Faculty of Mathematics and Informatics

VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
MODELLING AND DATA ANALYSIS
MASTER'S STUDY PROGRAMME

FAULT DETECTION IN THE SOLAR POWER GENERATION PROCESS

GEDIMŲ IDENTIFIKAVIMAS SAULĖS ELEKTRINIŲ ELEKTROS GAMYBOS PROCESE

Master's thesis

Author: Monika Galinytė
VU email address: monika.galinyte@mif.stud.vu.lt
Supervisor: dr. Jolita Bernatavičienė

Vilnius

2024

Abstract

In this paper, statistical and machine learning methods are employed to detect faults in the solar power generation process. The primary objective is to analyze unlabeled data and identify strings affected by shading. The behavior of 143 strings that generate solar electrical power is investigated, comparing them based on their similarities. Initially, the data is evaluated using mean, standard deviation, and correlation coefficients to detect any abnormalities. Following this, K-means clustering is applied to the data from the days with the highest power generation and compared with the results with those from average generation days. Next, the time series data is transformed into functional data and smoothed using Fourier basis functions. This functional data is then used to detect outliers and perform functional K-means clustering.

Keywords: solar power, outlier detection, fault detection, K-means clustering, Functional data analysis

Santrauka

Šiame darbe naudojami statistiniai ir mašininio mokymosi metodai gedimams saulės energijos gamybos proceso aptikti. Pagrindinis tikslas yra išanalizuoti neklasifikuotus duomenis ir nustatyti stygas, paveiktas šešėlių. Ištirtas 143 saulės elektros energiją generuojančių stygų elgesys, lyginant jas pagal panašumus. Iš pradžių duomenys įvertinami naudojant vidurkį, standartinį nuokrypį ir koreliacijos koeficientus, siekiant nustatyti esamus nukrypimus. Tuomet, K-means klasterizavimas taikomas dienoms, kai pastebėta didžiausia elektros galios generacija, ir palyginamas su dienomis, kai elektros galios generacija buvo vidutinė. Tada laiko eilučių duomenys paverčiami funkciniais duomenimis ir jiems pritaikoma Fourier transformacija. Apdirbti funkciniai duomenys naudojami išskirčių identifikavimui ir funkciniam K-means klasterizavimui.

Raktiniai žodžiai: saulės energija, išskirčių identifikavimas, gedimų indentifikavimas, K-means klasterizavimas, Funkcinių duomenų analizė

Notation

- **DT** - Decision tree
- **DC** - Direct current
- **EWMA** - Exponentially weighted moving average
- **FDA** - Functional data analysis
- **PV** - Photovoltaic
- **LG** - Line to ground
- **LL** - Line to line
- **MSE** - Mean square error
- **OC** - Open circuit
- **SD** - Standard deviation
- **t-SNE** - T-Distributed Stochastic Neighbor Embedding
- **W** - Watt

Contents

List of Figures

1 Introduction

Nowadays, solar energy is becoming increasingly popular, primarily for environmental and economic reasons. People are seeking ways to meet their energy needs with minimal environmental impact. Additionally, given the current economic situation, individuals are looking to reduce their electricity bills, while some view solar plants as a viable investment. According to the Official Statistics Portal of Lithuania [13], solar power plants generated 79.4% more electrical energy in 2022 compared to 2021. The International Renewable Energy Agency reported that solar energy consumption in Lithuania accounted for 12% of the total renewable energy consumption in 2021. Despite the rapid growth in the solar energy sector and extensive research, fault detection remains one of the most challenging aspects of solar power generation. This challenge is primarily due to limited data, noise caused by environmental variations, and incipient faults that are difficult to identify due to very minor energy losses. Furthermore, components such as inverters, which are responsible for converting current to usable electricity, are designed to shut down in the event of a failure, and these occurrences often go unregistered. Solar power is a clean energy source that plays a crucial role in environmental decarbonization, however, undetected faults can lead to inefficiencies that impact financial results and pose safety hazards, including the risk of electrical fires. Therefore, the purpose of this thesis is to address the fault detection challenge by employing statistical and machine learning methods.

Solar power plants are continuously exposed to many factors that significantly degrade their performance and efficiency. In order to successfully achieve our goal, we have to understand what kind of faults are possible in solar power plants. Scientists categorize faults of solar power systems in different ways, either by their duration or by their nature:

- By Duration [17]

    1. Temporary faults - short-term issues that can resolve on their own or through simple interventions. They often do not cause lasting damage to the system.
    2. Permanent Faults - long-lasting or persistent issues that require significant repair or replacement of components. They can cause substantial damage if not addressed promptly.

- By Nature [5]

    1. Physical faults - involve physical damage to the solar panels or other components, such as cracks or soiling (accumulation of dirt and debris).
    2. Electrical faults - involve issues in the electrical circuitry, such as short circuits, open circuits, or inverter malfunctions.
    3. Environmental faults - caused by external environmental factors, such as shading from nearby objects, extreme weather conditions, or temperature variations.

However, this research solely addresses the detection of environmental faults, particularly shading, which affects electrical power generation.

While numerous papers have been published focusing on various types of fault detection in the field of solar energy, many of them rely on simulated or experimental data where electrical or environmental

faults are artificially created. Some studies utilize real-life datasets with pre-labeled faults. Thus, we believe that further research in this area is relevant. To the best of our knowledge, the combination of time series clustering and functional data clustering has not been extensively explored on a case-by-case basis for fault detection. This paper demonstrates that, despite the longer computation time, analyzing case-specific scenarios provides an advantage in identifying more abnormalities. This is because examining longer intervals of data can obscure important features, for instance, the same string may compensate for poor generation results from the previous day, which were impacted by adverse weather conditions.

Our research indicates that statistical measurements, such as standard deviation or correlation, can suggest the possibility of shading affecting strings by examining the relationship between strings connected to the same panel. However, these assumptions should be supported by other methods. Therefore, we incorporated K-means clustering and FDA algorithms, which effectively identified strings which performance was negatively impacted by shading compared to others. To validate our findings, we referred to a report prepared by experts accompanying the dataset, which indicated that inverters 9, 8, 11, and 12 exhibited lower generation results due to shading. Nonetheless, we hypothesize that multiple high-performing strings connected to an inverter could compensate for the losses experienced by shaded strings connected to the same inverter. Consequently, we focused on implementing the selected methods on strings rather than inverters.

The rest of the work is organized as follows. Section 2 presents the literature review, encompassing the types of faults detected, the data utilized, and offering a brief summary of statistical and machine learning methodologies employed in the detection of solar energy faults. Furthermore, we examine the application of functional data clustering methods in addressing various issues identified in the literature. Section 3 consists of the theoretical aspects of the methods used in this research, while Section 4 introduces the dataset utilized in the experimental part, which is detailed further in Section 4. Results and conclusions are presented in Sections 5 and 6, respectively.

2 Literature review

Before starting a new research it is very important and helpful to get to know the fellow researches' work. Analyzing what has been done in the field helps to understand the challenges of similar work, distinguish the methods that are the most common and accurate from the ones that are less suitable to the topic of interest. For that reason, the first part of this paper presents the reviews of scientific papers that contributes to improving and understanding solar power plants operations.

2.1 Fault detection

In order to make solar power plants more efficient, we need to acknowledge the weakest parts of it. Two the most common reasons of decrease in solar power output are weather conditions and faulty elements. This paper's focus is a fault detection of solar array.

In their work [28], Y. Zhao et al. explored Decision Tree (DT) based models. With this method, scientists successfully achieved high accuracy at detection and classification of pre-defined faults. While authors celebrated the simplicity and accuracy of DT models, they also acknowledged the limitations of this method. High cost of training and poor performance on a data that is significantly different from training set encourages to seek for better approach. Later Y. Zhao et al. [30] introduced three different outlier detection rules in order to identify 5 faults of solar power plant at different contamination levels. While 3-Sigma rule failed at all cases, Hampel identifier had the best accuracy at higher contamination level due to method's outlier-sensitivity being the lowest. However, Boxplot rule performed better at half as small contamination level as Hampel identifier. A group of researches proposed combining one-diode model with an exponentially weighted moving average (ODM-EWMA) to detect changes that identify as faults in photovoltaic (PV) systems [6]. This method was applied to five cases - normal operating conditions, study of open-circuit (OC) PV strings, study of a short circuit (SC) in a string of PV modules, study of shading fault and study of multiple faults. The differences between predicted and measured values of maximum current, voltage and power generated values were loaded to EWMA chart, which is used to detect faults. Overall, authors concluded that proposed method successfully detects and identifies faults. In 2021, fault detection using statistical tool based on William Gosset's (Student's) T-Test [10] was introduced. Experiment started from inducing faults in actual PV plant by switching the relays at certain times to short-circuit in PV module and raising translucent sheet in front of the panels to cause partial shading or soiling. After that, theoretical PV plant was simulated using irradiance and temperature measured in actual PV plant. The direct current (DC) capacity of individual strings within the actual solar power plant were assessed against the mean DC capacity of strings in the simulated PV plant using an introduced statistical tool, Z_Score (the difference of string powers of actual PV plant and the mean of string powers of simulated PV plant divided by standard deviation of simulated PV plant). Based on the predefined thresholds, faults were classified either as PV modules failure, partial shading or soiling loss, or PV string failure.

Another method for fault identification involves employing machine learning techniques. To address challenges such as non-linearity, the expense of training data, and the need for case-by-case analysis, a separate study [29] introduced a graph-based semi-supervised learning approach (GBSSL). This method was proposed for detecting accidental short-circuiting between two points in the array with

different potentials, as well as identifying accidental disconnections in normal current-carrying conductors. Choosing semi-supervised learning method, requires significantly smaller amount of labeled data, which results in decrease of resources. Researches managed to reach 100% accuracy in detection and classification of fault in photovoltaic arrays. [9] paper's goal was to investigate fault detection problem on the DC side with the help of multi-resolution signal decomposition (MSD) and a fuzzy inference system (FIS) focusing on line to line (LL) and line to ground (LG) shortcomings. For training purposes, researches used simulated events, in total 3888 faults and 220 cases of normal activity were generated. Later the performance of proposed method was verified on experimental PV array with only LL faults. Both, simulation and actual cases studies reached satisfactory accuracy. In their second paper on fault detection [20] S. Rao et al. continued exploring customized neural network algorithms. Researchers' goal was to improve the reliability and efficiency of utility scale solar arrays. Using K-means algorithm on voltage, current and temperature, let to identify such faults as ground, arc, degradation, complete or partial shading, variation in temperature, standard test conditions with irradiance at 1000 W/m^2 and a module temperature of 25°C. The main part of the research was implementing multilayer feed-forward neural network also called multilayer perceptron (MLP). By using this fault detection and identification method on experimentally generated data, authors managed to detect and label 8 types of faults with overall accuracy of 99.7%. Acknowledgment that incipient faults are extremely hard to detect due to insignificant losses of generated electricity led to research [31] where long-term differences between actual and predicted values are analyzed in order to detect faults. The paper's main idea is to use collaborative filtering (CF) method to accurately predict current values based on the similarity with historical values. With the help of Euclidean distance, the similarity within data was calculated, the next step was to apply CF technique to predict string's current values. The fault relevancy was evaluated by calculating the residuals between actual and predicted values, as well as, analyzing the span of the residual. Once the relevance is established, auto-threshold model [32] is applied for fault detection.Experimental part proved the efficiencies of proposed model and highlighted Euclidean distance superiority against Cosine and Pearson based models. 5000 I-V curves (PV generator current vs. voltage) simulated under uniform conditions and 5000 I-V curves simulated under different mismatched conditions were used in [16] paper. Multilayer Perceptron algorithm (MLP) was applied to the training set. In order to find the best possible model, training stage was completed multiple times with different number of neurons in the hidden layers. Despite a few classification errors, MLP showed promising results. However, second evaluation of the model was performed using 2000 experimental I-V curves and based on this part of the research, the network with 15 neurons surpassed any other variation of MLP. A research [18] for hourly fault detection based on actual PV systems located in Australia and China could be summarized in three steps - data was pre-processed by correcting measured data using linear fitting lines between global radiation and power under standard test condition (this step eliminated abnormal data), then combination of K-means Cluster and Multiclass-gradient boosting decision three-logistic regression (Multiclass-GBDT-LR) was introduced in order to recognise hourly weather status patterns. The last step was applying two layers fitting model for hourly faults monitoring. For the first layer K-Nearest Neighbors (KNN), Gradient Boosting (GBR), MLP and Random Forest (RF) methods were used as a base learners, while Linear Regression was used in the second layer. Overall, an unsupervised hourly weather status patter recognition and blending fitting model showed its significant

advantages by promptly detecting faults. Comparison of three machine - learning models is presented in the work of Qais Ibrahim Ahmed et al. [1]. To identify irregularities of the solar power plant in India, authors implemented physical model, Support Vector Machine (SVM) regression model and SVM classification model. Grey wolf optimizer (GWO) was used for tuning the hyper parameters for all three cases. By comparing sensitivity and specificity rates, researchers showed that combination of GWO and SVM classification model outperformed other two models by classifying inverter states into normal and fault categories. Sufyan Ali Memon et al. [14] introduced a paper where simulated labeled data was fed to Convolutional Neural Networks (CNN) in order to identify whether PV array works under normal conditions, faces degradation, LL fault, OC (open circuit) fault or is partially shaded. The CNN performance was evaluated using learning curves, accuracy curves and confusion matrix. It was concluded that using CNN model on simulated data achieved overall accuracy of 95.20%. To solve anomaly detection exercise, Eleonora Arena et al. [2] introduced Monte Carlo based pre-processing technique. This method's benefit is that it intrinsically deals with outlier substitution and takes into account the temporal locality of subsequent samples. Following pre-processing, the authors proceeded to train an anomaly detection model utilizing Principal Component Analysis. The proposed method was tested under standard and abnormality conditions. It accurately predicted equipment failure nearly two weeks in advance, while also showcasing resilience against false alarms during normal conditions. Summary of discussed fault detection in solar power plants methods is presented in the table 5.

2.2 Functional clustering

Functional data analysis has a wide range of applications. F. Ieva et al. [8] used multivariate functional clustering method for an analysis of electrocardiograph (ECG). Data set consists of 198 subjects, from which 97 subjects are diagnosed with either right bundle branch block (RBBB) or left bundle branch block (LBBB). To smooth the ECG data, Daubechies wavelet-based smoothing method was chosen. Authors highlighted that each curve has its own biological time so the same feature can appear at different times among patients. To solve this problem monotone interpolation method was used. By using such registration technique, morphological information is separated from duration of the different segments of the ECG. The goal of the research was to distinguish cases by different pathologies considering only the shape of the curves. For that purpose authors used k-means clustering. The main distance used int his work was natural distance in the Hilbert space (d_1). For evaluation and comparison reasons, authors also used natural seminorm ($\tilde{d_1}$) and norm (d_2) in the Hilbert space. By using silhouette plots, it was identified, that optimal number of clusters is 3. Results of cross-validation identified that using d_1 and d_2 distances, provide slightly better results than using $\tilde{d_1}$. Paper by J. Jacques and C. Preda [11] also considered the problem of clustering multivariate functional data. As a solution, authors introduced principal component analysis for multivariate functional data and then assumed a cluster-specific Gaussian distribution for the principal component scores (further called Funclust). This research was supported by three evaluation strategies: Funclust method was compared to other clustering methods applied to univariate functional data, Funclust method was compared to simulated multivariate functional data, as a third strategy climatology data set was introduced which was used to present geographical interpretations of the clustering results using Funclust and K-means methods. For this thesis, the third case is the most relevant. Canadian temperature and precipitation

curves were smoothed using Fourier basis with 65 knots. Funclust method was compared with already mentioned kmeans-d_1 and kmeans-d_2 algorithms ([8]). By comparing these methods, authors concluded that kmeans-d_1 and kmeans-d_2 seems less pertinent than Funclust since two out of four identified clusters includes Atlantic and Pacific countries. To conclude the results, expectation-maximization algorithm was proposed. Both real data and simulated cases illustrated the efficiency of proposed method. Another paper that implemented functional data clustering was written by A.Z. Zambom et al. [27]. Authors proposed new variation of functional data K-means clustering method based on a measure of similarity of curves that is composed of a combination of the t test statistic and an ANOVA-type test statistic for parallelism. Experimental part was implemented using 2 cases of simulated data and 4 cases of real life data. Results of the experiment was compared with multiple different variations of functional data clustering introduced by different authors, such as already mentioned Funclust method introduced by [8]. Real data used for experimental part is very well known amongst those, who are interested in functional data analysis methods. Canadian weather, Growth, Yeast Gene, and Fat spectrum data. Experiments on both, simulated and real data, showed that proposed strategy performed competitively, in some cases achieving the smallest rate of error. [15] paper introduced a novel approach (Deep-FDA) that combines functional data analysis (FDA) and neural networks. Performance of the proposed method was evaluated by comparing two scenarios - model-driven simulation where random samples were generated reproducing the model and real three months worth of traffic network data. Simulated data was divided to balanced and imbalanced clusters cases, which were then divided into different sample sizes and fed to functional K-means clustering, principal component analysis (PCA), functional principal component analysis (FPCA) and Autoencoders (AE) methods. Results of the experiment were evaluated using precision, recall, L^1 and L^2 distances. It was concluded that results of AE are very similar to PCA and FPCA, however PCA and FPCA outperformed AE when applied to small sample size. The same methods were tested on a real data set to detect imperceptible patterns. Data frame was divided into 1 month, 2 months and 3 months samples and 4 network segments based on the services network provides. In this case all four methods gave very similar results, however based on Functional Silhouette Coefficient (SC) and Functional Davies-Bouldin Index (DBI), K-means clustering performed better than projection based algorithms in most cases, while FPCA demonstrated its superiority against PCA by better differentiating two clusters that are very close to each other. Overall, the proposed combination of FDA and AE showed that it can be valuable tool in capacity planning and alarming. To detect anomaly in functional data, J. R. Sosa Donoso et al. [24] adapted Local Correlation Integral (LOCI) method. The implementation of this method was validated using hourly average humidity data obtained in Ecuador. This data set corresponds to 2190 days of information, where each day is considered as a function of 24 measurements. Performance of functional data LOCI (FD-LOCI) algorithm using L^2 metric was compared to FD-LOCI with elastic depth, Functional Boxplot, Elastic Depth and fdqcs.depth methods. It was evaluated that all methods performed similarly, yet based on an ability to detect anomalies, both FD-LOCI variations performed the best.

Shading effects to solar energy generation was analyzed in [3]. For comparison purposes, data with and without shading was simulated assuming that PV plant is located Rwanda, Africa. Location is important, because real weather observations were imported to generation process to get as accurate

simulation as possible. P-V (power - voltage curves) and I-V curves were investigated at different rates of irradiance and different numbers of shaded modules. Authors concluded that shadow affects the performance of PV systems and the rate at which the PV system is affected depends on the shaded area, position, and how much radiation reaches that shaded area. In conclusion, the reviewed literature highlights the critical importance of robust fault detection methods in ensuring the efficient and reliable operation of solar power generation systems. Despite advancements in fault detection methodologies, the impact of shading on system efficiency continues to pose significant hurdles. Moving forward, our study aims to contribute to this field by proposing a novel methodology that integrates machine learning techniques with functional data analysis to enhance fault detection accuracy in the presence of shading anomalies.

3 Methodology

This research is based on statistical and machine learning methods as illustrated in figure 1. Data set is explored through descriptive statistics, such as Mean, Standard Deviation (SD), and Correlation. These measurements help identify patterns, relationships between variables, and anomalies. Based on this initial evaluation, K-means clustering and outlier detection are used to identify strings affected by shading. To further support current research, time series data is converted to a functional dataset and apply Functional Data Analysis (FDA) methods. This section outlines the methods used and provides key definitions and formulas.

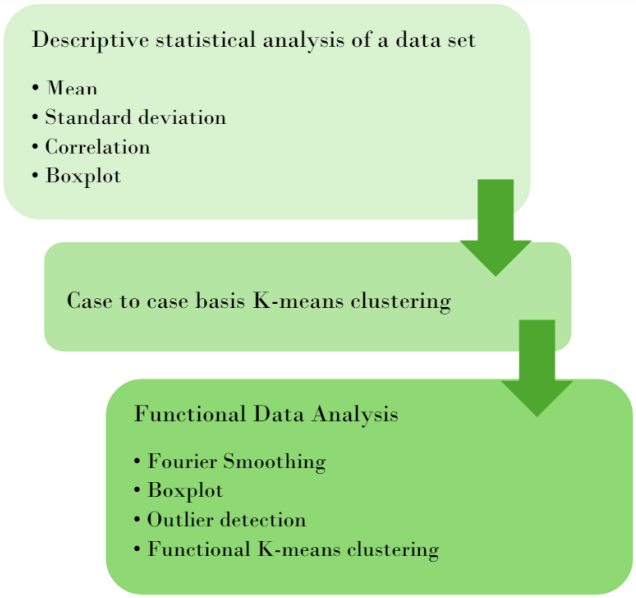

Figure 1: Research methodology workflow

3.1 Descriptive statistics

Descriptive statistics consists of multiple different measurements, that provides essential information about a dataset, such as central tendencies, variability, and distributional properties. Statistical metrics, used in experimental part are presented below.

- Mean

$$\tilde{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i)\right), \tag{1}$$

  where n - number of items in the sample, $x_1, x_2, ..., x_n$ - sample

- Standard deviation (SD)

$$s_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \tilde{x})^2}, \tag{2}$$

  where N - size of the sample.

- Correlation coefficient

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \tilde{x})(y_i - \tilde{y})}{(n-1)s_x s_y} \tag{3}$$

- Boxplot

  Boxplots [23] are a simple yet effective way to summarize data using five key values: the minimum (extreme lower), the maximum (extreme upper), the first quartile (Q1), the median (second quartile, Q2), and the third quartile (Q3). This technique allows us to calculate the range (Max − Min) and the interquartile range (IQR, Q3 − Q1). The IQR helps us set boundaries for distinguishing normal data from outliers.

3.2 K-means clustering

K-means clustering is an unsupervised machine learning algorithm, which groups data points based on the distance [22]. Algorithm assigns data point to a cluster based on its similarity to other data point (similarity within a cluster should be maximal, while similarity between separate clusters should be minimal). Number of clusters has to be predefined. Algorithm works in tree steps - sets initial center point, assigns data points that are closest to respective center point based on chosen distance metric, specifies center point according to formed clusters. A successful K-means implementation depends on chosen distance metrics, which goal is to obtain an appropriate similarity function. In this case we use Euclidean distance, which is defined below.

$$d(X, Y) = \sqrt{\sum_{i=1}^{m}(x_1 - y_1)^2} \tag{4}$$

3.2.1 Elbow method

There are two common ways to identify optimal number of clusters - Elbow method [4] and Silhouette score [21]. Elbow method is a visual technique. It plots a graph shaped like an elbow. The graph represents the variance explained by different numbers of clusters. Variance within each cluster (eq. 5) is plot against various k values. We consider k to be an optimal number of clusters at the point were graph bends like an elbow.

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2, \tag{5}$$

where C_i is the set of points in cluster i, and μ_i is the centroid of cluster i.

3.2.2 Silhouette score

Similar to Elbow method, the purpose of Silhouette score is So evaluate how well clusters are arranged. The range of silhouette score is between 1 and -1. Score equal or close to 1 means that clusters are clearly arranged, 0 means that differences between clusters are insignificant, and 1 means that clusters are not properly arranged. Silhouette score is calculated by eq. 6.

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}, \text{ where} \tag{6}$$

$a(i)$ is average dissimilarity of object i to all other objects of cluster A to which object i has been assigned and $b(i)$ is average dissimilarity of object i to all objects of B (cluster B is the second-best choice for object i).

3.3 T-Distributed Stochastic Neighbor Embedding

T-Distributed Stochastic Neighbor Embedding (t-SNE) [26] is intended for high-dimensional data visualization. As a part of unsupervised machine learning algorithms, it does not require labeled data. This method is a variation of Stochastic Neighbor Embedding [7], it gives each data point a location in two or three-dimensional map by reducing the tendency to pound data points together in the center of the map. t-SNE works in following steps:

1. It calculates conditional probabilities reflecting similarities between data points using distance metrics. We use Euclidean distance to do this exercise.

$$p_{j|i} = \frac{exp\left(-\frac{||x_i - x_j||^2}{2\sigma_i^2}\right)}{\sum\limits_{k \neq i} exp\left(-\frac{||x_i - x_k||^2}{2\sigma_i^2}\right)}, \text{ where}$$

   $||x_i - x_j||^2$ is Euclidean distance and σ_i^2 is the variance of the Gaussian kernel centered at x_i. Since we are interested only in modeling pairwise similarities, we set $p_{i|j} = 0$.

2. Data points are randomly placed in smaller dimension, where conditional probabilities are calculated:

$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum\limits_{k \neq i} exp(-||y_i - y_k||^2)}$$

   If the map points y_i and y_j correctly model the similarity between the high dimensional data points x_i and x_j, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal.

3. By minimizing the sum of Kullback-Leibler divergences (eq. 7) over all data points using a gradient descent method (eq. 8), SNE aims to minimize the mismatch between $p_{j|i}$ and $q_{j|i}$.

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}, \text{ where} \tag{7}$$

   P_i is conditional probability distribution over all other data points given point x_i, Q_i is the conditional probability distribution over all other map points given point y_i.

$$\frac{\delta C}{\delta y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j). \tag{8}$$

3.4 Functional Data Analysis

This section introduces the concepts of Functional Data Analysis [19], [12] utilized in this thesis.

The main idea of this statistical method is to analyze data that is the form of functions, curves or images. FDA focuses on data that are best represented as continuous functions such time and space. Functional data sample is defined in eq. 9.It means that N curves are observed on an interval $[T_1, T_2]$ and J_i is a number of data points for curve i.

$$x_n(t_{j,n}) \in \mathbb{R}, t_{j,n} \in [T_1, T_2], n = 1, 2, ..., N, j = 1, ..., J_n. \tag{9}$$

The fundamental idea of FDA is that we study smooth curves for which the values $x_n(t)$ exist at any point t, but are observed only at selected points $t_{j,n}$ (10

$$\{x_n(t) : t \in [T_1, T_2], n = 1, 2, ..., N\}. \tag{10}$$

3.4.1 Fourier Smoothing

Usually, the first step in working with functional data is its expression through basis expansion (eq. 11). Smoothing reconstructs function $x_j(t)$ to remove excessive noise, improve statistical inference or estimation.

$$X_n(t) \approx \sum_{m=1}^{M} c_{nm} B_m(t), 1 \leq n \leq N, \tag{11}$$

where B_m are chosen basis function.

Due to data structure, Fourier basis functions (eq. 12) are used in this research. Fourier basis is periodic, which is suitable for seasonal data, it fits best cases where beginning and end of an interval has approximately the same values.

$$x(t) \approx \lambda_0 + \lambda_1 sin(wt) + \lambda_2 cost(wt) + ... + \lambda_{2M-1} sin(Mwt) + \lambda_{2M} cos(Mwt) \tag{12}$$

composed by $K = 2M + 1$ basis functions:

$$\psi_0(t) = 1, \psi_{2r-1}(t) = sin(rwt), \psi_{2r}(t) = cos(rwt),\ r = 1, 2, ..., M.$$

Parameter w determines the period $2\pi/w$ which is equal to the length of the interval τ ($w = 2\pi/T if \tau = [0, T]$).

3.4.2 Cross-validation

To determine optimal K for Fourier smoothing we use Cross-validation. This technique is used to evaluate the performance of the model on unseen data. General idea is that we split our data set into multiple folds, that are used in training and testing the model. We iterate through this process each

time calculating Mean Squared Error (MSE) (eq. 13) using specific K in order to evaluate how well K fits our data. The lower MSE is, the more suitable K is to our basis function.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{13}$$

3.4.3 Functional Boxplot

In the classical Boxplot, the box itself represents the middle of 50% of the data, while in FDA, data ordering progresses from the center outwards. The 50% central region is defined by the band which is delimited by the $\alpha(0 < \alpha < 1)$ proportion of the deepest curves from the sample is used to estimate the α central region. This region's border is marked by the envelope representing the box in a traditional Boxplot. Similar to the IQR, this 50% central region offers insight into the spread of the central 50% of the curves. It's robust for interpretation as it remains unaffected by outliers or extreme values, providing a less biased visualization of the curves' spread. The observation within the box signifies the median, a robust statistic for measuring centrality. The "whiskers" of the Boxplot are the vertical lines extending from the box, denoting the maximum envelope of the dataset excluding outliers.

3.4.4 Outlier detection

Outliers in functional data set can be two types:

1. curves with gross errors, such as mistakes in typing, measurements or recording;
2. curves that are somehow abnormal, do not follow the same patter as the rest of the curves.

The first type of outlier typically needs to be addressed through elimination or correction to maintain data integrity. However, the second type of outlier can offer valuable insights into the dataset, aiding in the identification of abnormalities such as faults, diseases, or other deviations from the norm. A curve can be an outlier if it is significantly far away from the expected function or is in different shape than the rest of the curves during the same period of time. To identify outliers in functional data we use functional depths. If there is a curve representing an outlier, it will have a significantly low depth. Depth measures how deep (central) is a datum point in respect to the rest of the population. In univariate data, the median would be the deepest point. There are multiple methods to calculate band depth, how to calculate the halfspace and the simplicial depths of x_i is provided in the eq. 14.

$$\begin{aligned} HD_n(x_i) &= min\{F_n(x_i), 1 - F_n(x_i)\} \\ SD_n(x_i) &= 2F_n(x_i)(1 - F_n(x_i)), \end{aligned} \tag{14}$$

where F_n is the empirical cumulative distribution function of the sample $x_i, ..., x_n$:

$$F_n(x) = \frac{1}{n}\sum_{k=1}^{n}\mathbf{1}\{x_k \leq x\}, x \in \mathbb{R}.$$

3.4.5 Functional K-means clustering

Functional K-means clustering [25] extends the traditional K-means clustering algorithm to handle functional data. Unlike traditional K-means clustering, which deals with scalar data points, functional K-means clustering works with data that are functions over a continuum, such as time or space. To perform clustering, a suitable distance measure between functions is required. Here, we use L^2 norm (eq. 15), commonly referred to as the Euclidean norm. Similarly to K-means algorithm described in section 3.2, functional K-means randomly selects initial centroids or use methods such as hierarchical clustering for initialization. Next step is to assign each function to the nearest centroid based on chosen distance measurement. Once all functions are assigned, centroids are recalculated by averaging the functions in each cluster.

$$||f - g||_2 = \left(\int_a^b (f(t) - g(t))^2 dt \right)^{\frac{1}{2}}, \tag{15}$$

where $f(t)$ and $g(t)$ are defined on the interval $[a, b]$.

4 Experimental results

4.1 Data

The data set used in this research consist of 15 minutes resolution observations of 143 strings, each string is connected to one of 12 inverters, which are then connected to one of 3 blocks. Structure of this solar panel is detailed in Table 1. This photovoltaic system is installed in the eastern part of Lithuania. Observations cover period from 2022-03-01 until 2023-10-04 with an exception of a few days where solar power plant was turned off due to maintenance work - this issue is further explained in the pre-processing section. Original data set contains observations of inclined irradiance, string current measured in ampere (A) and string voltage measured in volts (V), however this work is based on electrical power observations, which is a product of current and voltage. It is know that all parts of this PV plant have the same technical parameters, thus assumption is that each string generates the same amount of energy under the same conditions. From data visualizations (figures 62, 63 and 64), it is clear, that we are dealing with seasonal time series data with significant decrease in electrical power generation during winter months.

Block	Inverter	String
No.1	INV-1	1-1 string to 1-13 string
	INV-2	2-1 string to 2-13 string
	INV-3	3-1 string to 3-13 string
	INV-9	9-1 string to 9-13 string
	INV-10	10-1 string to 10-13 string
No.2	INV-4	4-1 string to 4-12 string
	INV-5	5-1 string to 5-12 string
	INV-8	8-1 string to 8-12 string
	INV-11	11-1 string to 11-12 string
No.3	INV-6	6-1 string to 6-10 string
	INV-7	7-1 string to 7-10 string
	INV-12	12-1 string to 12-10 string

Table 1: Structure of Solar Power Plant and annotations

4.1.1 Data pre-processing

It is clear from unprocessed data's illustrations (figures 62, 63 and 64) that chosen data set contains some missing values. We know that solar power plant was turned off for maintenance from 2022-05-24 until 2022-06-02. It was decided to remove all observations until 2020-06-02. Missing values were also noticed on 2022-09-15 between 2 p.m. and 3 p.m. and on 2023-06-29 between 12 p.m. and 3 p.m, however reason behind this is unknown. To estimate these missing points spline interpolation method was used. Final step in determining data frame for further analysis was removing night time hours. Since sun is essential in solar power generation, removing night hours significantly reduces size of the data set without compromising data quality. Removing observations measured between 8 p.m. and 6 a.m. reduced the size of data set almost in half. To get better understanding of each string, zoomed in

view on period starting from 2023-06-12 and ending on 2023-06-18 are also included (figures 2, 3 and 4. From these illustrations, we can see that some strings generate lower amount of electrical power than other strings connected to the same inverter.

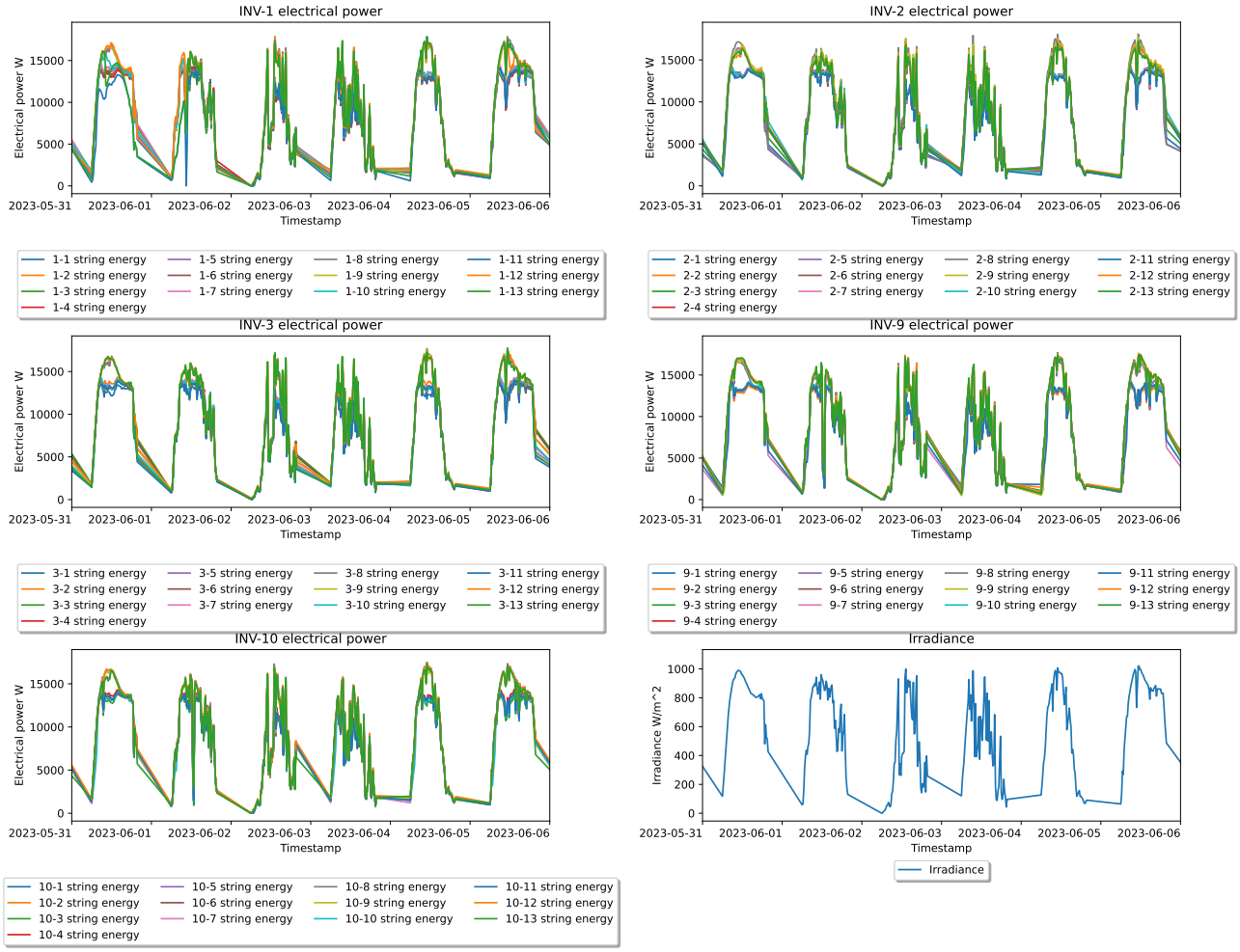

Figure 2: Block no.1 electrical power and irradiance after pre-processing

4.1.2 Descriptive statistics

To better understand data patterns we look deeper into descriptive statistics. Table 2 represents ten days each with the highest mean, the lowest mean, the highest standard deviation (SD) and the lowest SD. While it may seem intuitive that days with the highest mean of electrical power correspond to days when the most solar energy was generated, this assumption may not always be true. To test this, we calculated how much electrical power was generated daily. Computations proved that days with the highest mean are the same as days with highest generation results. It is interesting that out of 10 days where electrical power generation was the highest, 9 days are from 2023 either May or beginning of June. Based on Lithuanian Hydrometeorological Service data, average duration of solar radiance in May 2023 was 346.8 hours and in June 2023 it was 312 hours while in May 2022 it was 237.7 and in June 2022 it was 280 hours. By looking into days with the lowest mean we can see that more days are from 2022 than from 2023. It was reported that average duration of solar radiance in November 2022

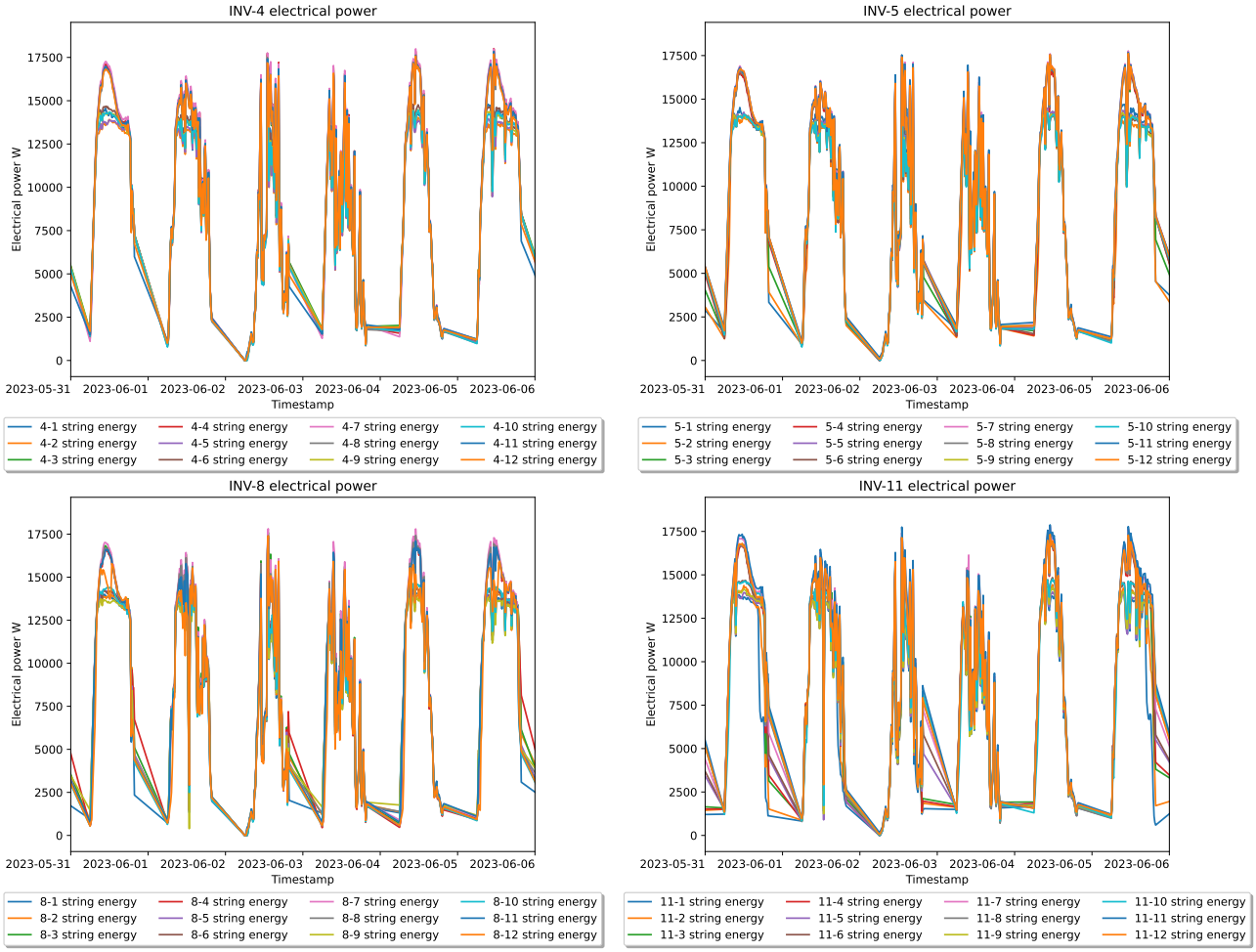

Figure 3: Block no.2 electrical power after pre-processing

was 18.2 hours, in December 2022 it was 19 hours while in January 2023 it was 14 hours. January 2023 stands out because a new record was set of 5.1 hours of average duration of solar radiance in Vilnius.

Another important measurement is SD. It provides valuable insights into the dispersion of our data. From the table 2 we can see that data points are spread out over a wider range from the mean in months with high solar radiance, while in months with low solar radiance data values are significantly closer to the mean. This implies that electrical power generation fluctuates widely over time. This inconsistency is mostly because of different seasons, but also could be impacted by weather conditions, clouds coverage, shading or other environmental factors. High dispersion may also suggest inefficiencies or performance issues in solar energy systems. It could indicate flaws in system design, maintenance issues, or limitations in technology that affect the consistency and reliability of power generation. Further steps of this research will show the impact of faults such as shading to electrical power generation.

To illustrate descriptive statistic of our data set we use Box plots (figures 5 and 6). We can see that for all strings, median is quite small, it is around 2500 W, while the upper extreme is more than 17500 W. This suggests that our data set is positively skewed, indicating that observations of strings generating very high amount of electrical power are less frequent. Interquartile Range (IQR) looks

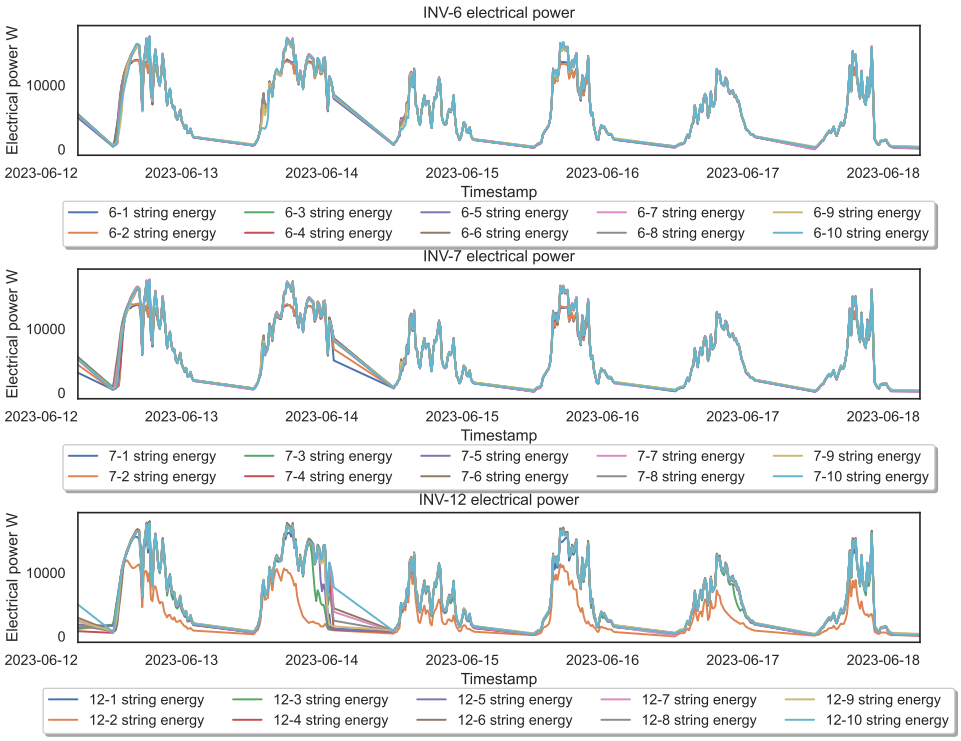

Figure 4: Block no.3 electrical power after pre-processing

Day	**Highest mean**	**Day**	**Lowest mean**	**Day**	**Highest SD**	**Day**	**Lowest SD**
2023-06-06	12461.2	2022-11-21	142.4	2022-06-22	1798.96	2023-01-03	41.67
2023-06-05	12449.3	2023-01-09	139.5	2022-06-24	1738.95	2022-12-30	41.58
2023-05-31	12330.0	2022-12-06	138.0	2022-06-23	1704.32	2023-01-02	41.29
2023-06-09	12324.4	2023-01-11	127.8	2022-06-25	1592.2	2022-12-30	40.51
2023-05-28	12146.5	2022-12-20	127.0	2022-06-26	1582.12	2023-01-02	39.54
2023-05-07	12117.8	2022-12-11	124.5	2022-06-27	1492.47	2022-12-31	39.32
2022-06-22	12076.1	2022-11-23	109.0	2022-07-21	1109.79	2023-01-10	39.09
2023-05-06	12033.7	2022-11-27	108.9	2022-07-20	1083.95	2023-01-17	38.31
2023-05-08	11935.4	2022-12-07	82.3	2022-07-22	1041.19	2023-01-13	38.04
2023-06-07	11899.7	2022-12-09	78.8	2022-08-11	1041.04	2023-01-01	34.74

Table 2: Descriptive statistics

very consistently, 50 % of the time generated electrical power is between 0 W and 10000 W. Despite the similarities of the strings, there are also very significant differences. We can see that performance of some strings connected to the same inverter can be inferior, for instance, string no.1 generates less power than 15000 W, while other 8 strings connected to the same inverter no.7 generates more than 17500 W. Nevertheless, string no.2 connected to inverter no.12 stands out the most. With median around 1200 W, IQR between 0 W and 4000 W, it also has outliers. From this we can assume, that string 12-2 is affected by outside factors.

As already mentioned, we have inverters with the same technical parameters that are installed in the same location, hence we expect that they would generate almost the same amount of power under the

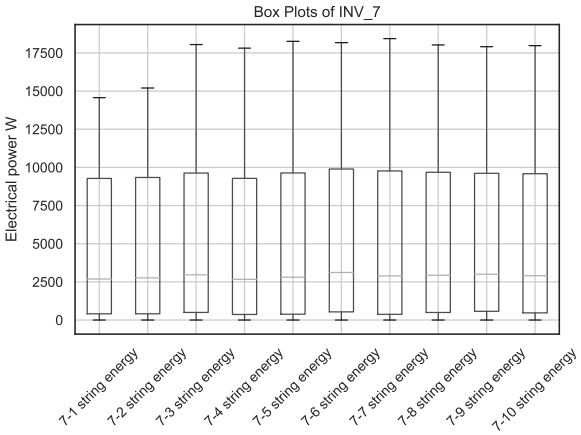

Figure 5: Box plot of strings connected to inverter no.7

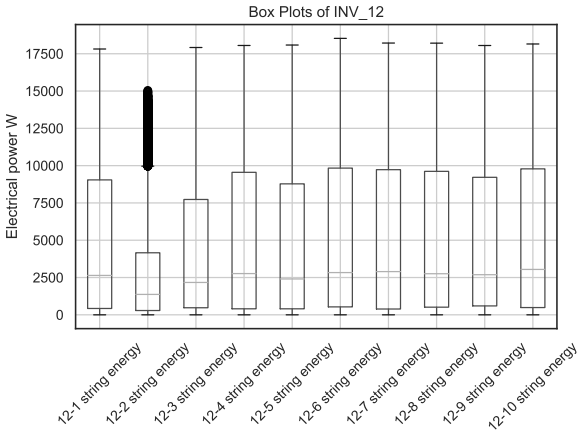

Figure 6: Box plot of strings connected to inverter no.12

same irradiance. To examine that relationship we use correlation heatmaps. While majority of strings (figure 7 demonstrate strong relationship with correlation equal or close to 1, string no.2 connected to inverter no.12 (figure 8) demonstrates deviation from perfect correlation. This exception suggests that string 12-2 has a weaker relationship with other strings and its generation might be impacted by other conditions such as shading.

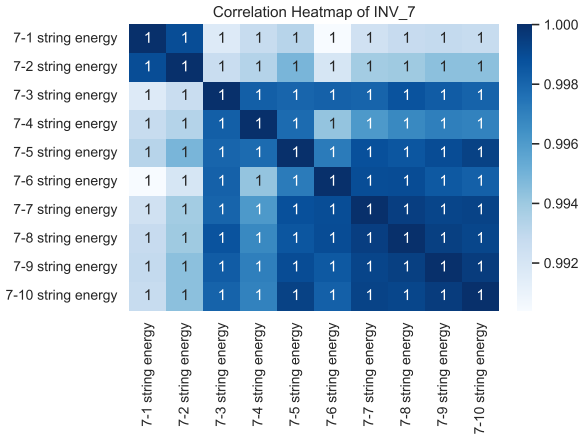

Figure 7: Correlation of strings connected to inverter no.7

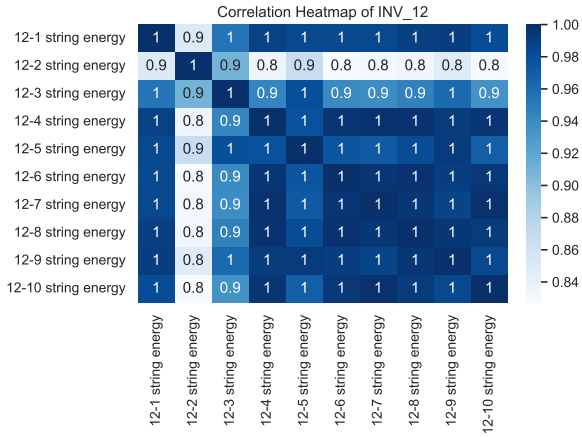

Figure 8: Correlation of strings connected to inverter no.12

Descriptive statistics provided some important insights about our data. We investigated mean, median, SD, IQR and correlation. It proved that some strings can generate significantly smaller amount of power despite identical technical parameters and the same irradiance. This suggests that there is a possibility of faults which impact generation process. To investigate this further we use different clustering methods.

4.2 K-means clustering

Clustering part of the experiment is done on case to case basis. We take 10 days on which electrical power generation was the highest and apply K-means algorithm. To find the optimal k for K-means algorithm both elbow and silhouette methods are used. Assuming identical technical characteristics for each string and consistent irradiance levels, we anticipate uniform electrical power output across all strings. To validate this assumption, we calculate the covariance, anticipating it to approach 1. Minor deviations are expected, as factors such as uneven terrain or varying tilt angles may slightly influence this measurement. In this part strings are not divided to inverters or blocks. Case to case analysis presented in the descending order according to table 2.

- **2023-06-06**

  Based on elbow test (figure 9), it is not very clear whether we should choose 3 or 4 as optimal k for K-means clustering, however silhouette method (figure 10) suggest 4 as an optimal k. From the K-means results (figure 11) we can see that majority of strings are distributed evenly through classes 0 and 1, but there are way less strings that were assigned to class 2 and only 1 string was assigned to class 3. It appears that **12-2 string** was classified to 3 group. From the descriptive statistics we know, that this string stands out in comparison to others. Following 6 strings were assigned to class 2 - 11-1 string, 11-2 string, 12-1 string, 12-3 string, 12-5 string and 12-9 string. By comparing how much electrical power each string generated during the day (from 6 a.m. until 8 p.m.), we see that 12-2 string generated the least amount of power (362677 W), while string 11-11 generated the most electrical power (758248 W). Strings classified to group 2 generated electrical power from 593395 W to 682983 W. By scrutinizing the correlation outcomes across all pairs of strings, we pinpoint 319 pairs exhibiting correlations below 0.7. Specifically, the correlation between string 12-2 and strings 7-1, 7-2, 9-11, and 10-10 falls below 0.3. This observation suggests that despite uniform irradiance conditions, these string pairs demonstrate diverse behaviors.

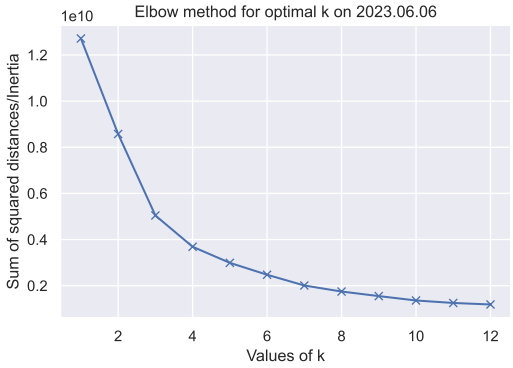

Figure 9: Elbow analysis for optimal k

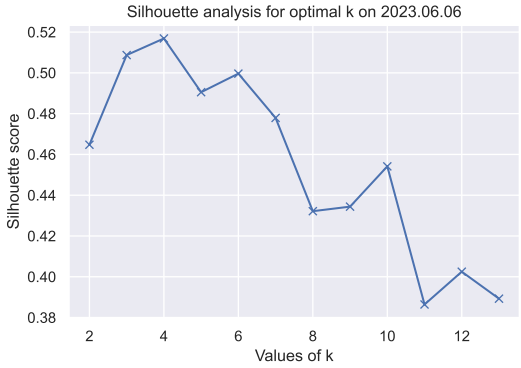

Figure 10: Silhouette analysis for optimal k

- **2023-06-05**

  Results of the day before (figure 12) look very similar to the first case. With the help of elbow and silhouette methods we determine that optimal number of clusters is 4. Again, we see that majority of strings are divided between clusters 0 and 1 and distribution of remaining clusters is the same

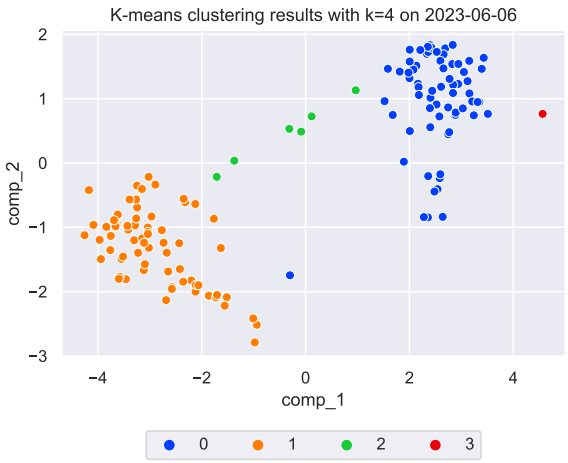

Figure 11: 2023.06.06 K-means clustering results

as in the first case. However, there are slight changes in generation of electrical power - **string 12-2** remains the one that generates the smallest amount of power during the day (333558 W), string 1-8 generated the most power (768848 W), strings classified to group 2 generated between 576057 W and 684957 W. We see that while string 12-2 generated less electrical power than the day before, best performing string generated more power than the day before. On this particular day, we observe 332 pairs of strings with correlations below 0.7. Among these, there are 24 unique strings that correlate with string 12-2, each with a correlation smaller than 0.2.

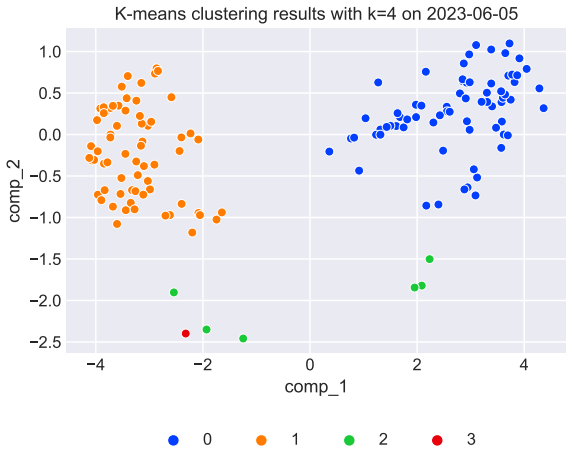

Figure 12: 2023.06.05 K-means clustering results

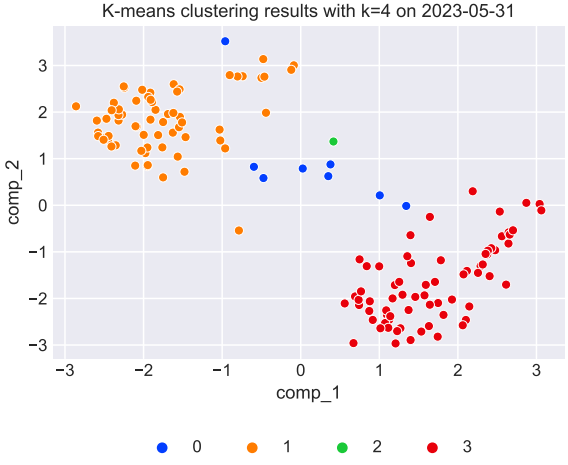

Figure 13: 2023.05.31 K-means clustering results

- **2023-05-31**

  On this day we have a little bit different distribution of clusters (figure 13). Majority of strings are grouped to either 1 or 3 cluster. 8 strings were assigned to cluster class 0 and **12-2 string** is classified to 2 class alone. Group 0 consist of these strings - 1-11, 1-13, 11-1, 11-2, 12-1, 12-3, 12-5 and 12-9 strings. Electrical power generation of 12-2 string was higher (376491 W) than on 5 and 6 of June. Power generation of strings assigned to cluster 0 is between 551329 W and 670178 W. String 11-11 generated the most electrical power (753891 W) on this day, but slightly lower amount than on 5 and 6 of June. We have identified a total of 181 pairs with correlation coefficients smaller than 0.7. The lowest correlation coefficient was observed between String 12-2 and 17 other strings linked to inverters 2, 6, 7, 9, or 10, resulting in correlation coefficients ranging from 0.4 to 0.3.

- **2023-06-09**

  In this case, silhouette results contradict the results of elbow method for optimal k (figures 14 and 15), thus we compromise and take the middle value, in this case we choose k equal to 3. From figure 11 we can see that majority of strings are either in clusters 0 or 1, remaining three strings belong to cluster 2, it includes **11-1, 12-2** and **12-3 strings**. Electrical power generation results of the day are as follows: string 12-2 generated 346751 W, string 12-3 generated 572583 W and string 11-1 generated 581024 W, the rest of the strings generated between 626049 W and 753132 W. The biggest amount of energy was generated by 1-8 string. On this day, we identify 331 pairs of strings with correlation coefficient lower than 0.7. String 12-2 continues to demonstrate notably low correlation, with 12 pairs showing coefficients lower than 0.3.

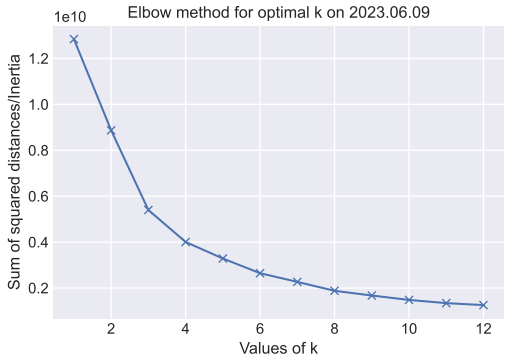

Figure 14: Elbow analysis for optimal k

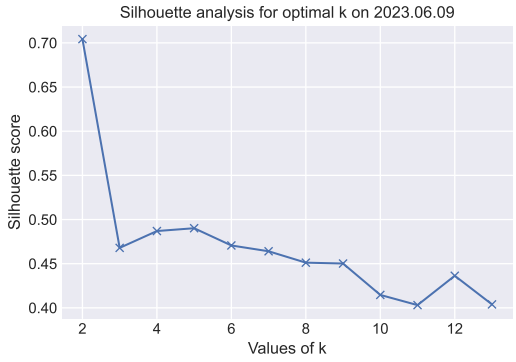

Figure 15: Silhouette analysis for optimal k

- **2023-05-28**

  This case (figure 17) is very similar to 2023.05.31. Chosen k is equal to 4, **string 12-2** is once again classified as cluster 2, majority of strings are assigned to clusters 1 and 3, cluster 0 consist of 11-1, 11-2, 12-1, 12-3, 12-5 and 12-9 strings. String 12-2 generated the smallest amount of power (373757 W) during the day, while string 11-11 generated the highest amount of power (738361 W). Strings assigned to cluster 0 generated electrical power between 585535 W and 664404 W. Moreover, we observe 193 pairs of strings with correlations coefficient below 0.7. Among these,

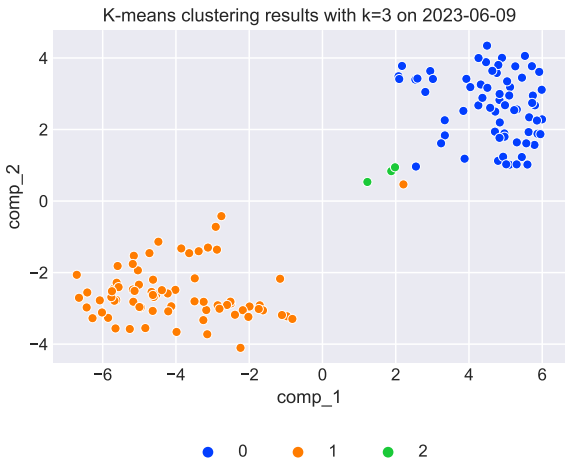

Figure 16: 2023.06.09 K-means clustering results

there are 30 unique strings that correlate with string 12-2, each registering a correlation smaller than 0.4, but not exceeding than 0.3.

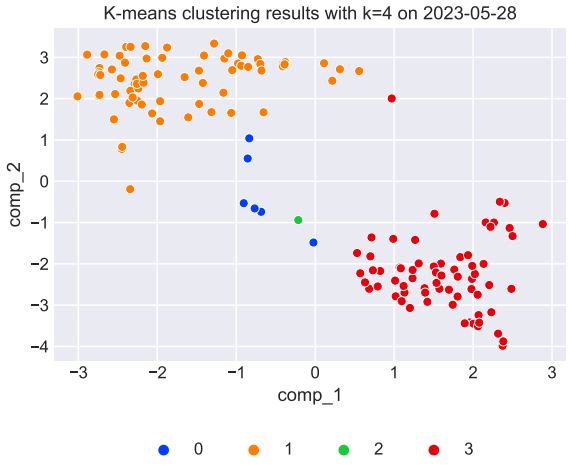

Figure 17: 2023.05.28 K-means clustering results

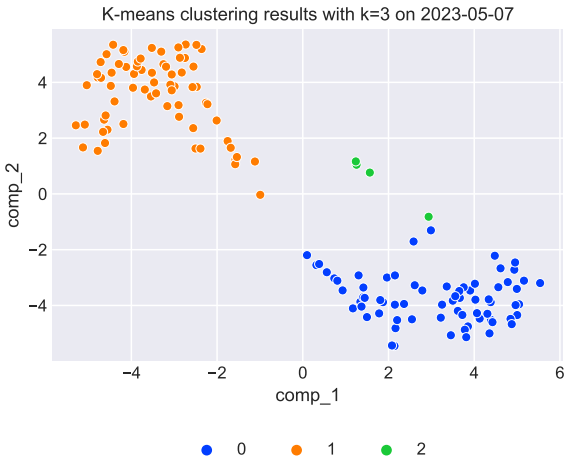

Figure 18: 2023.05.07 K-means clustering results

- **2023-05-07**

  Similarly to 2023.05.28 case, we choose $k = 3$ (figure 18). With such k, we get that majority of strings belong either to cluster 0 or 1. Four strings were recognized as part of cluster 2: **11-1, 12-2, 12-3** and **12-5 strings.** As per previous cases, string 12-2 generated the least electrical power (422143 W), while string 4-7 generated the most power (741253 W). Strings 11-1 and 12-3 generated 588876 W and 611332 W respectively, the rest of the strings generated more than 169214 W. We have identified a total of 164 pairs with correlation coefficients smaller than 0.7. The lowest correlation coefficient was observed between String 12-2 and 11 other strings linked to inverters 1, 2, 3, 9, or 10, resulting in correlation coefficients ranging from 0.4 to 0.3.

- **2022-06-22**

  The only case from the year 2022 that is among 10 the highest generating days. For this case, elbow and silhouette methods' results are not completely consistent (figures 19 and 20), however we choose $k = 2$. From figure 21, we see that majority of strings are assigned to cluster 0, while 11 strings are assigned to cluster 1. All 11 string from cluster 1 are connected to **inverter no.8**, only string 8-12 is assigned to class 0. Strings from cluster 1 generated from 367863 W to 394143 W. It is worth to mention that string 8-12 generated 545155 W of electrical power, string 12-2 generated 369514 W despite being classified to cluster 0. String 11-11 outperformed other strings by generating 782379 W of electrical power during the day. In this case, there are 164 pairs of strings with correlation coefficient lower than 0.7. 11 strings are in pair with string 12-2 with correlation between 0.4 and 0.3.

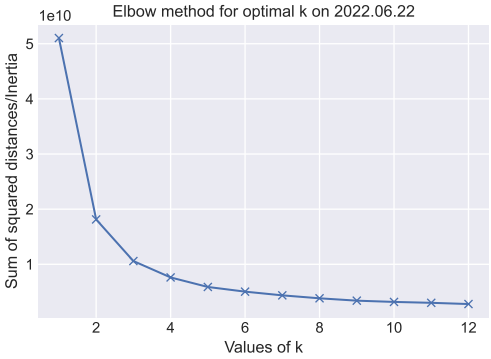

Figure 19: Elbow analysis for optimal k

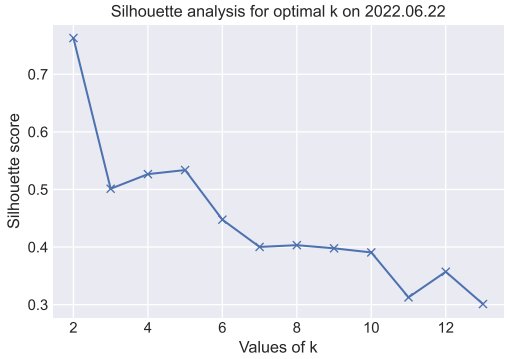

Figure 20: Silhouette analysis for optimal k

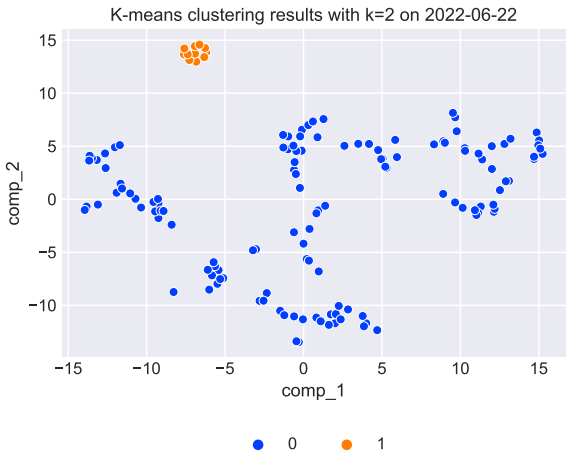

Figure 21: 2022-06-22 K-means clustering results

- **2023-05-06**

  For this day we get that optimal $k = 3$. From figure 22, we see that majority of strings are divided between cluster classes 0 and 1. **Strings 11-1, 12-2, 12-3 and 12-5** are assigned to cluster 2. Looking into daily electrical power generation, we see that string 12-2 generated the smallest amount of power (422407 W), string 11-11 generated the most power (745542 W), strings

11-1, 12-3 and 12-5, generated 567111 W, 613764 W and 652273 W respectively. The rest of the strings generated between 608156 W and 741911 W. In this case, we have identified 164 pairs of strings with correlation coefficients below 0.7, where 24 strings are paired with string 12-2, each displaying correlation coefficients lower than 0.2.

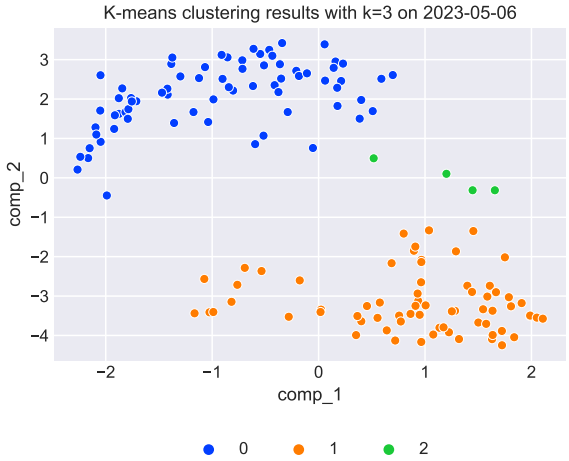

Figure 22: 2023.05.06 K-means clustering results

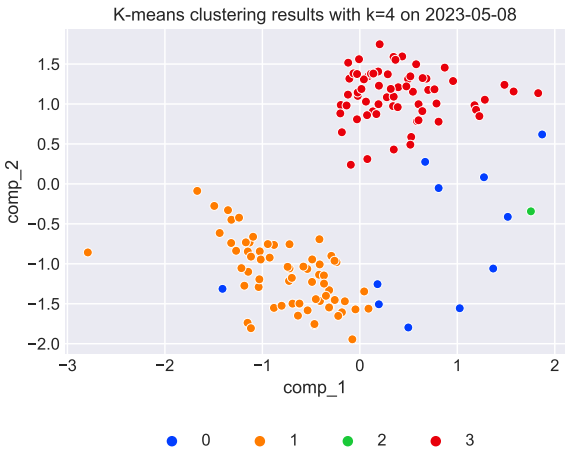

Figure 23: 2023.05.08 K-means clustering results

- **2023-05-08**

  From results of elbow and silhouette methods, we determine $k = 4$. From the figure 23 we see, that majority of strings are divided between classes 1 and 3. 11 strings (8-8, 8-10, 8-11, 8-12, 11-1, 11-2, 11-3, 12-1, 12-3, 12-5 and 12-9) are classified to cluster 0. Once again the only string assigned to cluster 2 is **string 12-2** with daily electrical power generation of 412719 W. On this day, string 4-7 generated the highest amount of power (721553 W). By scrutinizing the correlation outcomes across all pairs of strings, we pinpoint 141 pairs exhibiting correlations below 0.7. Specifically, the correlation between string 12-2 and string 9-11 falls below 0.4 but not exceeding 0.3.

- **2023-06-07**

  To cluster this case, it was decided to use $k = 2$. K-means clustering results (figure 24) show that majority of clusters are assigned to class 0 and **strings 11-1, 12-2, 12-3 and 12-5** are assigned to class 1. Daily generated amount by string 12-2 is 337730 W, while string 11-11 generated the most electrical power (724301 W.) String 11-1, 12-3 and 12-5 produced 564765 W, 576164 W and 629315 W respectively. For the current day, we have found 312 pairs of strings exhibiting correlation coefficients below 0.7. We observe that correlation of strings 9-11 and 12-2 is between 0.4 and 0.3, while correlation of strings 6-10 and 12-2 is lower than 0.3.

The analysis of K-means clustering reveals several patterns. It becomes evident that string 12-2 frequently experiences interruptions due to external factors, such as shading, leading to reduced electrical power generation. Moreover, we observe significant variations in power generation among strings connected to the same inverter, despite identical irradiance levels. However, it is essential to note that the preceding section highlights a limited number of instances, most of which occurred within

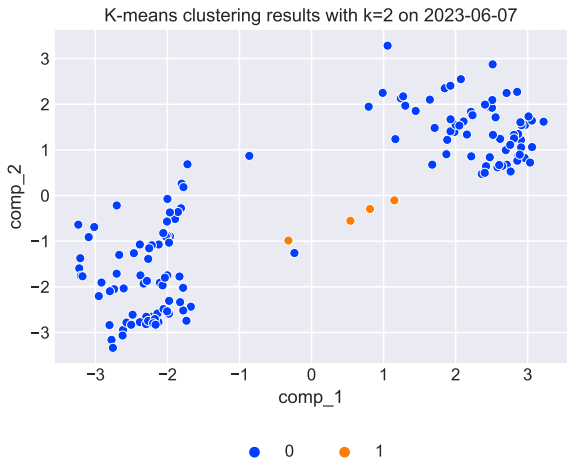

Figure 24: 2023.06.07 K-means clustering results

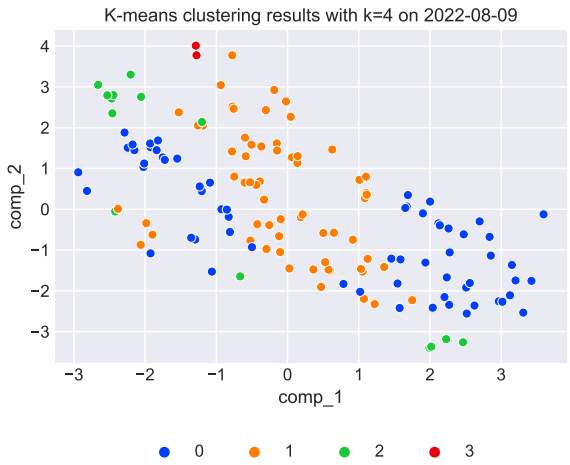

Figure 25: 2022.08.09 K-means clustering results

a one-month period. For a more comprehensive evaluation of K-means clustering ability to identify outliers, we must examine data from other days, months, or even seasons. Therefore, we pinpoint five days exhibiting an average level of daily electrical power generation and we delve into the results of K-means clustering applied to these selected days.

- **2022-08-09**

  After analyzing the results obtained from the elbow and silhouette methods, we determined that the optimal number of clusters is $k = 4$. Figure 25 illustrates that the majority of strings are divided between classes 0 and 1. **Strings 2-9** and **12-2** are classified as cluster 3, while 14 strings (5-1 string, 8-1 string, strings 1, 2, 3, 4 and 6 connected to inverter no. 11, strings 1,3,4,5,6,8,9 connected to inverter no. 12) are assigned to cluster 2. Notably, strings 2-9 and 12-2 exhibited the lowest power generation, producing 238474 W and 160381 W, respectively. On this day, string 11-11 performed the best by generating 296063 W of power.

- **2022-09-25**

  In figure 26, we observe 4 clusters, where strings are equally distributed between classes 0 and 1, slightly smaller amount of strings are assigned to class 2. Remaining strings allocated to class 3 are **strings 8-11** and **8-12**. These two strings generated the smallest amount of power (207936 W and 166856 W respectively). 2-8 string generated the most electrical power (295515 W) throughout the day.

- **2023-06-17**

  Following evaluation with silhouette and elbow methods, we opt for $k = 3$ clusters. The distribution of strings is relatively even across clusters 0 and 1, with only string 12-2 assigned to cluster 2 (figure 27). Daily electrical power generation ranges from 130168 W to 286511 W. Notably, the lowest amount of power was generated by string 2, connected to inverter no. 12, while the highest was produced by string 11, connected to inverter no. 11.

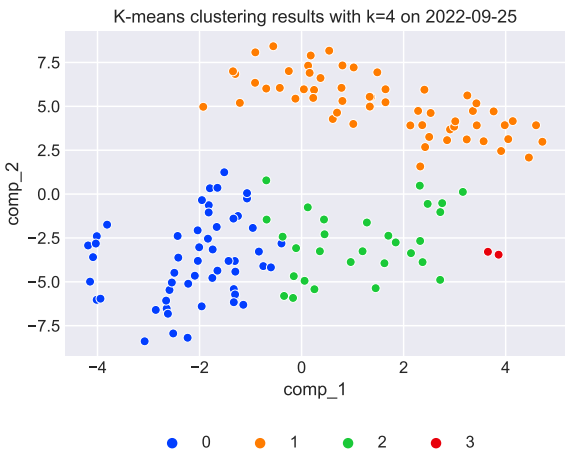

Figure 26: 2022.09.25 K-means clustering results

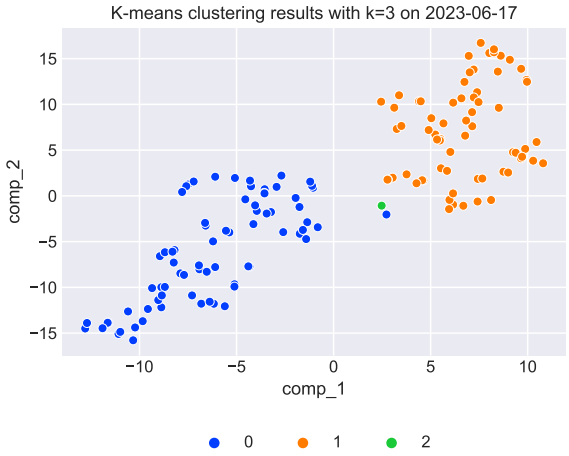

Figure 27: 2023.06.17 K-means clustering results

- **2023-07-06**

  Another case with optimal $k = 3$ (figure 24), identified relatively equal number of strings assigned to classes 0 and 1, 21 strings (strings 3 to 10 connected to inverter no. 6, strings 3 to 5 and 7 to 10 connected to inverter no. 7, strings 5 and 8 connected to inverter no.9 and strings 4, 6, 8, 9 connected to inverter no.12) assigned to cluster 2 and **12-2 string** classified to class 3. Lower limit equal to 177969 W of electrical power was generated by string 12-2, while upper limit of 290639 W was generated by string 1-12.

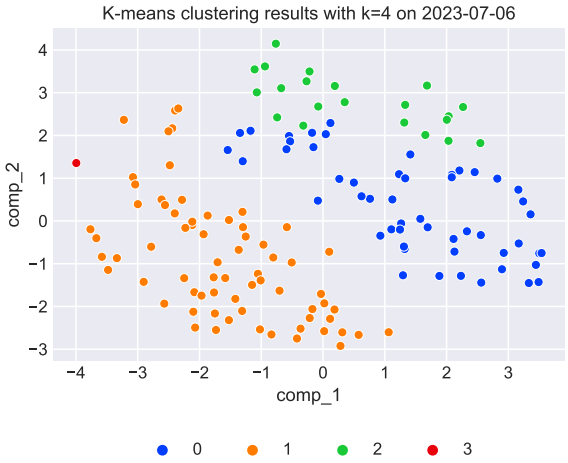

Figure 28: 2023.07.06 K-means clustering results

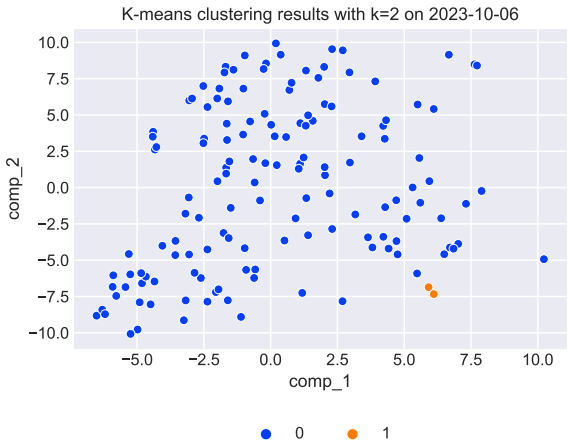

Figure 29: 2023.10.06 K-means clustering results

- **2023-10-06**

  Last but not least, we apply K-means clustering method with $k = 2$ on observation collected on October 6, 2023. From figure 29, we observe that majority of strings were assigned to class 0, while **strings 8-11** and **8-12** were classified to the second cluster. String 8-12 generated the least electrical power during the day (135967 W), while the most power (283874 W) produced string

2-8.

Day	String X	String Y	Correlation coefficient between strings X and Y
2022-08-09	String 8-12	String 12-2	0.76
2022-09-25	String 8-12	String 12-2	0.78
2023-06-17	String 8-12	String 12-2	0.89
2023-07-06	String 6-10	String 12-2	0.93
2023-10-06	String 8-12	String 12-2	0.22

Table 3: Lowest correlation coefficients

The K-means clustering results for five days characterized by the lowest mean electrical power generation are summarized in Table 4. During low irradiance periods, it is harder to notice abnormality in power generation. Low irradiance, coupled with shading or soiling, results in minimal or even nonexistent energy generation. From the table 4, we see that even neighboring strings that are connected to the same inverter, can produce very different amounts of electrical power. Overall, K-means clustering proves to be a valuable tool to investigate how each string behaves under the same irradiance, significantly lower generation indicates disturbance in the process. Faults such as soiling or shading are temporary yet cause a significant decrease in electrical power generation. Despite the uncomplicated implementation of K-means algorithm, it requires huge time resource in order to investigate case-by-case data, thus we continue to search for more efficient unsupervised methods in order to detect shading and soiling faults.

Day	k	K-means results	Daily generation interval in W	String that generated the most power	String that generated the least power
2022-11-21	2	The strings were evenly distributed between 2 clusters	[919.2; 14323]	4-7 string	1-11 string
2022-12-09	2	The strings were evenly distributed between 2 clusters	[0; 11683.3]	3-12 string	3-4 string
2022-12-20	2	The strings were evenly distributed between 2 clusters	[820.5; 13343.9]	11-11 string	9-4 string
2023-01-09	2	The strings were evenly distributed between 2 clusters	[0; 15346.3]	1-12 string	1-11 string
2023-01-11	2	The strings were evenly distributed between 2 clusters	[0;14161.7]	10-12 string	1-4 string and 1-11 string

Table 4: Summary of K-means clustering results of cases with lowest electrical power generation mean

4.3 Functional data analysis

In this section, we continue to monitor the behaviour of strings on days with high and average irradiance. We transform time series to functional data in order to detect faults on a daily basis. Data is smoothed with $k - basis = 11$, the number of clusters applied is the same as applied for traditional K-means clustering. We provide illustrations for outliers detection and functional K-means clustering.

- **2023-06-06**

  Outliers detection identifies (fig. 32), that strings 1 and 2 connected to inverter no. 11 and strings 1, 2, 3, 5 and 9 connected to inverter no. 12 act as an outliers. However, these strings are not assigned to the same cluster, Strings 1,2,3 and 5 connected to inverter no. 12 are assigned to the smallest cluster and are represented by the purple lines in figure 33.

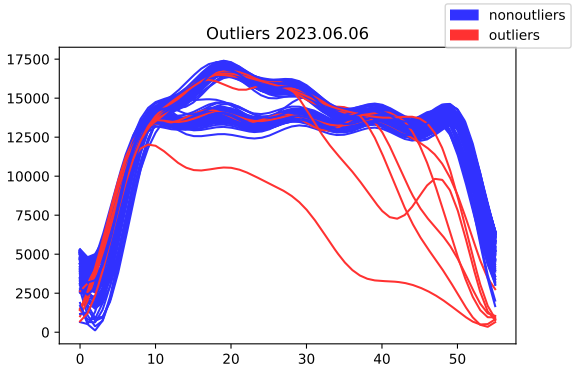

Figure 30: 2023.06.06 outliers detection

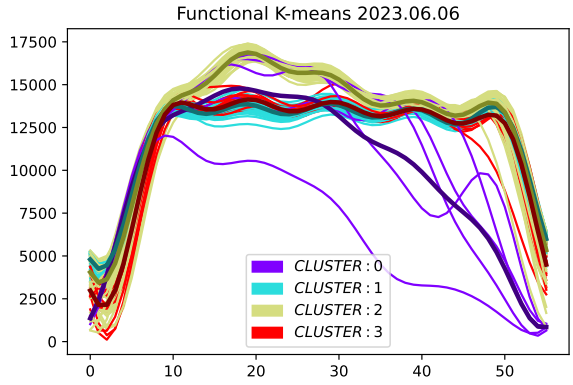

Figure 31: 2023.06.06 Functional K-means clustering

- **2023-06-05**

  Similar to the previous day, outlier detection reveals that strings 6-10, 1, and 2 connected to inverter no. 11, along with strings 2, 3, and 5 connected to inverter no. 12, are identified as outliers (figure 32). However, these strings are distributed across different clusters. Specifically, strings 11-1, 12-2, 12-3, and 12-5 belong to the smallest cluster, which is depicted by the purple lines in figure 33.

- **2023-05-31**

  From the figure 34, we observe that strings 11 and 13 connected to inverter no.1, string 1 an 2 connected to inverter no.2 and strings 1, 2, 3, 5 and 9 connected to inverter no. 12 act as an outliers. However, these strings are distributed across different clusters. Specifically, strings 1-1, 1-13, 11-1, 11-2, 12-1, 12-2, 12-3, and 12-5 belong to the smallest cluster, which is depicted by the purple lines in figure 35.

- **2023-06-09**

  Figure 36 illustrates that strings 11-1, 11-2, 12-1, 12-2, 12-3, 12-5 and 12-9 are outliers. Only, strings 11-1, 12-1, 12-2, 12-3 and 12-5 are assigned to the smallest cluster, which is depicted by the purple lines in figure 37.

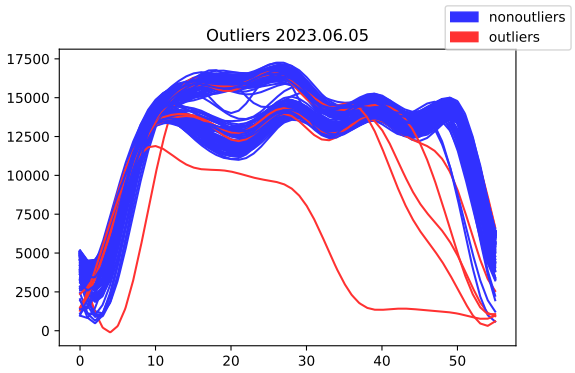

Figure 32: 2023.06.05 outliers detection

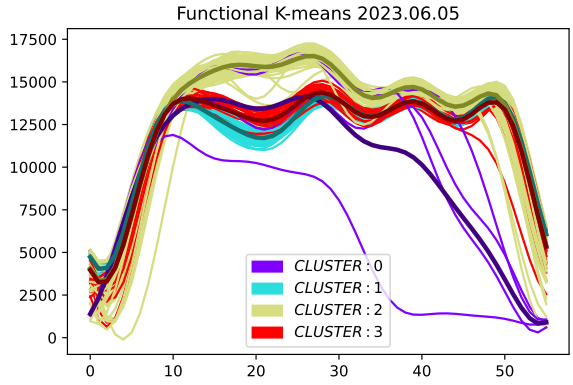

Figure 33: 2023.06.05 Functional K-means clustering

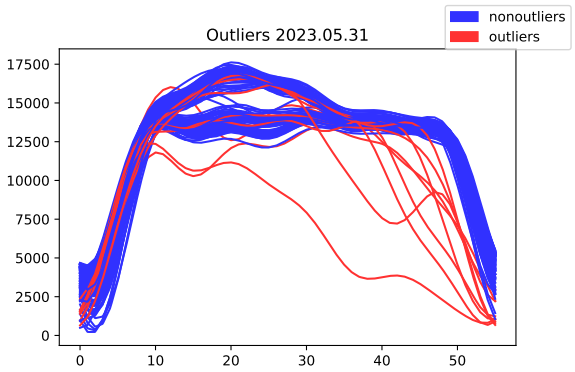

Figure 34: 2023.05.31 outliers detection

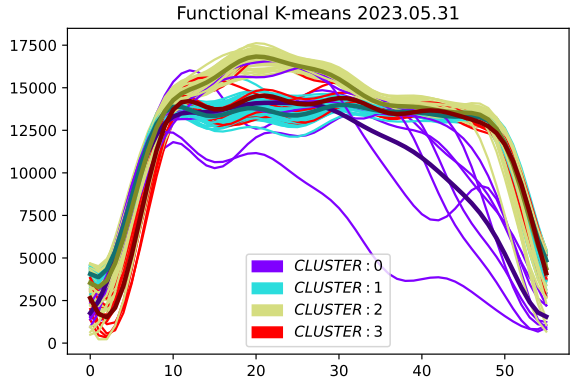

Figure 35: 2023.05.31 Functional K-means clustering

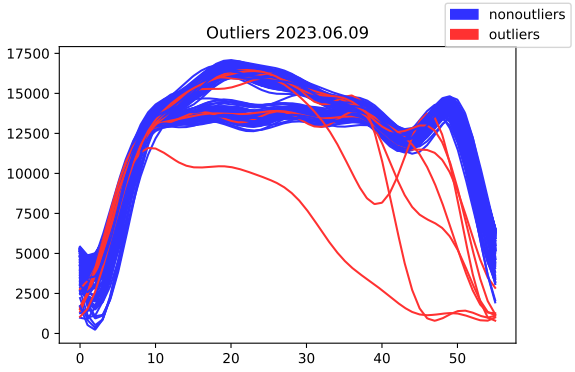

Figure 36: 2023.06.09 outliers detection

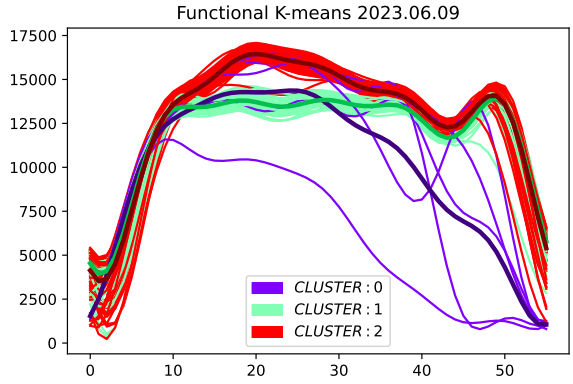

Figure 37: 2023.06.09 Functional K-means clustering

- **2023-05-28**

  On this day, we observe that strings 11-1, 11-2, 12-1, 12-2, 12-3, 12-5 and 12-9 are identified as outliers (figure 38). But, strings 11-1, 12-1, 12-2, 12-3 and 12-5 are assigned to the smallest cluster, which is illustrated by the blue lines in figure 39.

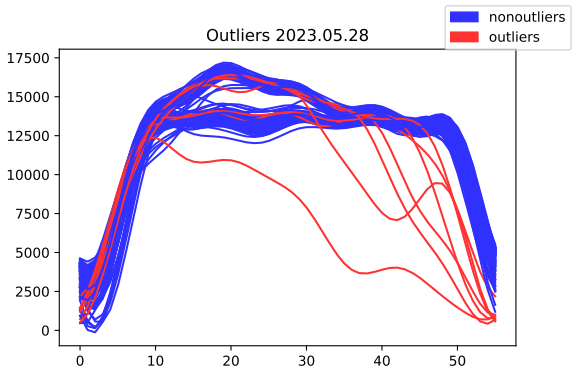

Figure 38: 2023.05.28 outliers detection

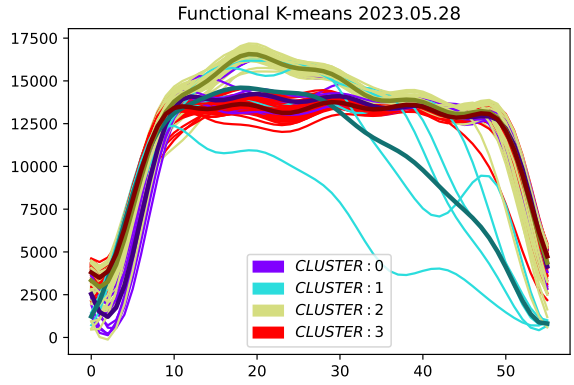

Figure 39: 2023.05.28 Functional K-means clustering

- **2023-05-07**

  On this day, we observe that strings 11-1, 11-2, 12-2, 12-3, 12-5 and 12-9 are identified as outliers (figure 40). However, strings 11-1, 12-2, 12-3, and 12-5 are assigned to the smallest cluster, illustrated by the purple lines in figure 41.

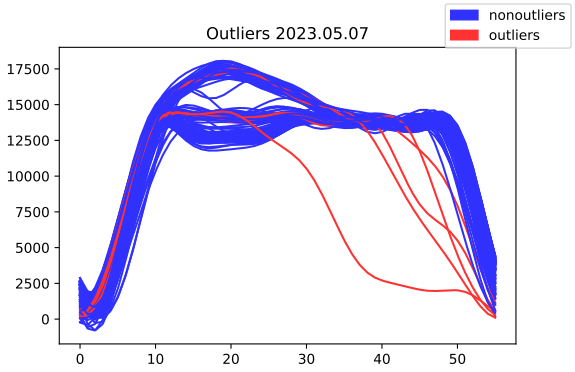

Figure 40: 2023.05.07 outliers detection

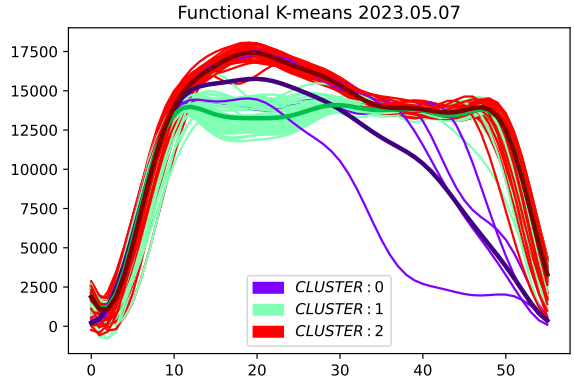

Figure 41: 2023.05.07 Functional K-means clustering

- **2022-06-22**

  Despite being the only day from 2022 amongst highest generating days, we can still see some similarities to 2023. As an outliers, all strings connected to inverter no. 8 except string 12 are recognised (figure 42). Also strings 11-1, 12-2 and 12-3 are an outliers. Functional K-means clustering assigned all strings connected to inverter no. 8 except string 12 (figure 43).

- **2023-05-06**

  On this day, we observe only three strings identified as outliers - strings 11-1, 12-2 and 12-3 (figure 44). However, clustering results show that in the smallest cluster, represented by purples lines, string 12-5 was assigned together with strings classified as outliers (figure 45).

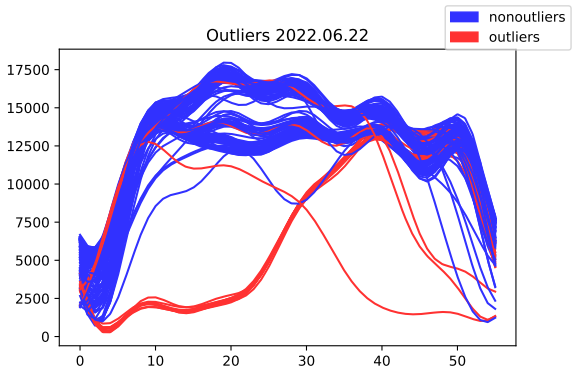

Figure 42: 2022.06.22 outliers detection

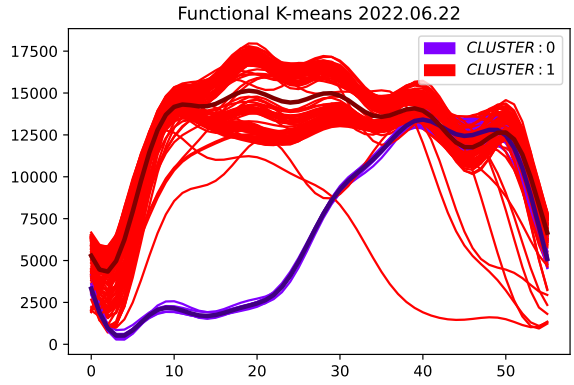

Figure 43: 2022.06.22 Functional K-means clustering

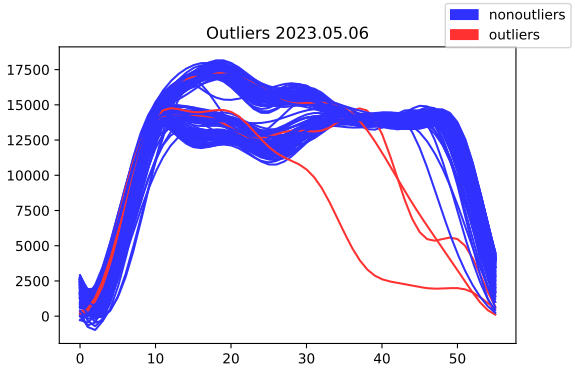

Figure 44: 2023.05.06 outliers detection

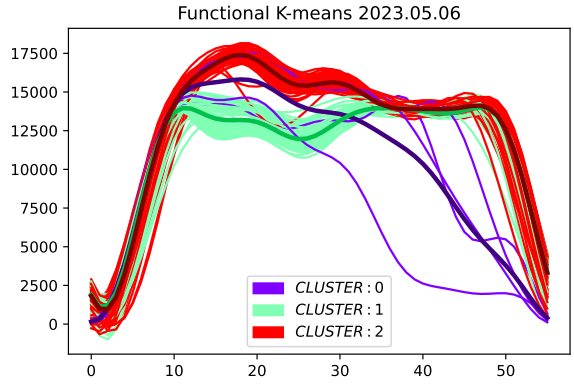

Figure 45: 2023.05.06 Functional K-means clustering

- **2023-05-08**

  On this day, we observe four strings identified as outliers - strings 11-1, 11-2, 12-2, 12-3 and 12-4 (figure 48). However, clustering results show that in the smallest cluster, represented by purples lines, strings 11-1, 12-2, 12-3 and 12-5 are grouped (figure 49).

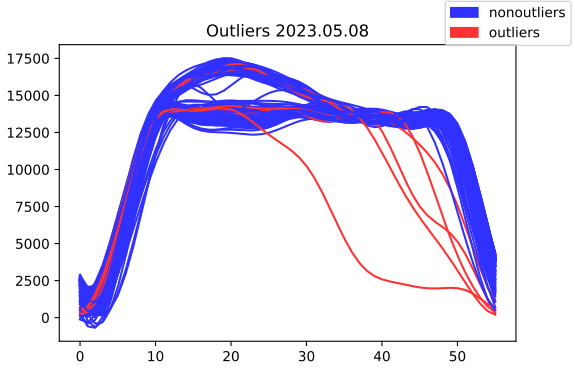

Figure 46: 2023.05.08 outliers detection

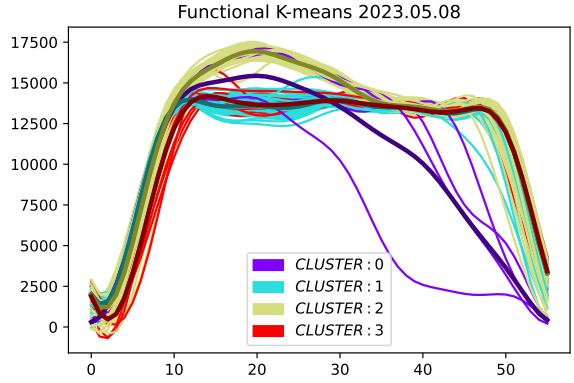

Figure 47: 2023.05.08 Functional K-means clustering

- **2023-05-08**

  On this day, we observe four strings identified as outliers - strings 11-1, 11-2, 12-2, 12-3 and 12-4 (figure 48). However, clustering results show that in the smallest cluster, represented by purples lines, strings 11-1, 12-2, 12-3 and 12-5 are grouped (figure 49).

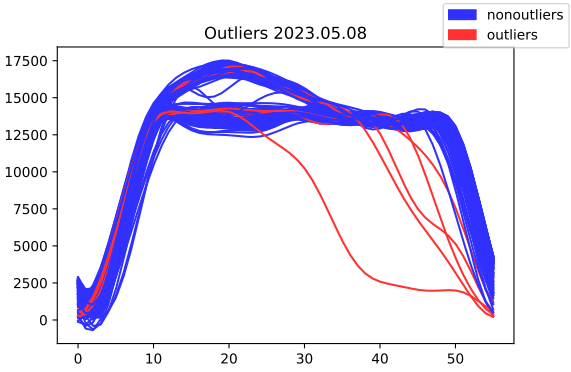

Figure 48: 2023.05.08 outliers detection

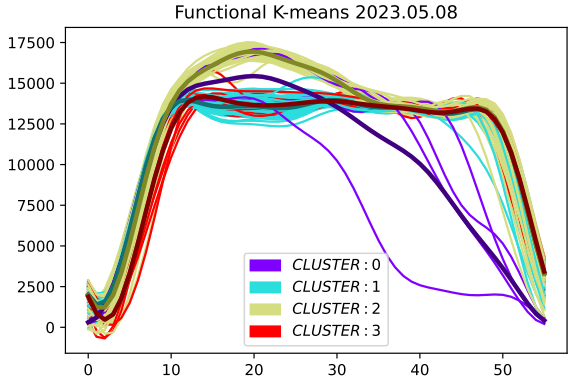

Figure 49: 2023.05.08 Functional K-means clustering

- **2023-06-07**

  From the figure 50, we observe that strings 11-1, 11-2, 12-1, 12-2, 12-3, 12-5 and 12-9 are recognised as outliers. However, clustering results show that in the smallest cluster, represented by purples lines, string 12-9 is not included (figure 51).

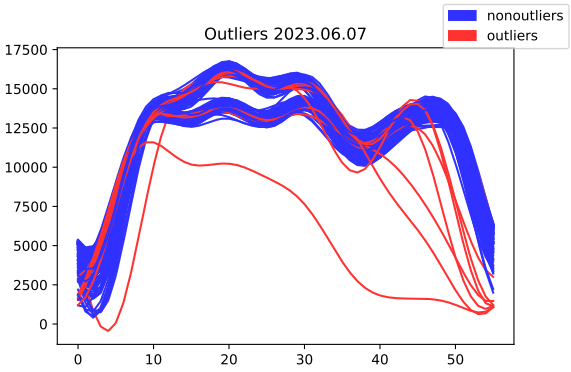

Figure 50: 2023.06.07 outliers detection

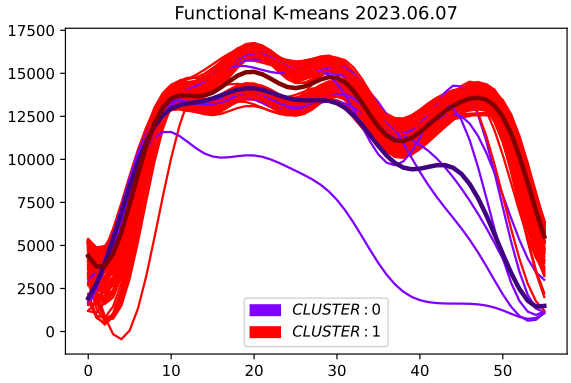

Figure 51: 2023.06.07 Functional K-means clustering

- **2022-08-09**

  Moving on to the days with average irradiance, we see that new strings are identified as outliers. From the figure 52, we observe that strings 2-9, 8-12 and 12-2 are outliers, while figure 53 reveals that strings 2-9 and 12-2 are the closes and thus grouped to one cluster represented by the purple lines.

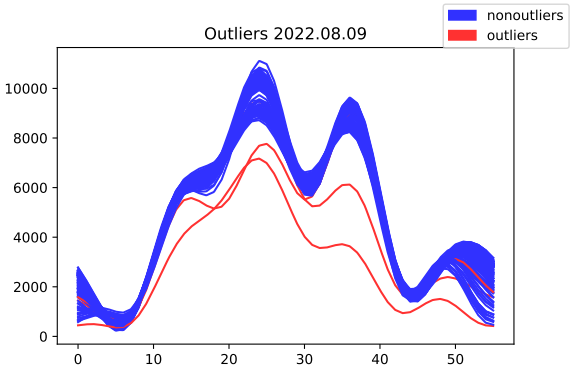

Figure 52: 2022.08.09 outliers detection

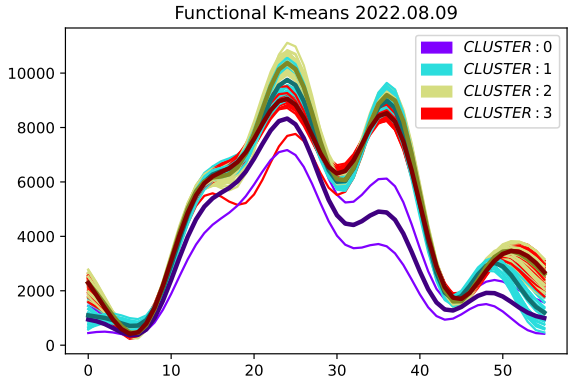

Figure 53: 2022.08.09 Functional K-means clustering

- **2022-09-25**

  From the figure 54, we observe that strings 8-11, 8-12 and 12-2 are outliers, while figure 55 reveals that strings 8-11 and 8-12 are the closes and thus grouped to one cluster represented by the purple lines.

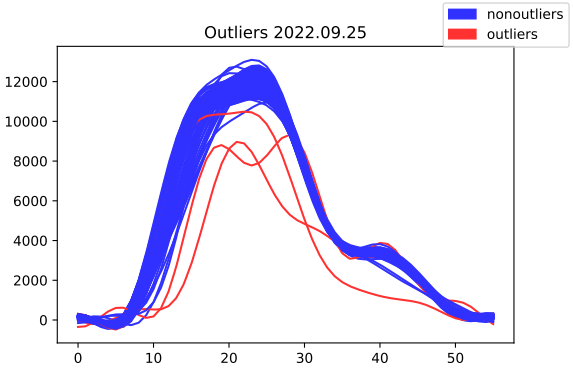

Figure 54: 2022.09.25 outliers detection

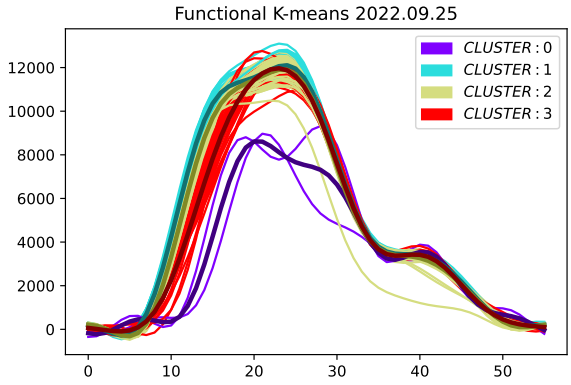

Figure 55: 2022.09.25 Functional K-means clustering

- **2023-06-17**

  Figure 56 depicts that strings 8-12 and 12-2 are outliers, while figure 57 reveals that string 12-2 is the only one assigned to the separated cluster (represented by the purple line).

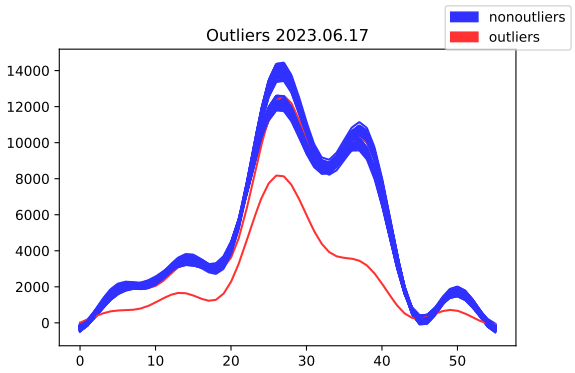

Figure 56: 2023.06.17 outliers detection

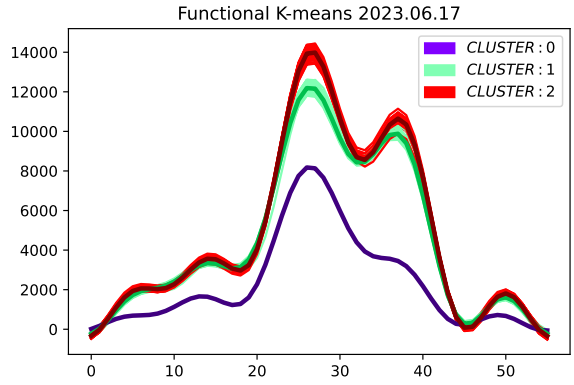

Figure 57: 2023.06.17 Functional K-means clustering

- **2023-07-06**

  Outlier detection (figure 58) of this day gave the same results as the day above, strings 8-12 and 12-2 are outliers. However, functional clustering results are different (58). We observe that strings 6-1, 6-2, 7-1, 7-2, 8-12, 9-7, 9-11, 10-10 and 12-2 (represented by the purple line) are assigned to the smallest cluster .

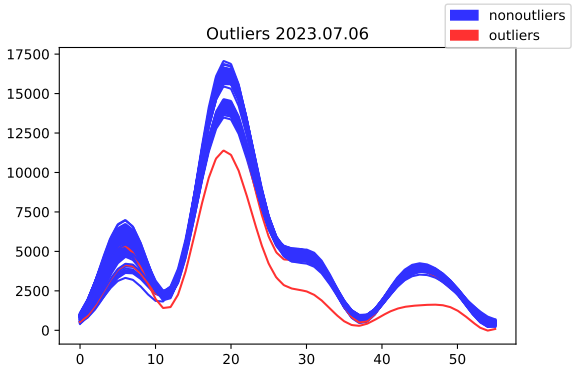

Figure 58: 2023.07.06 outliers detection

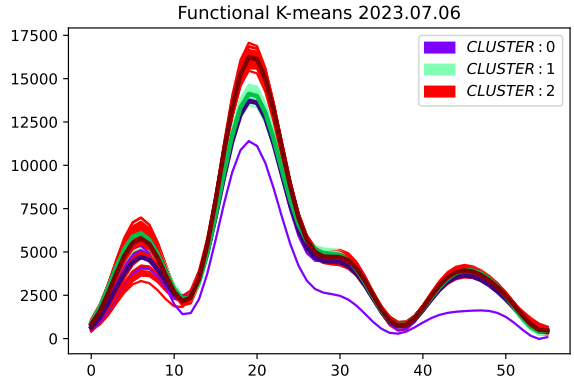

Figure 59: 2023.07.06 Functional K-means clustering

- **2023-10-06**

  Figure 60 illustrates that strings 8-11, 8-12, 9-12, 11-1, 12-2 and 12-3 are outliers. However, functional clustering method reveals that outliers are grouped in two separate clusters together with non-outliers, based on their distances (figure 37).

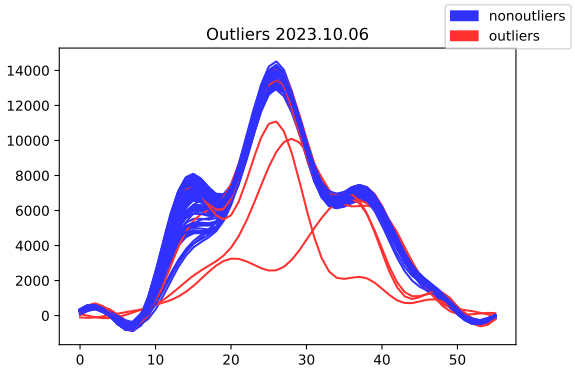

Figure 60: 2023.10.06 outliers detection

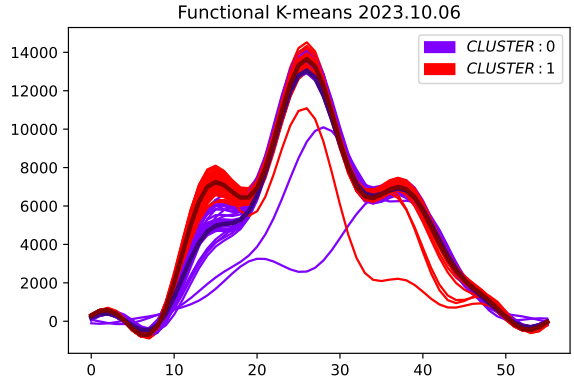

Figure 61: 2023.10.06 Functional K-means clustering

5 Results

In this section, we discuss the results of our research. We selected 10 days with the highest mean daily electrical power and 5 days with average daily generation. From the descriptive statistics, we observed initial signs of abnormalities. A high SD (table 2) indicates that the electrical power generated by each string is spread over a wider range, suggesting more variability in the data. Conversely, a low standard deviation on days with lower generated power suggests that values are close to the mean, indicating the data points are relatively consistent.

Examining the correlation coefficients, we found that the correlation between strings is usually very close to 1. This suggests that each string generates approximately the same amount of energy under the same irradiance and technical conditions. However, we noticed that string 12-2 has a lower correlation coefficient (figure 8). This indicates that this string might be affected by external conditions, such as shading, which impacts its production.

In the next step, we visualized the strings using a boxplot. While the boxplots of most strings are consistent with a median around 2500 W, string 12-2 (figure 6) stands out with a median around 1200 W, an (IQR) between 0 W and 4000 W, and it also has outliers.

Following our findings from descriptive statistics, we proceed with clustering the strings on a case-by-case basis using the K-means algorithm. By forming cluster groups, we identify strings that are similar in terms of electrical power generation. Consistent with our initial analysis, string 12-2 is assigned to a smaller, separate cluster in all cases. Occasionally, on days when electrical power generation is the highest, string 12-2 is joined by strings 11-1, 12-3, and 12-5.

We also observe that the correlation between pairs of strings decreases drastically on these high-generation days, with correlation values closer to 0 than 1. Comparing these strings to others, we note significantly lower generation outputs and higher variance.

When irradiance decreases, the results of the K-means clustering changes accordingly. On these days, strings 2-9, 8-11, 8-12, and 12-2 are assigned to separate, smaller clusters, indicating higher variance from the mean. Although a decrease in correlation is also noticeable on these lower irradiance days, it is not as significant as on high-performing days.

In the final stage of the experiment, we apply FDA methods to support our previous findings. We transform the time series data into functional data series and smooth it using the Fourier basis method. We then use outlier detection and functional K-means clustering to identify strings affected by shading. Outlier detection reveals that on high irradiance days, outliers are observed in inverters 1, 2, 6, 11, and 12, while on average irradiance days, outliers are found within inverters 2, 8, 9, 11, and 12. However, these outliers are not necessarily grouped into the same cluster. Overall, the results from functional data analysis methods align with traditional K-means clustering, identifying strings 11-1, 12-2, 12-3, and 12-5 as the strings most frequently shaded. It's important to highlight that changes in shaded inverters over time are normal. This occurs due to the changing position of the sun, which affects the size and placement of shade. As the sun moves throughout the day and across seasons, the shading patterns on solar panels can vary, leading to fluctuations in the shading of inverters.

6 Conclusion

This paper aimed to address the challenge of fault detection in solar power generation by employing statistical and machine learning methods. Through an extensive analysis of 143 strings generating solar electrical power, we investigated abnormalities and identified strings affected by shading, a crucial environmental fault impacting power generation.

Our research utilized a combination of descriptive statistics, K-means clustering, and Functional Data Analysis methods. Descriptive statistics, including mean, standard deviation, and correlation coefficients, provided initial insights into the data, revealing abnormal patterns and indicating potential shading effects. K-means clustering further confirmed these findings, highlighting strings consistently impacted by shading, such as string 12-2.

Furthermore, our study incorporated FDA methods, transforming time series data into functional data series and applying Fourier basis smoothing to detect outliers and perform functional K-means clustering. These methods aligned with traditional K-means clustering, reinforcing the identification of shaded strings, particularly strings 11-1, 12-2, 12-3, and 12-5.

Our results underscore the significance of environmental faults, such as shading, in solar power generation systems. By detecting and addressing these faults, we can enhance the efficiency and reliability of solar energy production, contributing to environmental sustainability and economic viability.

It is important to note that changes in shaded inverters over time are normal, driven by the dynamic movement of the sun. As the sun's position changes throughout the day and across seasons, shading patterns on solar panels fluctuate, impacting power generation.

In summary, our study provides valuable insights into fault detection methodologies for solar power generation systems. By leveraging statistical and machine learning techniques, we can better understand and mitigate environmental faults, ultimately advancing the adoption and effectiveness of solar energy as a clean and renewable energy source.

In addition, our study highlights the computational demands associated with case-by-case basis calculations, particularly in the context of analyzing daily observations. Given the considerable time resources required for such analyses, a potential avenue for future research involves exploring advanced Functional Data Analysis (FDA) methods to identify shaded strings without the need for splitting data into daily observations. By leveraging more sophisticated FDA techniques, such as functional principal component analysis (FPCA) or functional regression, researchers can streamline the analysis process and gain insights into shading patterns across longer time intervals. This approach not only reduces computational burden but also allows for a more comprehensive understanding of shading effects on solar power generation. Implementing further FDA methods offers the opportunity to uncover nuanced relationships between environmental factors, such as shading, and their impact on solar energy production. By harnessing the full potential of functional data analysis, future studies can enhance the efficiency and accuracy of fault detection in solar power systems, ultimately contributing to the advancement of renewable energy technologies.

References

[1] Qais Ibrahim Ahmed et al. "Development of a hybrid support vector machine with grey wolf optimization algorithm for detection of the solar power plants anomalies". In: *Systems* 11.5 (2023), p. 237.

[2] Eleonora Arena et al. "Anomaly detection in photovoltaic production factories via Monte Carlo pre-processed principal component analysis". In: *Energies* 14.13 (2021), p. 3951.

[3] Dushengere Bernadette et al. "Analysis of shading effects in solar PV system". In: *Int. J. Sustain. Green Energy* 10.2 (2021), pp. 47–62.

[4] Mengyao Cui et al. "Introduction to the k-means clustering algorithm based on the elbow method". In: *Accounting, Auditing and Finance* 1.1 (2020), pp. 5–8.

[5] Zeynep Bala Duranay. "Fault Detection in Solar Energy Systems: A Deep Learning Approach". In: *Electronics* 12.21 (2023), p. 4397.

[6] Elyes Garoudja et al. "Statistical fault detection in photovoltaic systems". In: *Solar Energy* 150 (2017), pp. 485–499.

[7] Geoffrey E Hinton and Sam Roweis. "Stochastic neighbor embedding". In: *Advances in neural information processing systems* 15 (2002).

[8] Francesca Ieva et al. "Multivariate functional clustering for the morphological analysis of electrocardiograph curves". In: *Journal of the Royal Statistical Society Series C: Applied Statistics* 62.3 (2013), pp. 401–418.

[9] Zhehan Yi and Amir H Etemadi. "Fault detection for photovoltaic systems based on multi-resolution signal decomposition and fuzzy inference systems". In: *IEEE Transactions on Smart Grid* 8.3 (2016), pp. 1274–1283.

[10] Muhammad Saad Iqbal et al. "Real-time fault detection system for large scale grid integrated solar photovoltaic power plants". In: *International Journal of Electrical Power & Energy Systems* 130 (2021), p. 106902.

[11] Julien Jacques and Cristian Preda. "Model-based clustering for multivariate functional data". In: *Computational Statistics & Data Analysis* 71 (2014), pp. 92–106.

[12] P. Kokoszka and M. Reimherr. *Introduction to Functional Data Analysis*. CRS Press, 2017.

[13] Official Statistics Portal of Lithuania. *Energy statistics 2022*. 2023. URL: `https://osp.stat.gov.lt/informaciniai-pranesimai?articleId=11069641`.

[14] Sufyan Ali Memon et al. "A machine-learning-based robust classification method for PV panel faults". In: *Sensors* 22.21 (2022), p. 8515.

[15] Daniel Perdices, Jorge E López de Vergara, and Javier Ramos. "Deep-fda: Using functional data analysis and neural networks to characterize network services time series". In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 986–999.

[16] Michel Piliougine and Giovanni Spagnuolo. "Mismatching and partial shading identification in photovoltaic arrays by an artificial neural network ensemble". In: *Solar Energy* 236 (2022), pp. 712–723.

[17] R Prasanna et al. "Comprehensive Review on Modelling, Estimation, and Types of Faults in Solar Photovoltaic System". In: *International Journal of Photoenergy* 2022 (2022).

[18] Jiaqi Qu et al. "An unsupervised hourly weather status pattern recognition and blending fitting model for PV system fault detection". In: *Applied Energy* 319 (2022), p. 119271.

[19] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis. Second edition.* Springer-Verlag: New York, 2005.

[20] Sunil Rao, Andreas Spanias, and Cihan Tepedelenlioglu. "Solar array fault detection using neural networks". In: *2019 IEEE international conference on industrial cyber physical systems (ICPS)*. IEEE. 2019, pp. 196–200.

[21] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[22] Archana Singh, Avantika Yadav, and Ajay Rana. "K-means with Three different Distance Metrics". In: *International Journal of Computer Applications* 67.10 (2013).

[23] Abir Smiti. "A critical overview of outlier detection methods". In: *Computer Science Review* 38 (2020), p. 100306.

[24] Jorge R Sosa Donoso et al. "Local correlation integral approach for anomaly detection using functional data". In: *Mathematics* 11.4 (2023), p. 815.

[25] Thaddeus Tarpey and Kimberly KJ Kinateder. "Clustering functional data." In: *Journal of classification* 20.1 (2003).

[26] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[27] Adriano Zanin Zambom, Julian AA Collazos, and Ronaldo Dias. "Functional data clustering via hypothesis testing k-means". In: *Computational Statistics* 34 (2019), pp. 527–549.

[28] Ye Zhao et al. "Decision tree-based fault detection and classification in solar photovoltaic arrays". In: *2012 Twenty-Seventh Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*. IEEE. 2012, pp. 93–99.

[29] Ye Zhao et al. "Graph-based semi-supervised learning for fault detection and classification in solar photovoltaic arrays". In: *IEEE Transactions on Power Electronics* 30.5 (2014), pp. 2848–2858.

[30] Ye Zhao et al. "Outlier detection rules for fault detection in solar photovoltaic arrays". In: *2013 Twenty-Eighth Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*. IEEE. 2013, pp. 2913–2920.

[31] Yingying Zhao et al. "Collaborative fault detection for large-scale photovoltaic systems". In: *IEEE Transactions on Sustainable Energy* 11.4 (2020), pp. 2745–2754.

[32] Yingying Zhao et al. "Hierarchical anomaly detection and multimodal classification in large-scale photovoltaic systems". In: *IEEE Transactions on Sustainable Energy* 10.3 (2018), pp. 1351–1361.

Appendix A

References	Year	Data	Fault types	Horizon	Model	Results
[28]	2012	A small-scale grid-connected PV system located in Boston, USA	LL between string 1 and negative bus bar (LL), LL with fault impedance $R_f = 20$ ohms (LL-20), OC on string 2 (OC), partial shading. Faults were intentionally generated through experimentation	2 days	DT based detection and classification	Accuracy (%): fault detection 93.56%, classification 85.43%. Class-label accuracy (%): LL 99.99%, normal 99.48%, OC 89.78%, shade 21.3%, LL-20 0%
[30]	2013	A small-scale grid-connected PV system	LL between the middle of one string and negative bus bar (LL), LL with fault impedance $R_f = 20$ ohms (LL-20), OC on a string (OC), degradation fault on a PV module (DF), partial shading. Faults were intentionally generated through experimentation	Not available	Outlier detection rules: 3-Sigma, Hampel identifier and Boxplot.	3-Sigma rule - vulnerable to outliers, failed to detect the faults in all cases; Hampel identifier - failed to detect DF, but has the best tolerance to outliers; Boxplot rule - outperformed other two rules by detecting all types of faults, has low outlier sensitivity

Table 5: Summary of fault detection methods

References	Year	Data	Fault types	Horizon	Model	Results
[6]	2017	Grid-connected PV system located in Algeria	Short circuits (SC), open circuits and partial shading	1 second resolution	ODM - EWMA	Normal operating conditions - data points are within 95% confidence limits in the EWMA charts; OC conditions - concluded that proposed method is efficient; SC conditions - concluded that proposed method works; Shading - EWMA chart is able to detect the faults, but it cannot classify it; multiple faults - EWMA chart accurately detected and classified multiple faults
[10]	2021	Actual data - PV power plant installed in Pakistan, simulated data - linked to actual data to produce similar power as the actual data	PV modules failure, partial shading or soiling loss, single or multiple PV string failure	Actual data is in 15 minutes resolution	Z_Score	Proposed method is able to detect faults, however it does not distinguish faults that has vary similar impact of power reduction
[29]	2014	A small-scale grid-connected PV system	LL and OC	Not provided	GBSSL	Faults that do not overlap were detected and classified with 100 % accuracy

Table 5: Summary of fault detection methods (continued)

References	Year	Data	Fault types	Horizon	Model	Results
[9]	2016	Simulated PV system and small-scale grid-connected PV array	LL and LG (only for simulated data)	Not provided	MSD and FIS	Simulation results reach 100% accuracy with mismatch of 30-60%, 97.69% accuracy with mismatch of 20% and 65.05% accuracy with 10% of mismatch for LL faults and 100% accuracy with mismatch of 10-40%, 97.22% accuracy with mismatch of 50% and 92.13% accuracy with 60% mismatch for LG faults. Experimental results reach 96.1% accuracy with mismatch of 50% and 84.9% accuracy with 25% mismatch
[20]	2019	Experimental data developed through Smart Monitoring Devices	standard test conditions (STC), short circuit (SC), varying temperature, partial shading, complete shading, soiling, ground fault (Gnd), arc fault (Arc)	Not provided	MLP	Overall 99.7% accuracy. Varying temperature - 99.9%, complete shading - 98.7%, soiling - 98.8%, STC - 100%, SC - 100%, Gnd - 100%, Arc - 100% accuracy
[31]	2020	Real data of sites A (flat terrain) and B(mountainous area) located in China.	Not provided	Not provided	CF	Detection accuracy of site A is 96.3%, of site B - 97.1%
[2]	2021	Data set of photovoltaic production factory located in Catania, Italy	N/A	N/A	Monte Carlo (MC) simulation for the preprocessing, PCA for anomaly detection	In the preprocessing stage, MC method surpassed Interquartile Range method. MC combined with PCA showed accurate results and resilience to false alarms.

Table 5: Summary of fault detection methods (continued)

References	Year	Data	Fault types	Horizon	Model	Results
[16]	2022	10000 I-V curves simulated under uniform or mismatched conditions and experimental curves from a real PV generator	Not provided	Not provided	MLP	The best results were achieved with 15 neurons - MSE $6.2e^{-9}$. 1 neuron - MSE $8.2e^{-10}$, 10 neurons - MSE $2.0e^{-}10$
[18]	2022	DKASC PV system in Australia (system A) and a PV system in China (system B).	DHSFc1- sunny weather without clouds, extraterrestrial radiation and low volatility; DHSFc2 - light cloud weather with swift clouds movements, high volatility; DHSFc3 - cloudy weather with more clouds and low volatility	1 hour resolution	Classification methods: Multiclass-GBDT-LR (MGL), DT, Xgboost, RF and stacking, detection method: blending fitting models	System A: in classification task MGL surpassed DT, Xgboost, RF and stacking with accuracy of 98.85%, recall 98.81%, F1 98.83%, detection accuracy 97.71%; System B: in classification task MGL surpassed DT, Xgboost, RF and stacking with accuracy of 98.19%, recall 98.15%, F1 98.18%, detection accuracy 99.29%
[14]	2022	Simulated data set	LL, degradation, OC, shadowing	N/A	CNN	Training accuracy 97.64%, testing accuracy 95.20 %
[1]	2023	Solar power plants located in India	External and internal factors	34 days at 15 min intervals	Physical model, SVM regression model (SVM-GW-R), SVM classification model (SVM-GW-C)	Sensitivity: SVM-GW-C 85.71%, SVM-GW-R 68.42%, physical model 52.63 %; Specificity: SVM-GW-C 99.21%, SVM-GW-R 94.57%, physical model 93.02%

Table 5: Summary of fault detection methods (continued)

Appendix B

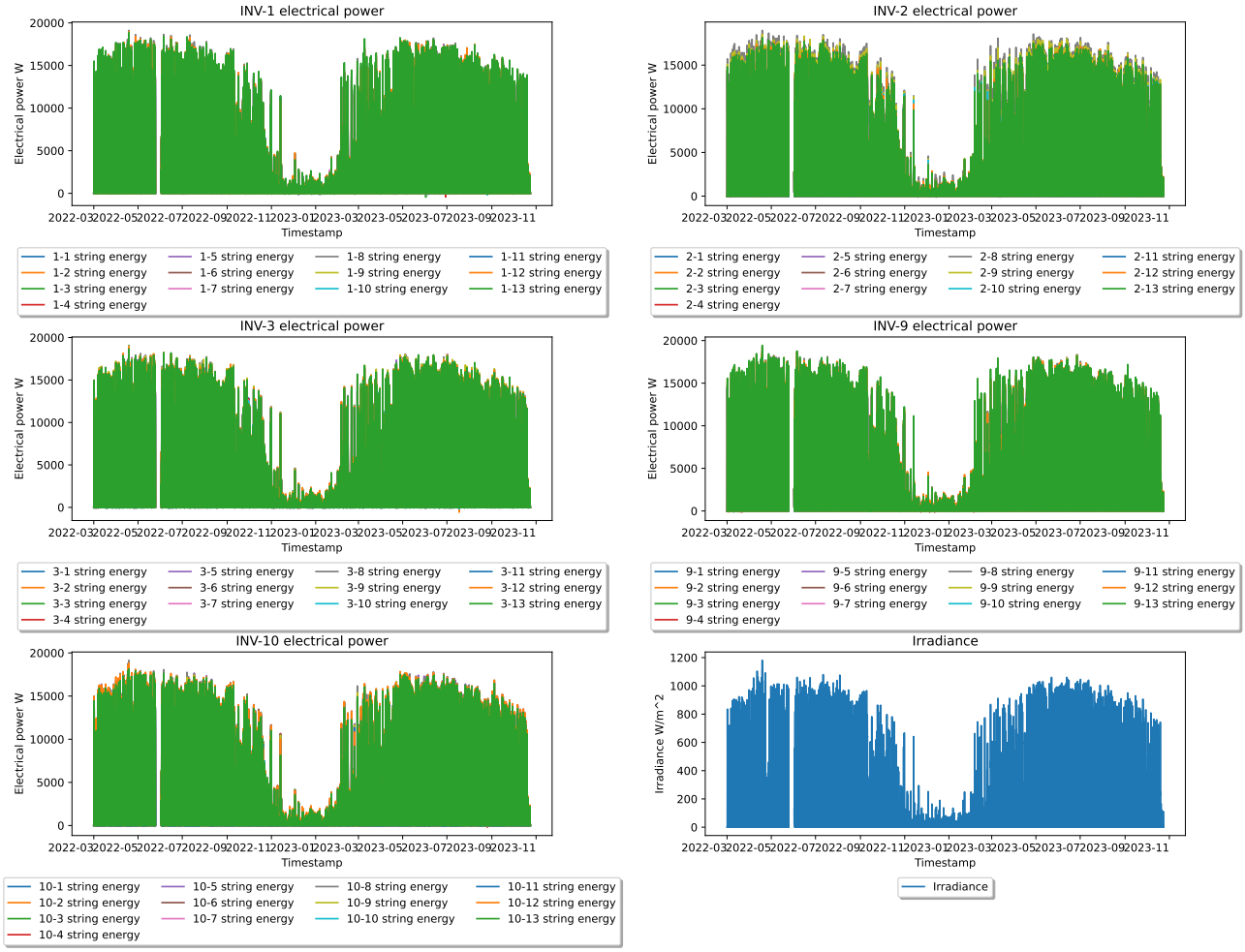

Figure 62: Block no.1 electrical power and irradiance

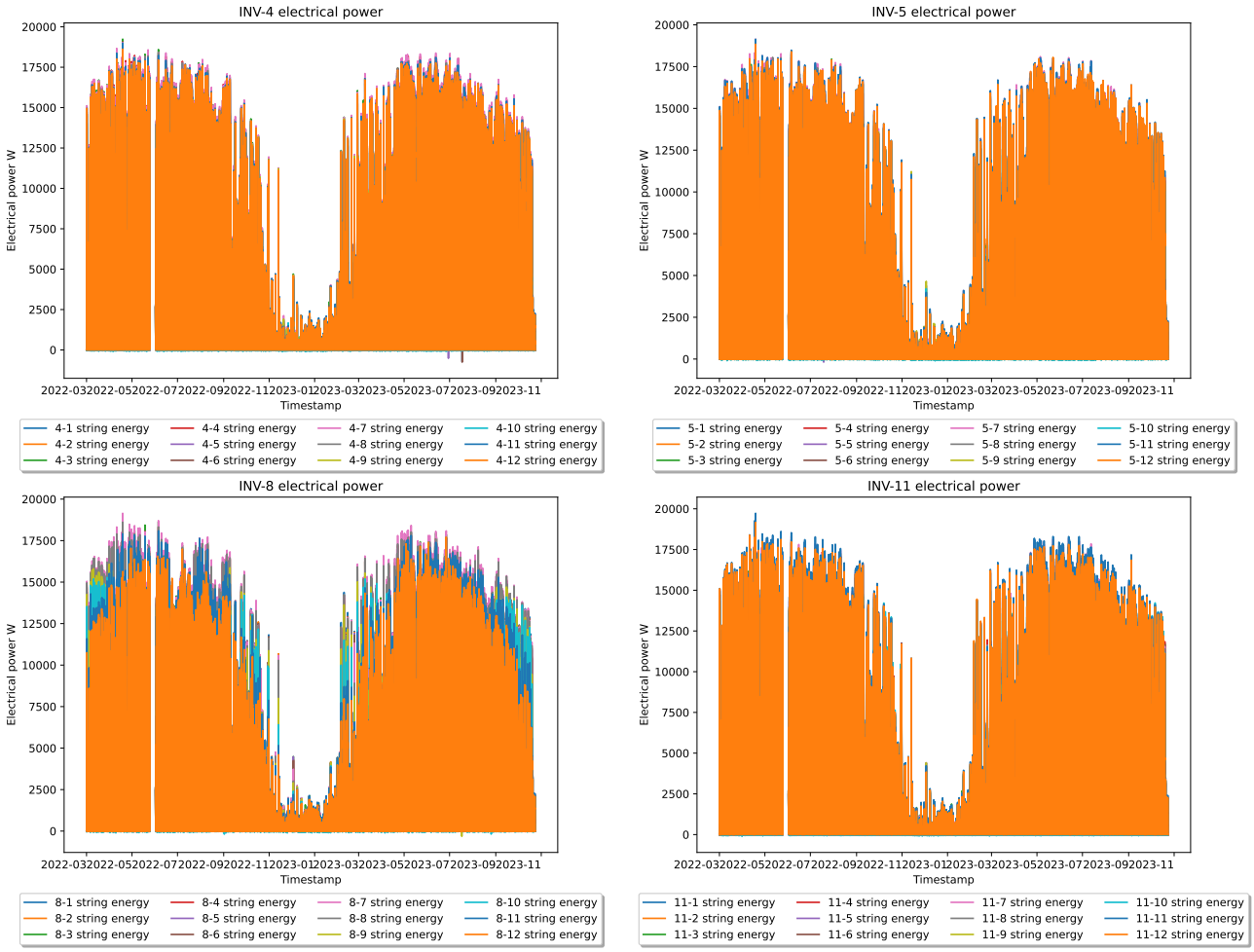

Figure 63: Block no.2 electrical power

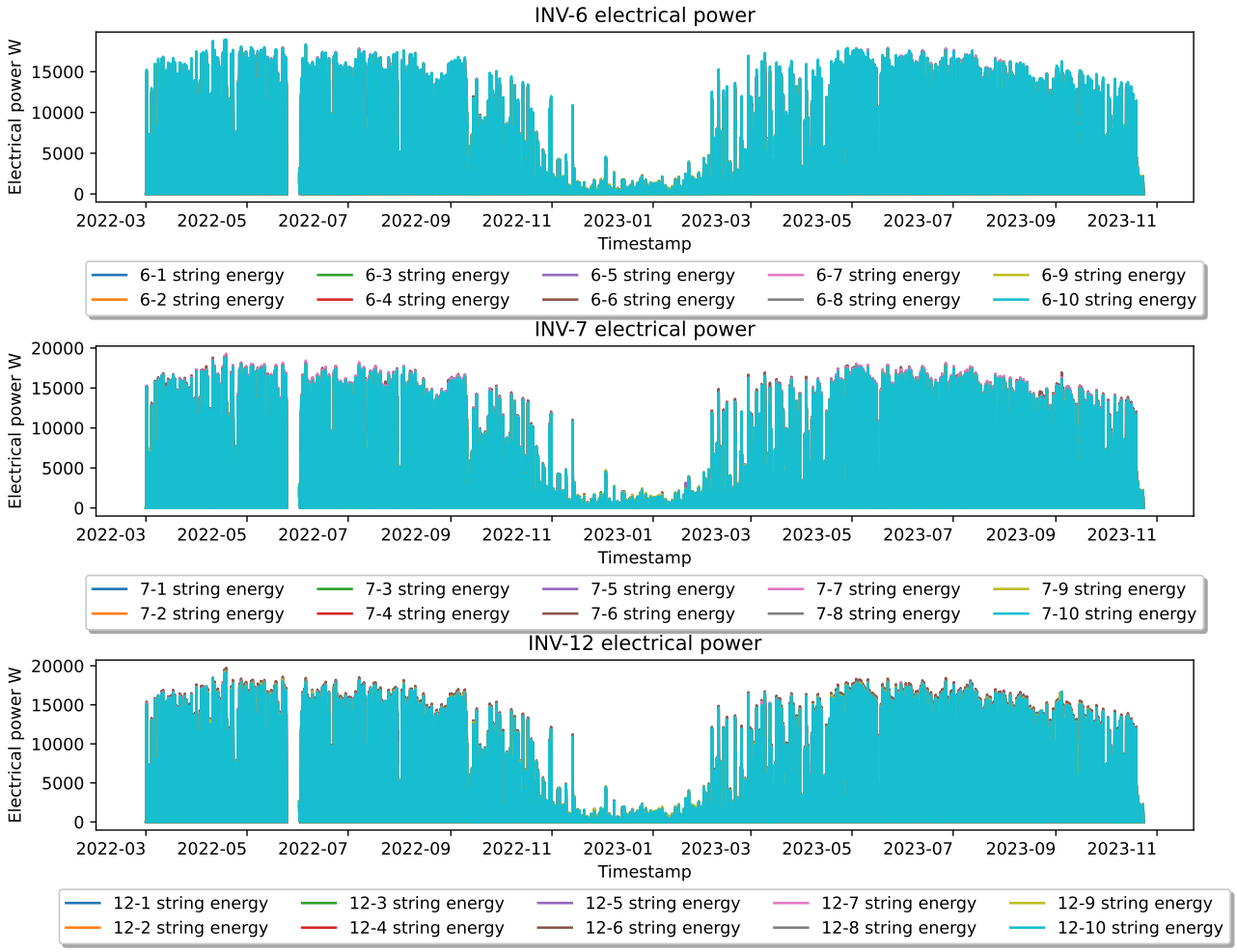

Figure 64: Block no.3 electrical power

Appendix C

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
from sklearn import manifold
import scipy.cluster.hierarchy as sch
import skfda
from skfda.ml.clustering import FuzzyCMeans, KMeans
from skfda.exploratory.visualization import Boxplot
from skfda.exploratory.visualization.clustering import (
    ClusterMembershipLinesPlot,
    ClusterMembershipPlot,
    ClusterPlot,
)
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

irradiance = pd.read_excel(r"C:\Users\hp\OneDrive\Desktop\Master thesis\Code\Data\irradiance.x

data['Timestamp'] = pd.to_datetime(data['Timestamp'])
irradiance['Timestamp'] = pd.to_datetime(irradiance['Timestamp'])
data.set_index('Timestamp', inplace=True) #sets time as an index
irradiance.set_index('Timestamp', inplace=True) #sets time as an index
data = data.reset_index() #resets index

#Renaming columns
data = data.rename(columns=new_column_names)

#double checking that every column's name was changed
columns_data = pd.DataFrame(data.columns, columns=['Column_Names'])

#-------------------------------------------------------------------------------------------
#Getting the energy data set

#Creating a new DataFrame with the same index as df
```

```
energy1_df = pd.DataFrame(index=data.index)
energy2_df = pd.DataFrame(index=data.index)
energy3_df = pd.DataFrame(index=data.index)
energy4_df = pd.DataFrame(index=data.index)
energy5_df = pd.DataFrame(index=data.index)
energy6_df = pd.DataFrame(index=data.index)
energy7_df = pd.DataFrame(index=data.index)
energy8_df = pd.DataFrame(index=data.index)
energy9_df = pd.DataFrame(index=data.index)
energy10_df = pd.DataFrame(index=data.index)
energy11_df = pd.DataFrame(index=data.index)
energy12_df = pd.DataFrame(index=data.index)

#Multiply respective columns

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'1-{i} String current'
    column2 = f'1-{i} String voltage'
    new_column_name = f'1-{i} string energy'
    energy1_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'2-{i} String current'
    column2 = f'2-{i} String voltage'
    new_column_name = f'2-{i} string energy'
    energy2_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'3-{i} String current'
    column2 = f'3-{i} String voltage'
    new_column_name = f'3-{i} string energy'
    energy3_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'4-{i} String current'
    column2 = f'4-{i} String voltage'
    new_column_name = f'4-{i} string energy'
    energy4_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'5-{i} String current'
```

```
    column2 = f'5-{i} String voltage'
    new_column_name = f'5-{i} string energy'
    energy5_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'6-{i} String current'
    column2 = f'6-{i} String voltage'
    new_column_name = f'6-{i} string energy'
    energy6_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'7-{i} String current'
    column2 = f'7-{i} String voltage'
    new_column_name = f'7-{i} string energy'
    energy7_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'8-{i} String current'
    column2 = f'8-{i} String voltage'
    new_column_name = f'8-{i} string energy'
    energy8_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'9-{i} String current'
    column2 = f'9-{i} String voltage'
    new_column_name = f'9-{i} string energy'
    energy9_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'10-{i} String current'
    column2 = f'10-{i} String voltage'
    new_column_name = f'10-{i} string energy'
    energy10_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
    column1 = f'11-{i} String current'
    column2 = f'11-{i} String voltage'
    new_column_name = f'11-{i} string energy'
    energy11_df[new_column_name] = data[column1] * data[column2]

for i in range(1, len(data.columns) // 2 + 1):
```

```
    column1 = f'12-{i} String current'
    column2 = f'12-{i} String voltage'
    new_column_name = f'12-{i} string energy'
    energy12_df[new_column_name] = data[column1] * data[column2]

#Joining new dataset
df = [energy1_df, energy2_df, energy3_df, energy4_df, energy5_df, energy6_df,
      energy7_df, energy8_df, energy9_df, energy10_df, energy11_df, energy12_df,
      irradiance]

energy = pd.concat(df, join='outer', axis=1)

#Visualizing unprocessed data
energy = energy.reset_index() #resets index

fig, axes = plt.subplots(3, 2, figsize=(16, 12), layout="constrained")

columns_inv1 = ['1-1 string energy', '1-2 string energy', '1-3 string energy',
                '1-4 string energy', '1-5 string energy', '1-6 string energy',
                '1-7 string energy', '1-8 string energy', '1-9 string energy',
                '1-10 string energy', '1-11 string energy', '1-12 string energy',
                '1-13 string energy']
energy.plot(x='Timestamp', y=columns_inv1, kind='line', ax=axes[0, 0])
axes[0, 0].set_ylabel('Electrical power W')
axes[0, 0].set_title('INV-1 electrical power')
axes[0, 0].legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), shadow=True, ncol=4)
axes[0, 0].set_xticklabels(axes[0, 0].get_xticklabels(), rotation=0, ha='right')

columns_inv2 = ['2-1 string energy','2-2 string energy', '2-3 string energy',
                '2-4 string energy', '2-5 string energy', '2-6 string energy',
                '2-7 string energy', '2-8 string energy', '2-9 string energy',
                '2-10 string energy', '2-11 string energy', '2-12 string energy',
                '2-13 string energy']
energy.plot(x='Timestamp', y=columns_inv2, kind='line', ax=axes[0, 1])
axes[0, 1].set_ylabel('Electrical power W')
axes[0, 1].set_title('INV-2 electrical power')
axes[0, 1].legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), shadow=True, ncol=4)
axes[0, 1].set_xticklabels(axes[0, 1].get_xticklabels(), rotation=0, ha='right')

columns_inv3 = ['3-1 string energy','3-2 string energy', '3-3 string energy',
                '3-4 string energy', '3-5 string energy', '3-6 string energy',
```

```
                '3-7 string energy', '3-8 string energy', '3-9 string energy',
                '3-10 string energy', '3-11 string energy', '3-12 string energy',
                '3-13 string energy']
energy.plot(x='Timestamp', y=columns_inv3, kind='line', ax=axes[1, 0])
axes[1, 0].set_ylabel('Electrical power W')
axes[1, 0].set_title('INV-3 electrical power')
axes[1, 0].legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), shadow=True, ncol=4)
axes[1, 0].set_xticklabels(axes[1, 0].get_xticklabels(), rotation=0, ha='right')

columns_inv9 = ['9-1 string energy','9-2 string energy', '9-3 string energy',
                '9-4 string energy', '9-5 string energy', '9-6 string energy',
                '9-7 string energy', '9-8 string energy', '9-9 string energy',
                '9-10 string energy', '9-11 string energy', '9-12 string energy',
                '9-13 string energy']
energy.plot(x='Timestamp', y=columns_inv9, kind='line', ax=axes[1, 1])
axes[1, 1].set_ylabel('Electrical power W')
axes[1, 1].set_title('INV-9 electrical power')
axes[1, 1].legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), shadow=True, ncol=4)
axes[1, 1].set_xticklabels(axes[1, 1].get_xticklabels(), rotation=0, ha='right')

columns_inv10 = ['10-1 string energy','10-2 string energy', '10-3 string energy',
                 '10-4 string energy', '10-5 string energy', '10-6 string energy',
                 '10-7 string energy', '10-8 string energy', '10-9 string energy',
                 '10-10 string energy', '10-11 string energy', '10-12 string energy',
                 '10-13 string energy']
energy.plot(x='Timestamp', y=columns_inv10, kind='line', ax=axes[2, 0])
axes[2, 0].set_ylabel('Electrical power W')
axes[2, 0].set_title('INV-10 electrical power')
axes[2, 0].legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), shadow=True, ncol=4)
axes[2, 0].set_xticklabels(axes[2, 0].get_xticklabels(), rotation=0, ha='right')

energy.plot(x='Timestamp', y='Irradiance', kind='line', ax=axes[2, 1])
axes[2, 1].set_ylabel('Irradiance W/m^2')
axes[2, 1].set_title('Irradiance')
axes[2, 1].legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), shadow=True, ncol=4)
axes[2, 1].set_xticklabels(axes[2, 1].get_xticklabels(), rotation=0, ha='right')

fig.savefig('Block1.pdf', format='pdf', dpi=1200, bbox_inches='tight')

plt.show()
```

```
#Data pre-processing

#Searching for missing values
missing_values = energy.isnull()
indices = missing_values.where(missing_values == True).stack().index

# Filter the DataFrame to keep only observations after the certain time point
certain_time_point = pd.Timestamp('2022-06-02 00:00:00')
energy = energy[energy['Timestamp'] >= certain_time_point]
energy = energy.set_index('Timestamp')

# Filter data between 6 am and 8 pm
energy = energy[(energy.index.hour >= 6) & (energy.index.hour < 20)]

#Filling in missing values
energy = energy.interpolate(option='spline')

# Replacing negative values with the absolute value
energy = energy.applymap(lambda x: abs(x))
energy = energy.reset_index()

#-------------------------------------------------------------------------
fig, axes = plt.subplots(2, 2, figsize=(15.6, 11.7), layout="constrained")

columns_inv4 = ['4-1 string energy','4-2 string energy', '4-3 string energy',
                '4-4 string energy', '4-5 string energy', '4-6 string energy',
                '4-7 string energy', '4-8 string energy', '4-9 string energy',
                '4-10 string energy', '4-11 string energy', '4-12 string energy']
energy.plot(x='Timestamp', y=columns_inv4, kind='line', ax=axes[0,0])
axes[0,0].set_ylabel('Electrical power W')
axes[0,0].set_title('INV-4 electrical power')
axes[0,0].legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), shadow=True, ncol=4)

columns_inv5 = ['5-1 string energy','5-2 string energy', '5-3 string energy',
                '5-4 string energy', '5-5 string energy', '5-6 string energy',
                '5-7 string energy', '5-8 string energy', '5-9 string energy',
                '5-10 string energy', '5-11 string energy', '5-12 string energy']
energy.plot(x='Timestamp', y=columns_inv5, kind='line', ax=axes[0,1])
axes[0,1].set_ylabel('Electrical power W')
axes[0,1].set_title('INV-5 electrical power')
axes[0,1].legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), shadow=True, ncol=4)
```

```
columns_inv8 = ['8-1 string energy','8-2 string energy', '8-3 string energy',
                '8-4 string energy', '8-5 string energy', '8-6 string energy',
                '8-7 string energy', '8-8 string energy', '8-9 string energy',
                '8-10 string energy', '8-11 string energy', '8-12 string energy']
energy.plot(x='Timestamp', y=columns_inv8, kind='line', ax=axes[1,0])
axes[1,0].set_ylabel('Electrical power W')
axes[1,0].set_title('INV-8 electrical power')
axes[1,0].legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), shadow=True, ncol=4)

columns_inv11 = ['11-1 string energy','11-2 string energy', '11-3 string energy',
                 '11-4 string energy', '11-5 string energy', '11-6 string energy',
                 '11-7 string energy', '11-8 string energy', '11-9 string energy',
                 '11-10 string energy', '11-11 string energy', '11-12 string energy']
energy.plot(x='Timestamp', y=columns_inv11, kind='line', ax=axes[1,1])
axes[1,1].set_ylabel('Electrical power W')
axes[1,1].set_title('INV-11 electrical power')
axes[1,1].legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), shadow=True, ncol=4)

for ax in axes.flat:
    ax.tick_params(axis='x', rotation=0)

#Zoomed in view
axes[0,0].set_xlim(pd.Timestamp('2023-05-31'), pd.Timestamp('2023-06-06'))
axes[0,1].set_xlim(pd.Timestamp('2023-05-31'), pd.Timestamp('2023-06-06'))
axes[1,0].set_xlim(pd.Timestamp('2023-05-31'), pd.Timestamp('2023-06-06'))
axes[1,1].set_xlim(pd.Timestamp('2023-05-31'), pd.Timestamp('2023-06-06'))
fig.savefig('Block2 - after pre-processing zoomed.pdf', format='pdf', dpi=1200, bbox_inches='t

#fig.savefig('Block2 after pre-processing.pdf', format='pdf', dpi=1200, bbox_inches='tight')

plt.show()
#-------------------------------------------------------------------------------
#Descriptive statistics of pre-processed data to identify days with the highest generated ener

energy = energy.set_index('Timestamp')
energy.drop(columns=['Irradiance'], inplace=True)

daily_energy = energy.resample('D').mean()

daily_describtion = daily_energy.T.describe()
```

```
daily_describtion_T = daily_describtion.T

# Finding days with the highest total power generation
daily_energy['Total_power'] = daily_energy.sum(axis=1)
dates_with_highest_power = daily_energy['Total_power'].nlargest(10).index

total_observations = daily_energy.sum(axis=1)
total_observations = total_observations.reset_index()

sorted_total_observations = total_observations.sort_values(by=0, ascending=False)
sorted_total_observations = sorted_total_observations.set_index('Timestamp')
total_observations = sorted_total_observations.reset_index()

#-------------------------------------------------------------------------
#Dividing data set based on the inverters

INV_1 = pd.DataFrame(index=energy.index)
INV_2 = pd.DataFrame(index=energy.index)
INV_3 = pd.DataFrame(index=energy.index)
INV_4 = pd.DataFrame(index=energy.index)
INV_5 = pd.DataFrame(index=energy.index)
INV_6 = pd.DataFrame(index=energy.index)
INV_7 = pd.DataFrame(index=energy.index)
INV_8 = pd.DataFrame(index=energy.index)
INV_9 = pd.DataFrame(index=energy.index)
INV_10 = pd.DataFrame(index=energy.index)
INV_11 = pd.DataFrame(index=energy.index)
INV_12 = pd.DataFrame(index=energy.index)

INV_1 = energy[['1-1 string energy', '1-2 string energy', '1-3 string energy', '1-4 string ene
               '1-5 string energy', '1-6 string energy', '1-7 string energy', '1-8 string ener
               '1-9 string energy', '1-10 string energy', '1-11 string energy', '1-12 string e
               '1-13 string energy']].copy()
INV_2 = energy[['2-1 string energy','2-2 string energy', '2-3 string energy', '2-4 string ener
                '2-5 string energy', '2-6 string energy', '2-7 string energy', '2-8 string ene
                '2-9 string energy', '2-10 string energy', '2-11 string energy', '2-12 string
                '2-13 string energy']].copy()
INV_3 = energy[['3-1 string energy','3-2 string energy', '3-3 string energy', '3-4 string ener
                '3-5 string energy', '3-6 string energy', '3-7 string energy', '3-8 string ene
                '3-9 string energy', '3-10 string energy', '3-11 string energy', '3-12 string
                '3-13 string energy']].copy()
```

```
INV_4 = energy[['4-1 string energy','4-2 string energy', '4-3 string energy', '4-4 string ener
                '4-5 string energy', '4-6 string energy', '4-7 string energy', '4-8 string ene
                '4-9 string energy', '4-10 string energy', '4-11 string energy', '4-12 string
INV_5 = energy[['5-1 string energy','5-2 string energy', '5-3 string energy', '5-4 string ener
                '5-5 string energy', '5-6 string energy', '5-7 string energy', '5-8 string ene
                '5-9 string energy', '5-10 string energy', '5-11 string energy', '5-12 string
INV_6 = energy[['6-1 string energy','6-2 string energy', '6-3 string energy', '6-4 string ener
                '6-5 string energy', '6-6 string energy', '6-7 string energy', '6-8 string ene
                '6-9 string energy', '6-10 string energy']].copy()
INV_7 = energy[['7-1 string energy','7-2 string energy', '7-3 string energy', '7-4 string ener
                '7-5 string energy', '7-6 string energy', '7-7 string energy', '7-8 string ene
                '7-9 string energy', '7-10 string energy']].copy()
INV_8 = energy[['8-1 string energy','8-2 string energy', '8-3 string energy', '8-4 string ener
                '8-5 string energy', '8-6 string energy', '8-7 string energy', '8-8 string ene
                '8-9 string energy', '8-10 string energy', '8-11 string energy', '8-12 string
INV_9 = energy[['9-1 string energy','9-2 string energy', '9-3 string energy', '9-4 string ener
                '9-5 string energy', '9-6 string energy', '9-7 string energy', '9-8 string ene
                '9-9 string energy', '9-10 string energy', '9-11 string energy', '9-12 string
                '9-13 string energy']].copy()
INV_10 = energy[['10-1 string energy','10-2 string energy', '10-3 string energy', '10-4 string
                 '10-5 string energy', '10-6 string energy', '10-7 string energy', '10-8 strin
                 '10-9 string energy', '10-10 string energy', '10-11 string energy', '10-12 st
                 '10-13 string energy']].copy()
INV_11 = energy[['11-1 string energy','11-2 string energy', '11-3 string energy', '11-4 string
                 '11-5 string energy', '11-6 string energy', '11-7 string energy', '11-8 strin
                 '11-9 string energy', '11-10 string energy', '11-11 string energy',
                 '11-12 string energy']].copy()
INV_12 = energy[['12-1 string energy', '12-2 string energy', '12-3 string energy', '12-4 strin
                 '12-5 string energy', '12-6 string energy', '12-7 string energy', '12-8 strin
                 '12-9 string energy', '12-10 string energy']].copy()

#Correlation

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_1 = INV_1.corr()
heatmap = sns.heatmap(corr_INV_1, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_1')
plt.savefig('heatmap_INV1.pdf', format='pdf', dpi=1200)
plt.show()
```

```
plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_2 = INV_2.corr()
heatmap = sns.heatmap(corr_INV_2, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_2')
plt.savefig('heatmap_INV2.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_3 = INV_3.corr()
heatmap = sns.heatmap(corr_INV_3, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_3')
plt.savefig('heatmap_INV3.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_4 = INV_4.corr()
heatmap = sns.heatmap(corr_INV_4, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_4')
plt.savefig('heatmap_INV4.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_5 = INV_5.corr()
heatmap = sns.heatmap(corr_INV_5, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_5')
plt.savefig('heatmap_INV5.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_6 = INV_6.corr()
heatmap = sns.heatmap(corr_INV_6, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_6')
plt.savefig('heatmap_INV6.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
```

```
sns.set_theme(style="white")
corr_INV_7 = INV_7.corr()
heatmap = sns.heatmap(corr_INV_7, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_7')
plt.savefig('heatmap_INV7.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_8 = INV_8.corr()
heatmap = sns.heatmap(corr_INV_8, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_8')
plt.savefig('heatmap_INV8.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_9 = INV_9.corr()
heatmap = sns.heatmap(corr_INV_9, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_9')
plt.savefig('heatmap_INV9.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_10 = INV_10.corr()
heatmap = sns.heatmap(corr_INV_10, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_10')
plt.savefig('heatmap_INV10.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
corr_INV_11 = INV_11.corr()
heatmap = sns.heatmap(corr_INV_11, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_11')
plt.savefig('heatmap_INV11.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
sns.set_theme(style="white")
```

```
corr_INV_12 = INV_12.corr()
heatmap = sns.heatmap(corr_INV_12, annot=True, cmap="Blues", fmt='.1g')
heatmap.set_title('Correlation Heatmap of INV_12')
plt.savefig('heatmap_INV12.pdf', format='pdf', dpi=1200)
plt.show()

#-------------------------------------------------------------------------------
#Boxplots

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_1.boxplot()
plt.title('Box Plots of INV_1')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV1.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_2.boxplot()
plt.title('Box Plots of INV_2')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV2.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_3.boxplot()
plt.title('Box Plots of INV_3')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV3.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_4.boxplot()
plt.title('Box Plots of INV_4')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV4.pdf', format='pdf', dpi=1200)
plt.show()
```

```
plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_5.boxplot()
plt.title('Box Plots of INV_5')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV5.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_6.boxplot()
plt.title('Box Plots of INV_6')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV6.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_7.boxplot()
plt.title('Box Plots of INV_7')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV7.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_8.boxplot()
plt.title('Box Plots of INV_8')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV8.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_9.boxplot()
plt.title('Box Plots of INV_9')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV9.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
```

```
INV_10.boxplot()
plt.title('Box Plots of INV_10')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV10.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_11.boxplot()
plt.title('Box Plots of INV_11')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV11.pdf', format='pdf', dpi=1200)
plt.show()

plt.figure(figsize=(6.4,4.8), layout="constrained")
INV_12.boxplot()
plt.title('Box Plots of INV_12')
plt.ylabel('Electrical power W')
plt.xticks(rotation=45)
plt.savefig('boxplot_INV12.pdf', format='pdf', dpi=1200)
plt.show()

#-------------------------------------------------------------------------------
#CLUSTERING

energy.set_index('Timestamp', inplace=True) #sets time as an index
energy = energy.reset_index() #resets index
################################################################################
#6 June, 2023

start_date = '2023-06-06 6:00:00'
end_date = '2023-06-06 20:00:00'

day_060623 = energy.loc[start_date:end_date]

#Transforming dataframes to np array and transposing it
day_060623_np = day_060623.to_numpy()
day_060623_np = np.transpose(day_060623_np)

#Choosing number of clusters
```

```
#Sillhouette method
range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
silhouette_avg = []

for num_clusters in range_n_clusters:

 # initialise kmeans
 kmeans = KMeans(n_clusters=num_clusters)
 kmeans.fit(day_060623_np)
 cluster_labels = kmeans.labels_

 # silhouette score
 silhouette_avg.append(silhouette_score(day_060623_np, cluster_labels))

plt.plot(range_n_clusters, silhouette_avg, 'bx-')
plt.xlabel('Values of k')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis for optimal k on 2023.06.06')
plt.savefig('20230606_silhoutte.pdf', format='pdf', dpi=1200)
plt.show()

#Elbow method
Sum_of_squared_distances = []
K = range(1,13)

for num_clusters in K :
 kmeans = KMeans(n_clusters=num_clusters)
 kmeans.fit(day_060623_np)
 Sum_of_squared_distances.append(kmeans.inertia_)

plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('Values of k')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow method for optimal k on 2023.06.06')
plt.savefig('20230606_elbow.pdf', format='pdf', dpi=1200)
plt.show()

#Based on the analysis, we choose number of clusters
num_clusters = 4
```

```
# Create and fit the KMeans model
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
cluster_labels_060623 = kmeans.fit_predict(day_060623_np)

# Add cluster labels as a new column in your DataFrame
day_060623 = day_060623.T
day_060623['Cluster'] = cluster_labels_060623

#Visualizing clusters

day_060623_y = day_060623['Cluster']
day_060623_y = day_060623_y.reset_index() #resets index
day_060623_y = day_060623_y.drop(['index'], axis=1)
C_day_060623 = day_060623.copy()
day_060623 = day_060623.drop(['Cluster'], axis=1)
day_060623_sc = StandardScaler().fit_transform(day_060623)
day_060623_sc = pd.DataFrame(day_060623_sc)
day_060623_sc.head()

tsne = manifold.TSNE(n_components = 2, verbose=0, perplexity=10, n_iter=300, random_state = 42
day_060623_p10 = tsne.fit_transform(day_060623_sc)
day_060623_p10 = pd.DataFrame(day_060623_p10)
day_060623_p10.rename(columns = {0:'comp_1', 1:'comp_2'}, inplace = True)
day_060623_p10['label'] = day_060623_y

tsne = manifold.TSNE(n_components = 2, verbose=0, perplexity=30, n_iter=300, random_state = 42
day_060623_p30 = tsne.fit_transform(day_060623_sc)
day_060623_p30 = pd.DataFrame(day_060623_p30)
day_060623_p30.rename(columns = {0:'comp_1', 1:'comp_2'}, inplace = True)
day_060623_p30['label'] = day_060623_y

tsne = manifold.TSNE(n_components = 2, verbose=0, perplexity=40, n_iter=300, random_state = 42
day_060623_p40 = tsne.fit_transform(day_060623_sc)
day_060623_p40 = pd.DataFrame(day_060623_p40)
day_060623_p40.rename(columns = {0:'comp_1', 1:'comp_2'}, inplace = True)
day_060623_p40['label'] = day_060623_y

tsne = manifold.TSNE(n_components = 2, verbose=0, perplexity=50, n_iter=300, random_state = 42
day_060623_p50 = tsne.fit_transform(day_060623_sc)
day_060623_p50 = pd.DataFrame(day_060623_p50)
day_060623_p50.rename(columns = {0:'comp_1', 1:'comp_2'}, inplace = True)
```

```
day_060623_p50['label'] = day_060623_y

tsne = manifold.TSNE(n_components = 2, verbose=0, perplexity=70, n_iter=300, random_state = 42
day_060623_p70 = tsne.fit_transform(day_060623_sc)
day_060623_p70 = pd.DataFrame(day_060623_p70)
day_060623_p70.rename(columns = {0:'comp_1', 1:'comp_2'}, inplace = True)
day_060623_p70['label'] = day_060623_y

tsne = manifold.TSNE(n_components = 2, verbose=0, perplexity=100, n_iter=300, random_state = 4
day_060623_p100 = tsne.fit_transform(day_060623_sc)
day_060623_p100 = pd.DataFrame(day_060623_p100)
day_060623_p100.rename(columns = {0:'comp_1', 1:'comp_2'}, inplace = True)
day_060623_p100['label'] = day_060623_y

fig, axes = plt.subplots(2, 3, figsize = (16,12))
fig.suptitle('Dataset projections with different perplexities 2023.06.06', size=24)

sns.set_style("darkgrid") # darkgrid, whitegrid, dark, white, ticks

sns.scatterplot(ax=axes[0,0],
  data = day_060623_p10,
  hue = day_060623_p10['label'].tolist(),
  palette = sns.color_palette("Paired"),
  x = "comp_1", y = "comp_2")
axes[0,0].set_title('Perplexity = 10', size=16)
axes[0,0].get_legend().remove()

sns.scatterplot(ax=axes[0,1],
  data = day_060623_p30,
  hue = day_060623_p30['label'].tolist(),
  palette = sns.color_palette("Paired"),
  x = "comp_1", y = "comp_2")
axes[0,1].set_title('Perplexity = 30', size=16)
axes[0,1].get_legend().remove()

sns.scatterplot(ax=axes[0,2],
  data = day_060623_p40,
  hue = day_060623_p40['label'].tolist(),
  palette = sns.color_palette("Paired"),
  x = "comp_1", y = "comp_2")
axes[0,2].set_title('Perplexity = 40', size=16)
```

```
axes[0,2].get_legend().remove()

sns.scatterplot(ax=axes[1,0],
  data = day_060623_p50,
  hue = day_060623_p50['label'].tolist(),
  palette = sns.color_palette("Paired"),
  x = "comp_1", y = "comp_2")
axes[1,0].set_title('Perplexity = 50', size=16)
axes[1,0].get_legend().remove()

sns.scatterplot(ax=axes[1,1],
  data = day_060623_p70,
  hue = day_060623_p70['label'].tolist(),
  palette = sns.color_palette("Paired"),
  x = "comp_1", y = "comp_2")
axes[1,1].set_title('Perplexity = 70', size=16)
axes[1,1].get_legend().remove()


sns.scatterplot(ax=axes[1,2],
  data = day_060623_p100,
  hue = day_060623_p100['label'].tolist(),
  palette = sns.color_palette("Paired"),
  x = "comp_1", y = "comp_2")
axes[1,2].set_title('Perplexity = 100', size=16)
axes[1,2].get_legend().remove()

lines, labels = axes[0,0].get_legend_handles_labels()

fig.legend(lines, labels, loc = 'lower center', ncol=10)

plt.show()

sns.scatterplot(data = day_060623_p70, hue = day_060623_p70['label'].tolist(),
                palette = sns.color_palette("bright"), x = "comp_1", y = "comp_2")
plt.title('K-means clustering results with k=4 on 2023-06-06')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), ncol=4)
plt.savefig('20230606_TSNE.pdf', format='pdf', dpi=1200, bbox_inches='tight')
plt.show()

#-------------------------------------------------------------------------
```

```
#Summing up how much electrical power was generated during the day
total_observations_060623 = day_060623.sum(axis=1)

#Descriptive statistics
describtion_060623 = day_060623.T.describe()
describtion_060623_T = describtion_060623.T

#Splitting data into data frames based on class:

# Group the DataFrame by the 'classes' column
grouped = C_day_060623.groupby('Cluster')

# Create separate DataFrames for each class
Cluster0_060623 = pd.DataFrame()
Cluster1_060623 = pd.DataFrame()
Cluster2_060623 = pd.DataFrame()
Cluster3_060623 = pd.DataFrame()

# Iterate over the groups and assign each DataFrame to the appropriate variable
for class_name, group_data in grouped:
    if class_name == 0:
        Cluster0_060623 = group_data.copy()  # Create a copy of the group data
    elif class_name == 1:
        Cluster1_060623 = group_data.copy()  # Create a copy of the group data
    elif class_name == 2:
        Cluster2_060623 = group_data.copy()  # Create a copy of the group data
    elif class_name == 3:
        Cluster3_060623 = group_data.copy()  # Create a copy of the group data

Cluster0_060623 = Cluster0_060623.drop('Cluster', axis='columns')
Cluster1_060623 = Cluster1_060623.drop('Cluster', axis='columns')
Cluster2_060623 = Cluster2_060623.drop('Cluster', axis='columns')
Cluster3_060623 = Cluster3_060623.drop('Cluster', axis='columns')

Cluster0_060623 = Cluster0_060623.T
Cluster1_060623 = Cluster1_060623.T
Cluster2_060623 = Cluster2_060623.T
Cluster3_060623 = Cluster3_060623.T

describtion_060623_C0 = Cluster0_060623.describe()
describtion_060623_C0_T = describtion_060623_C0.T
```

```
describtion_060623_C1 = Cluster1_060623.describe()
describtion_060623_C1_T = describtion_060623_C1.T

describtion_060623_C2 = Cluster2_060623.describe()
describtion_060623_C2_T = describtion_060623_C2.T

describtion_060623_C3 = Cluster3_060623.describe()
describtion_060623_C3_T = describtion_060623_C3.T

#Correlation

day_060623.drop(columns=['Timestamp'], inplace=True)
day_060623 = day_060623.T

plt.figure(figsize=(20,15))
sns.set_theme(style="white")
corr_060623 = day_060623.corr()
heatmap = sns.heatmap(corr_060623, annot=True, cmap="Blues", fmt='.1g')

#Let's group strings based on correlation result:

pairs_smaller_than_09 = []
pairs_smaller_than_08 = []
pairs_smaller_than_07 = []
pairs_smaller_than_06 = []
pairs_smaller_than_05 = []
pairs_smaller_than_04 = []
pairs_smaller_than_03 = []
pairs_smaller_than_02 = []

for i in range(len(corr_060623.columns)):
    for j in range(i + 1, len(corr_060623.columns)):
        if corr_060623.iloc[i, j] > 0.8 and corr_060623.iloc[i, j] < 0.9:
            pairs_smaller_than_09.append((corr_060623.columns[i], corr_060623.columns[j]))

for i in range(len(corr_060623.columns)):
    for j in range(i + 1, len(corr_060623.columns)):
        if corr_060623.iloc[i, j] <= 0.8 and corr_060623.iloc[i, j] > 0.7:
            pairs_smaller_than_08.append((corr_060623.columns[i], corr_060623.columns[j]))
```

```
for i in range(len(corr_060623.columns)):
    for j in range(i + 1, len(corr_060623.columns)):
        if corr_060623.iloc[i, j] <= 0.7 and corr_060623.iloc[i, j] > 0.6:
            pairs_smaller_than_07.append((corr_060623.columns[i], corr_060623.columns[j]))

for i in range(len(corr_060623.columns)):
    for j in range(i + 1, len(corr_060623.columns)):
        if corr_060623.iloc[i, j] <= 0.6 and corr_060623.iloc[i, j] > 0.5:
            pairs_smaller_than_06.append((corr_060623.columns[i], corr_060623.columns[j]))

for i in range(len(corr_060623.columns)):
    for j in range(i + 1, len(corr_060623.columns)):
        if corr_060623.iloc[i, j] <= 0.5 and corr_060623.iloc[i, j] > 0.4:
            pairs_smaller_than_05.append((corr_060623.columns[i], corr_060623.columns[j]))

for i in range(len(corr_060623.columns)):
    for j in range(i + 1, len(corr_060623.columns)):
        if corr_060623.iloc[i, j] <= 0.4 and corr_060623.iloc[i, j] > 0.3:
            pairs_smaller_than_04.append((corr_060623.columns[i], corr_060623.columns[j]))

for i in range(len(corr_060623.columns)):
    for j in range(i + 1, len(corr_060623.columns)):
        if corr_060623.iloc[i, j] <= 0.3 and corr_060623.iloc[i, j] > 0.2:
            pairs_smaller_than_03.append((corr_060623.columns[i], corr_060623.columns[j]))

for i in range(len(corr_060623.columns)):
    for j in range(i + 1, len(corr_060623.columns)):
        if corr_060623.iloc[i, j] < 0.2:
            pairs_smaller_than_02.append((corr_060623.columns[i], corr_060623.columns[j]))
#-------------------------------------------------------------------------------------------
#%% FDA

#STEP 1 - getting functional data

day_070523 = day_070523.reset_index()

data_matrix_070523 = day_070523.drop(columns=['Timestamp']).values
data_matrix_070523 = data_matrix_070523.T

grid_points_070523 = day_070523.index.values
```

```
#Creating FDataGrid object with specified grid_points
fd_070523 = skfda.FDataGrid(data_matrix=data_matrix_070523, grid_points=grid_points_070523)

labels_070523 = day_070523.columns.tolist()

fd_070523.plot()
plt.legend(labels_070523, loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=10)
plt.title('Functional Data 2023.05.07')
plt.show()

#BoxPlot

colormap = plt.cm.get_cmap('seismic')

fdBoxplot = Boxplot(fd_070523_smooth)
fdBoxplot.show_full_outliers = True

fdBoxplot.plot()

color = 0.3
outliercol = 0.7

fd_070523_smooth.plot(group=fdBoxplot.outliers.astype(int),
                      group_colors=colormap([color, outliercol]),
                      group_names=["nonoutliers", "outliers"])
plt.show()

#STEP 2 - Fourier smoothing
basis = skfda.representation.basis.FourierBasis((0, 6000), n_basis=24)
smoother = skfda.preprocessing.smoothing.BasisSmoother(basis)
fd_070523_smooth = smoother.fit_transform(fd_070523)

fd_070523_smooth.plot()
plt.legend(labels_070523, loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=10)
plt.title('Smoothed Functional Data 2023.05.07')
plt.show()

#Searching for optimal nbasis value

n_basis_values = range(5, 25)
```

```
# Spliting data into training and testing sets
fd_train, fd_test = train_test_split(fd_070523, test_size=0.2, random_state=42)
grid_points = fd_train.grid_points[0]

mse_values = []

# Loop over different values of n_basis
for n_basis in n_basis_values:
    basis = skfda.representation.basis.Fourier(n_basis=n_basis)
    fd_train_basis = fd_train.to_basis(basis)
    fd_train_smooth = fd_train_basis.evaluate(grid_points)

    fd_test_basis = fd_test.to_basis(basis)
    fd_test_smooth = fd_test_basis.evaluate(grid_points)
    mse = mean_squared_error(fd_test_smooth.flatten(), fd_test.data_matrix.flatten())
    mse_values.append(mse)

optimal_n_basis = n_basis_values[np.argmin(mse_values)]
print("Optimal n_basis:", optimal_n_basis)

#Plotting MSE values for different n_basis values
plt.plot(n_basis_values, mse_values, marker='o')
plt.xlabel('Number of Basis Functions (n_basis)')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('MSE vs. n_basis')
plt.grid(True)
plt.show()

#STEP 4 - outlier detection, boxplot

colormap = plt.cm.get_cmap('seismic')

fdBoxplot = Boxplot(fd_070523_smooth)
fdBoxplot.show_full_outliers = True
fdBoxplot.plot()
plt.show()

color = 0.3
outliercol = 0.7
fd_070523_smooth.plot(group=fdBoxplot.outliers.astype(int),
                      group_colors=colormap([color, outliercol]),
```

```
                    group_names=["nonoutliers", "outliers"])
plt.show()

#STEP 3 - clustering

#%% 6 June, 2023

#STEP 1 - getting functional data

day_060623 = day_060623.reset_index()

data_matrix_060623 = day_060623.drop(columns=['Timestamp']).values
data_matrix_060623 = data_matrix_060623.T

grid_points_060623 = day_060623.index.values

#Creating FDataGrid object with specified grid_points
fd_060623 = skfda.FDataGrid(data_matrix=data_matrix_060623, grid_points=grid_points_060623)

day_060623 = day_060623.drop(columns=['Timestamp'])
labels_060623 = day_060623.columns.tolist()

fd_060623.plot()
#plt.legend(labels_060623, loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=10)
plt.title('Functional Data 2023.06.06')
plt.savefig('FDA20230606.pdf', format='pdf', dpi=1200, bbox_inches='tight')
plt.show()

#STEP 2 - Fourier smoothing
basis = skfda.representation.basis.FourierBasis((0, 56), n_basis=11)
smoother = skfda.preprocessing.smoothing.BasisSmoother(basis)
fd_060623_smooth = smoother.fit_transform(fd_060623)

fd_060623_smooth.plot()
#plt.legend(labels_060623, loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=10)
plt.title('Smoothed Functional Data 2023.06.06')
plt.savefig('FDA20230606_smooth.pdf', format='pdf', dpi=1200, bbox_inches='tight')
plt.show()

#Searching for optimal nbasis value
```

```
n_basis_values = range(0,50)

# Spliting data into training and testing sets
fd_train, fd_test = train_test_split(fd_060623, test_size=0.2, random_state=42)
grid_points = fd_train.grid_points[0]

mse_values = []

# Loop over different values of n_basis
for n_basis in n_basis_values:
    basis = skfda.representation.basis.Fourier(n_basis=n_basis)
    fd_train_basis = fd_train.to_basis(basis)
    fd_train_smooth = fd_train_basis.evaluate(grid_points)

    fd_test_basis = fd_test.to_basis(basis)
    fd_test_smooth = fd_test_basis.evaluate(grid_points)
    mse = mean_squared_error(fd_test_smooth.flatten(), fd_test.data_matrix.flatten())
    mse_values.append(mse)

optimal_n_basis = n_basis_values[np.argmin(mse_values)]
print("Optimal n_basis:", optimal_n_basis)

#Plotting MSE values for different n_basis values
plt.plot(n_basis_values, mse_values, marker='o')
plt.xlabel('Number of Basis Functions (n_basis)')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('MSE vs. n_basis')
plt.grid(True)
plt.show()

#STEP 4 - outlier detection, boxplot

colormap = plt.cm.get_cmap('seismic')

fdBoxplot = Boxplot(fd_060623_smooth)
fdBoxplot.show_full_outliers = True
fdBoxplot.plot()
plt.title('fdBoxplot 2023.06.06')
plt.savefig('fdBoxplot_2023.06.06.pdf', format='pdf', dpi=1200, bbox_inches='tight')
plt.show()
```

```
color = 0.3
outliercol = 0.7
fd_060623_smooth.plot(group=fdBoxplot.outliers.astype(int),
                      group_colors=colormap([color, outliercol]),
                      group_names=["nonoutliers", "outliers"])
#plt.legend(labels_060623, loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=10)
plt.title('Outliers 2023.06.06')
plt.savefig('outliers_2023.06.06.pdf', format='pdf', dpi=1200, bbox_inches='tight')
plt.show()

#STEP 3 - clustering

kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(fd_060623_smooth)
print(kmeans.predict(fd_060623_smooth))

ClusterPlot(kmeans, fd_060623_smooth).plot()
#plt.legend(labels_060623, loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=10)
plt.title('Functional K-means 2023.06.06')
plt.savefig('fdK-means_2023.06.06.pdf', format='pdf', dpi=1200, bbox_inches='tight')
plt.show()
```