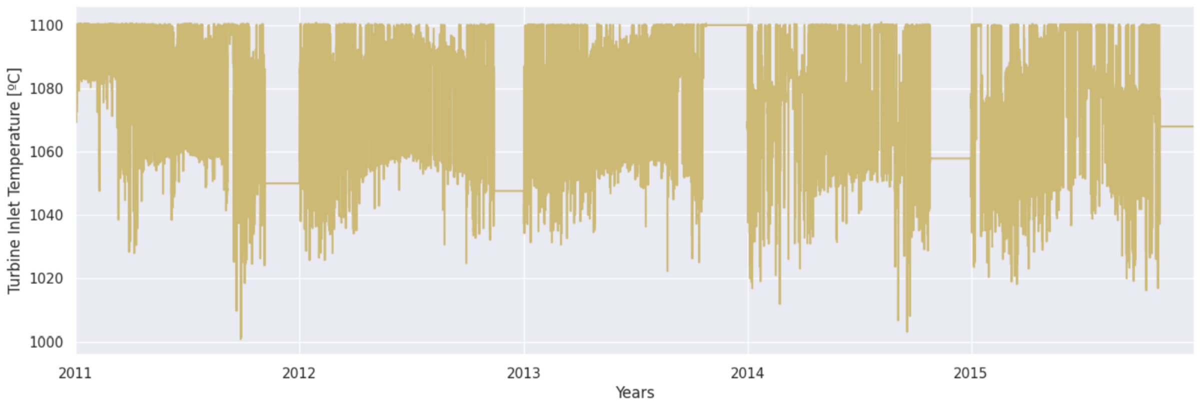


Figure 17: Turbine inlet temperature graph obtained with forward fill method for filling missing values

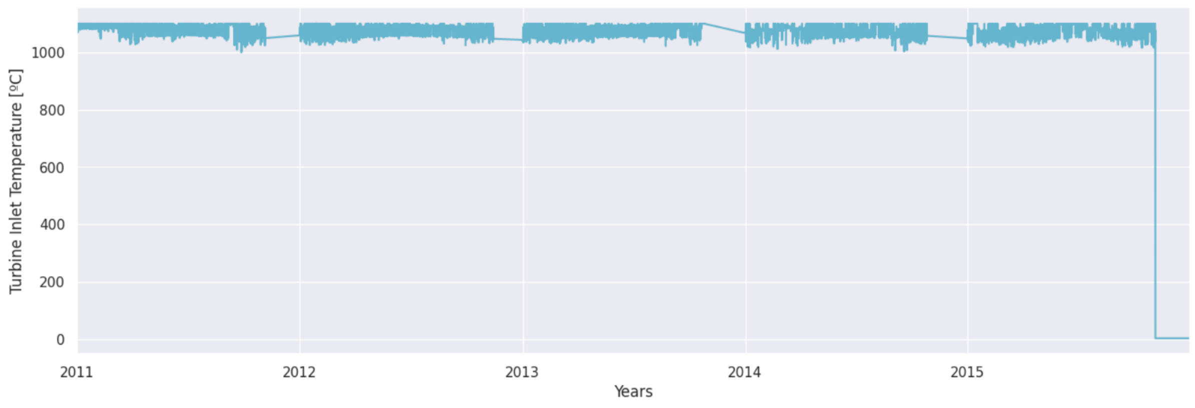


Figure 18: Turbine inlet temperature graph obtained with forward linear interpolation for filling missing values

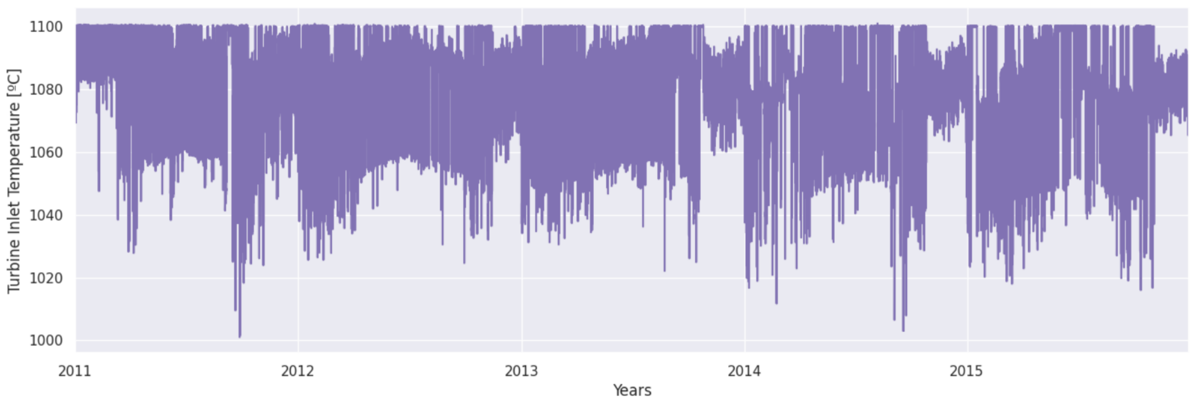


Figure 19: Turbine inlet temperature graph obtained with seasonal mean (n=2920) method for filling missing values

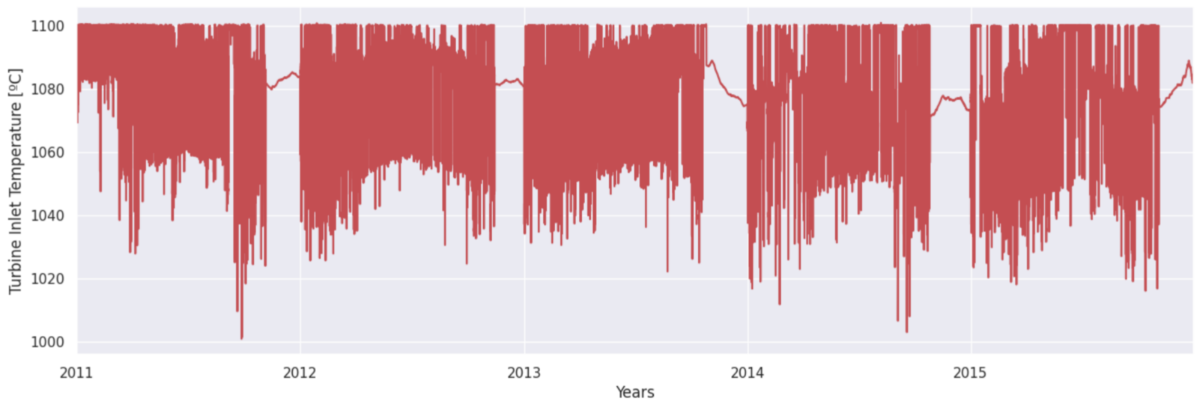


Figure 20: Turbine inlet temperature graph obtained with KNN mean (n=2920) method for filling missing values

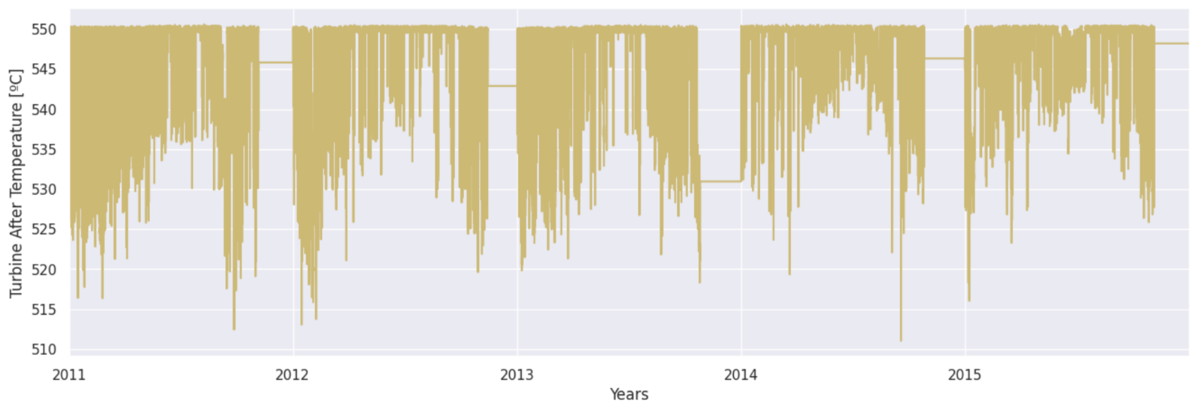


Figure 21: Turbine after temperature graph obtained with forward fill method for filling missing values

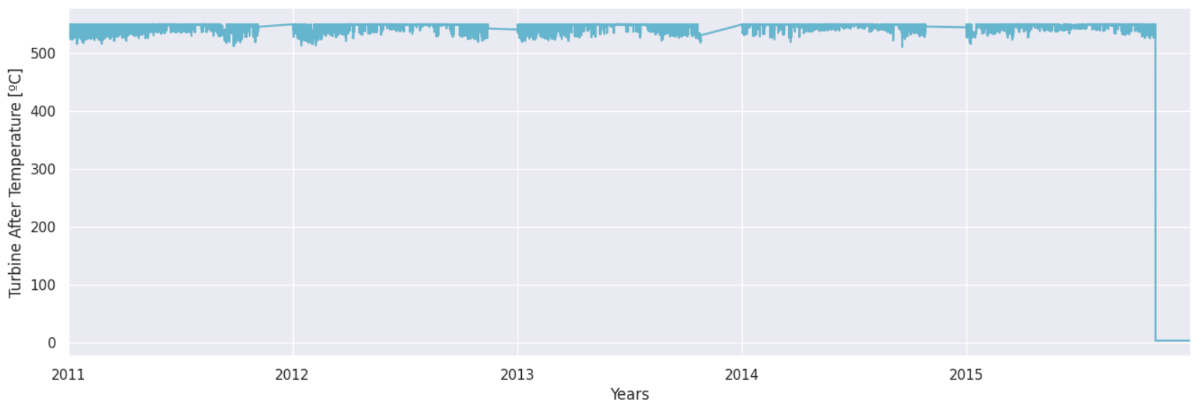


Figure 22: Turbine after temperature graph obtained with forward linear interpolation for filling missing values

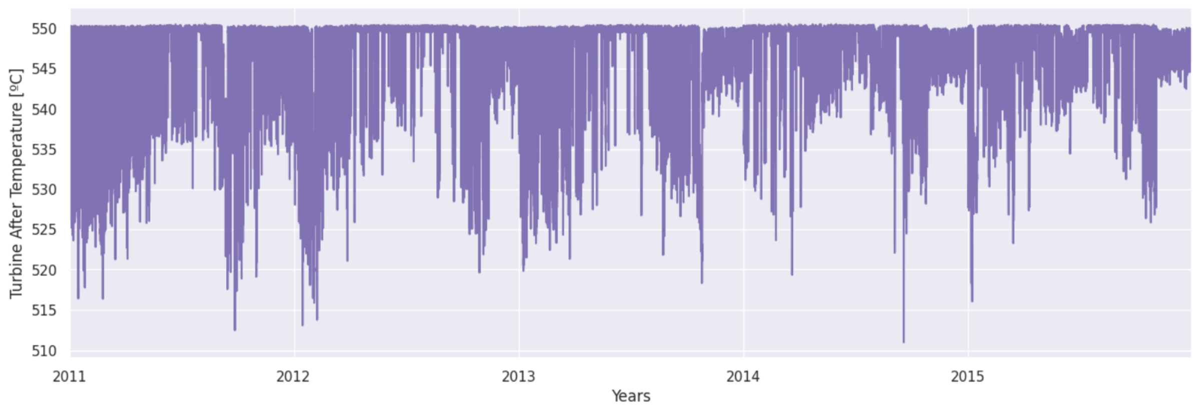


Figure 23: Turbine after temperature graph obtained with seasonal mean (n=2920) method for filling missing values

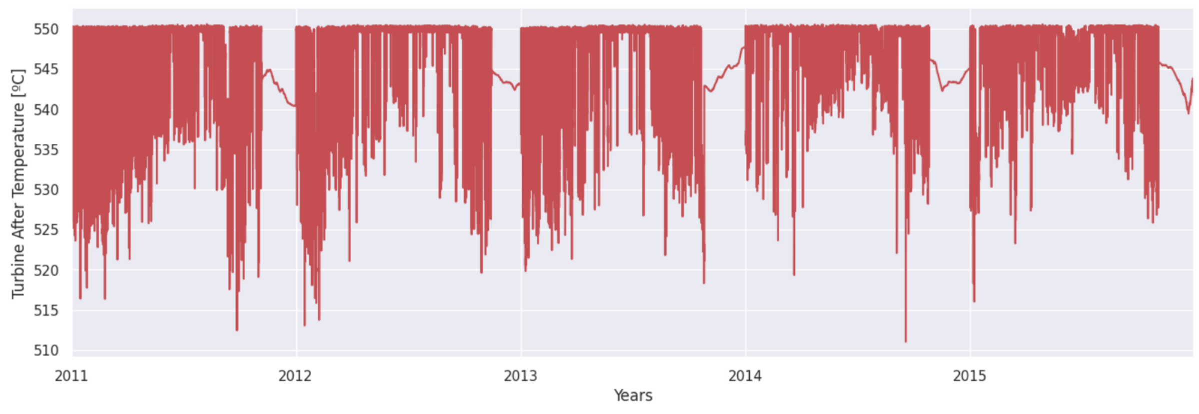


Figure 24: Turbine after temperature graph obtained with KNN mean (n=2920) method for filling missing values

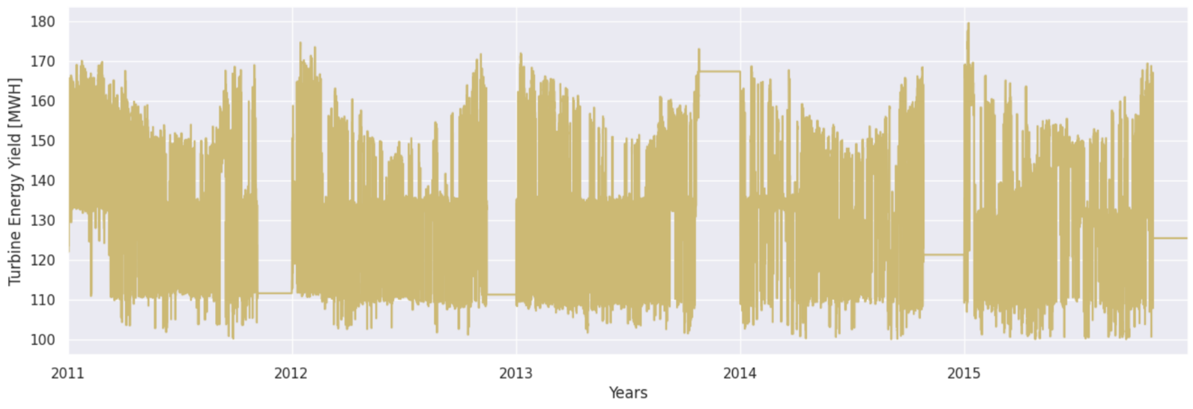


Figure 25: Turbine energy yield graph obtained with forward fill method for filling missing values

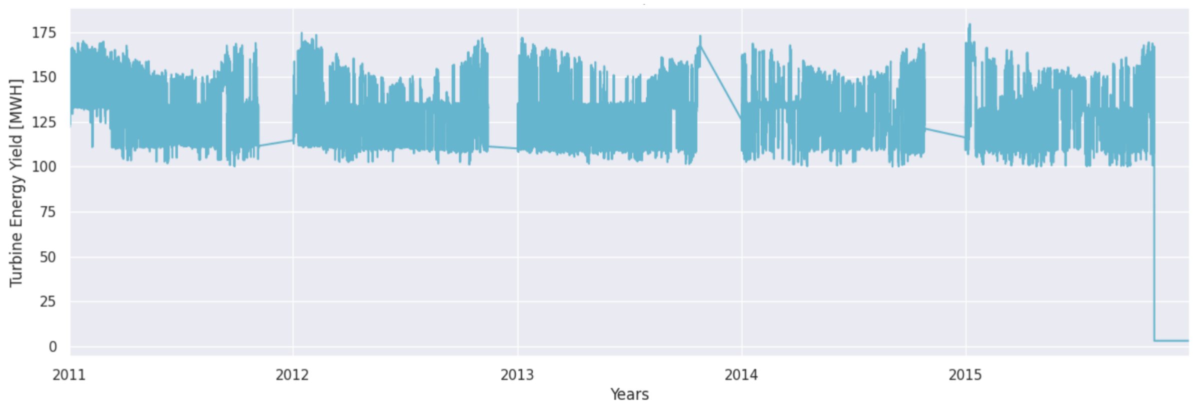


Figure 26: Turbine energy yield graph obtained with forward linear interpolation for filling missing values

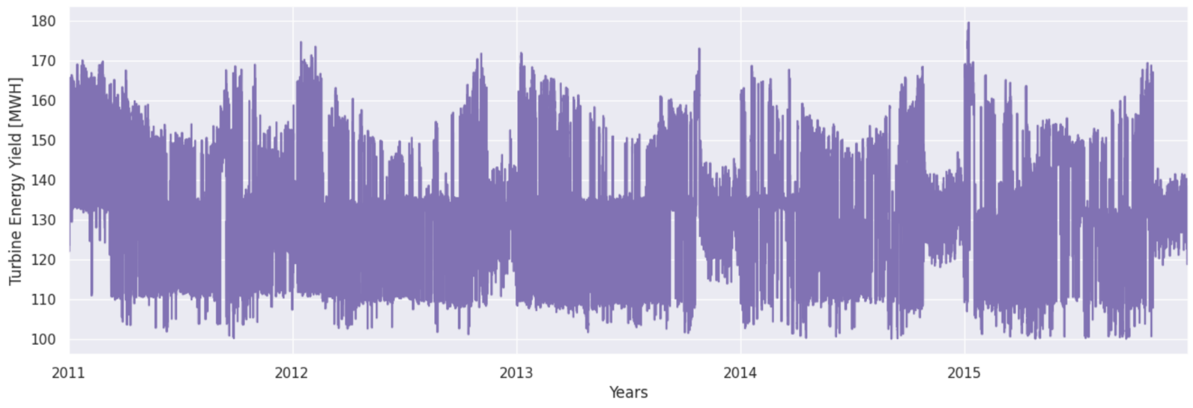


Figure 27: Turbine energy yield graph obtained with seasonal mean (n=2920) method for filling missing values

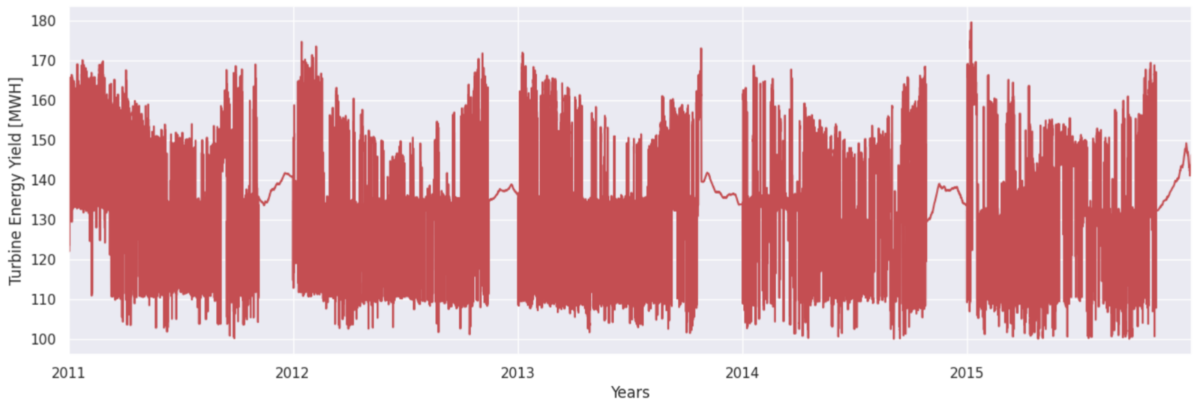


Figure 28: Turbine energy yield graph obtained with KNN mean (n=2920) method for filling missing values

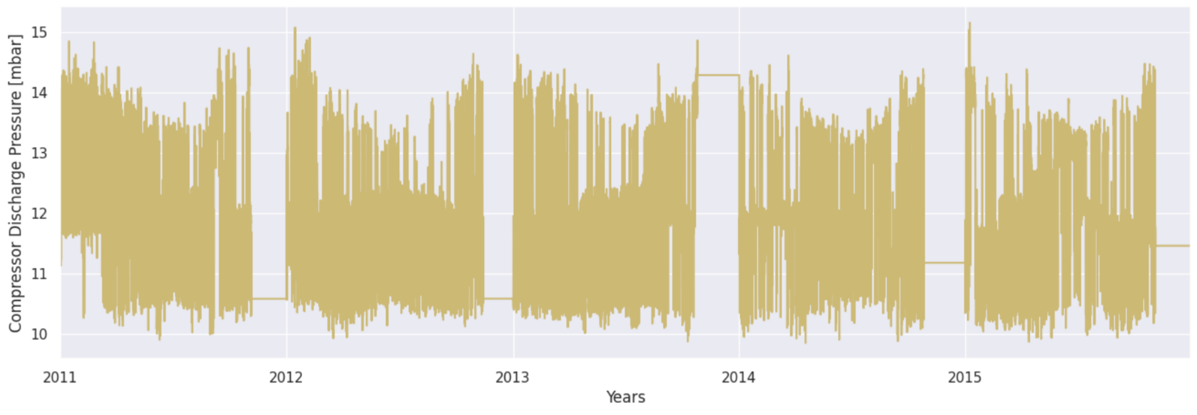


Figure 29: Compressor discharge pressure graph obtained with forward fill method for filling missing values

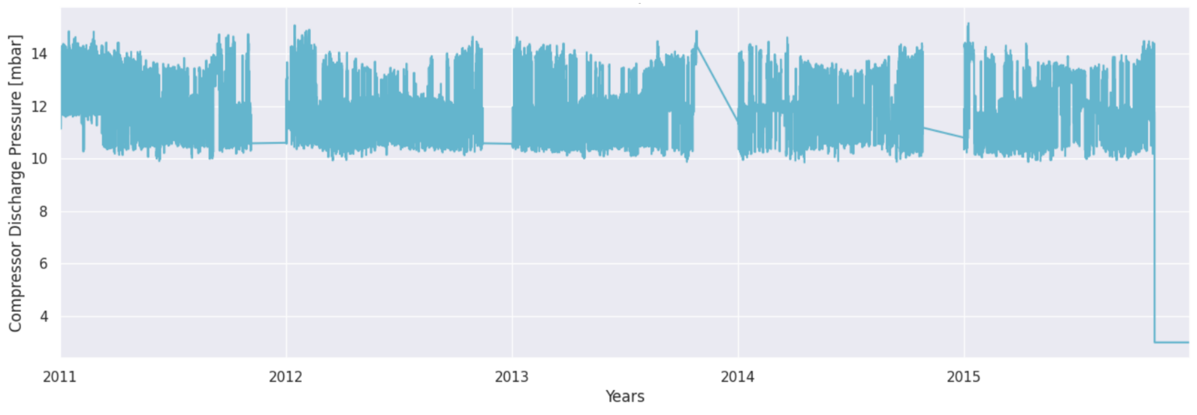


Figure 30: Compressor discharge pressure graph obtained with forward linear interpolation for filling missing values

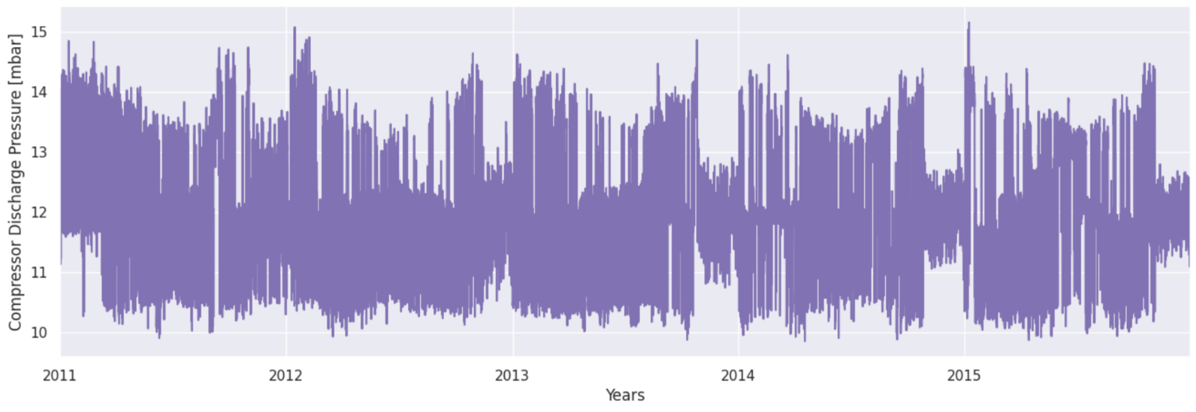


Figure 31: Compressor discharge pressure graph obtained with seasonal mean (n=2920) method for filling missing values

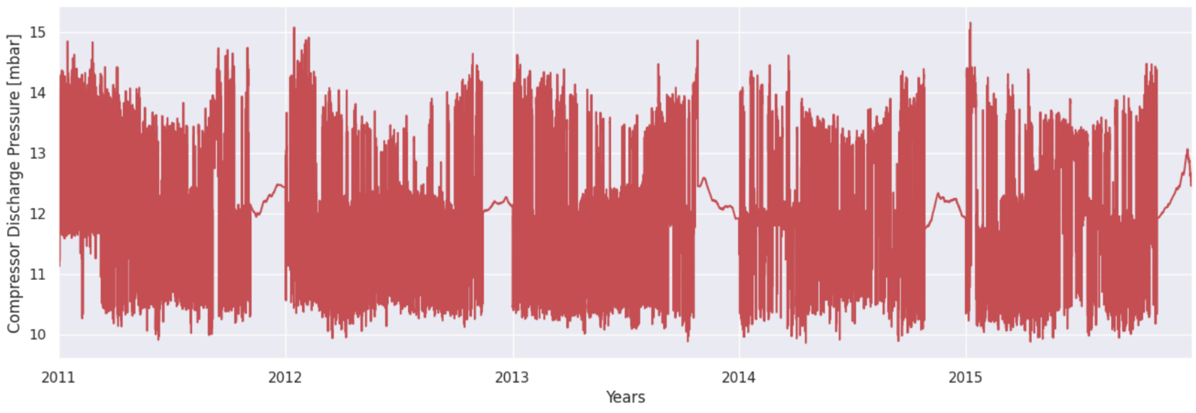


Figure 32: Compressor discharge pressure graph obtained with KNN mean (n=2920) method for filling missing values

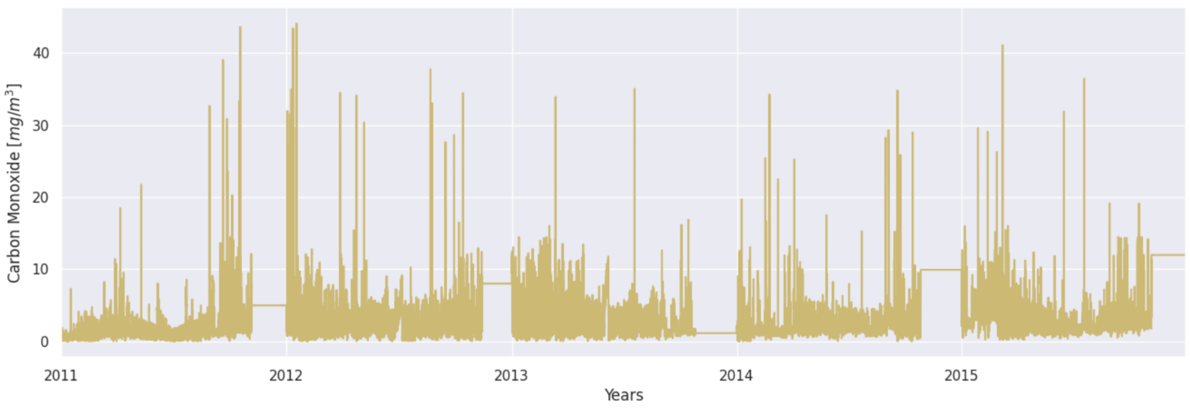


Figure 33: Carbon monoxide emissions graph obtained with forward fill method for filling missing values

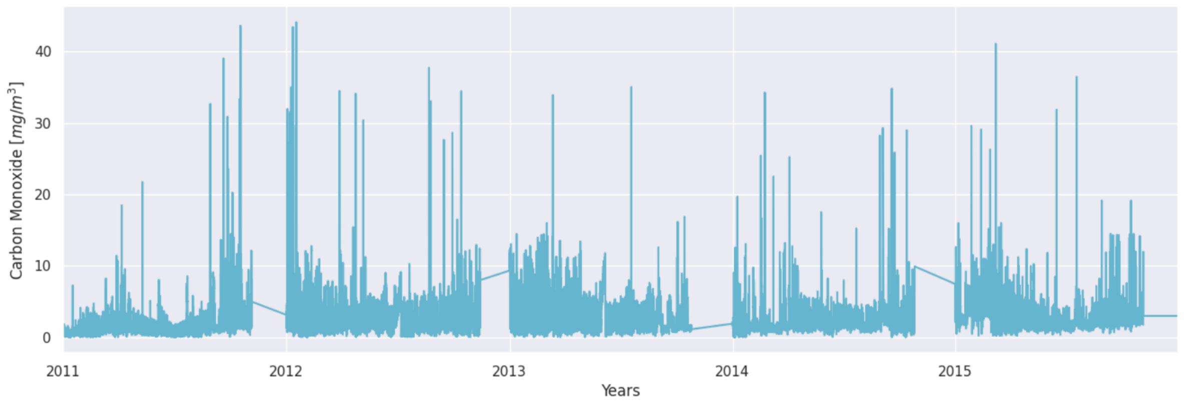


Figure 34: Carbon monoxide emissions graph obtained with forward linear interpolation for filling missing values

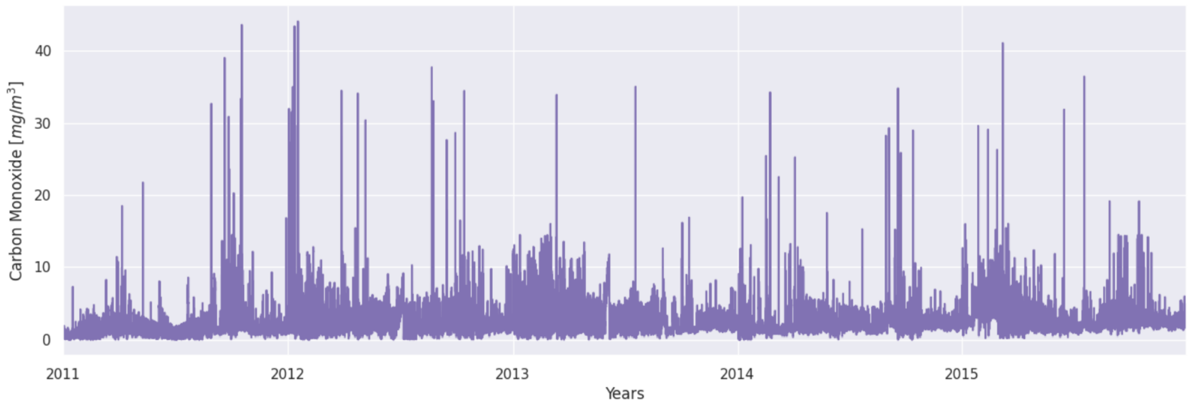


Figure 35: Carbon monoxide emissions graph obtained with seasonal mean (n=2920) method for filling missing values

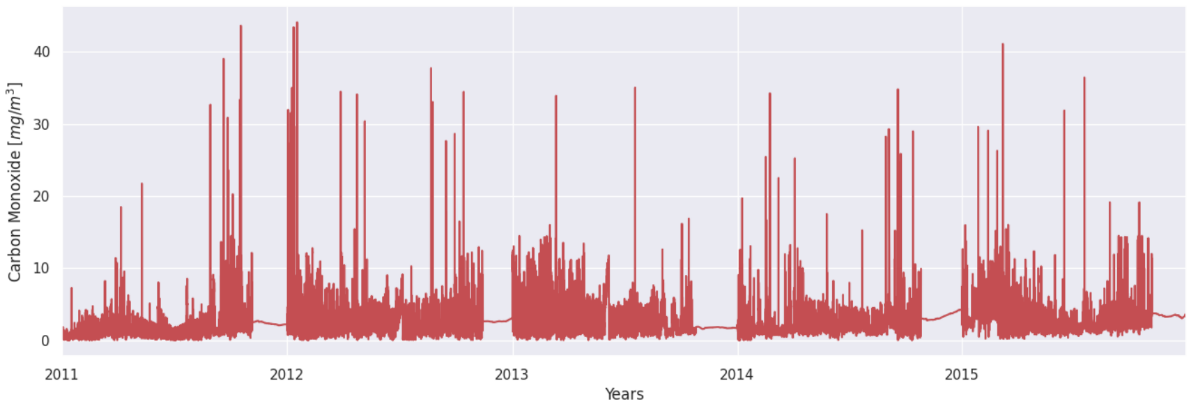


Figure 36: Carbon monoxide emissions graph obtained with KNN mean (n=2920) method for filling missing values

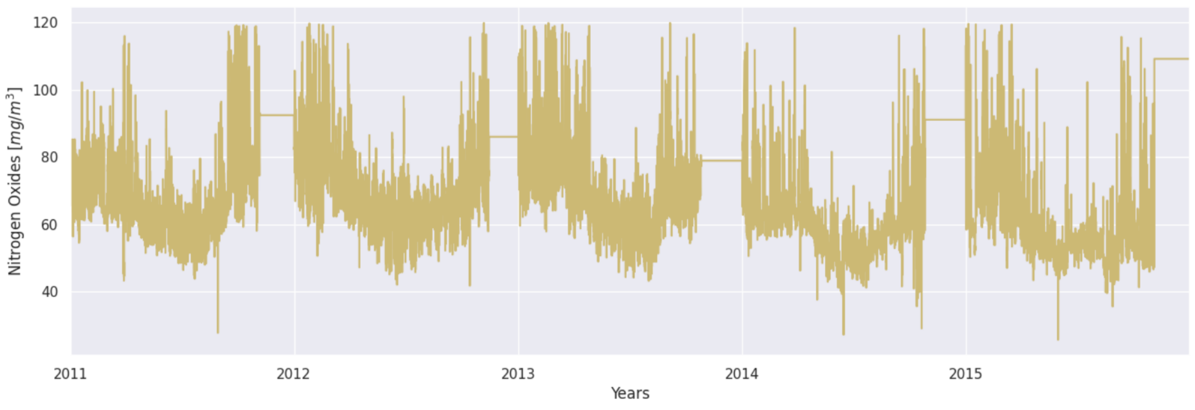


Figure 37: Nitrogen oxides emissions graph obtained with forward fill method for filling missing values

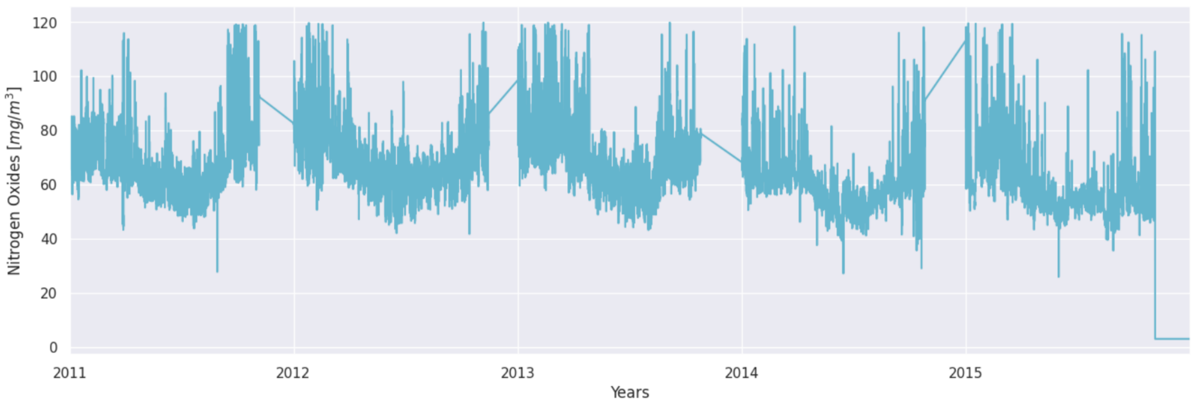


Figure 38: Nitrogen oxides emissions graph obtained with forward linear interpolation for filling missing values

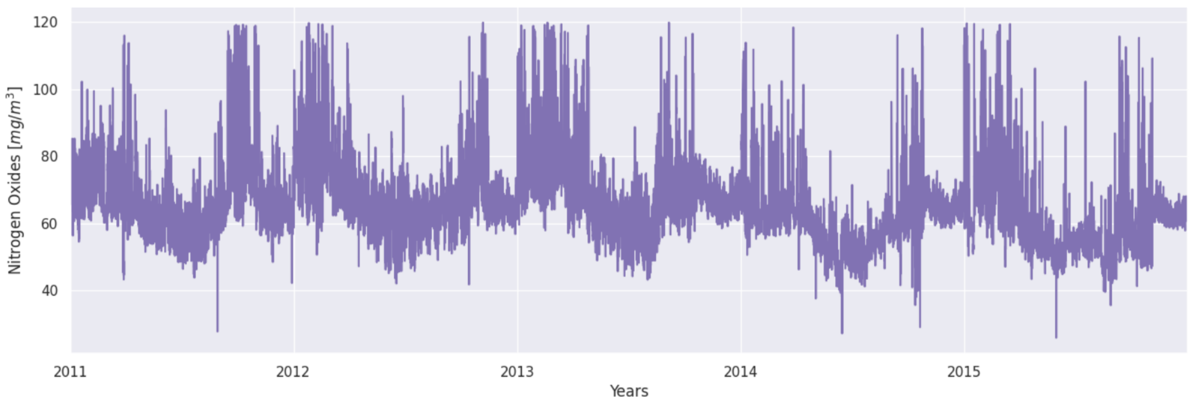


Figure 39: Nitrogen oxides emissions graph obtained with seasonal mean (n=2920) method for filling missing values

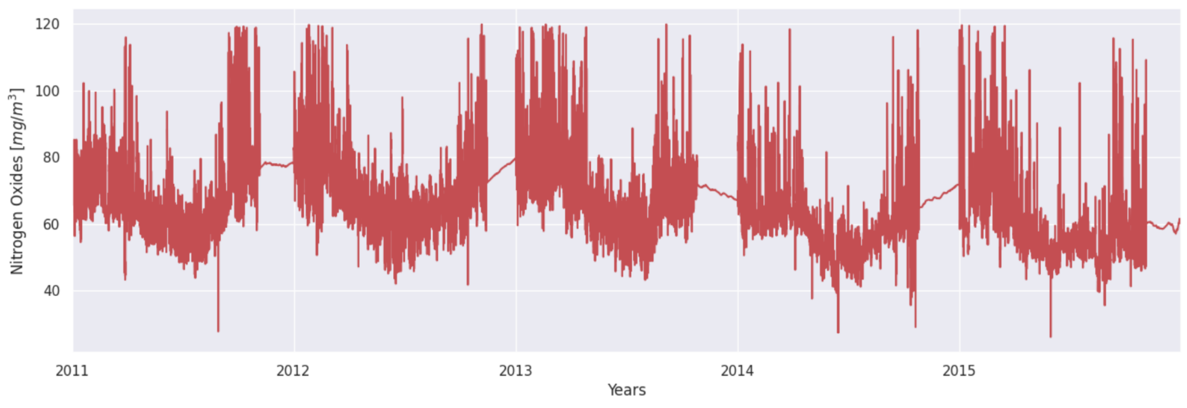


Figure 40: Nitrogen oxides emissions graph obtained with KNN mean (n=2920) method for filling missing values

	MSE	MAE
Forward Fill	526192.19	725.35
Linear Interpolation	357941.38	595.49
Seasonal Mean	479696.91	692.58
KNN Mean	402436.09	634.34

Table 1: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for ambient pressure feature

	MSE	MAE
Forward Fill	227.19	11.48
Linear Interpolation	694.29	16.72
Seasonal Mean	201.03	10.77
KNN Mean	207.60	11.02

Table 2: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for ambient humidity feature

	MSE	MAE
Forward Fill	0.59	0.61
Linear Interpolation	0.71	0.70
Seasonal Mean	0.57	0.58
KNN Mean	0.58	0.63

Table 3: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for air filter difference pressure feature

	MSE	MAE
Forward Fill	16.86	3.23
Linear Interpolation	53.52	4.82
Seasonal Mean	14.50	2.88
KNN Mean	14.36	2.90

Table 4: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for gas turbine exhaust pressure feature

	MSE	MAE
Forward Fill	497201.28	704.90
Linear Interpolation	409499.09	636.48
Seasonal Mean	560407.63	748.41
KNN Mean	431619.56	656.76

Table 5: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for turbine inlet temperature feature

	MSE	MAE
Forward Fill	44831.28	211.68
Linear Interpolation	37297.18	184.89
Seasonal Mean	28443.81	168.59
KNN Mean	33563.03	183.14

Table 6: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for turbine after temperature feature

	MSE	MAE
Forward Fill	202.77	11.05
Linear Interpolation	1494.37	20.93
Seasonal Mean	190.78	10.44
KNN Mean	191.30	10.46

Table 7: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for turbine energy yield feature

	MSE	MAE
Forward Fill	1.01	0.77
Linear Interpolation	7.13	1.45
Seasonal Mean	0.93	0.74
KNN Mean	0.93	0.73

Table 8: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for compressor discharge pressure feature

	MSE	MAE
Forward Fill	15.87	2.51
Linear Interpolation	7.35	1.70
Seasonal Mean	4.47	1.09
KNN Mean	4.48	1.22

Table 9: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for carbon monoxide emissions feature

	MSE	MAE
Forward Fill	348.81	16.24
Linear Interpolation	661.50	19.74
Seasonal Mean	144.74	9.90
KNN Mean	180.95	11.45

Table 10: Results of MSE and MAE obtained using different methods to fill missing values in the dataset for nitrogen oxides emissions feature

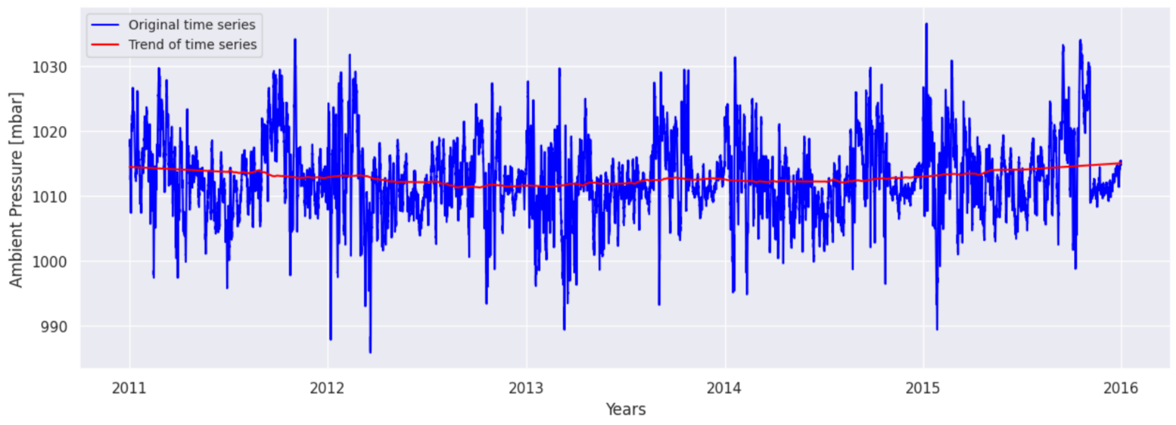


Figure 41: Original time series and trend for ambient pressure feature with additive decomposition

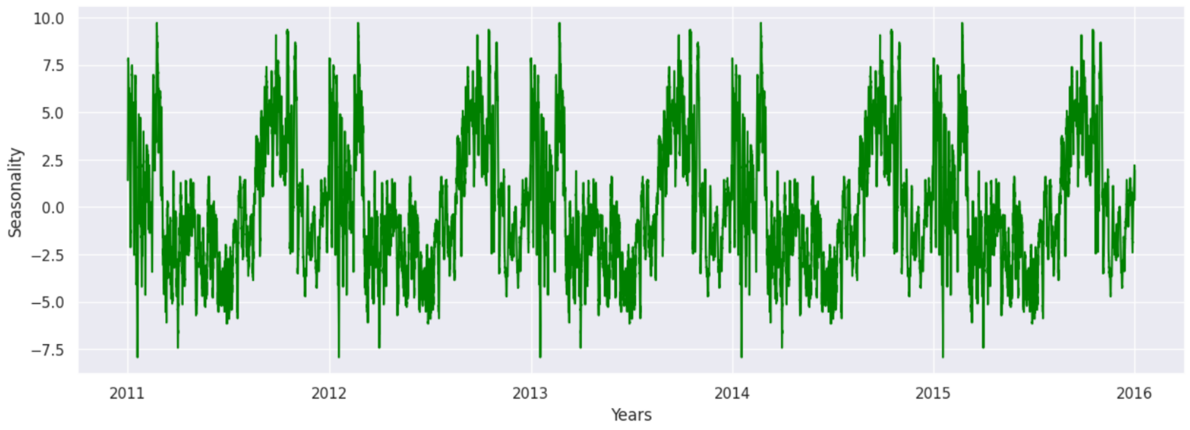


Figure 42: Seasonality plot for ambient pressure feature with additive decomposition

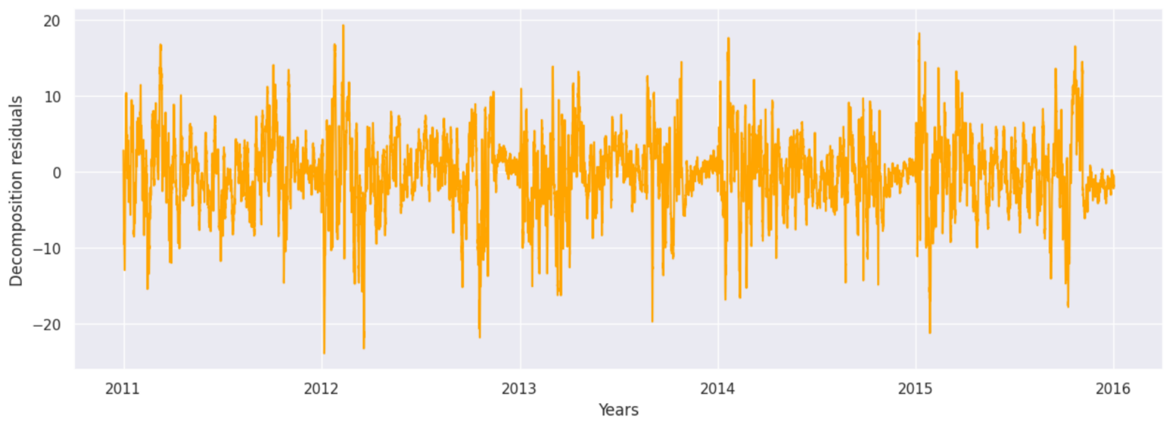


Figure 43: Decomposition residuals (noise) for ambient pressure feature with additive decomposition

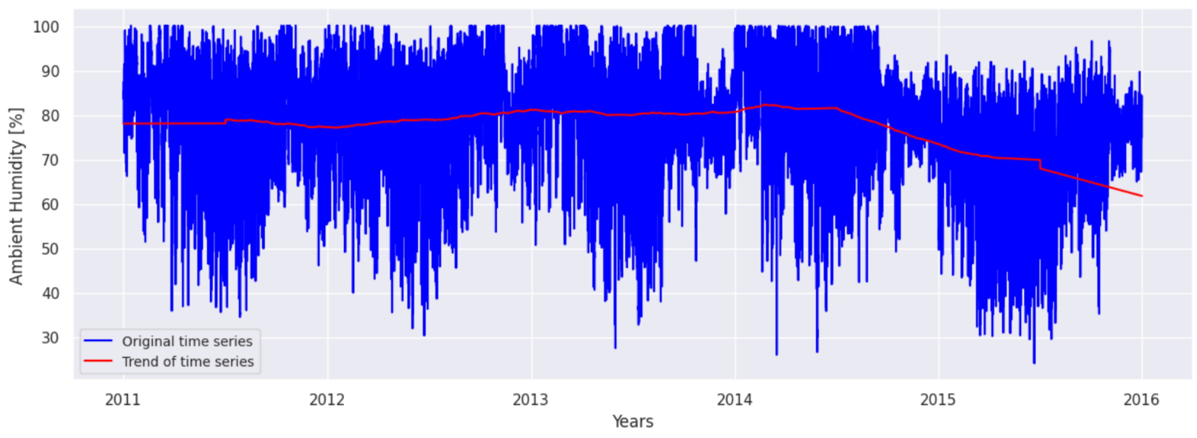


Figure 44: Original time series and trend for ambient humidity feature with additive decomposition

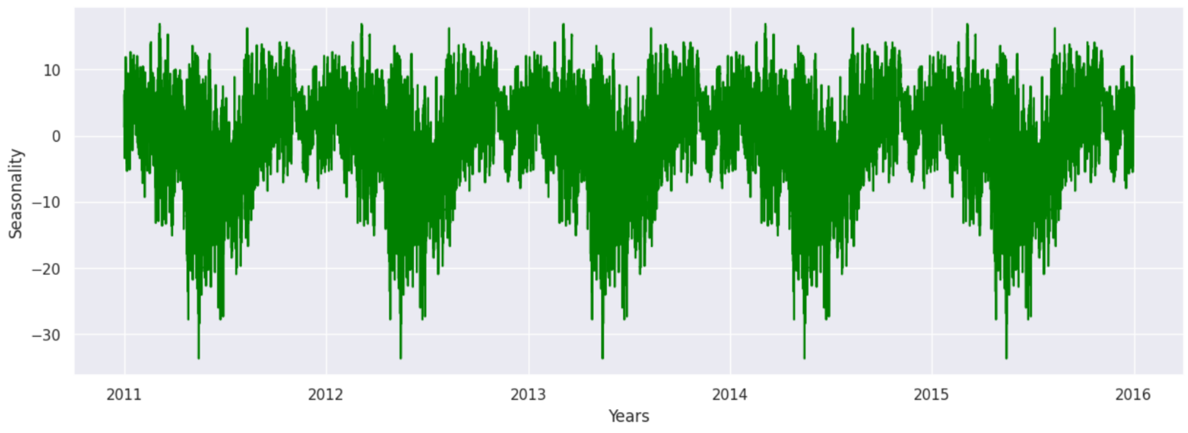


Figure 45: Seasonality plot for ambient humidity feature with additive decomposition

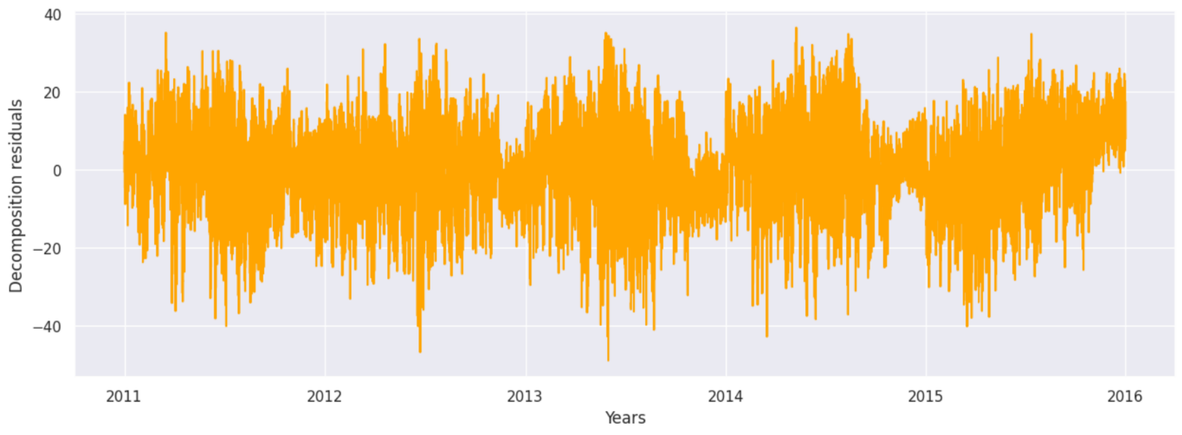


Figure 46: Decomposition residuals (noise) for ambient humidity feature with additive decomposition

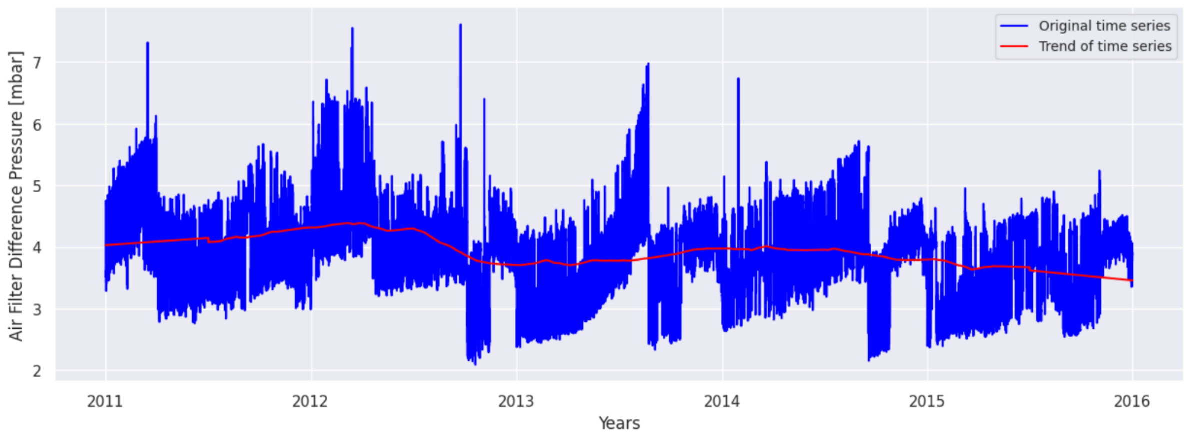


Figure 47: Original time series and trend for air filter difference pressure feature with additive decomposition

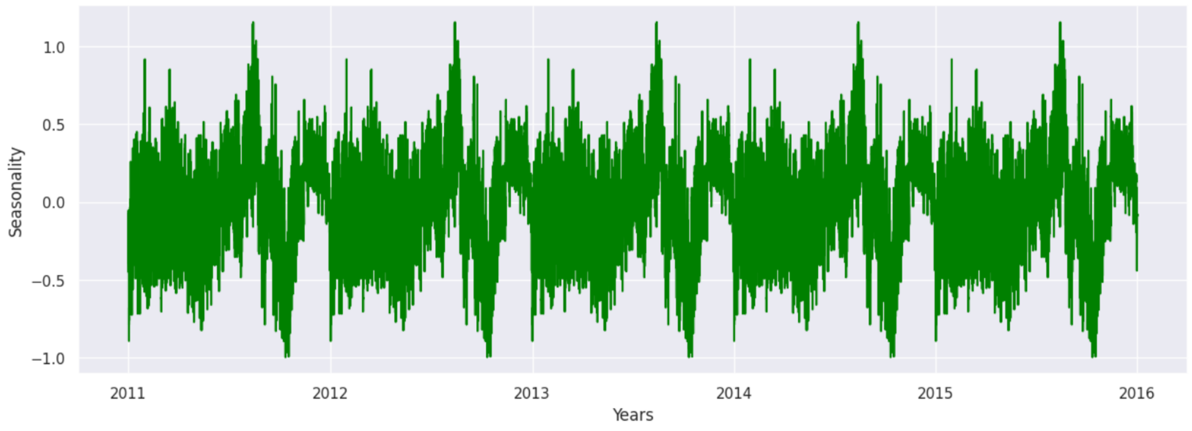


Figure 48: Seasonality plot for air filter difference pressure feature with additive decomposition

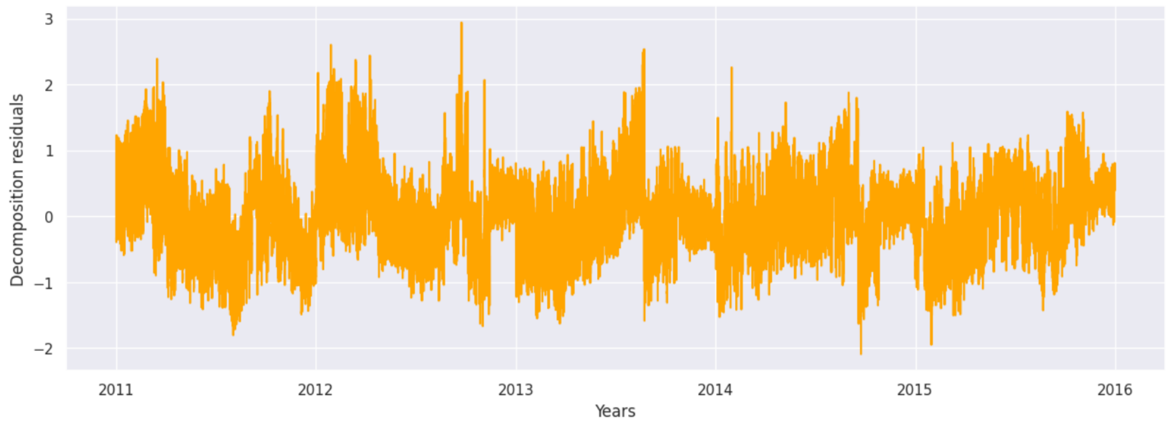


Figure 49: Decomposition residuals (noise) for air filter difference pressure feature with additive decomposition

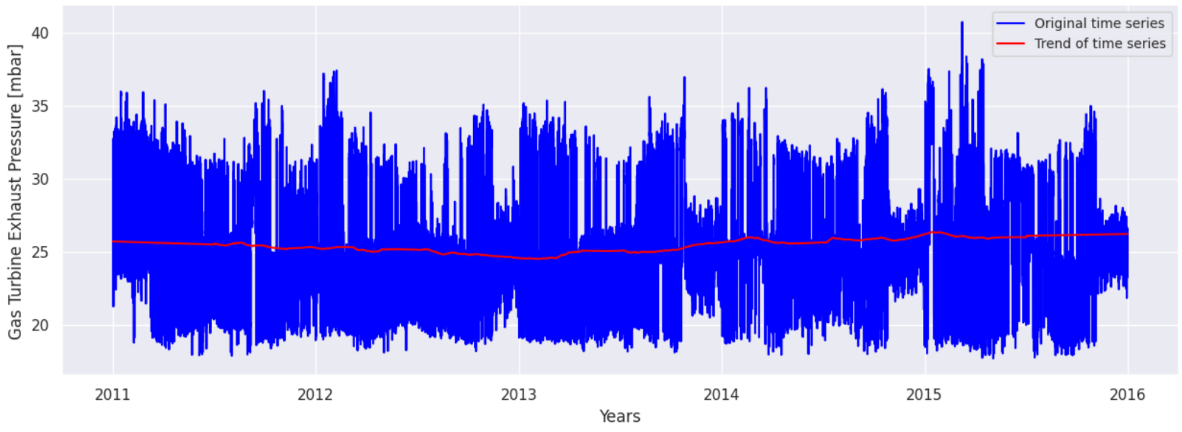


Figure 50: Original time series and trend for gas turbine exhaust pressure feature with additive decomposition

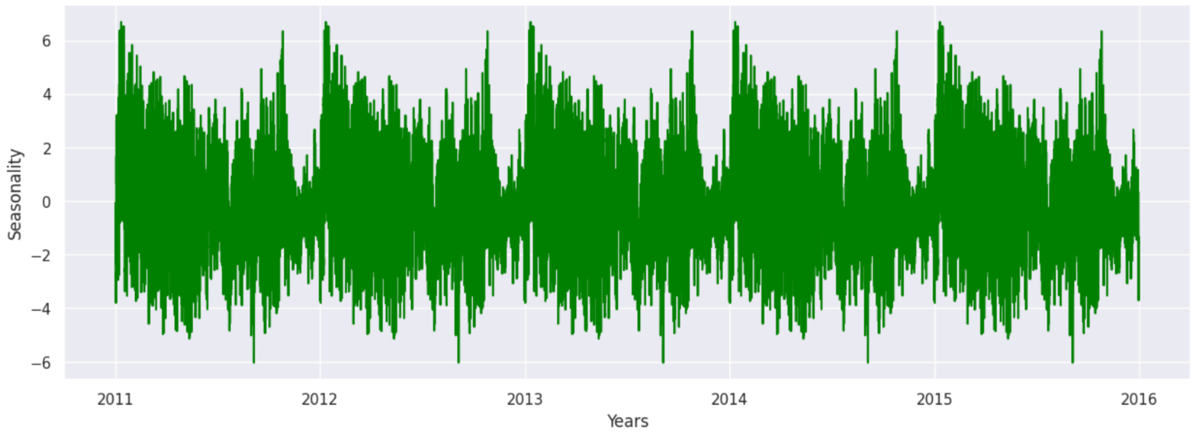


Figure 51: Seasonality plot for gas turbine exhaust pressure feature with additive decomposition

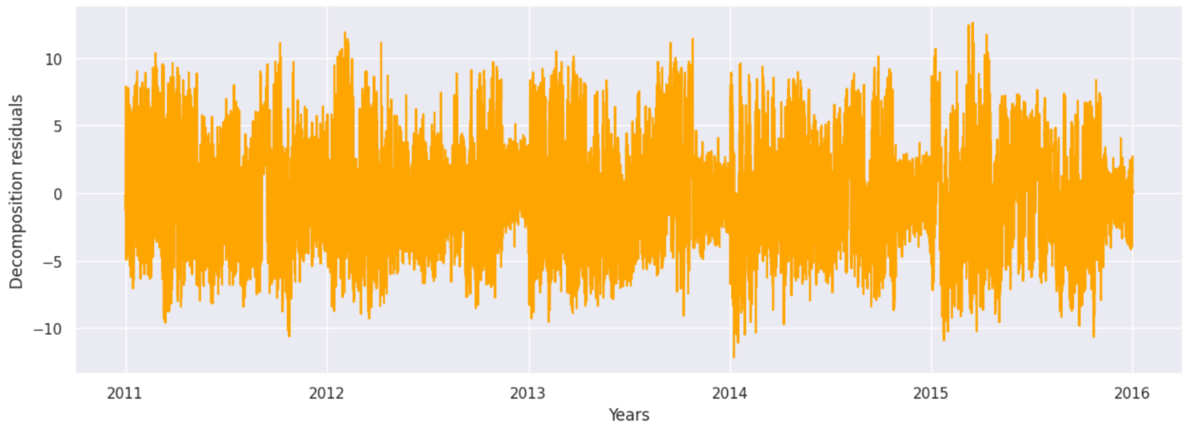


Figure 52: Decomposition residuals (noise) for gas turbine exhaust pressure feature with additive decomposition

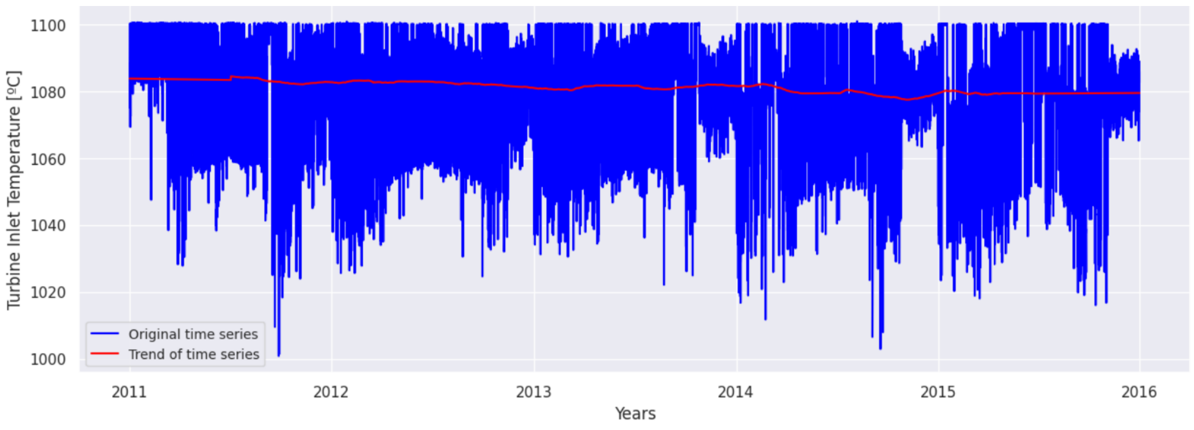


Figure 53: Original time series and trend for turbine inlet temperature feature with additive decomposition

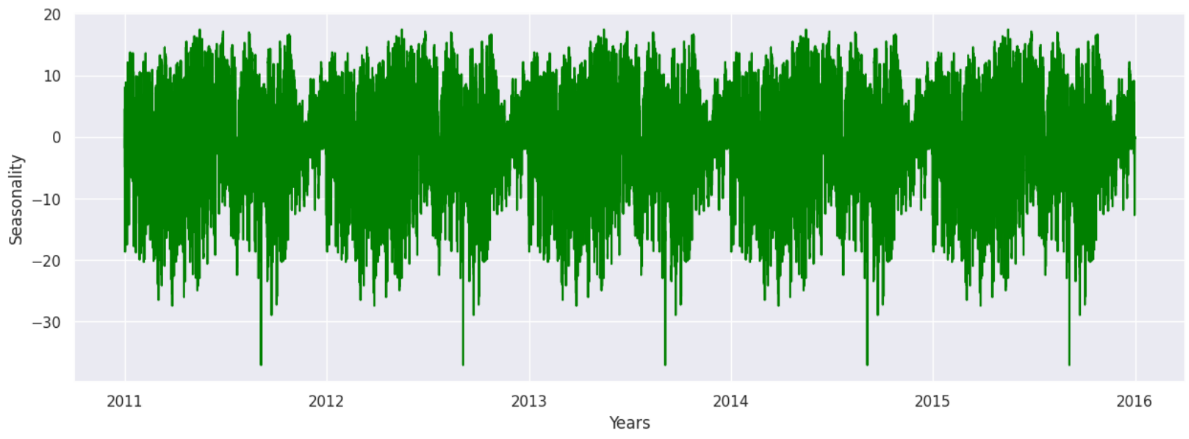


Figure 54: Seasonality plot for turbine inlet temperature feature with additive decomposition

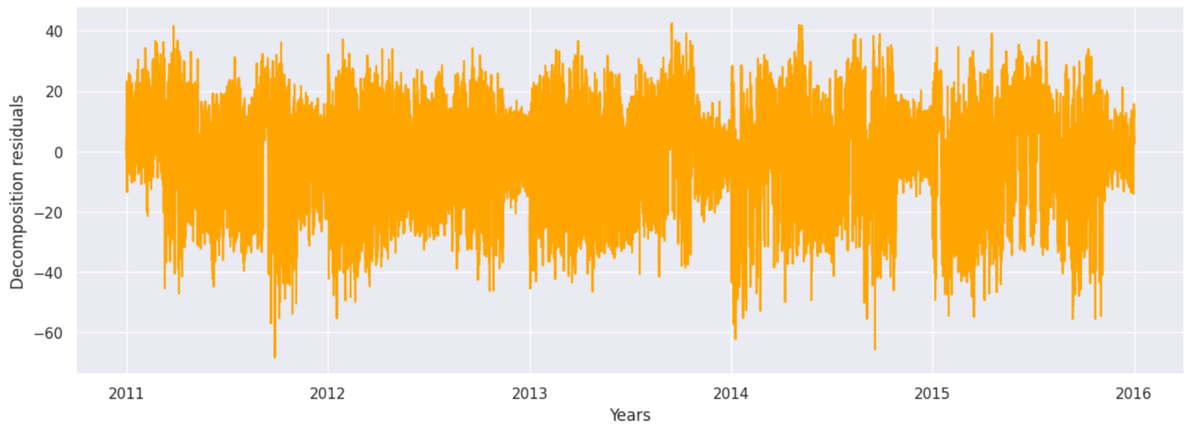


Figure 55: Decomposition residuals (noise) for turbine inlet temperature feature with additive decomposition

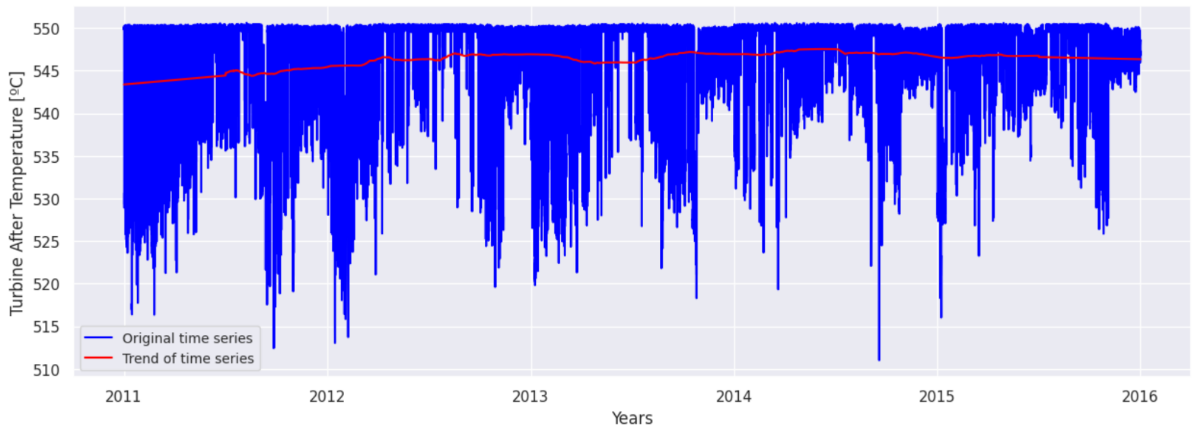


Figure 56: Original time series and trend for turbine after temperature feature with additive decomposition

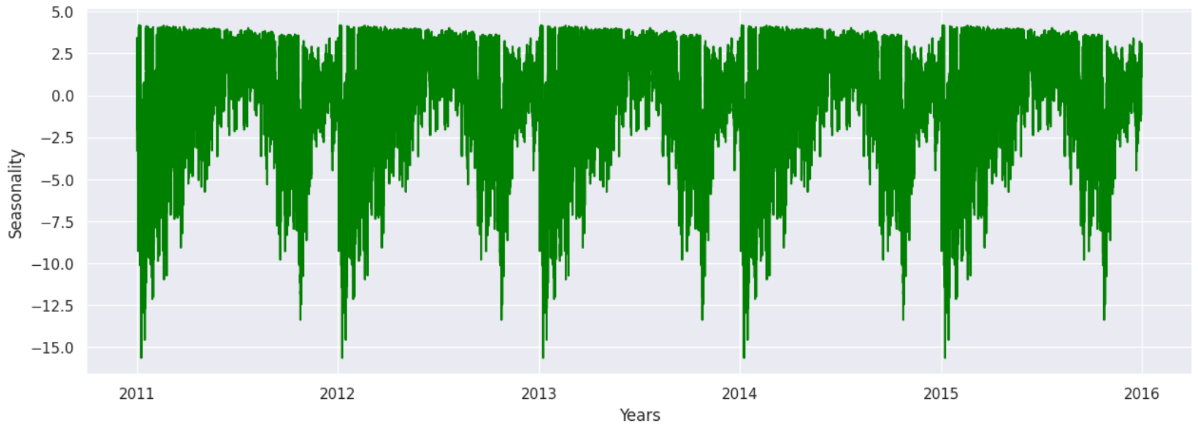


Figure 57: Seasonality plot for turbine after temperature feature with additive decomposition

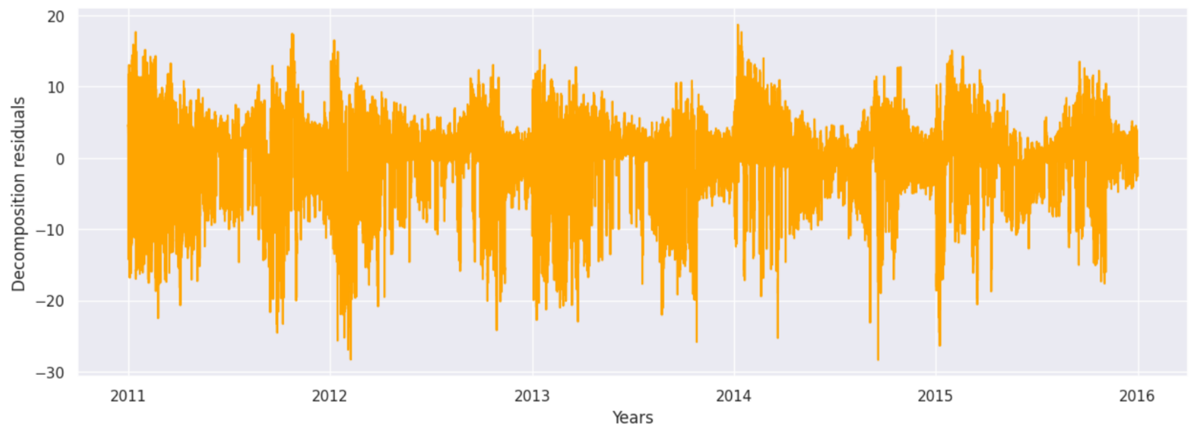


Figure 58: Decomposition residuals (noise) for turbine after temperature feature with additive decomposition

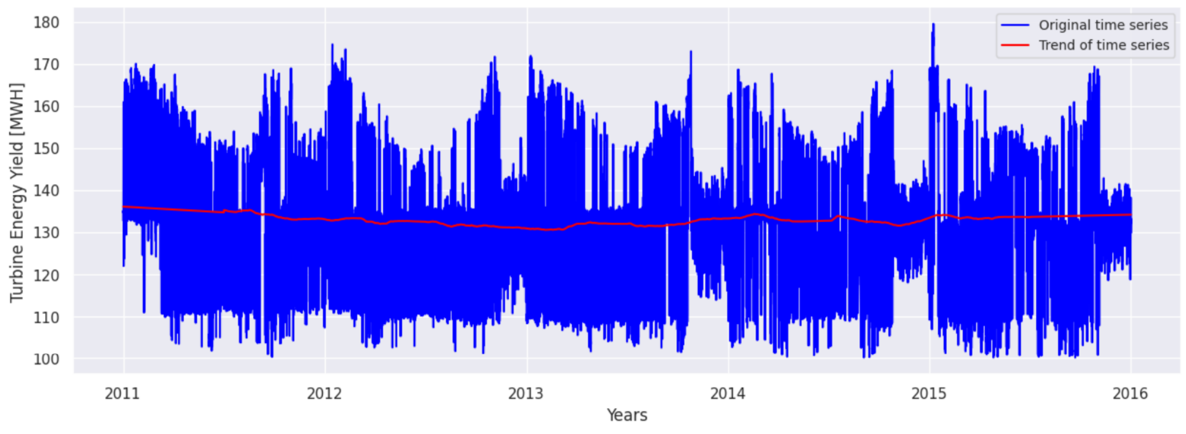


Figure 59: Original time series and trend for turbine energy yield feature with additive decomposition

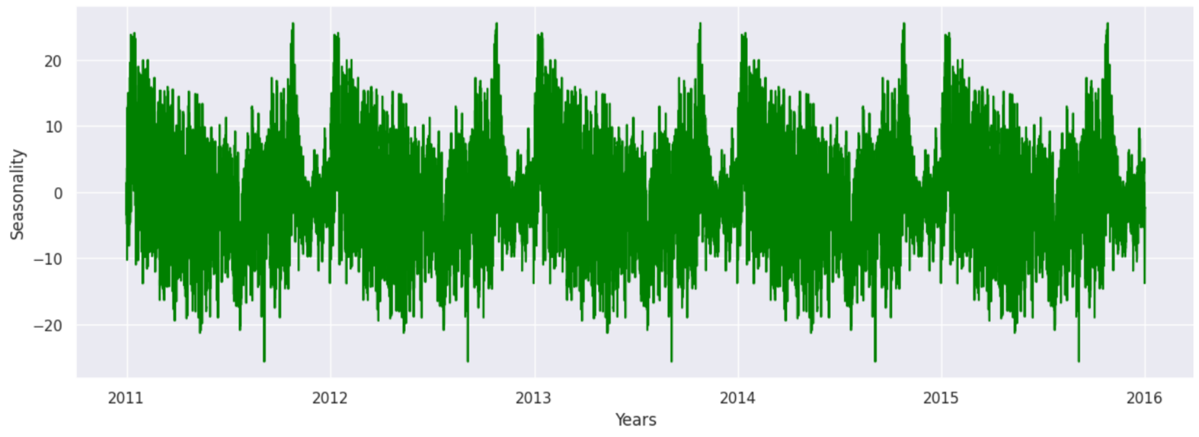


Figure 60: Seasonality plot for turbine energy yield feature with additive decomposition

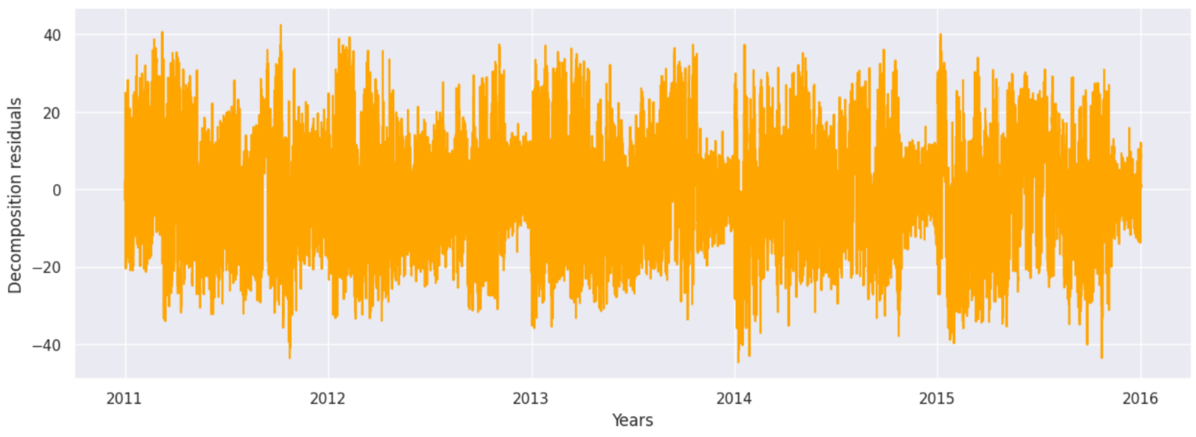


Figure 61: Decomposition residuals (noise) for turbine energy yield feature with additive decomposition

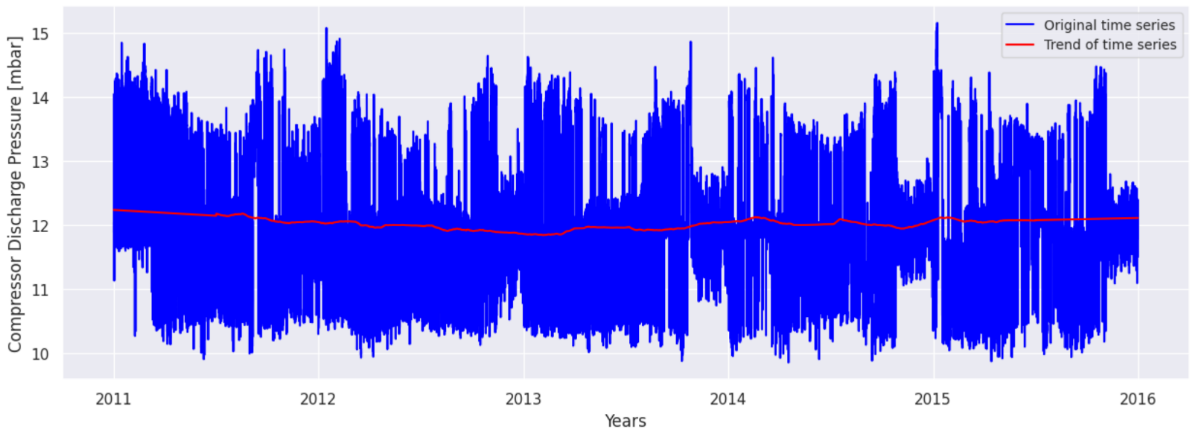


Figure 62: Original time series and trend for compressor discharge pressure feature with additive decomposition

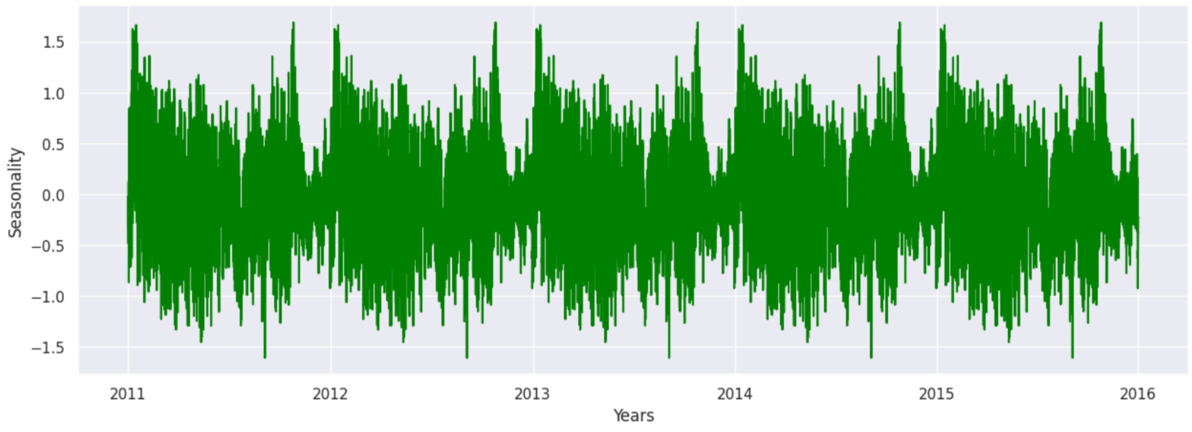


Figure 63: Seasonality plot for compressor discharge pressure feature with additive decomposition

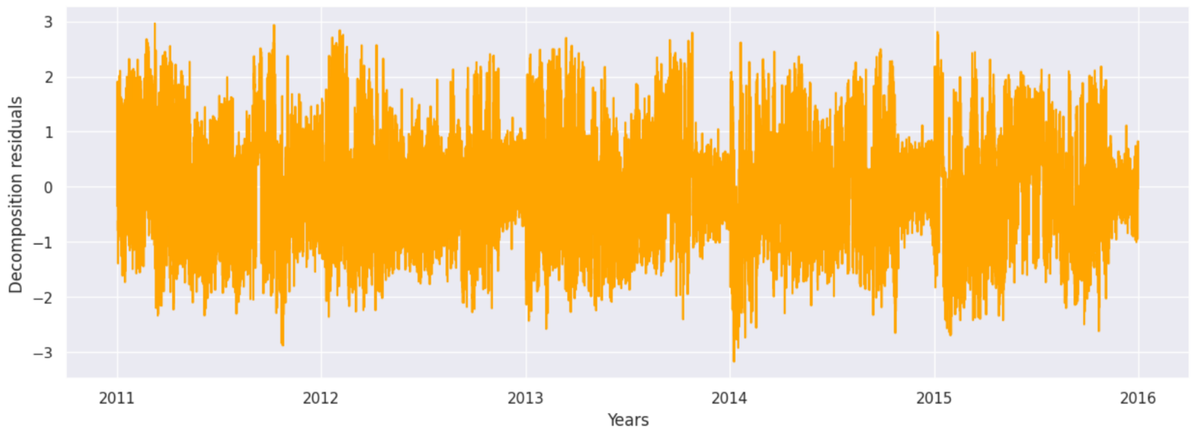


Figure 64: Decomposition residuals (noise) for compressor discharge pressure feature with additive decomposition

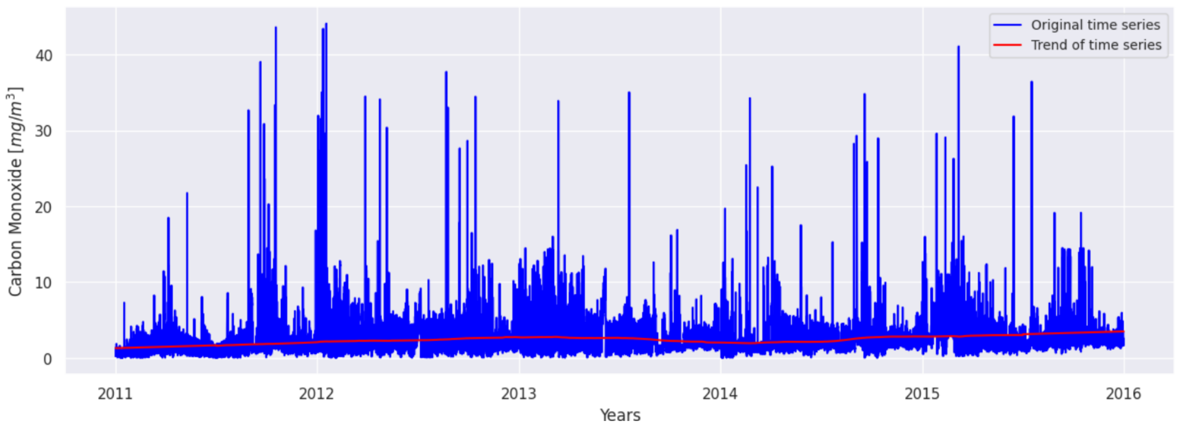


Figure 65: Original time series and trend for carbon monoxide emissions feature with additive decomposition

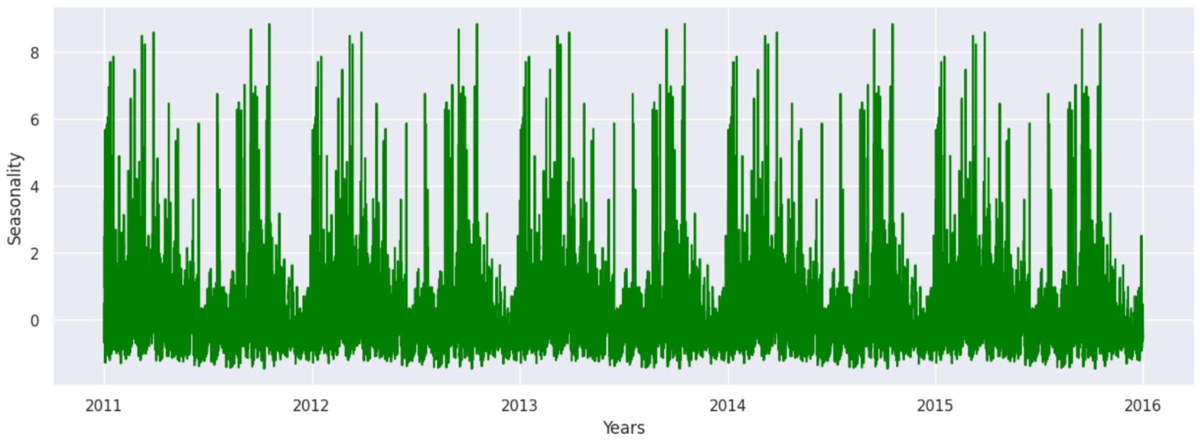


Figure 66: Seasonality plot for carbon monoxide emissions feature with additive decomposition

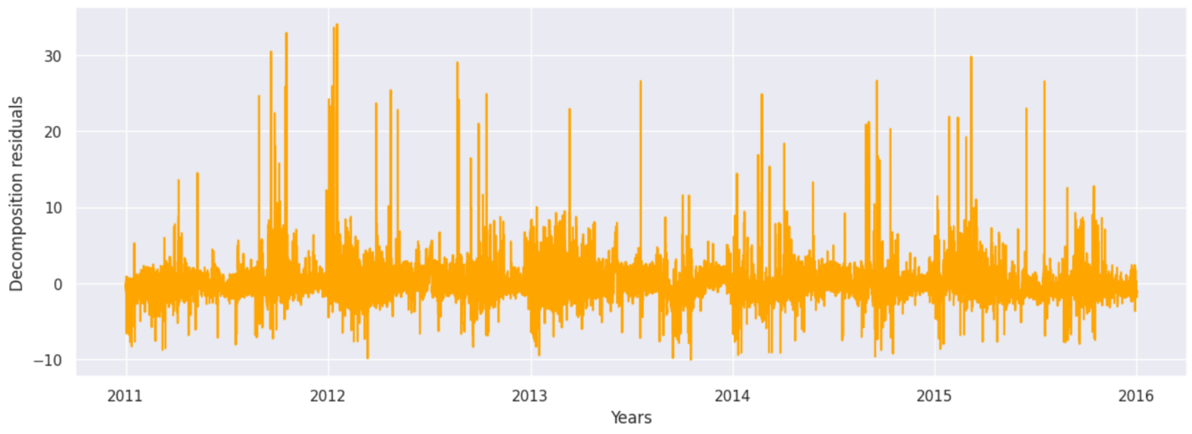


Figure 67: Decomposition residuals (noise) for carbon monoxide emissions feature with additive decomposition

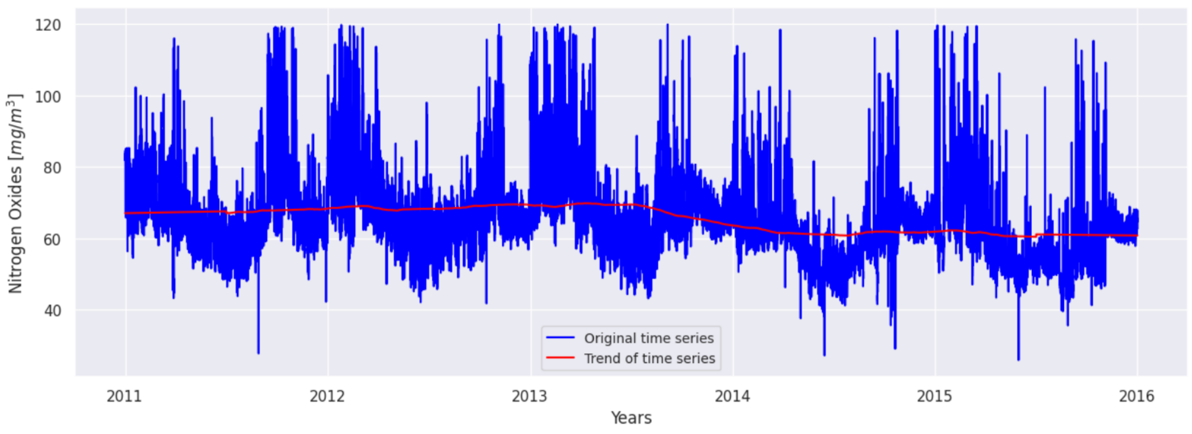


Figure 68: Original time series and trend for nitrogen oxides emissions feature with additive decomposition

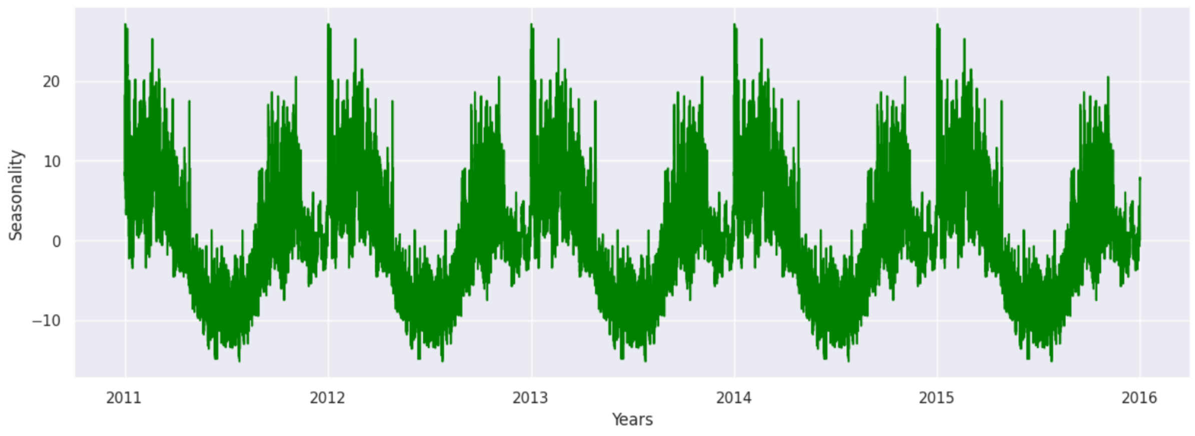


Figure 69: Seasonality plot for nitrogen oxides emissions feature with additive decomposition

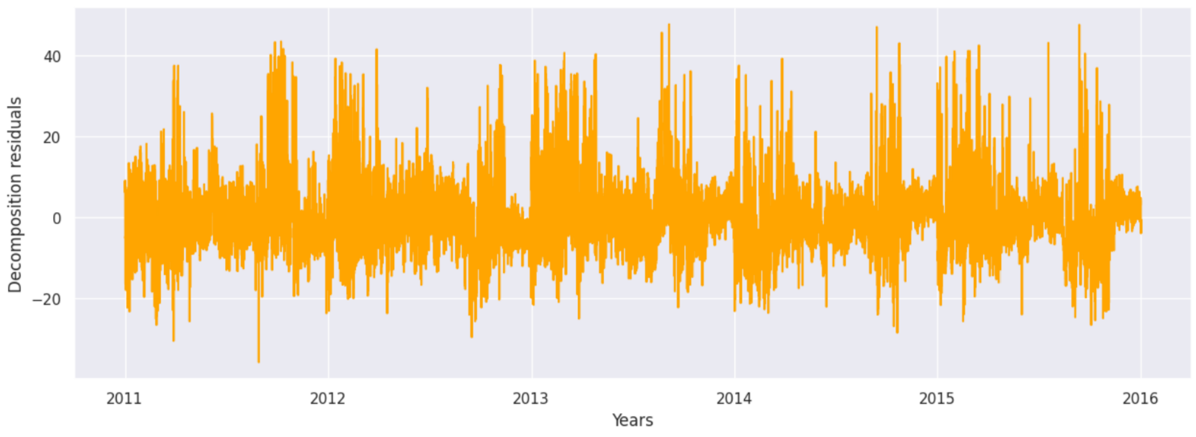


Figure 70: Decomposition residuals (noise) for nitrogen oxides emissions feature with additive decomposition

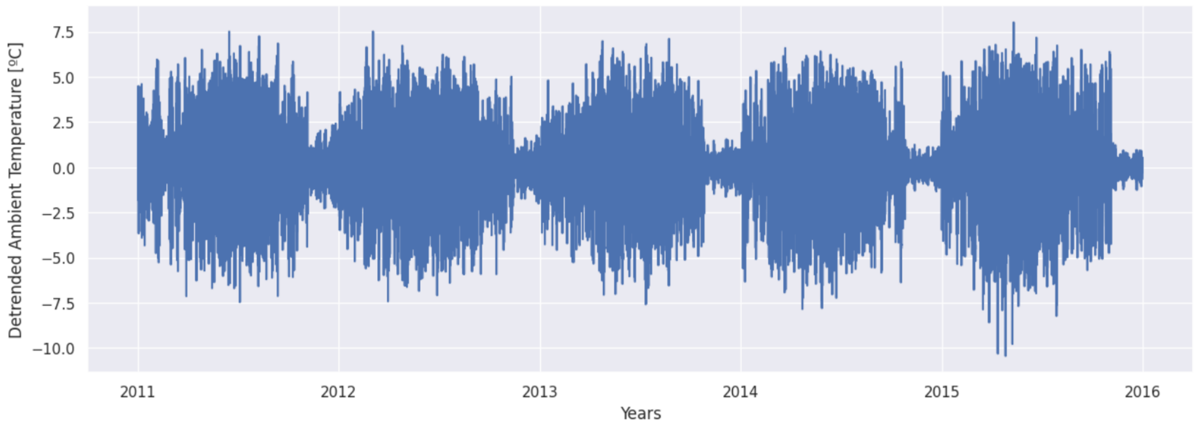


Figure 71: Detrended plot for ambient temperature feature

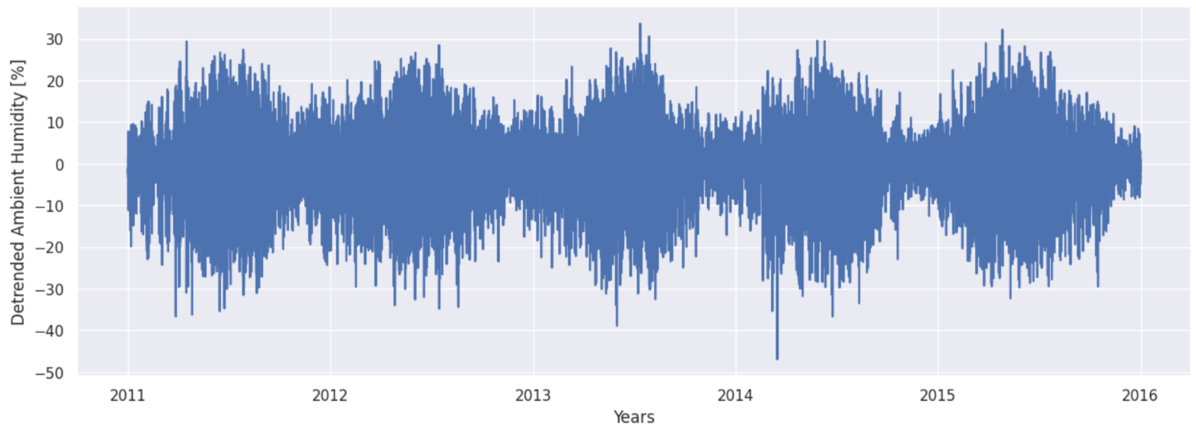


Figure 72: Detrended plot for ambient humidity feature

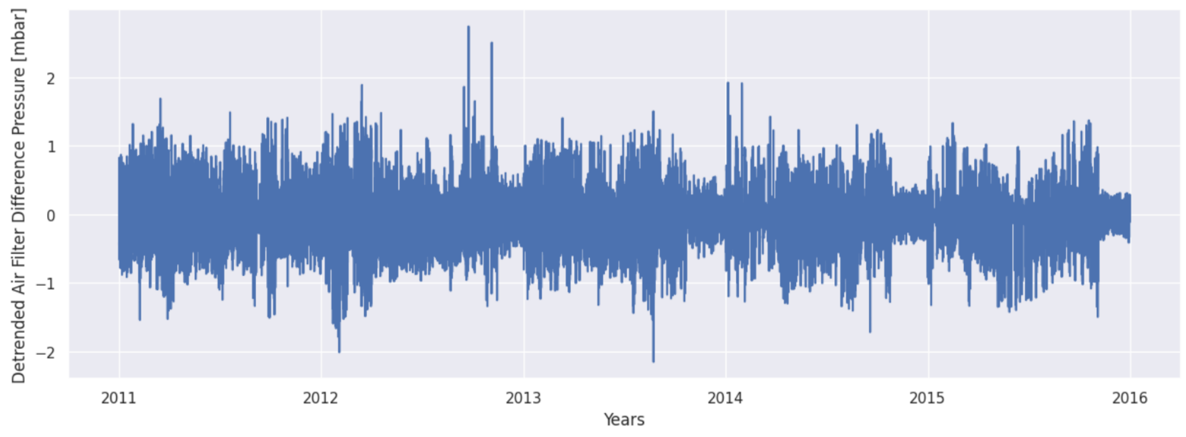


Figure 73: Detrended plot for air filter difference pressure feature

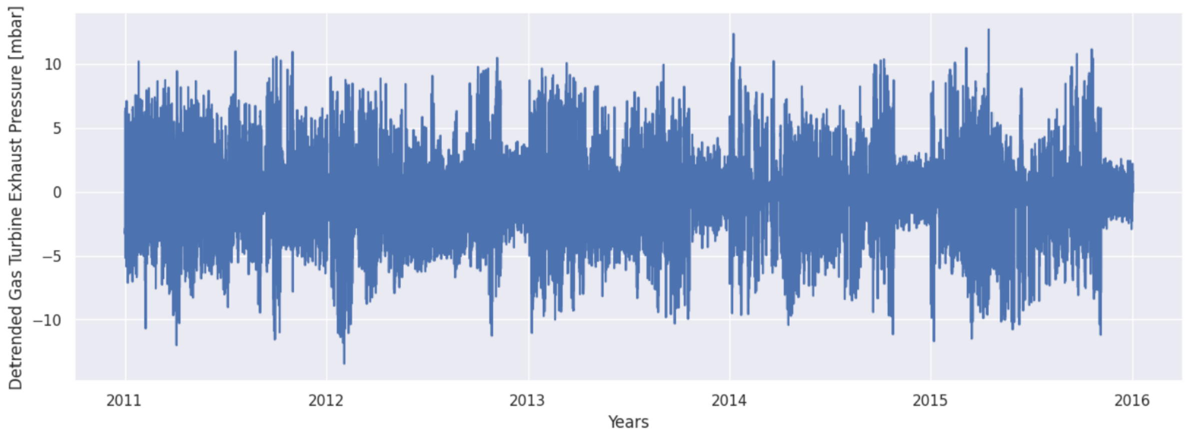


Figure 74: Detrended plot for gas turbine exhaust pressure feature

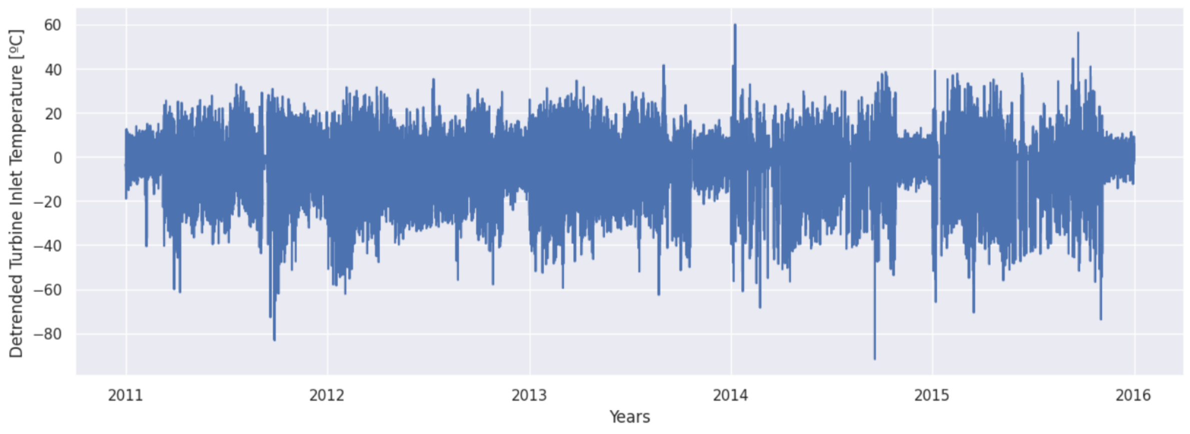


Figure 75: Detrended plot for turbine inlet temperature feature

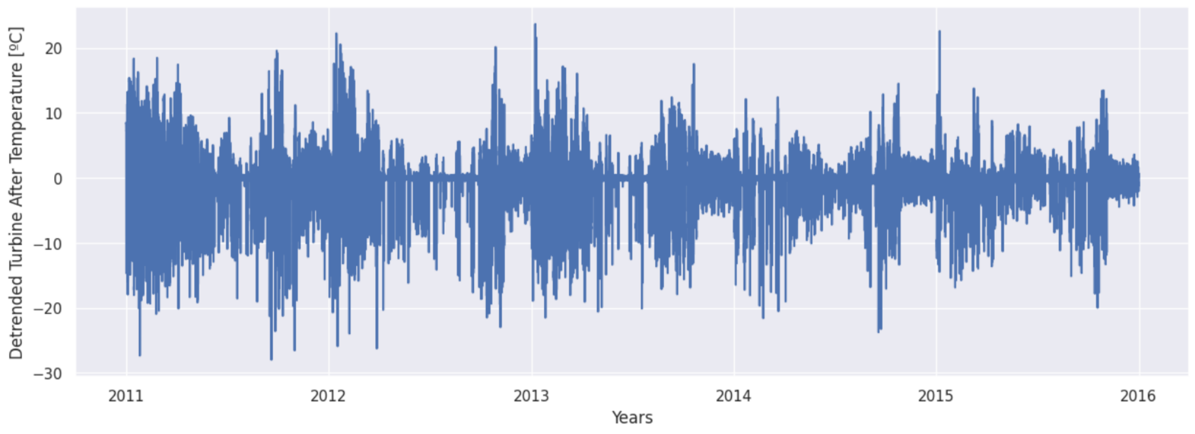


Figure 76: Detrended plot for turbine after temperature feature

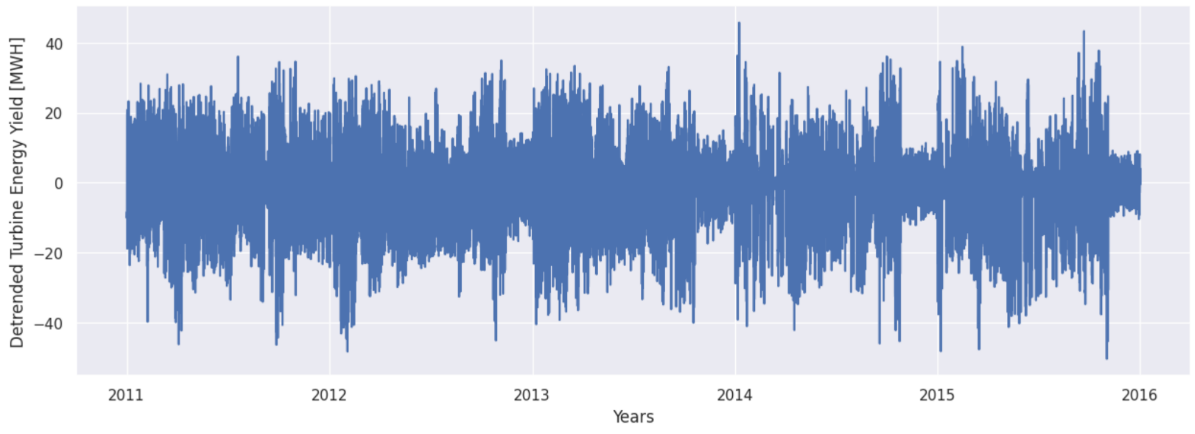


Figure 77: Detrended plot for turbine energy yield feature

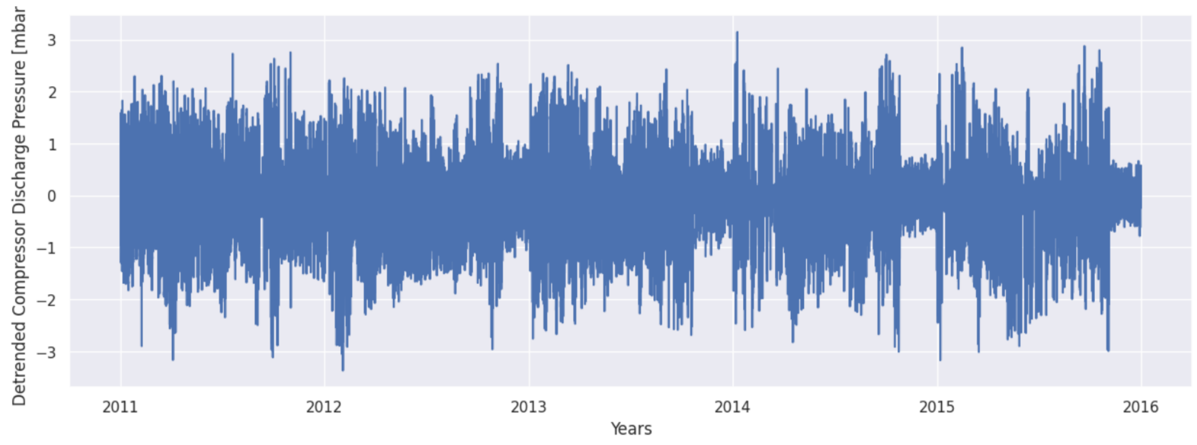


Figure 78: Detrended plot for compressor discharge pressure feature

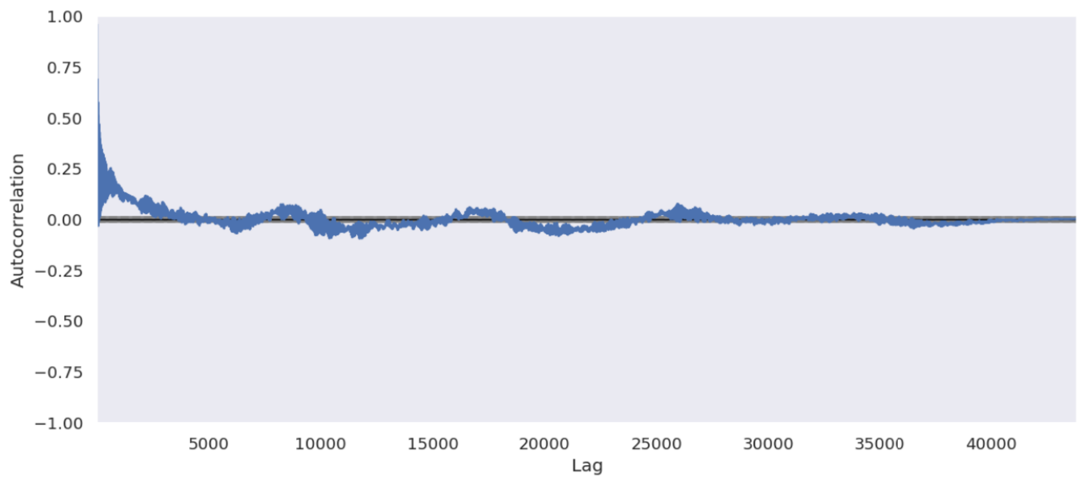


Figure 79: Seasonality plot for ambient humidity feature

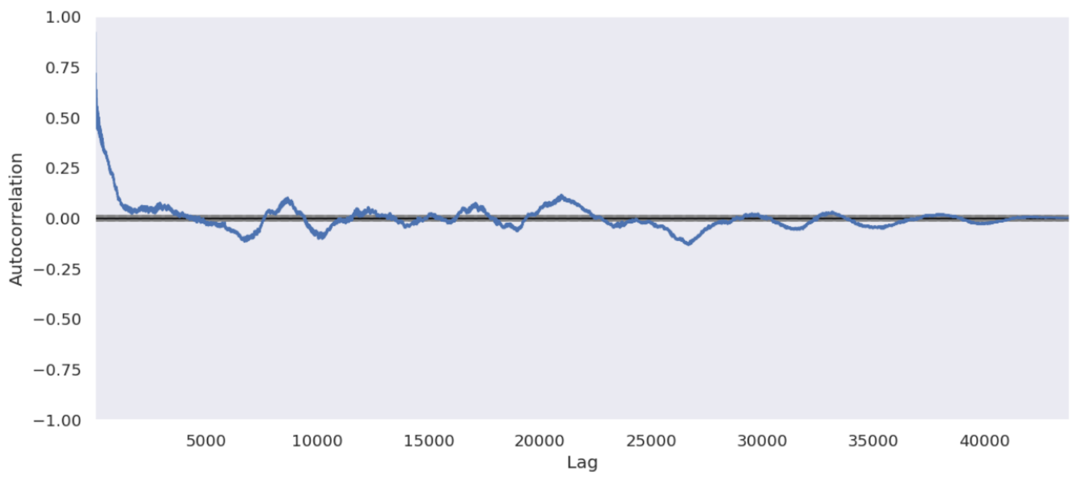


Figure 80: Seasonality plot for air filter difference pressure feature

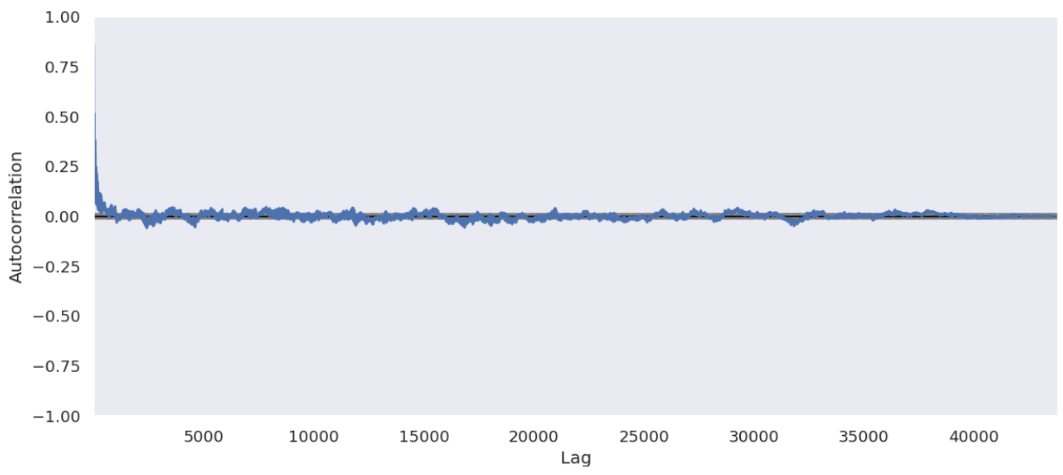


Figure 81: Seasonality plot for gas turbine exhaust pressure feature

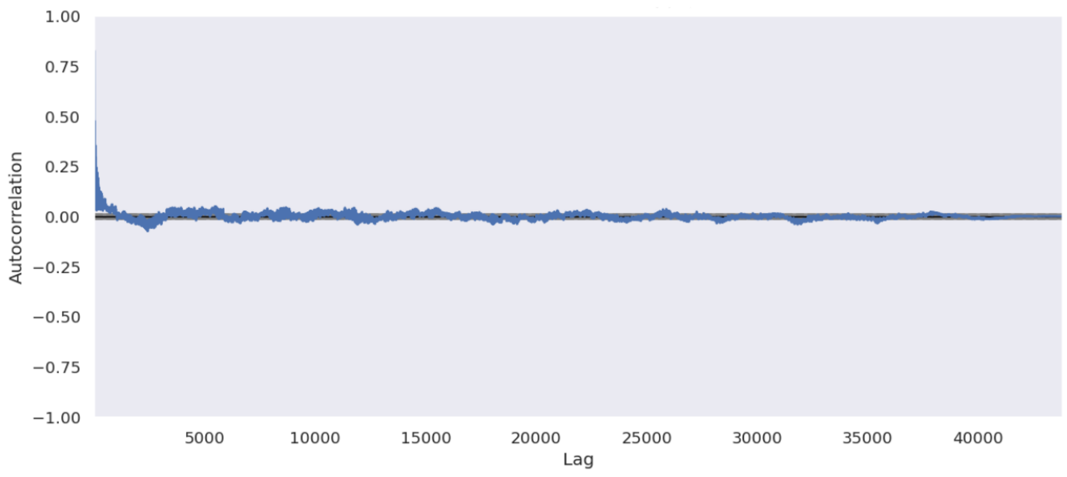


Figure 82: Seasonality plot for turbine inlet temperature feature

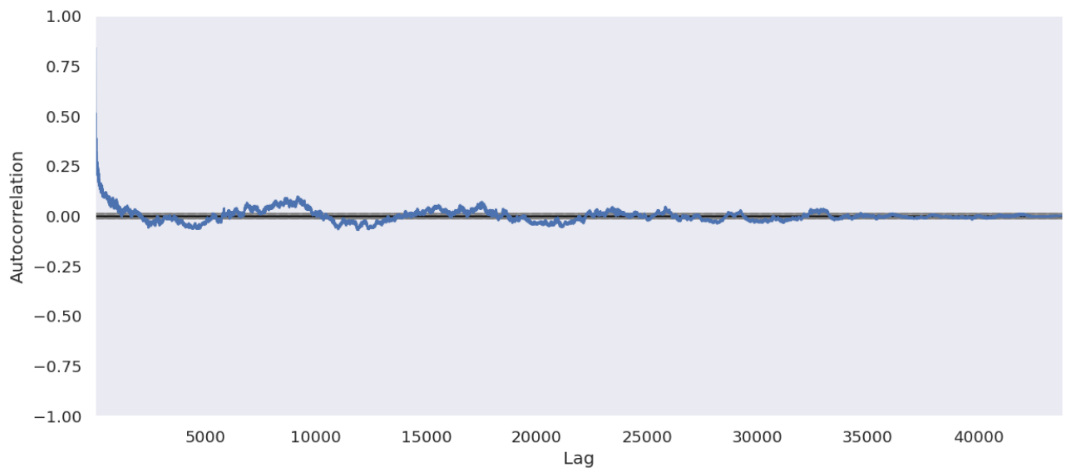


Figure 83: Seasonality plot for turbine after temperature feature

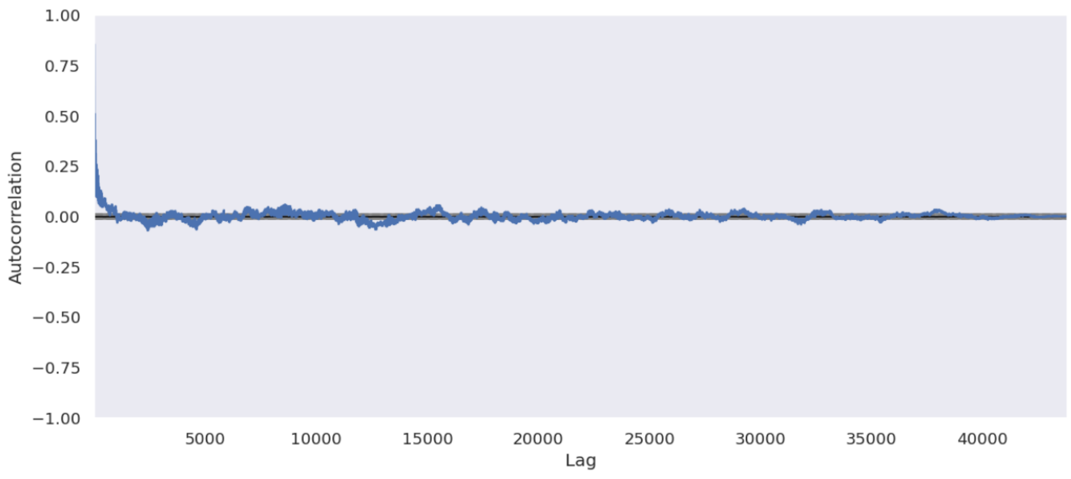


Figure 84: Seasonality plot for turbine energy yield feature

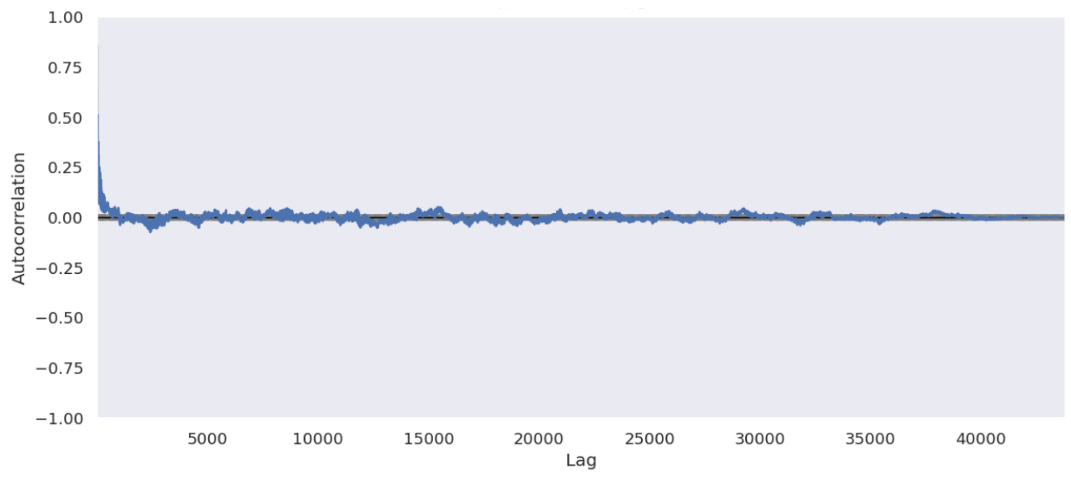


Figure 85: Seasonality plot for compressor discharge pressure feature

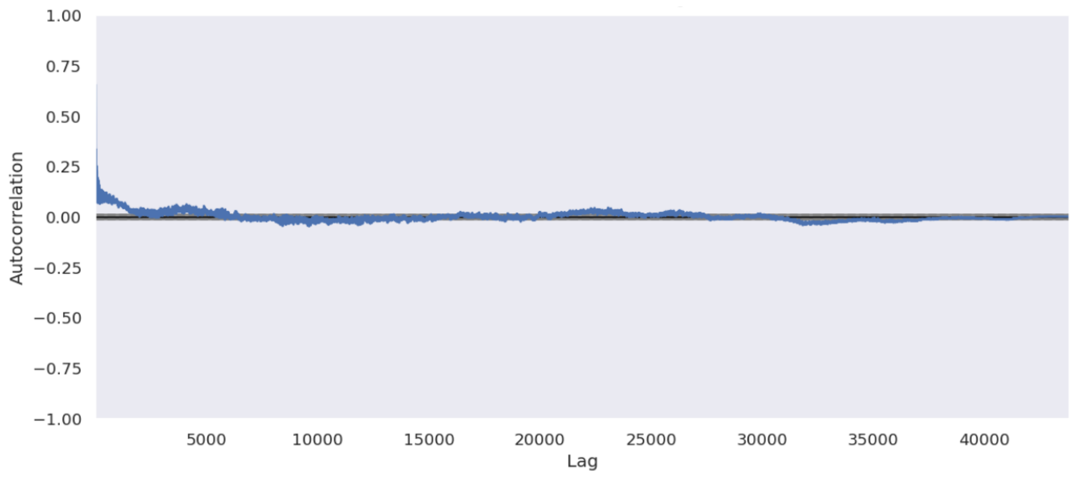


Figure 86: Seasonality plot for carbon monoxide emissions feature

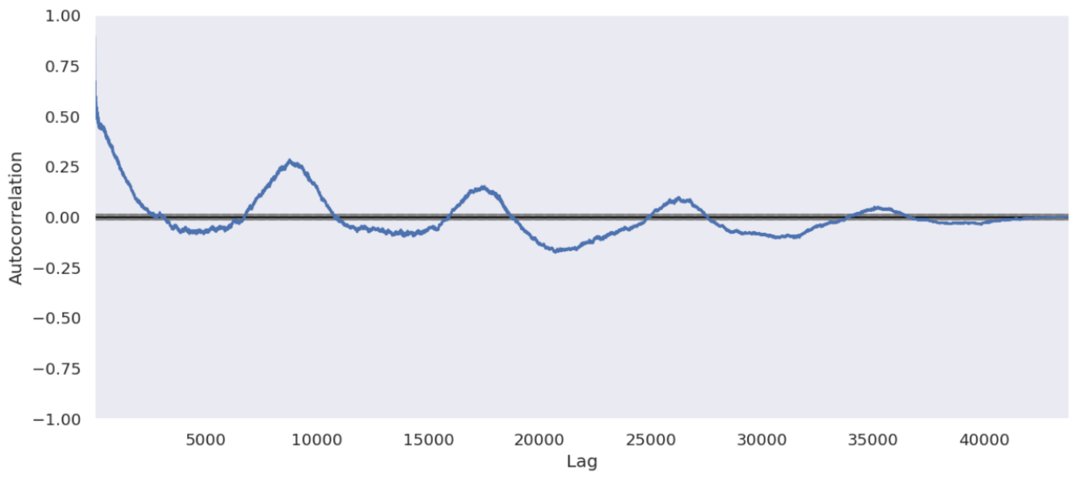


Figure 87: Seasonality plot for nitrogen oxides emissions feature

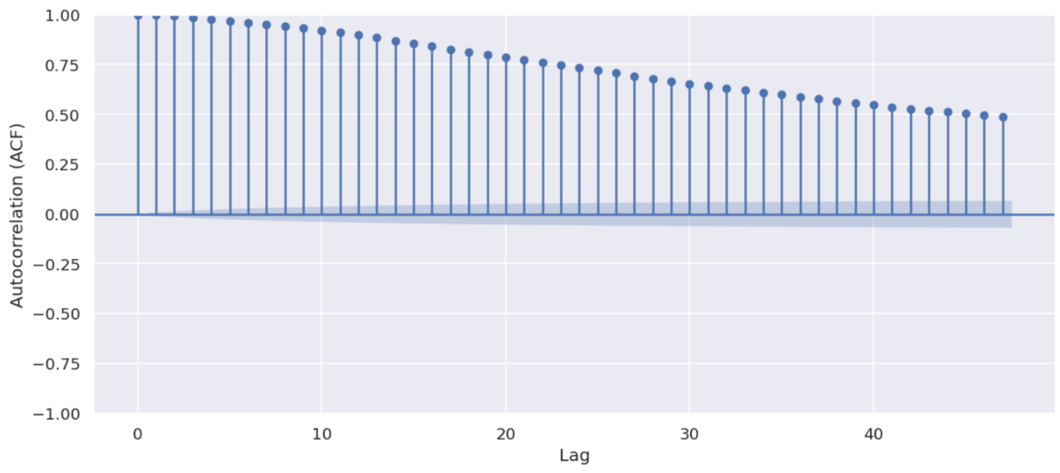


Figure 88: ACF plot for ambient pressure feature

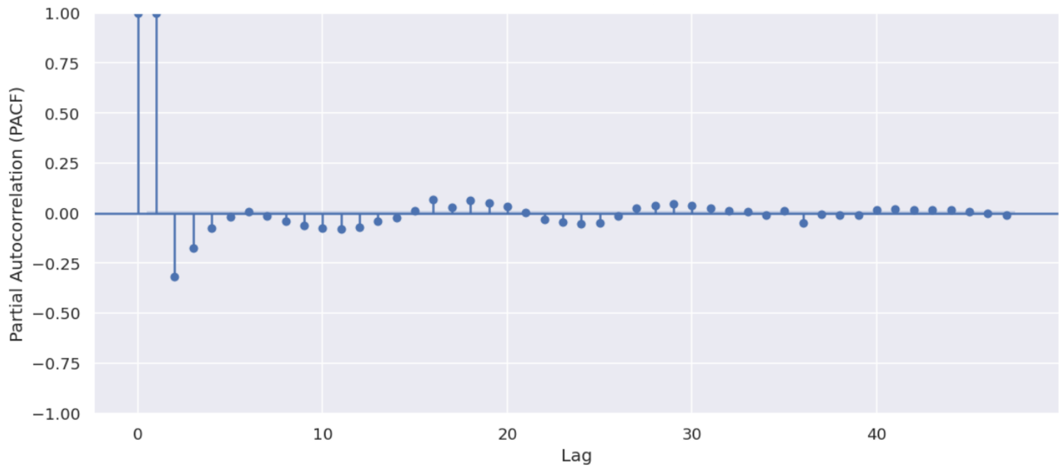


Figure 89: PACF plot for ambient pressure feature

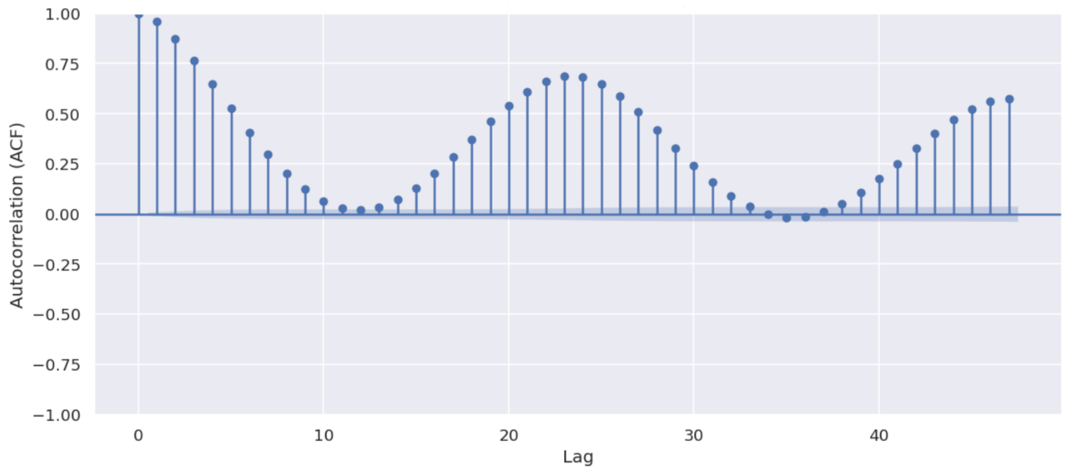


Figure 90: ACF plot for ambient humidity feature

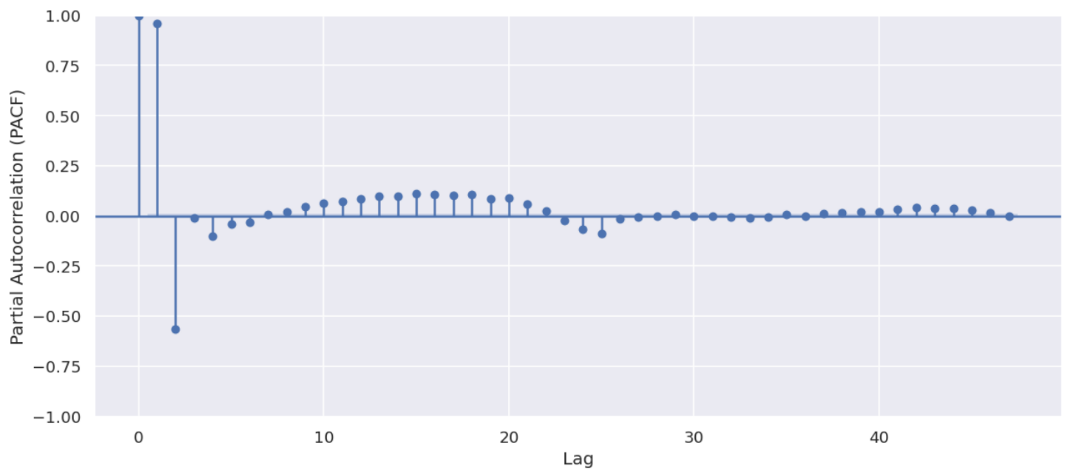


Figure 91: PACF plot for ambient humidity feature

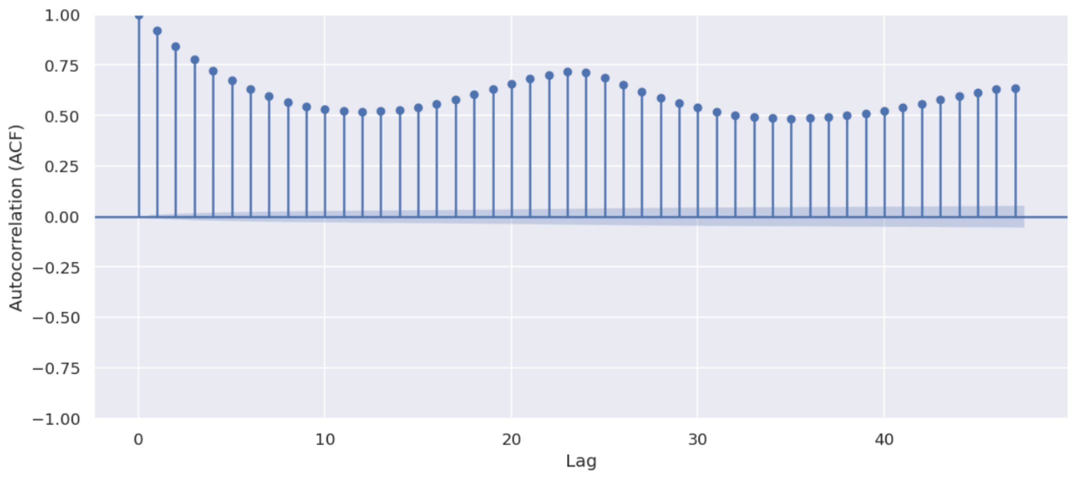


Figure 92: ACF plot for air filter difference pressure feature

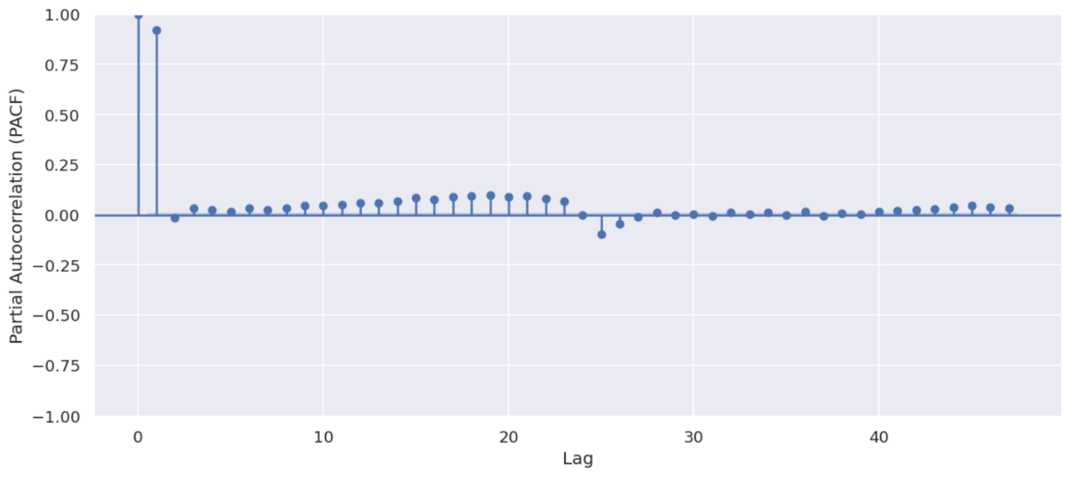


Figure 93: PACF plot for air filter difference pressure feature

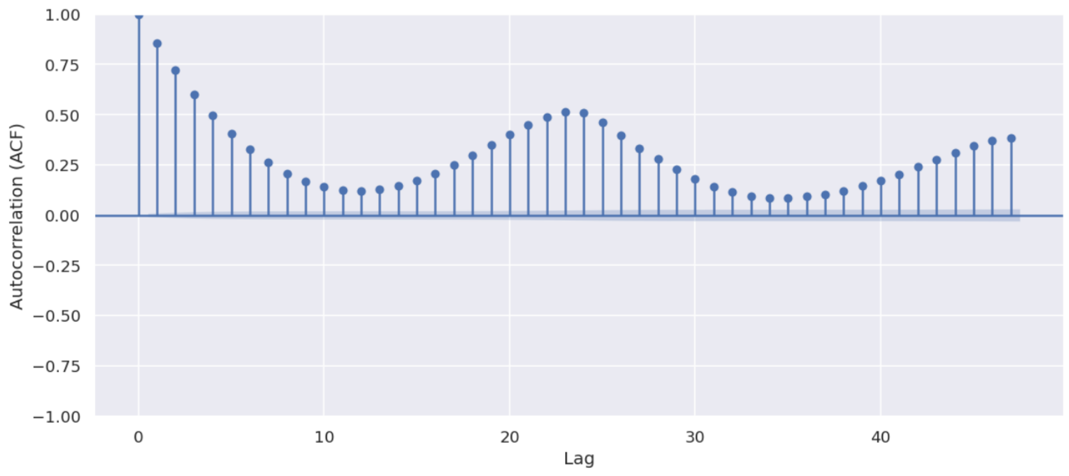


Figure 94: ACF plot for gas turbine exhaust pressure feature

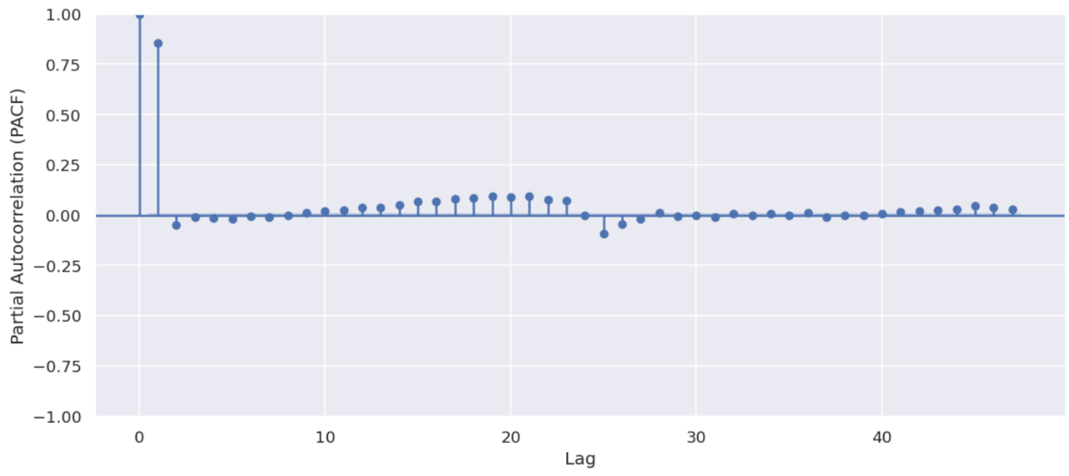


Figure 95: PACF plot for gas turbine exhaust pressure feature

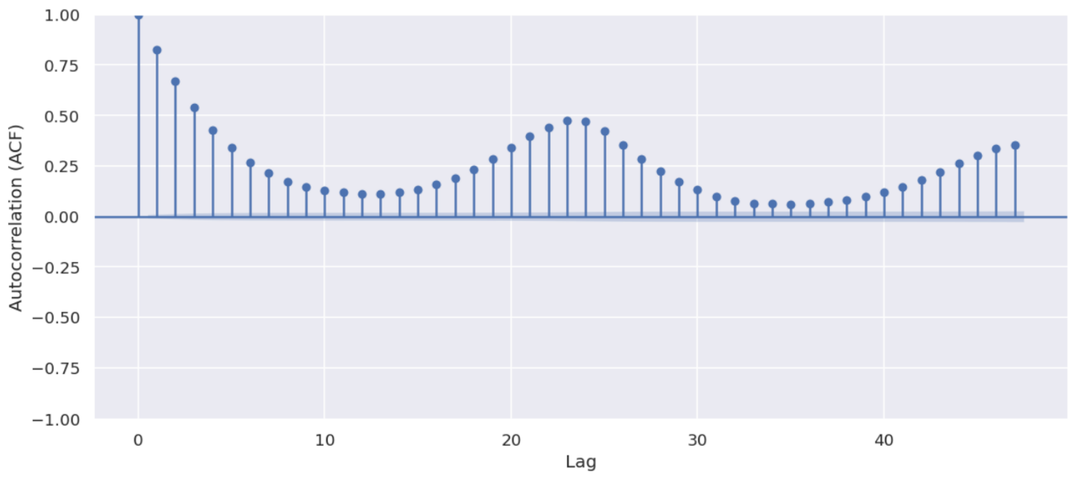


Figure 96: ACF plot for turbine inlet temperature feature

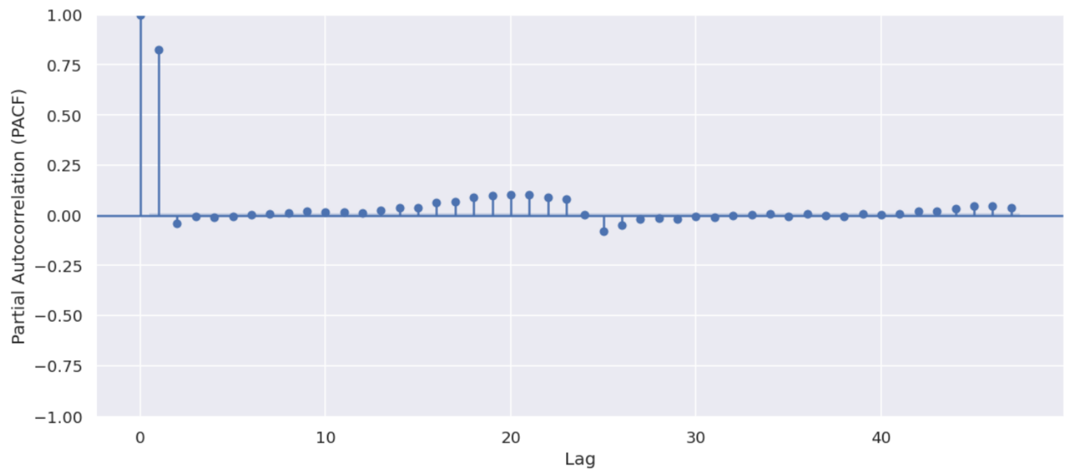


Figure 97: PACF plot for turbine inlet temperature feature

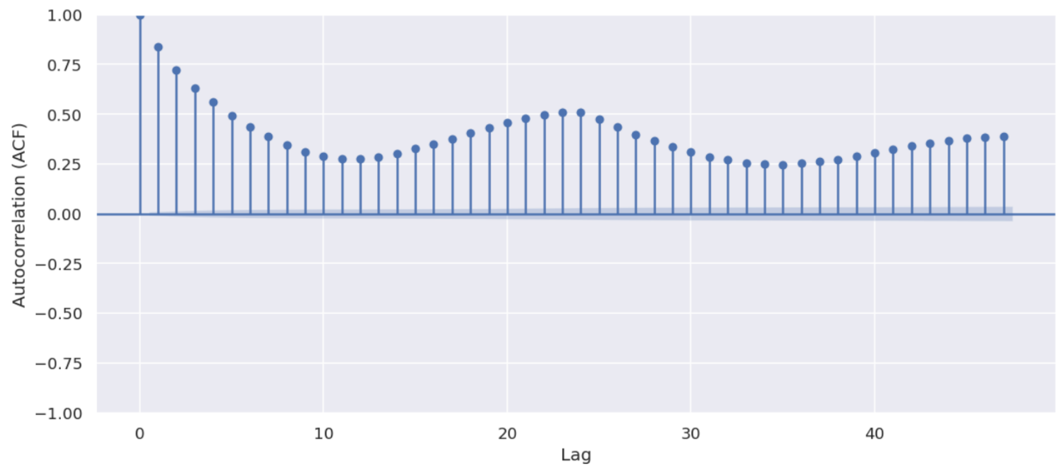


Figure 98: ACF plot for turbine after temperature feature

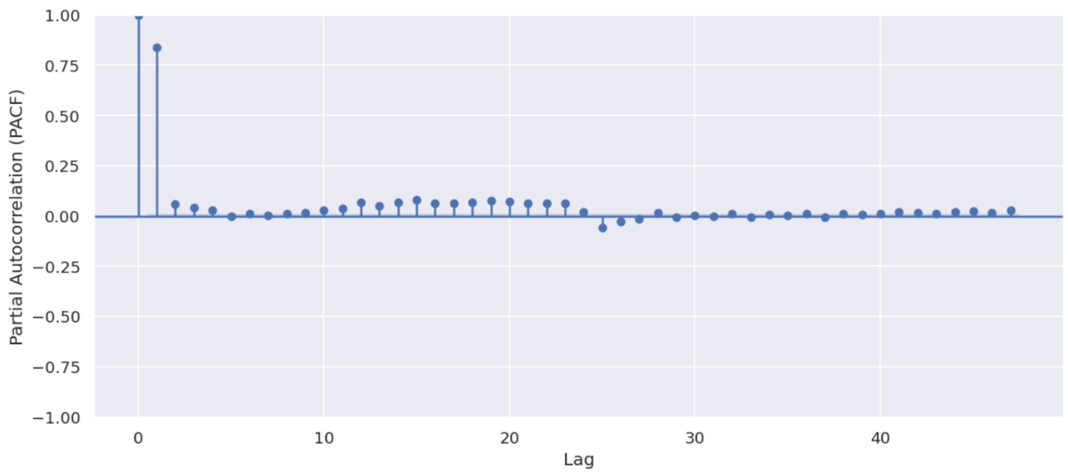


Figure 99: PACF plot for turbine after temperature feature

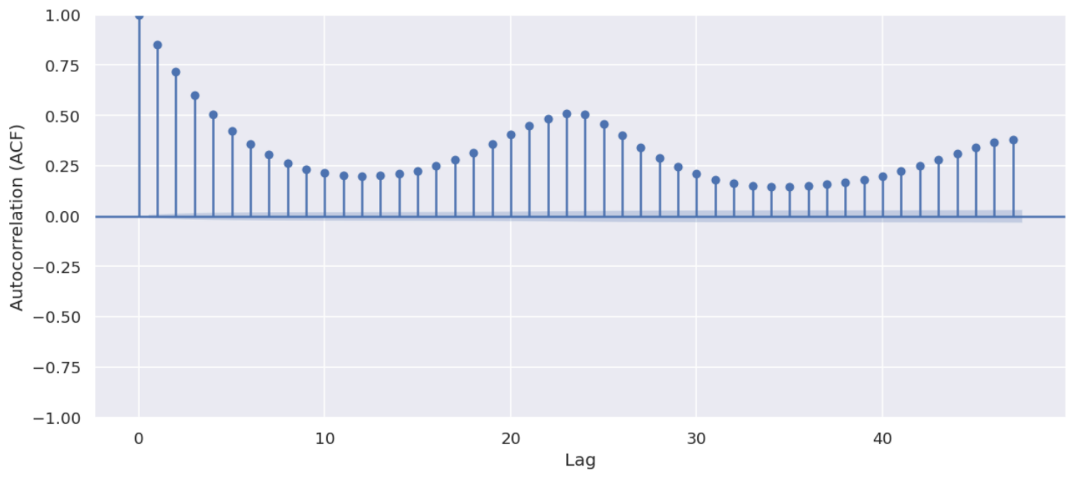


Figure 100: ACF plot for turbine energy yield feature

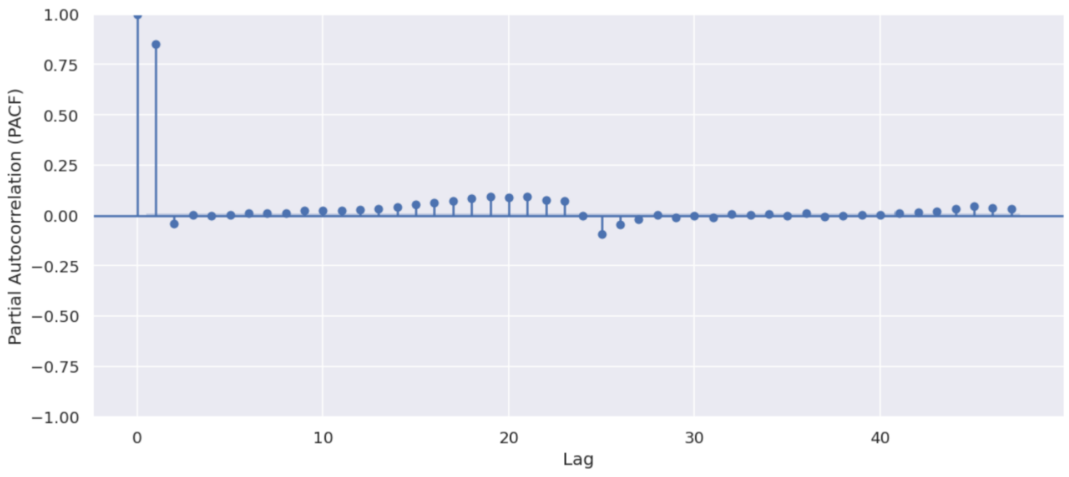


Figure 101: PACF plot for turbine energy yield feature

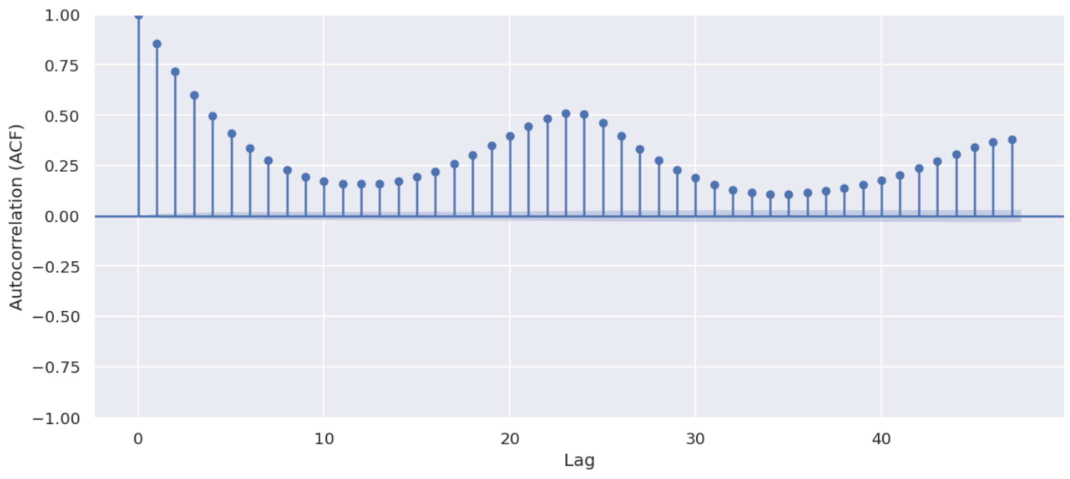


Figure 102: ACF plot for compressor discharge pressure feature

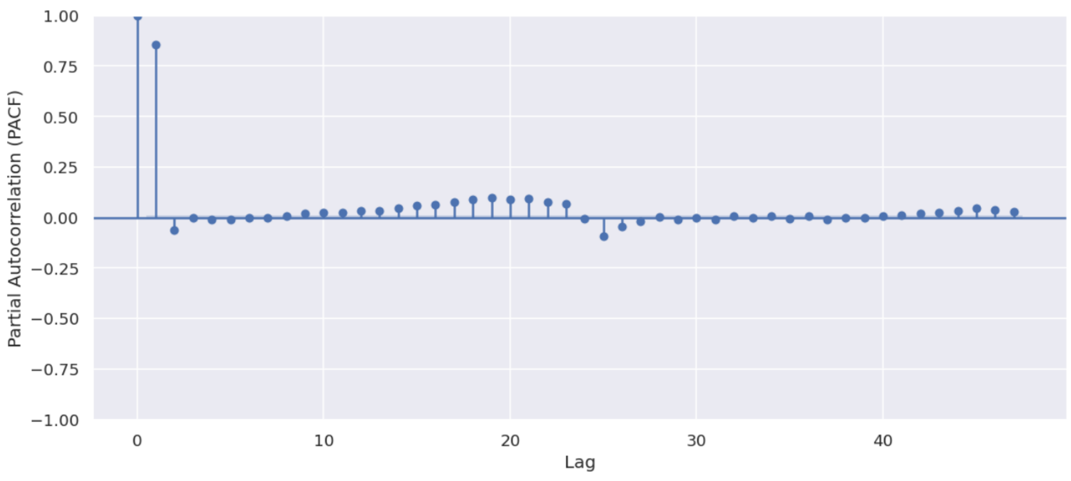


Figure 103: PACF plot for compressor discharge pressure feature

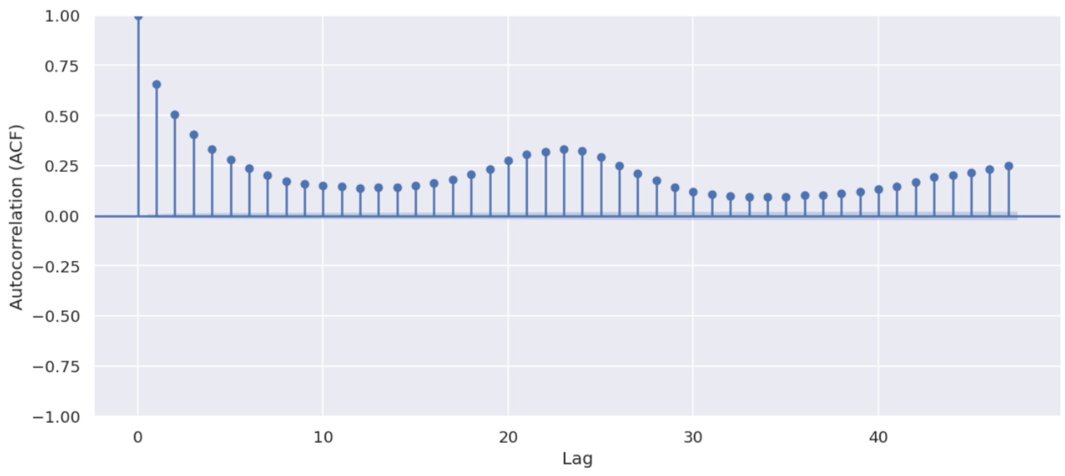


Figure 104: ACF plot for carbon monoxide emissions feature

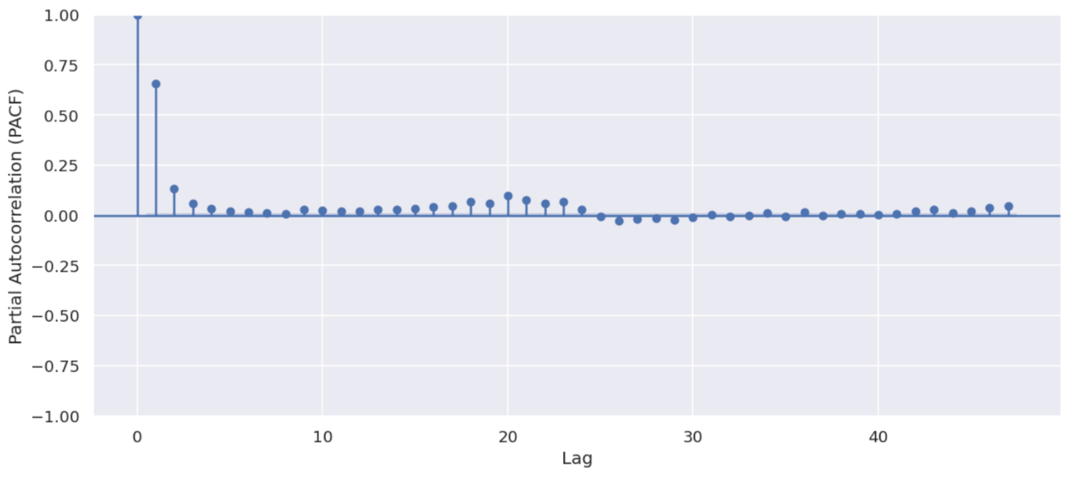


Figure 105: PACF plot for carbon monoxide emissions feature

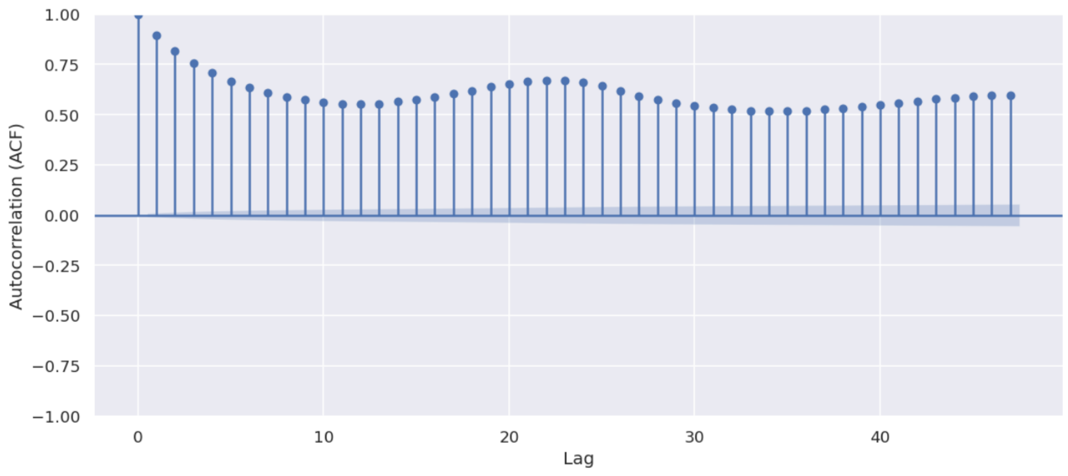


Figure 106: ACF plot for nitrogen oxides emissions feature

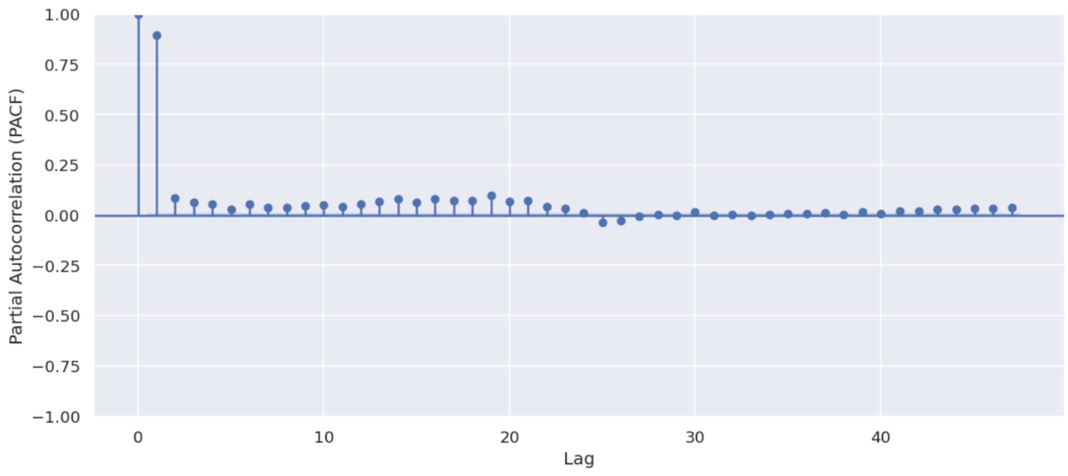


Figure 107: PACF plot for nitrogen oxides emissions feature

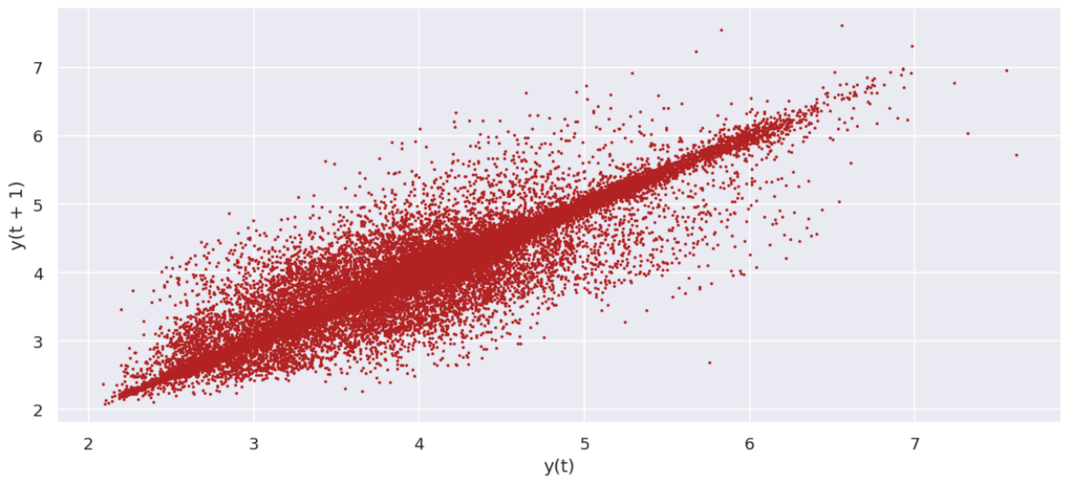


Figure 108: Lag plot for air filter difference pressure feature

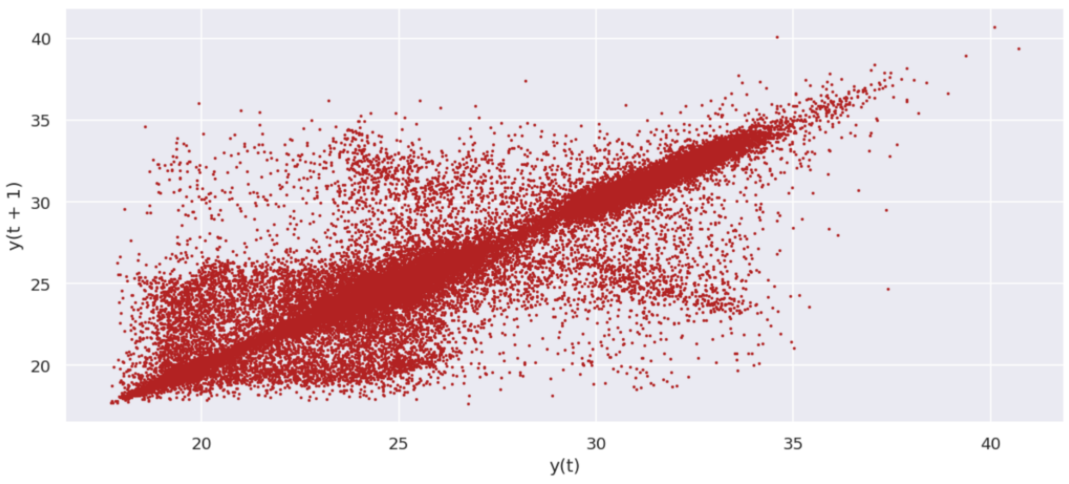


Figure 109: Lag plot for gas turbine exhaust pressure feature

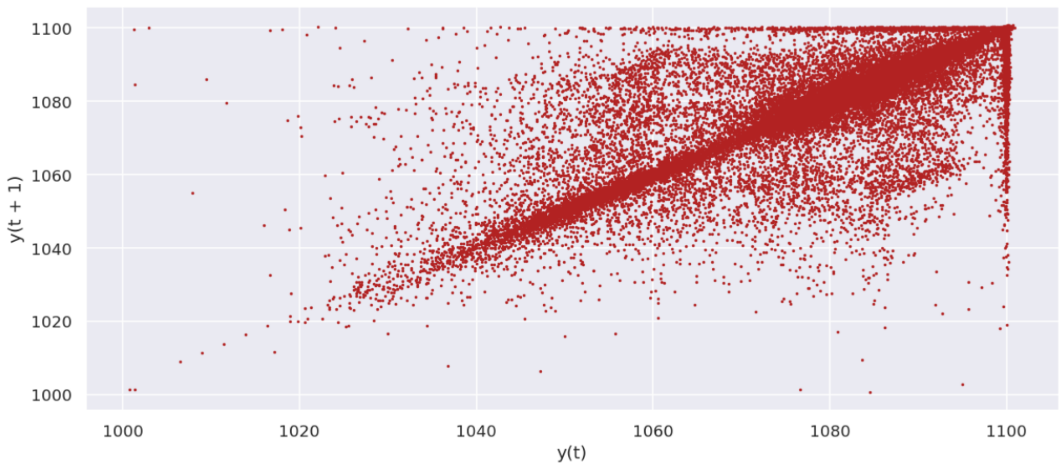


Figure 110: Lag plot for turbine inlet temperature feature

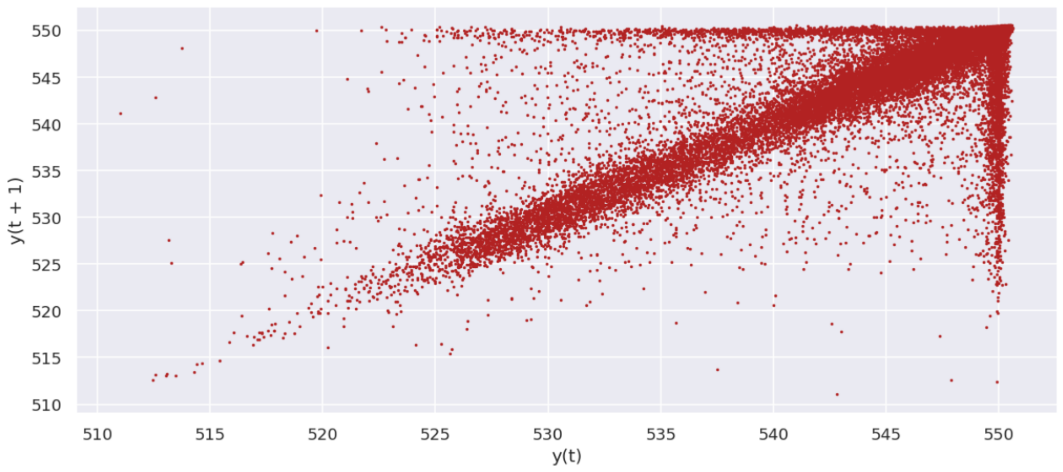


Figure 111: Lag plot for turbine after temperature feature

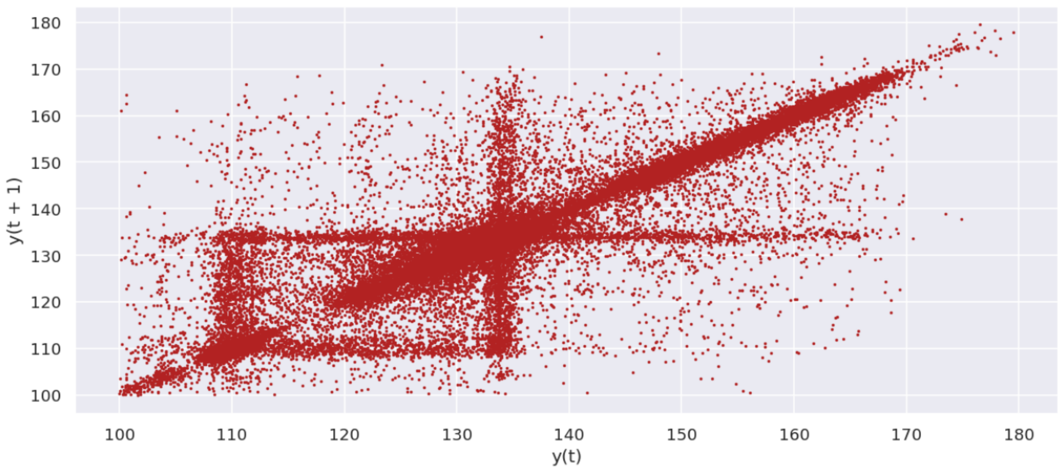


Figure 112: Lag plot for turbine energy yield feature

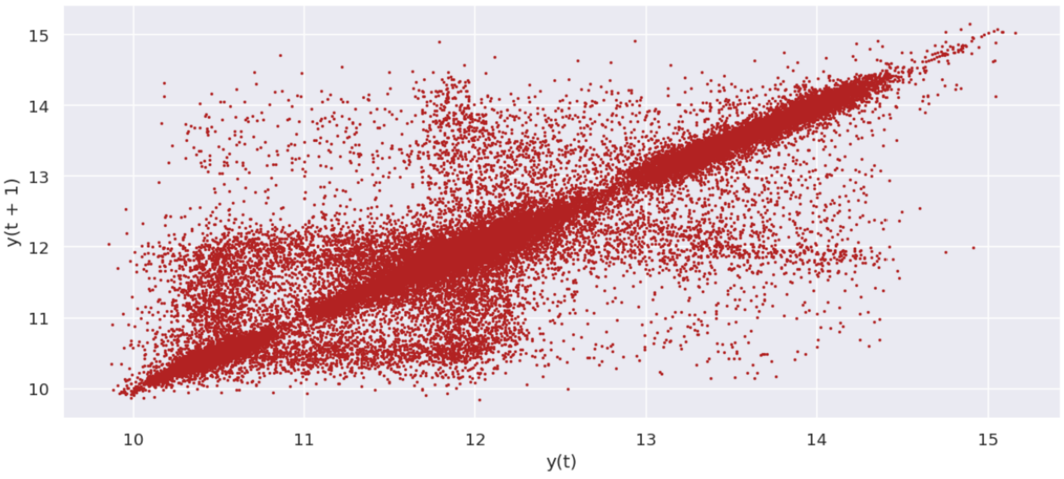


Figure 113: Lag plot for compressor discharge pressure feature

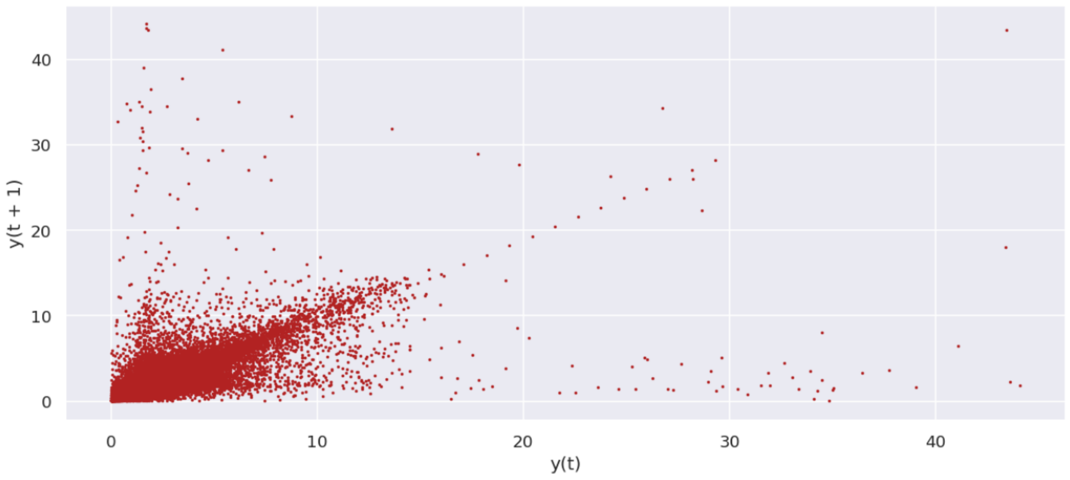


Figure 114: Lag plot for carbon monoxide emissions feature

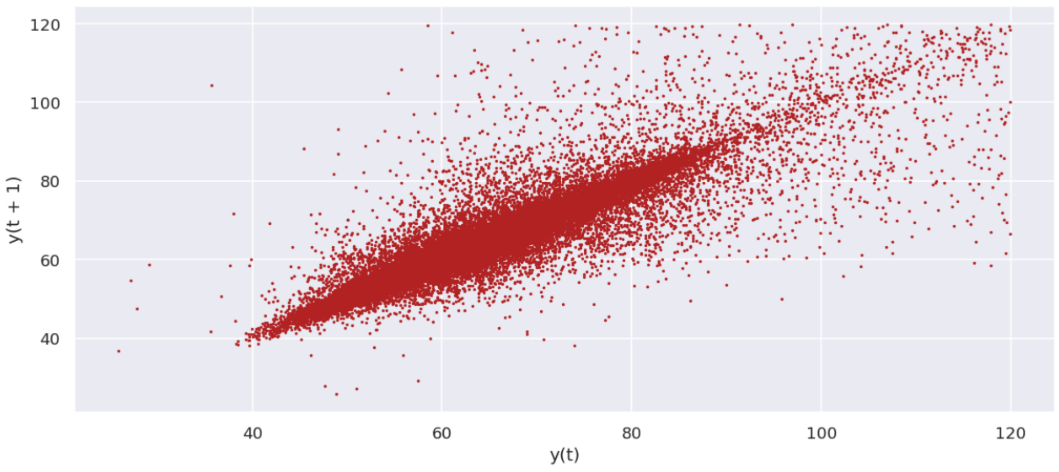


Figure 115: Lag plot for nitrogen oxides emissions feature

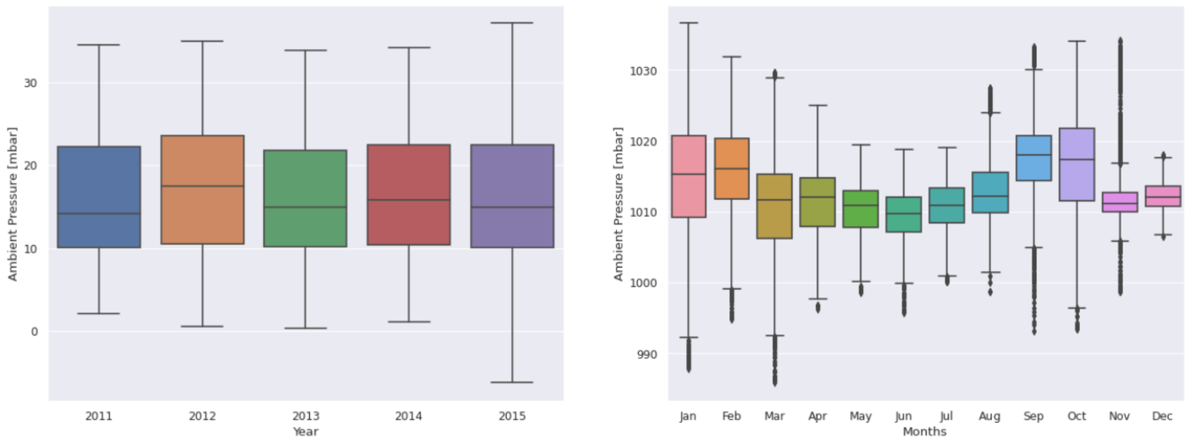


Figure 116: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for ambient pressure feature

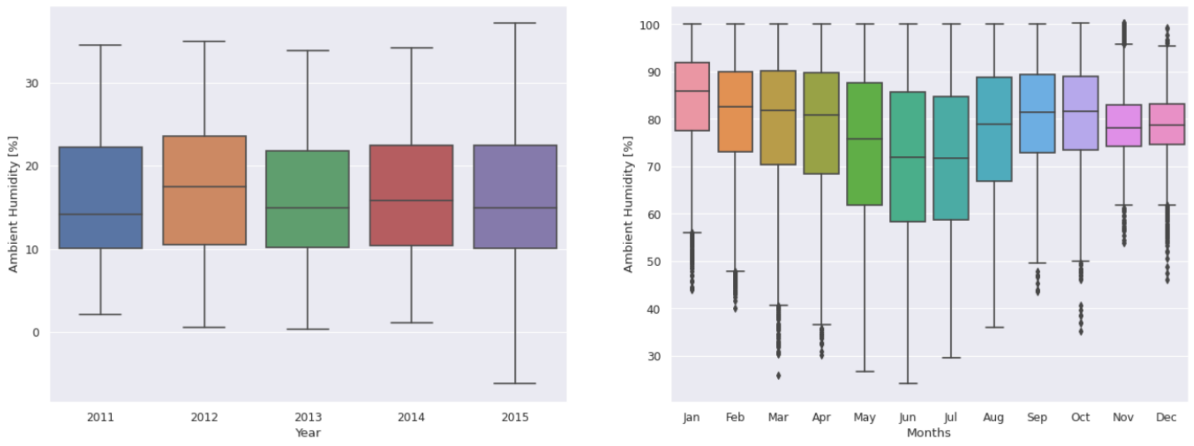


Figure 117: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for ambient humidity feature

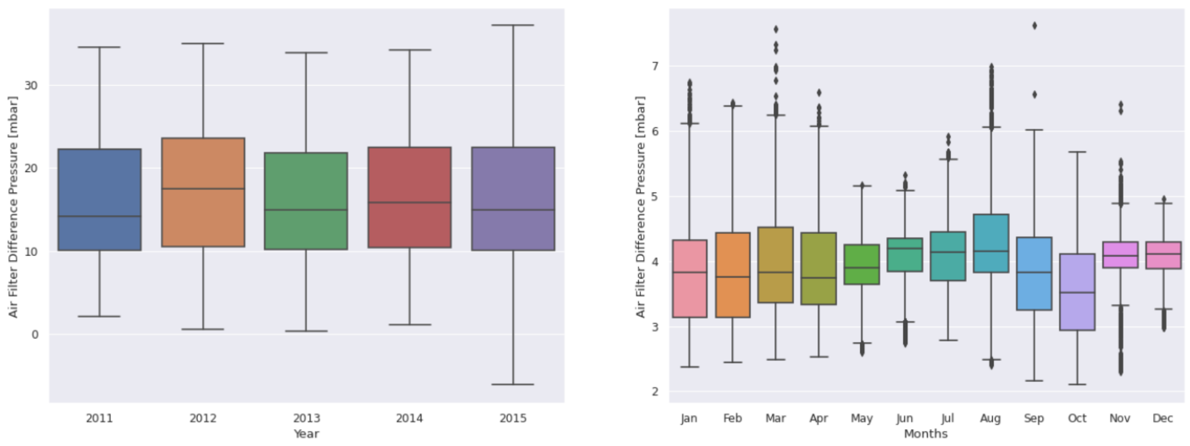


Figure 118: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for air filter difference pressure feature

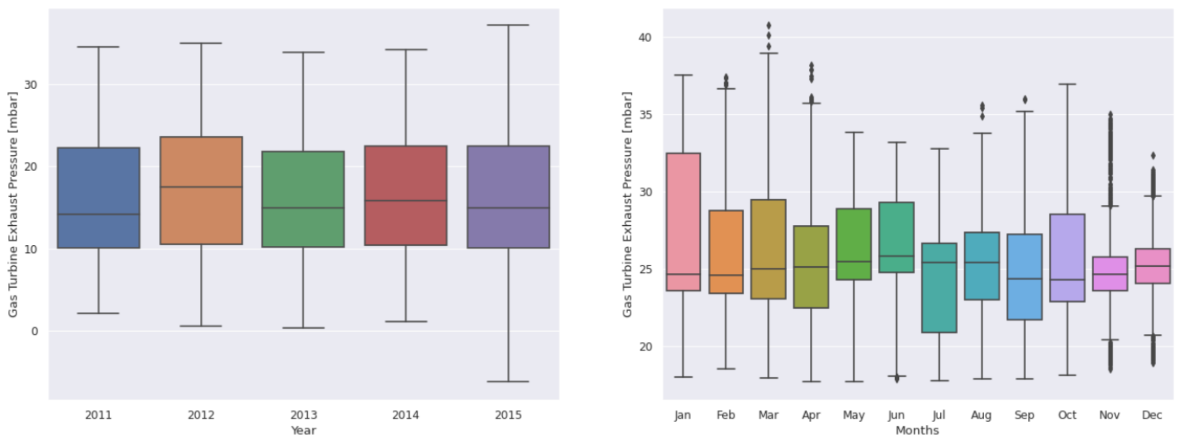


Figure 119: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for gas turbine exhaust pressure feature

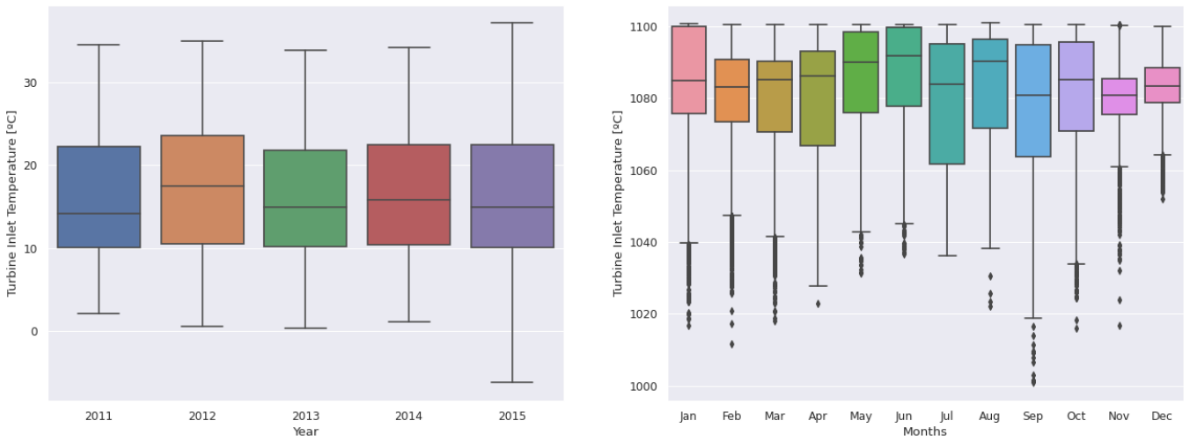


Figure 120: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for turbine inlet temperature feature

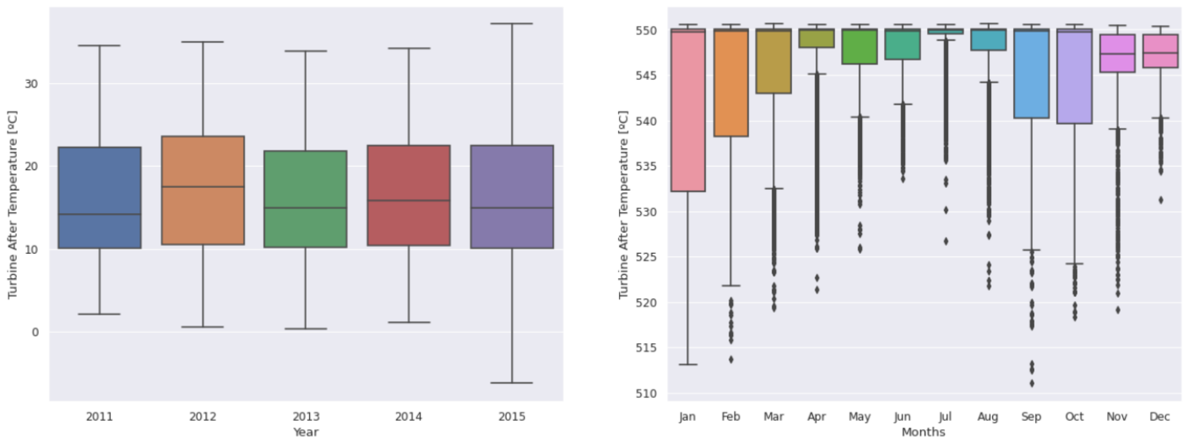


Figure 121: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for turbine after temperature feature

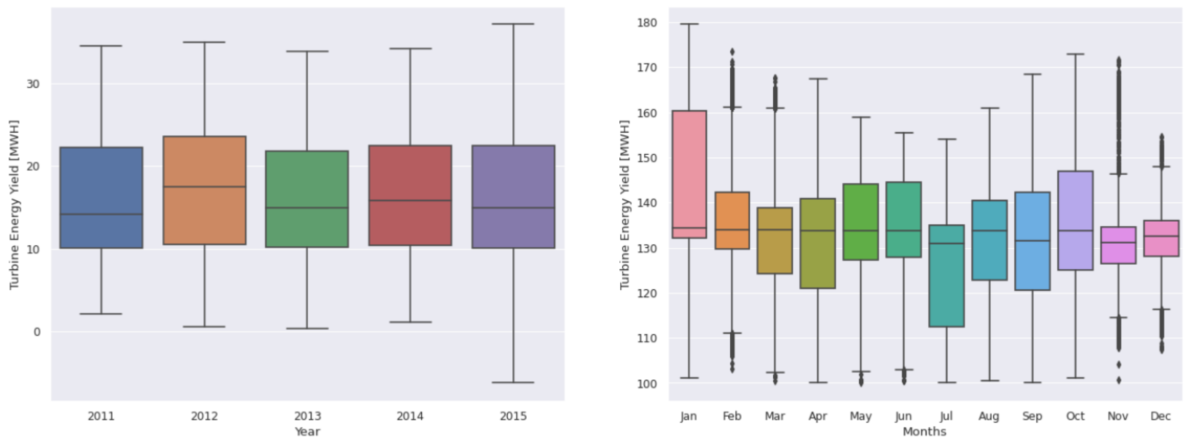


Figure 122: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for turbine energy yield feature

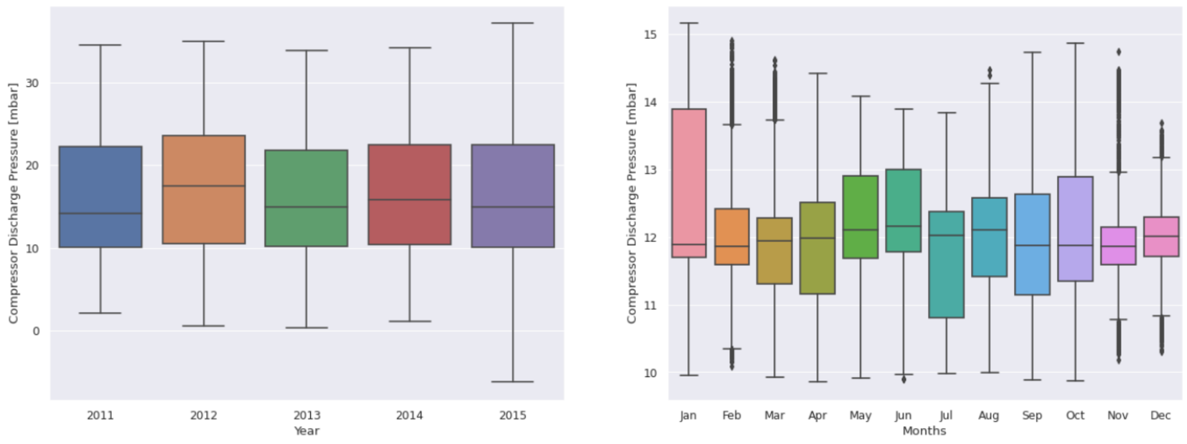


Figure 123: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for compressor discharge pressure feature

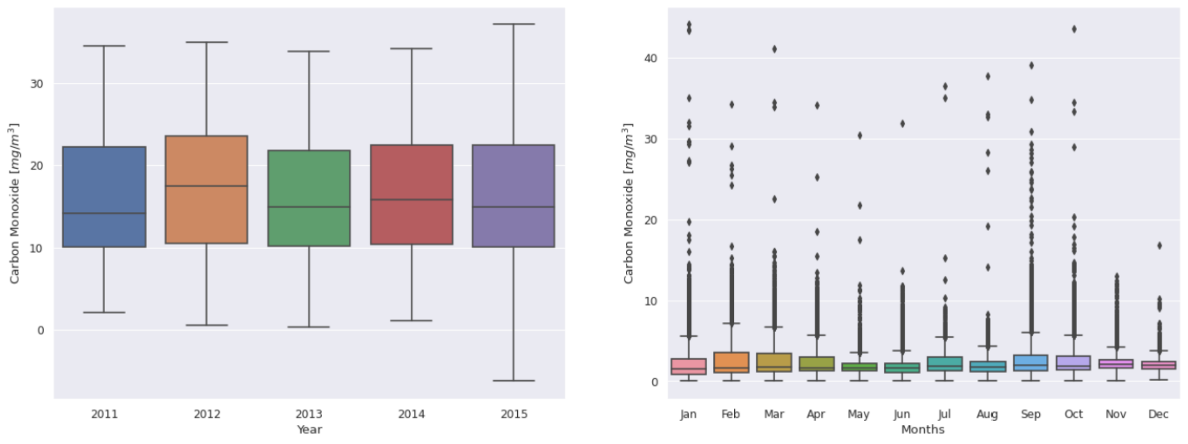


Figure 124: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for carbon monoxide emissions feature

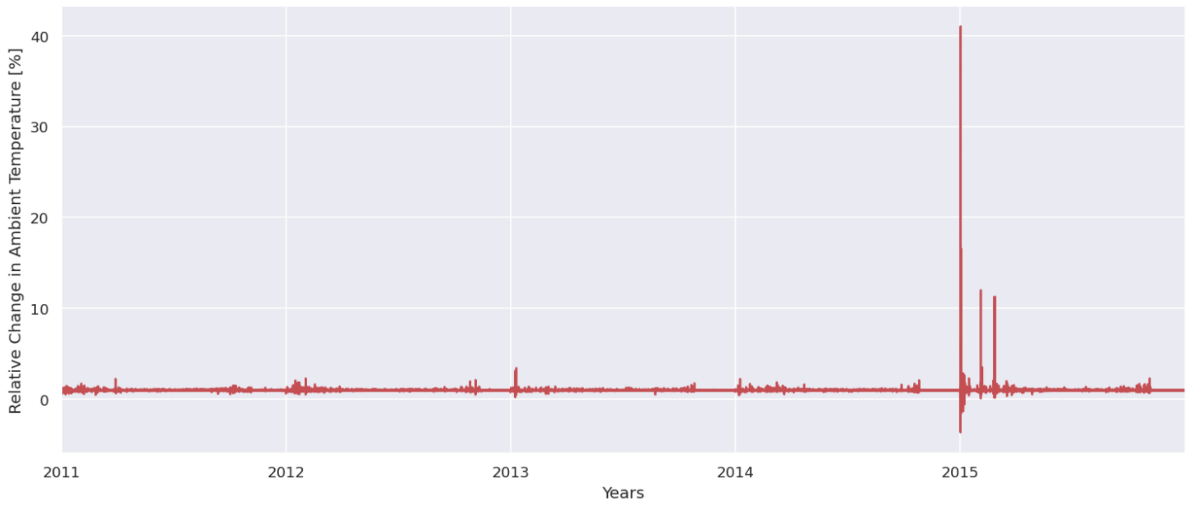


Figure 125: Relative change for ambient temperature feature over the years

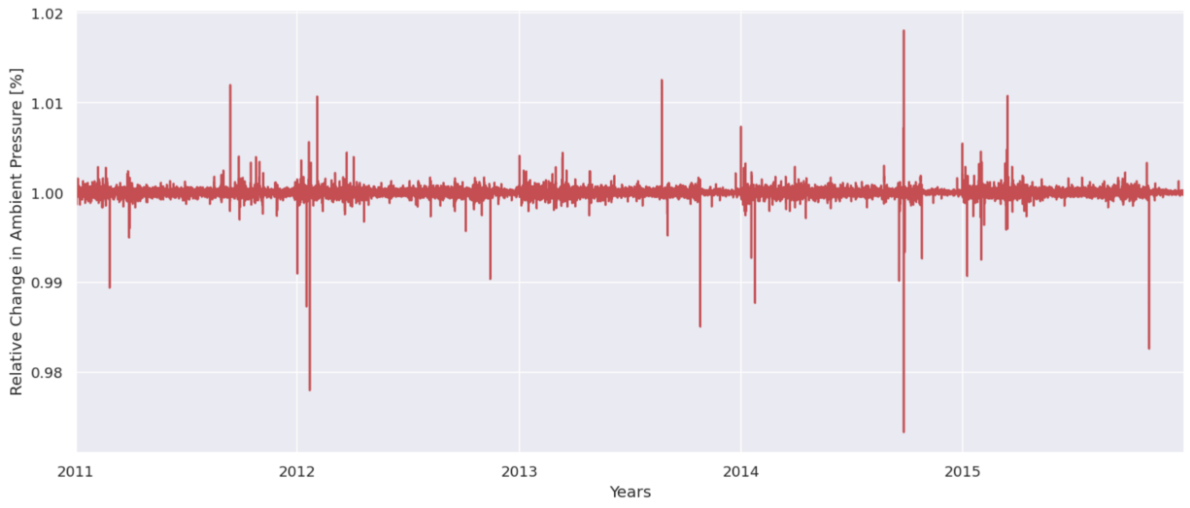


Figure 126: Relative change for ambient pressure feature over the years

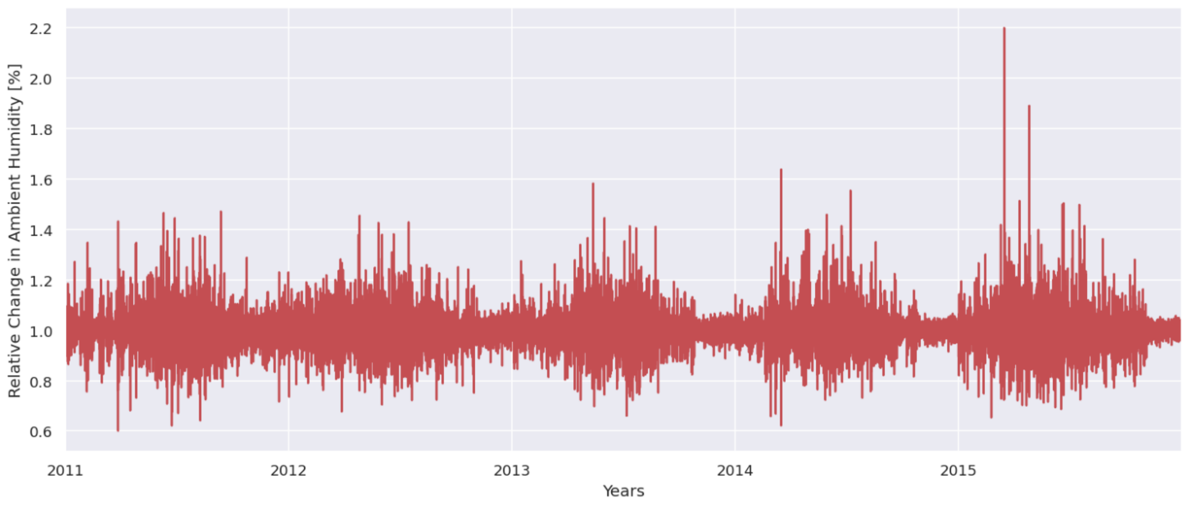


Figure 127: Relative change for ambient humidity feature over the years

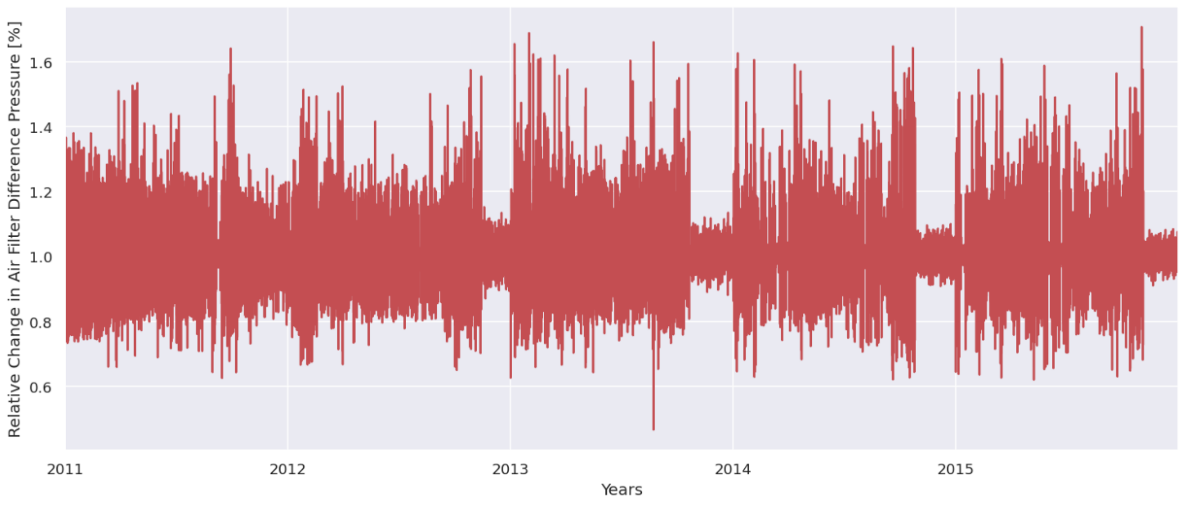


Figure 128: Relative change for air filter difference pressure feature over the years

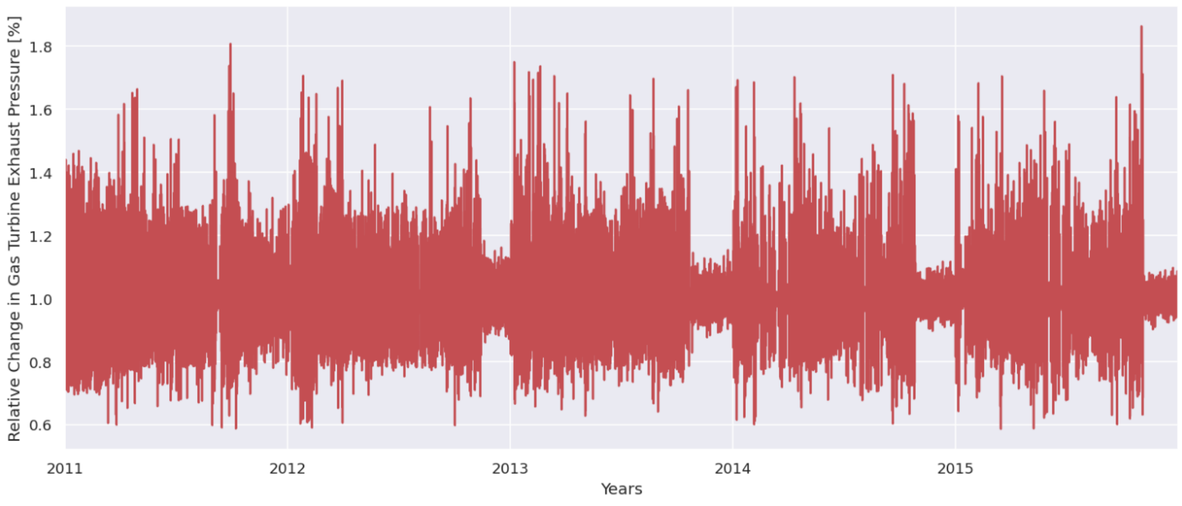


Figure 129: Relative change for gas turbine exhaust pressure feature over the years

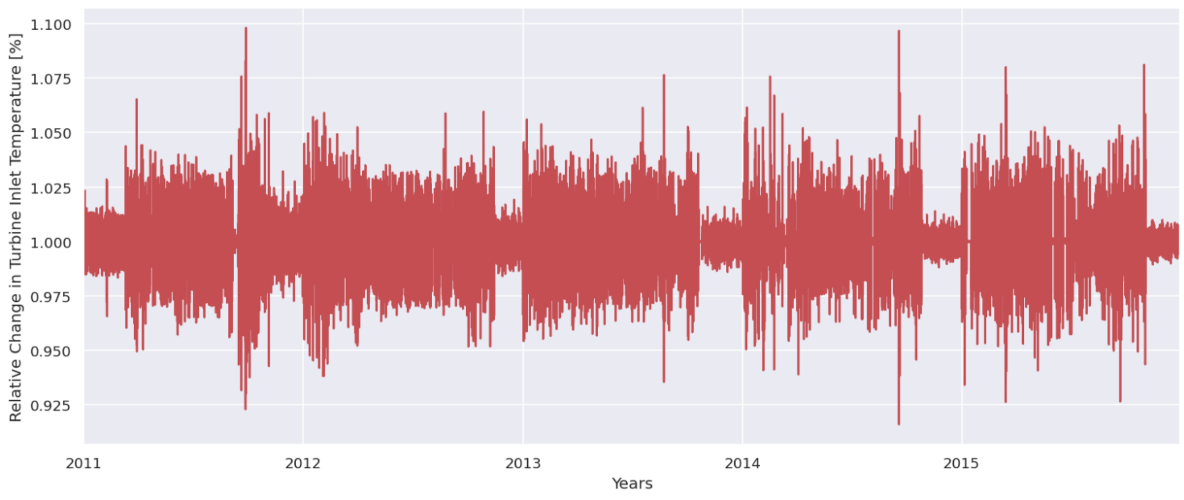


Figure 130: Relative change for turbine inlet temperature feature over the years

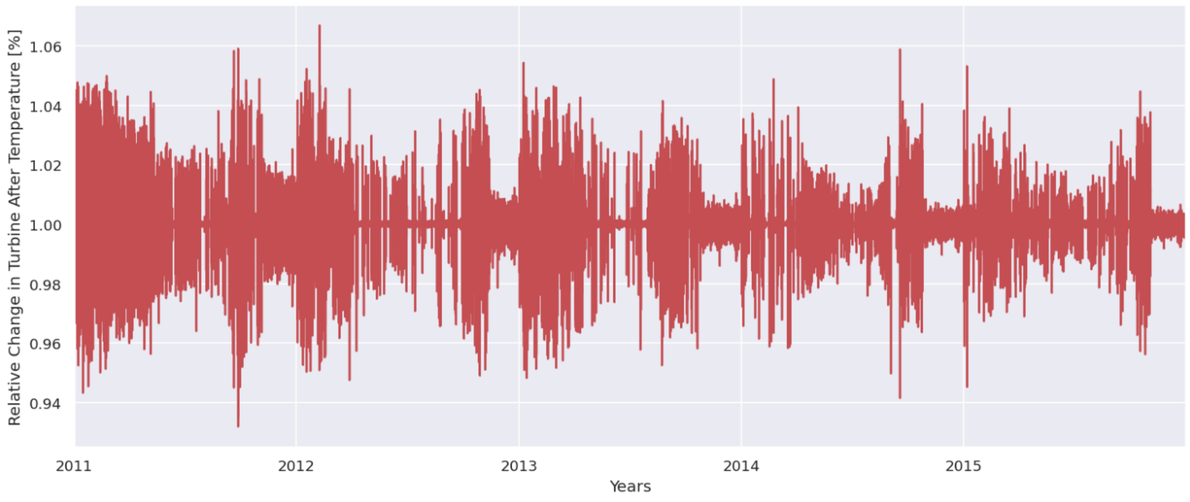


Figure 131: Relative change for turbine after temperature feature over the years

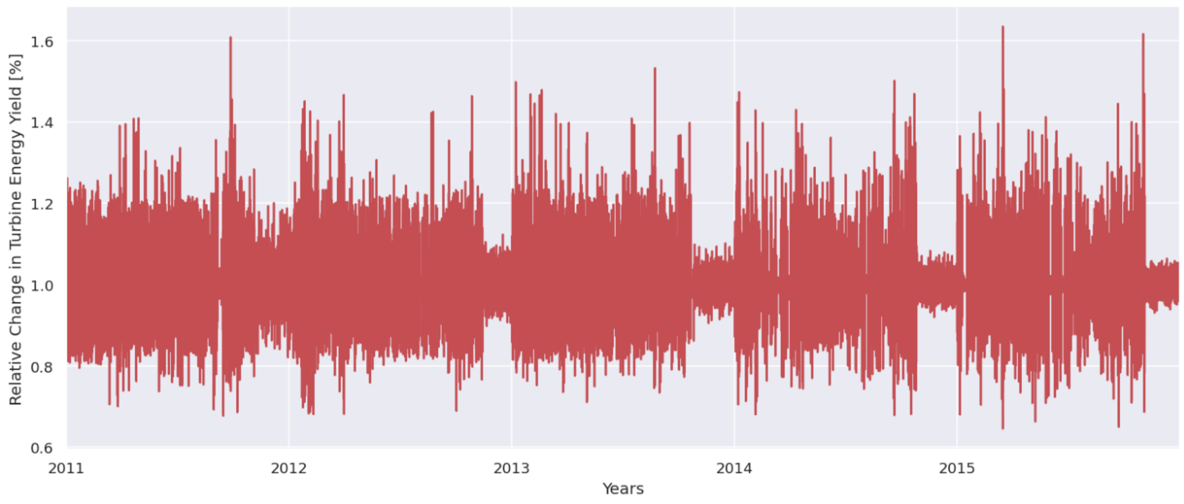


Figure 132: Relative change for turbine energy yield feature over the years

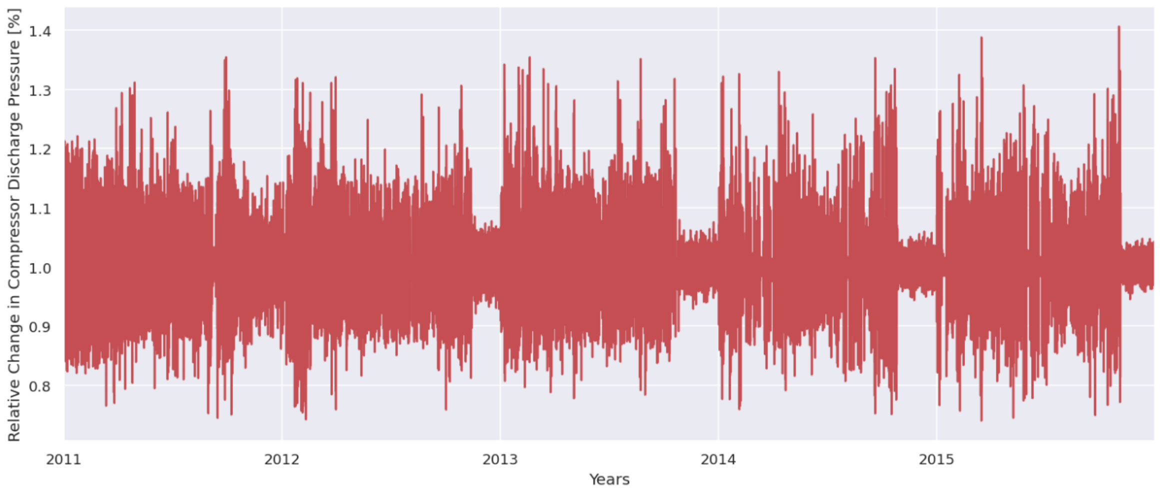


Figure 133: Relative change for compressor discharge pressure feature over the years

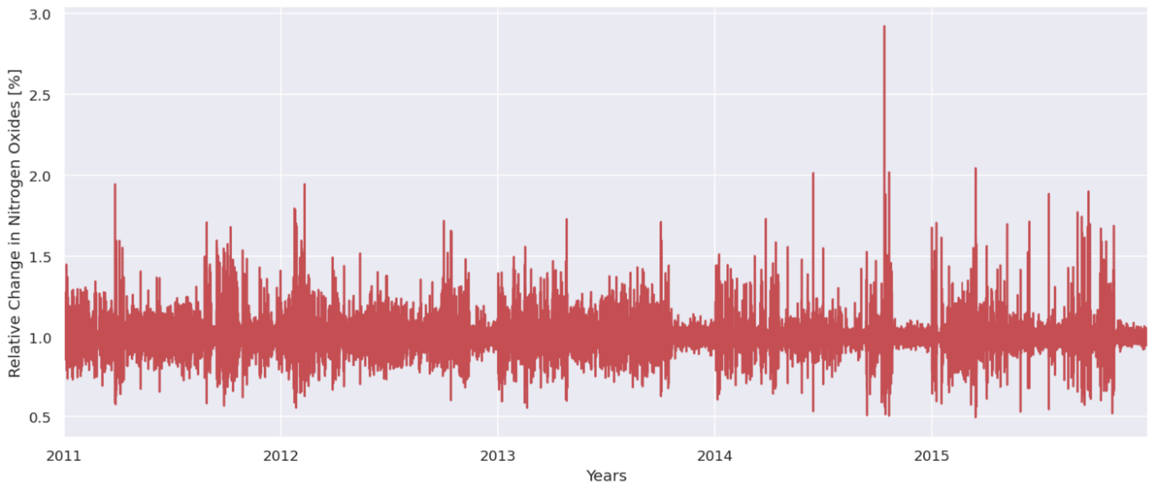


Figure 134: Relative change for nitrogen oxides emissions feature over the years

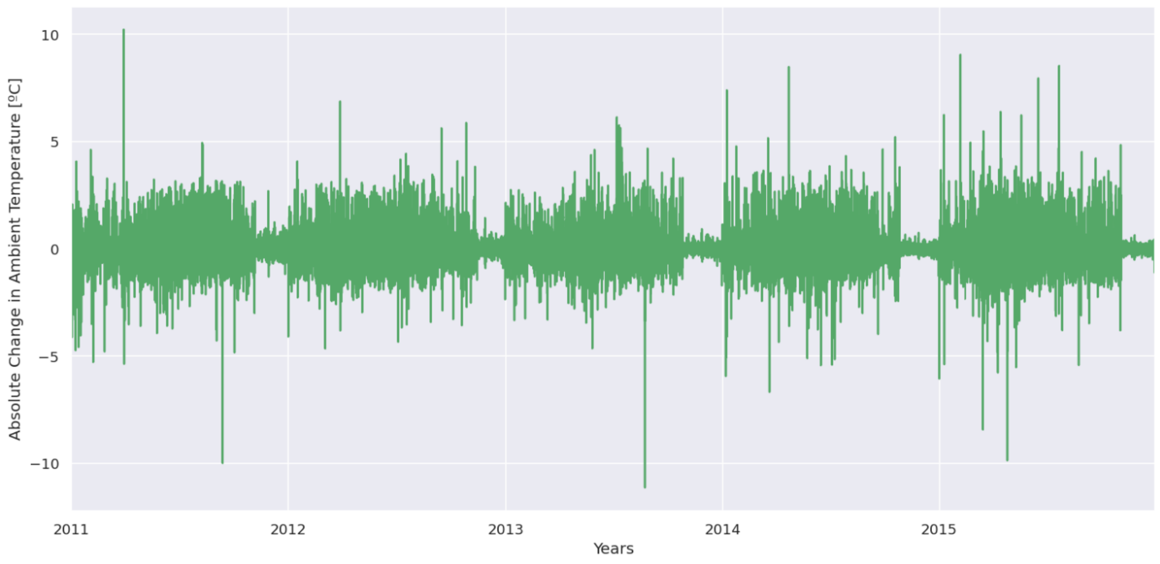


Figure 135: Absolute change for ambient temperature feature over the years

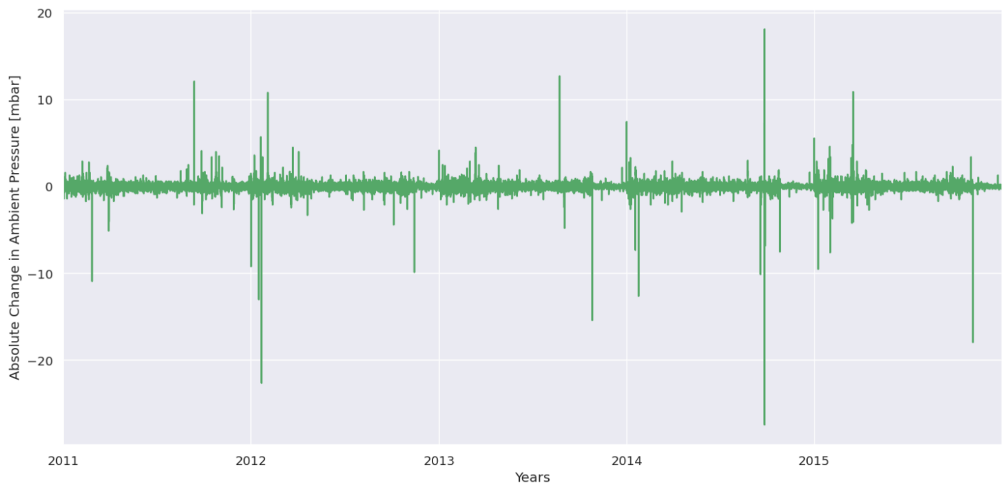


Figure 136: Absolute change for ambient pressure feature over the years

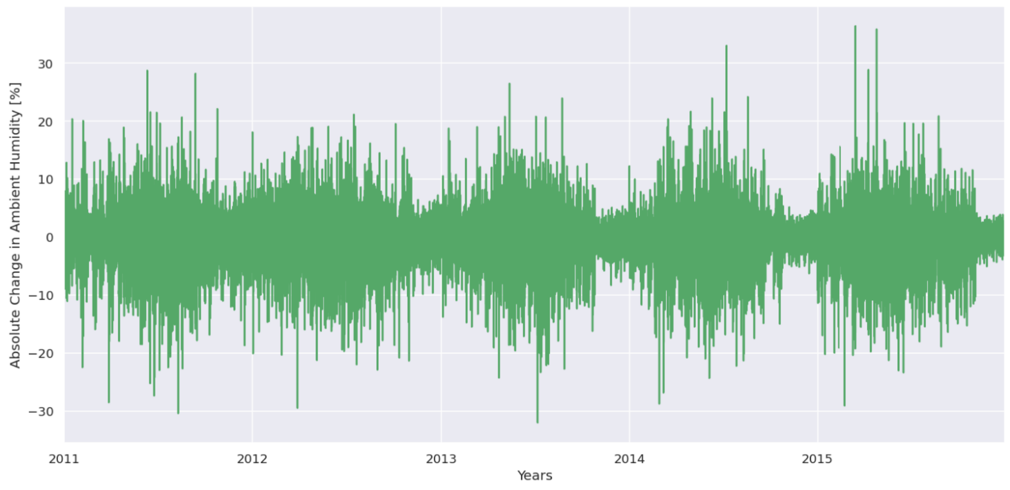


Figure 137: Absolute change for ambient humidity feature over the years

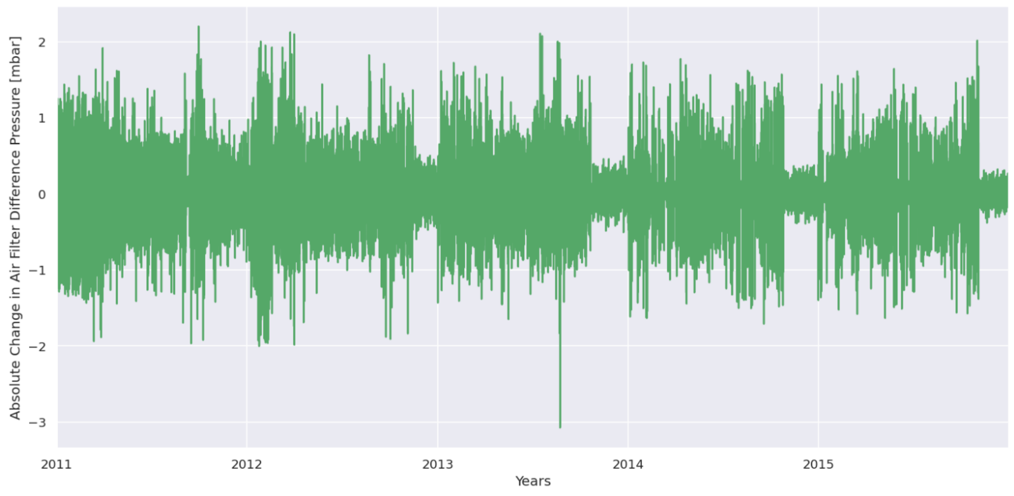


Figure 138: Absolute change for air filter difference pressure feature over the years

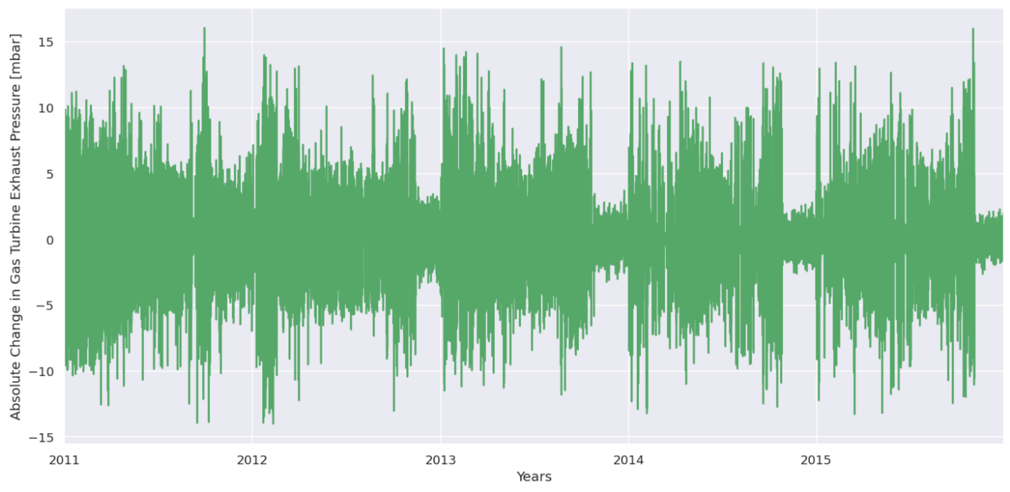


Figure 139: Absolute change for gas turbine exhaust pressure feature over the years

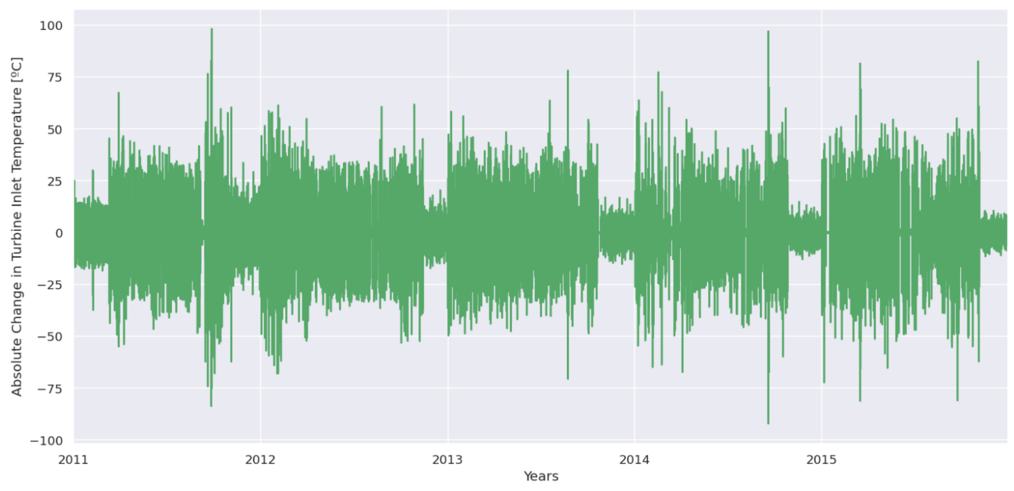


Figure 140: Absolute change for turbine inlet temperature feature over the years

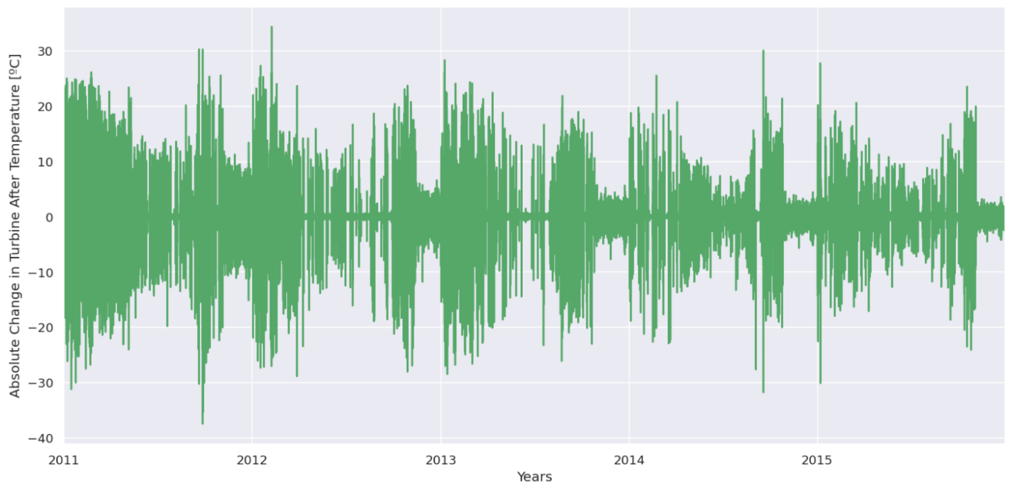


Figure 141: Absolute change for turbine after temperature feature over the years

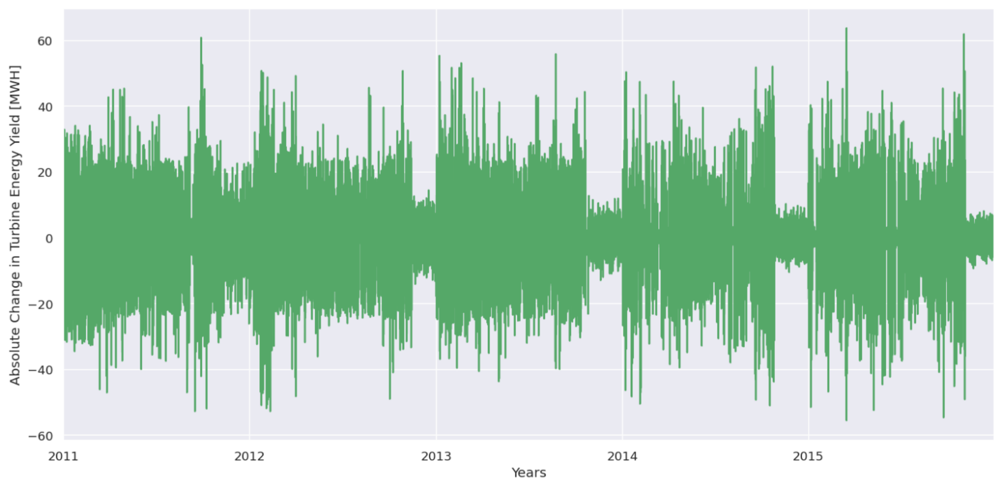


Figure 142: Absolute change for turbine energy yield feature over the years

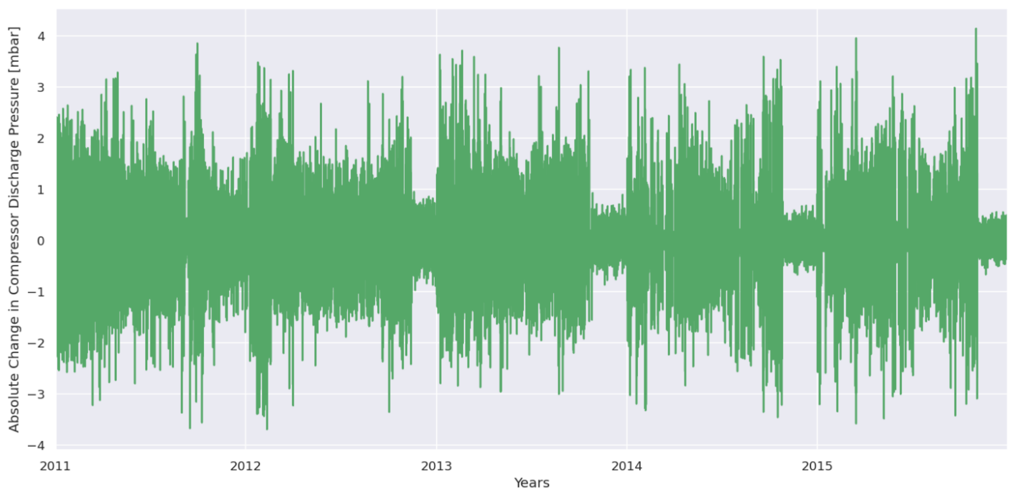


Figure 143: Absolute change for compressor discharge pressure feature over the years

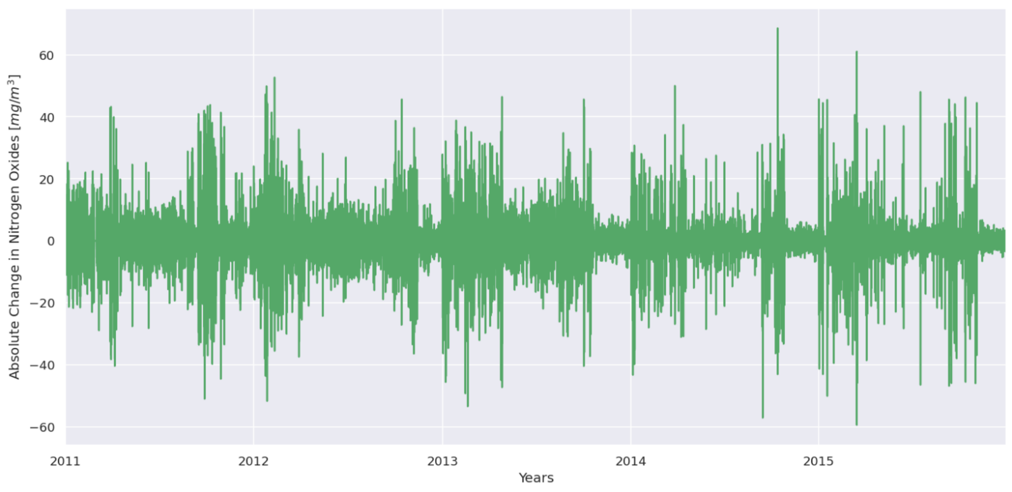


Figure 144: Absolute change for nitrogen oxides emissions feature over the years

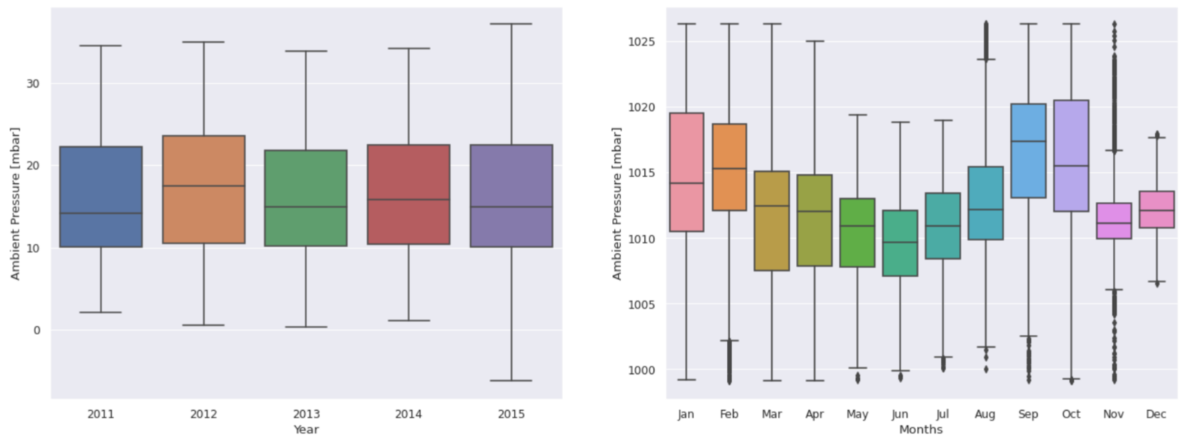


Figure 145: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for ambient pressure feature after outliers removal

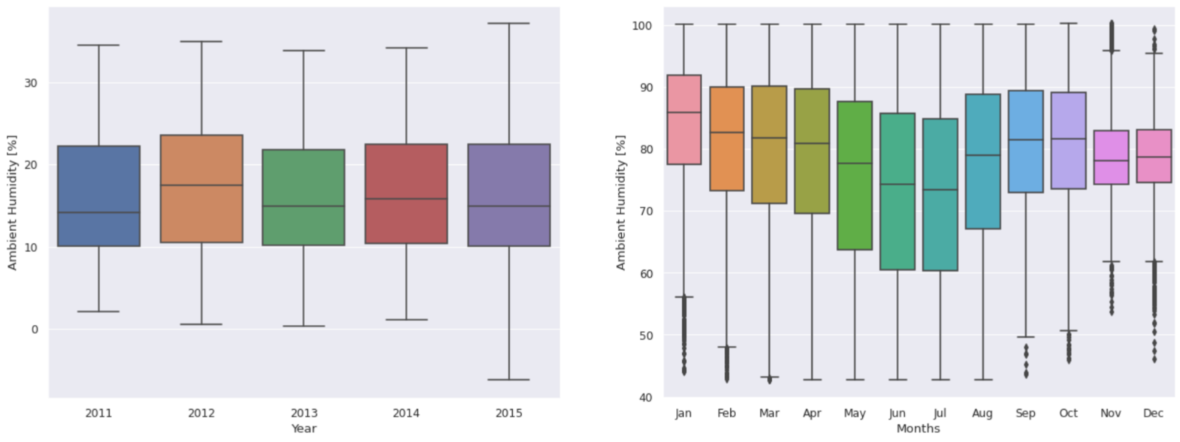


Figure 146: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for ambient humidity feature after outliers removal

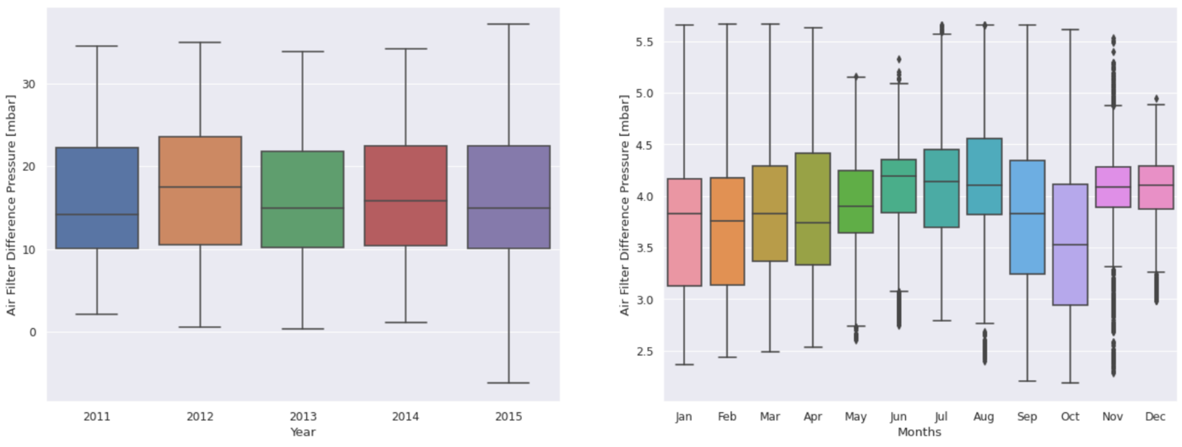


Figure 147: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for air filter difference pressure feature after outliers removal

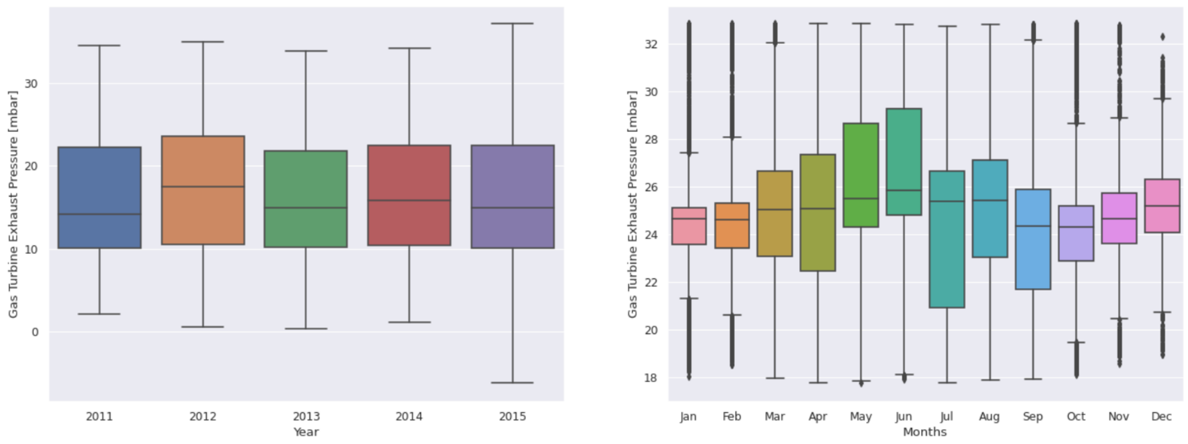


Figure 148: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for gas turbine exhaust pressure feature after outliers removal

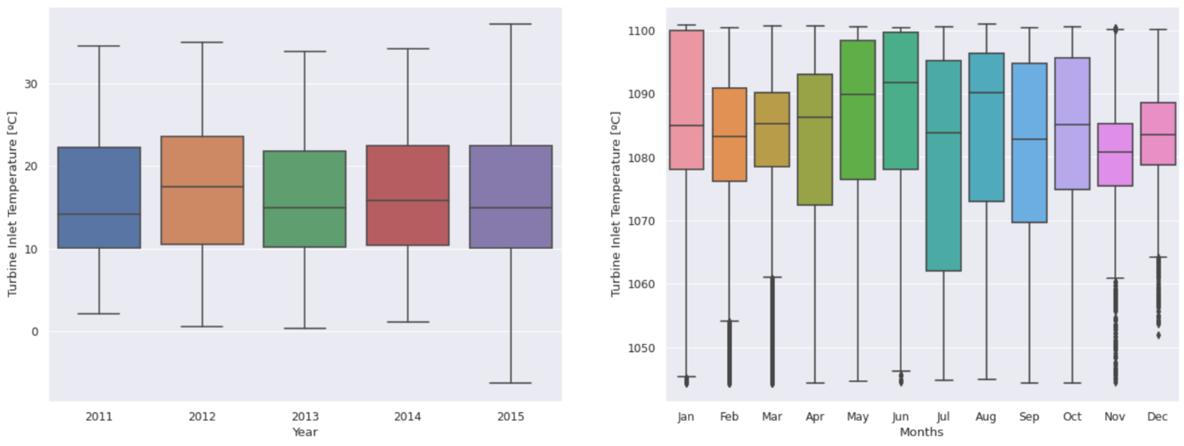


Figure 149: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for turbine inlet temperature feature after outliers removal

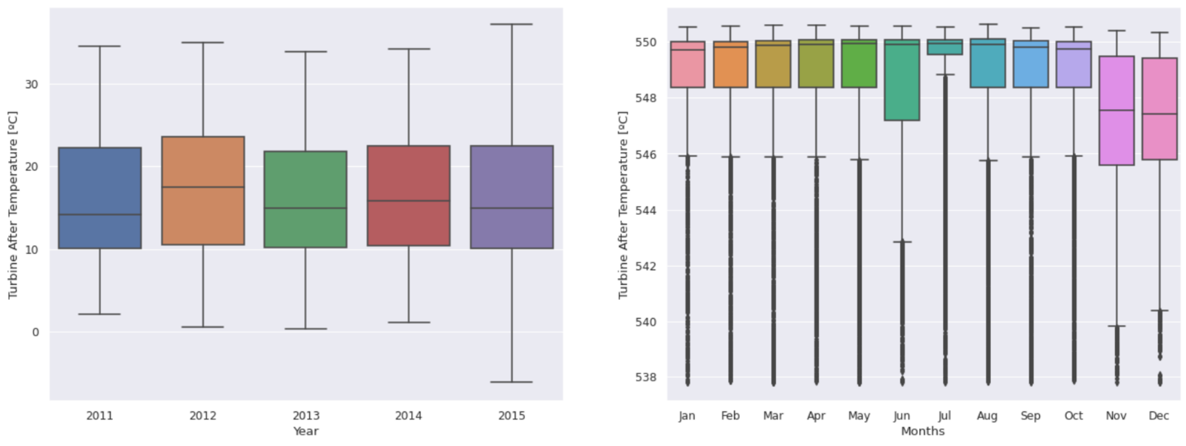


Figure 150: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for turbine after temperature feature after outliers removal

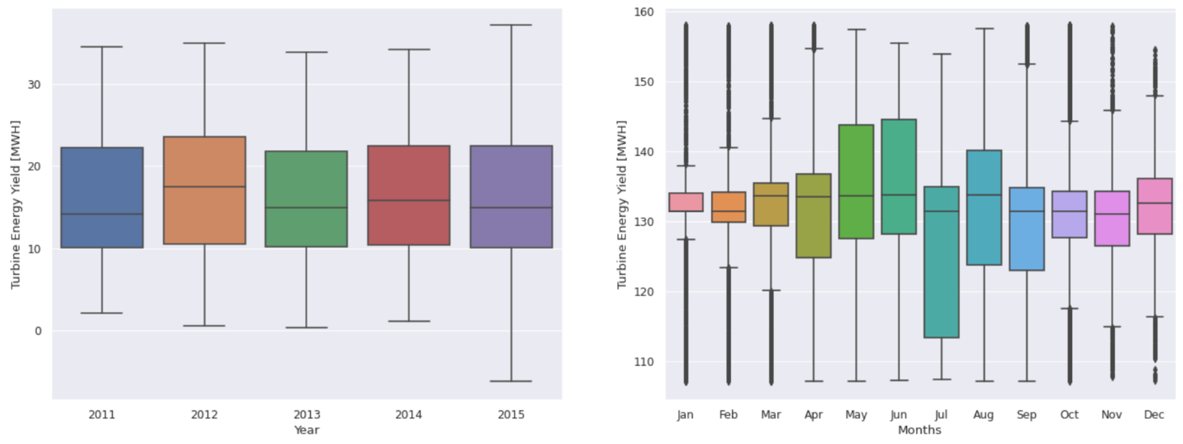


Figure 151: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for turbine energy yield feature after outliers removal

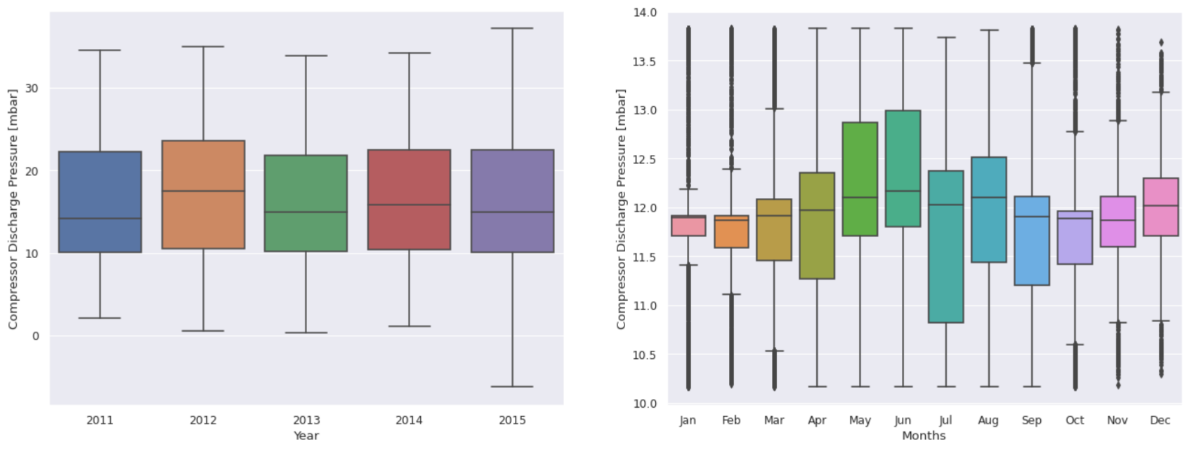


Figure 152: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for compressor discharge pressure feature after outliers removal

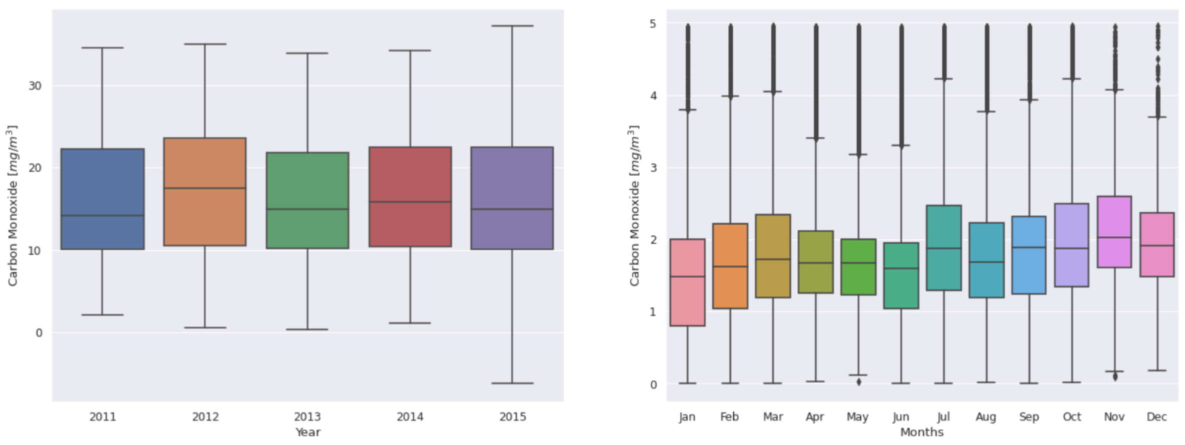


Figure 153: Boxplot of month-wise (seasonal) and year-wise (trend) distribution for carbon monoxide emissions feature after outliers removal

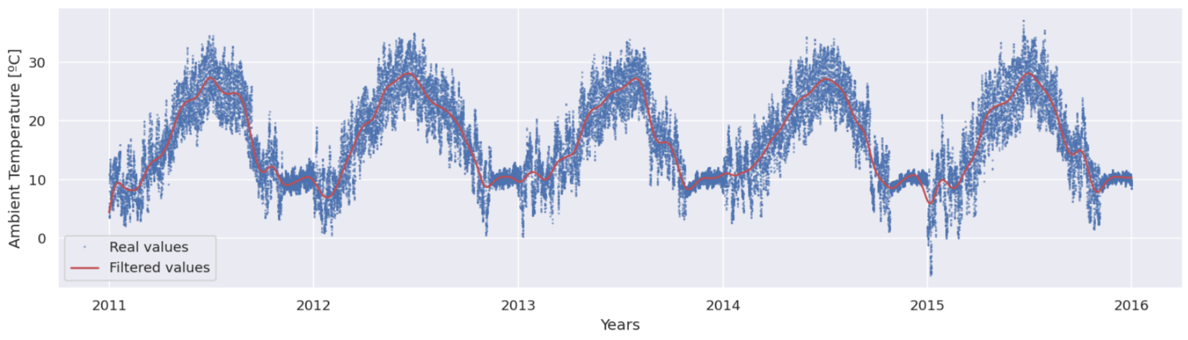


Figure 154: Ambient temperature feature and butterworth filtering after outliers removal

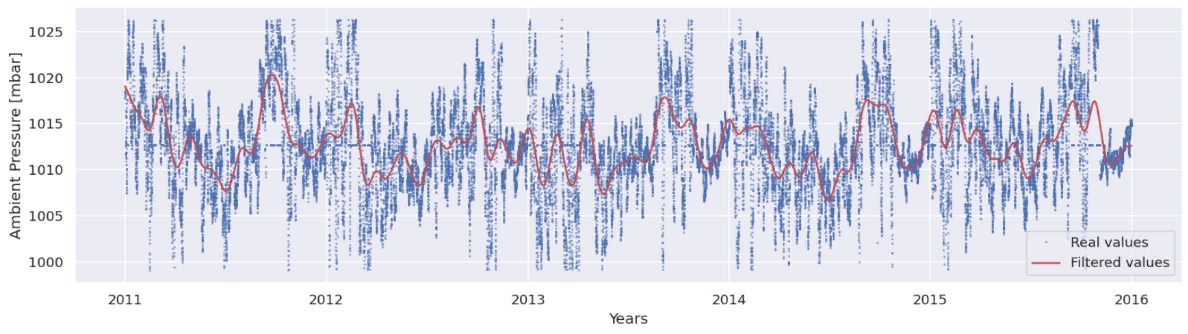


Figure 155: Ambient pressure feature and butterworth filtering after outliers removal

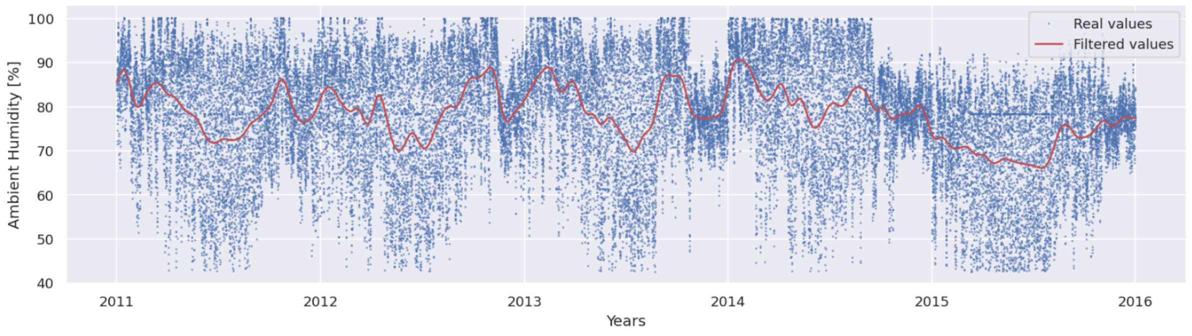


Figure 156: Ambient humidity feature and butterworth filtering after outliers removal

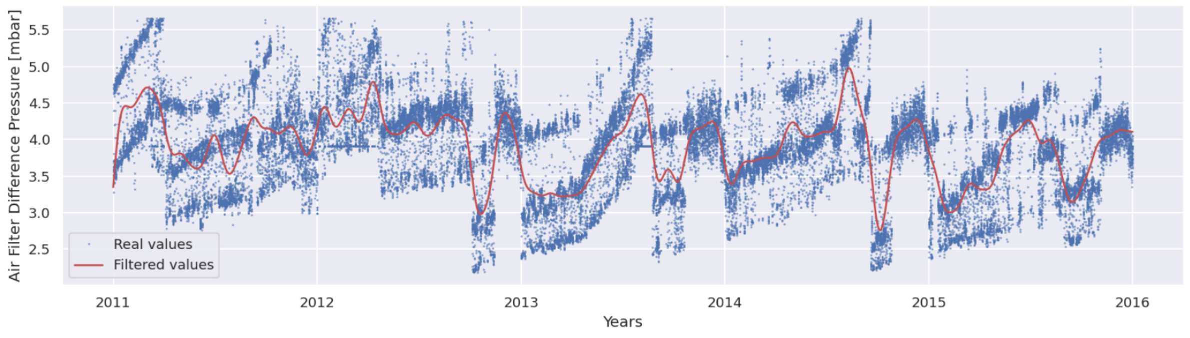


Figure 157: Air filter difference pressure feature and butterworth filtering after outliers removal

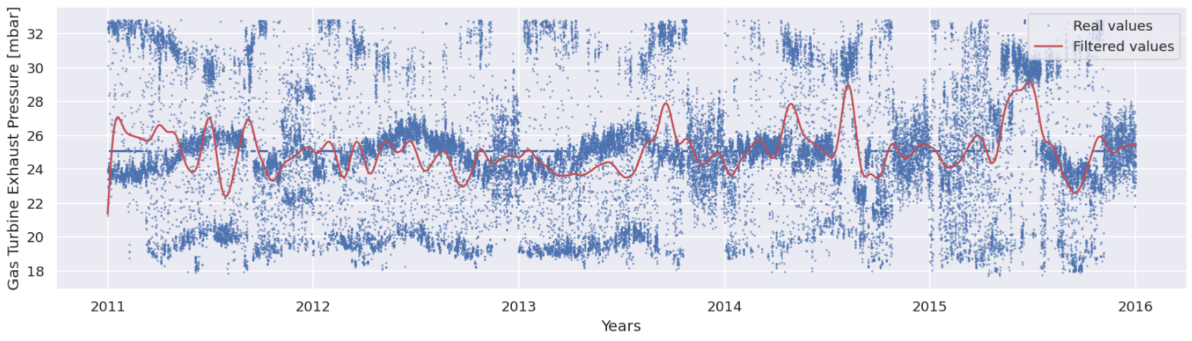


Figure 158: Gas turbine exhaust pressure feature and butterworth filtering after outliers removal

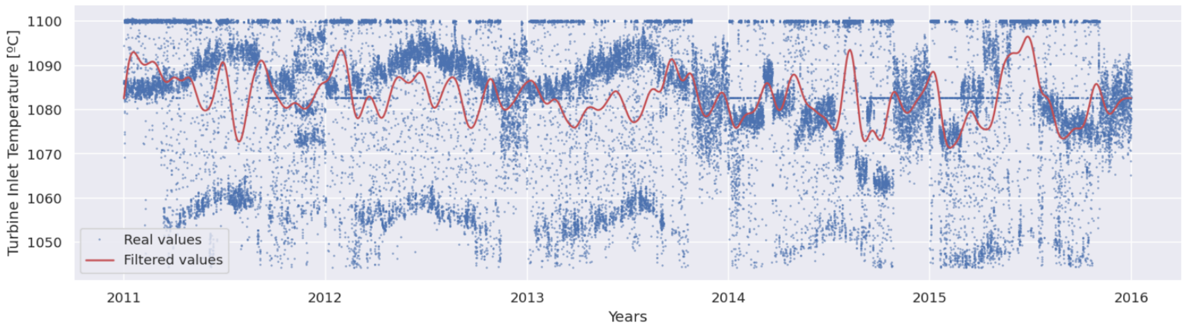


Figure 159: Turbine inlet temperature feature and butterworth filtering after outliers removal

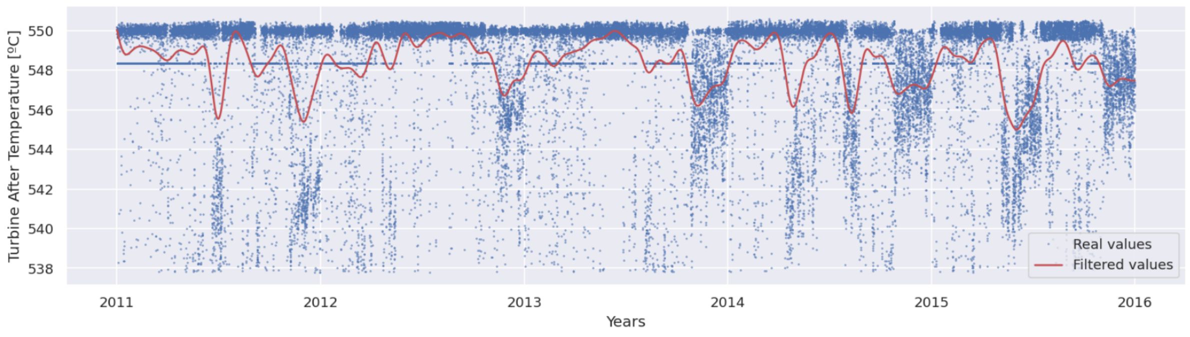


Figure 160: Turbine after temperature feature and butterworth filtering after outliers removal

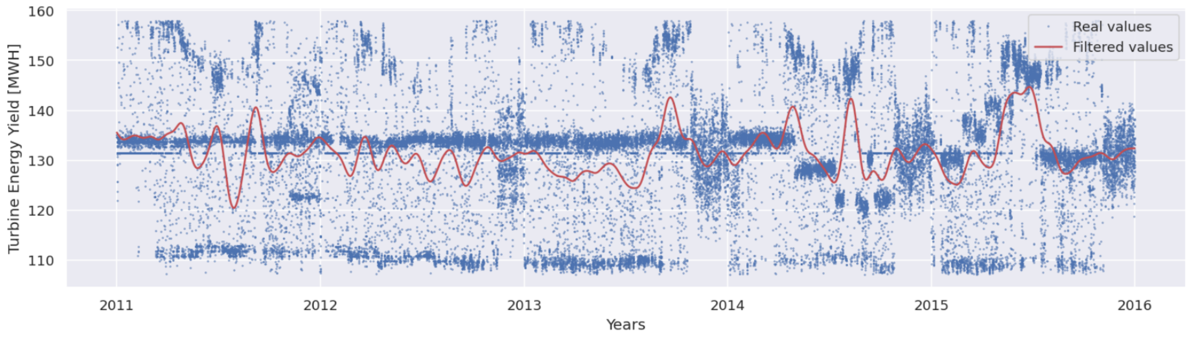


Figure 161: Turbine energy yield feature and butterworth filtering after outliers removal

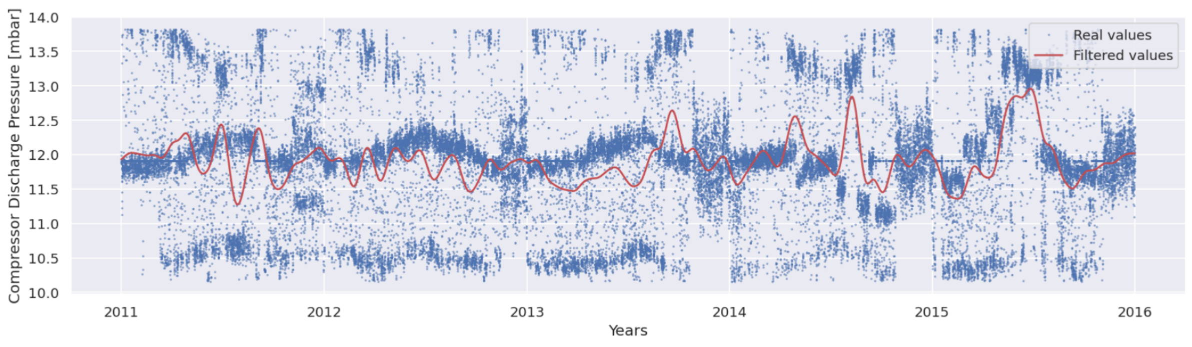


Figure 162: Compressor discharge pressure feature and butterworth filtering after outliers removal