Technical University of Crete, Greece

School of Electrical and Computer Engineering

A Dataset Generator for Smart Grid Ecosystems Populated with Electric Vehicles

Author:
Georgios Charalampidis

Thesis Committee:

Supervisor: Associate Professor Georgios Chalkiadakis (ECE)

Associate Professor Fotios Kanellos (ECE)

Associate Professor Vasileios Samoladas (ECE)

Chania, February 2022

Abstract

The widespread use of electric mobility technologies such as the Electric Vehicles (EVs), poses certain challenges both from the technical, and the socioeconomic points of view. In order to address these, research must utilize related data that originates from realistic sources. However, in the typical case, such data contains private and sensitive information that cannot be made available to the researchers. As a result, research makes excessive use of the few datasets that are publicly available. At the same time, the majority of the available data, contain only a few customers, while hundreds of thousands are required. To overcome this obstacle, synthetic data can be used, which nevertheless originates from models with relationships that sufficiently capture the properties of the actual real-world datasets.

In this thesis, we design a dataset generator for the domain of EVs charging management in Smart Grid settings. The generator (i) takes as input anonymized data, describing different energy generation and demand types, as well as charging profiles of EVs and corresponding trip and type information; (ii) employs a variety of models—in particular Histograms, Kernel Density Estimation, Generative Adversarial Networks, and Frequency Tables—using this data as training sets; and thus (iii) generates new synthetic data, not identical to the input, but adhering to the same principles, and relationships.

The proposed dataset generator also produces respective summarizations, which includes barplots and histograms to visualize the results, and different metrics in order to quantify a 'distance' between the distributions under comparison. These summarizations give us a complete picture of the generated data and they are particularly useful for detecting correspondence issues. Last but not least, the dataset generator is available via an online repository and it can readily be incorporated by third parties in their research activities.

Abstract in Greek

Η ευρεία χρήση ηλεκτρικών τεχνολογιών στον τομέα των μετακινήσεων, όπως τα Ηλεκτρικά Οχήματα (EVs), θέτει ορισμένες προκλήσεις τόσο από τεχνική όσο και από κοινωνικοοικονομική άποψη. Για να αντιμετωπιστούν οι συγκεκριμένες προκλήσεις, όσον αφορά τον τομέα της έρευνας, γίνεται επιτακτική η ανάγκη χρήσης δεδομένων τα οποία προέρχονται από ρεαλιστικές και αξιόπιστες πηγές. Ωστόσο, στην τυπική μέση περίπτωση, τέτοιου είδους δεδομένα περιέχουν ιδιωτικές και ευαίσθητες πληροφορίες που δεν μπορούν να διατεθούν στους ερευνητές. Ακόμη και τα λίγα δημόσια διαθέσιμα δεδομένα δεν είναι αρκετά για έρευνα στον τομέα, με αποτέλεσμα τα περισσότερα από αυτά να χρησιμοποιούνται υπερβολικά. Συγχρόνως, η πλειονότητα των διαθέσιμων δεδομένων, περιέχει λιγοστούς πελάτες, ενώ στην πραγματικότητα απαιτούνται εκατοντάδες χιλιάδες. Για να ξεπεραστεί αυτό το εμπόδιο, μπορούν να χρησιμοποιηθούν συνθετικά δεδομένα, τα οποία ωστόσο προέρχονται από μοντέλα με σχέσεις που αποτυπώνουν επαρκώς τις ιδιότητες των συνόλων δεδομένων του πραγματικού κόσμου.

Στην παρούσα διπλωματική εργασία, σχεδιάζουμε μια γεννήτρια δεδομένων για τον τομέα της διαχείρισης φόρτισης ηλεκτρικών οχημάτων σε οικοσυστήματα έξυπνων δικτύων ηλεκτροδότησης. Η γεννήτρια (α) λαμβάνει ως είσοδο ανώνυμα και με πιθανές ασυνέπειες δεδομένα, που περιγράφουν διαφορετικούς τύπους παραγωγής και ζήτησης ενέργειας, καθώς και διάφορα προφίλ φόρτισης ηλεκτρικών οχημάτων και αντίστοιχες πληροφορίες σχετικά με την διαδρομή και τον τύπο των EVs (β) εφαρμόζει ποικιλία μοντέλων —συγκεκριμένα Histograms, Kernel Density Estimation, Generative Adversarial Networks, και Frequency Tables —χρησιμοποιώντας αυτά τα δεδομένα ως σύνολα εκπαίδευσης και ως αποτέλεσμα καταφέρνει να (γ) δημιουργεί νέα συνθετικά δεδομένα, όχι πανομοιότυπα με τα δεδομένα εισόδου, αλλά τηρώντας τις ίδιες αρχές και σχέσεις.

Η προτεινόμενη γεννήτρια δεδομένων παράγει επίσης αντίστοιχα γραφήματα και ιστογράμματα για την οπτικοποίηση των αποτελεσμάτων και διαφορετικές μετρήσεις προκειμένου να ποσοτικοποιηθεί η «απόσταση» μεταξύ των κατανομών που συγκρίνουμε. Μέσω των συγκεκριμένων τρόπων απεικόνισης των αποτελεσμάτων, αποκτούμε μια πλήρη εικόνα των παραγόμενων συνθετικών δεδομένων και μπορούμε εύκολα να εντοπίσουμε ζητήματα αντιστοιχίας. Τέλος σημαντικό είναι πως η γεννήτρια δεδομένων είναι διαθέσιμη μέσω ενός διαδικτυακού αποθετηρίου, και μπορεί εύκολα να ενσωματωθεί από τρίτους στις ερευνητικές τους δραστηριότητες.

Acknowledgements

I would like to express my gratitude to those who helped make this thesis possible.

First and foremost, I would like to thank my supervisor Associate Professor Georgios Chalkiadakis for teaching me multi-agent systems and algorithmic game theory the past years and especially for all his support, guidance and trust throughout my thesis. Moreover, I am truly grateful to Dr. Charilaos Akasiadis for his guidance and comments throughout this work, the moral support and encouragement. I would also like to thank all my friends for motivating me, who were by my side and in their own way helped me during my studies. I cannot thank Malvina enough for all her love, patience and support all these years. Last but not least, I would especially like to thank my parents and sister; Christos, Varvara, and Zoe, who never stopped believing in me and loving me unconditionally.

Contents

List of Algorithms

LIST OF ALGORITHMS

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

Climate change is driving a transition towards fewer fossil fuel- consuming processes and thus more sustainable societies.The important sector of mobility is currently facing a transformation where Electric Vehicles (EVs) are promoted as an alternative to reduce pollution; due to the absence of direct emissions during operation, EVs are a promising technology that offers practical reductions of the CO_2 emissions. This of course only holds, provided that the additional energy that fulfills the increased demand originating from EVs is produced by renewable sources.

EV deployment has been growing rapidly over the past ten years, and the global stock of EV passenger vehicles passed 5 million in 2018, with an increase of 63% since 2017, while the rising trend continues [1]. Successfully establishing electric mobility solutions and maximizing the technological advantage from a societal aspect, requires a multilevel approach that involves car manufacturers and owners, building and civil infrastructure managers, and power system authorities to collaborate and maintain a common vision.

The operation of the EVs is two-folded: On the one hand, they can charge their batteries, preferably using energy from renewable sources that are characterized by high intermittency, operating in a Grid-to-Vehicle (G2V) mode [2]. On the other hand, they can discharge their batteries, in this way operating as temporal storage devices[3] in a Vehicle-to-Grid (V2G) mode [4], thus significantly increasing the storage capacity of the electricity network and reducing the energy that is wasted when the demand is lower than the supply. In both cases, the vehicles need to operate within the so-called Smart Grid which is equipped with algorithms that have the ability to efficiently manage large numbers of EVs [5, 6] both in the G2V [7, 8] and the V2G [9] operation of the EVs.

However, to that end, many challenges still arise in different levels, including the technical and financial ones. For instance, huge challenges lie ahead for the operation and design of the electricity distribution network [10]; the charging infrastructure needs to be properly placed to service large numbers of customers [11]; a variety of critical elements of vehicle-to-grid economics have to be accurately identified and dealt with [12]; while a grand challenge indeed arises for artificial intelligence and multiagent systems research [13], which is assigned the task, among others to optimize the aforementioned G2V/V2G operations, while respecting various interoperability and privacy issues [6, 14]. From an economic point of view the critical elements of vehicle-to-grid economics are explored in the relatively recent report of the U.S Department of Energy in [12]. As such, there is a profound need for thorough interdisciplinary research in this rising market. This in turn calls for efficient and effective simulators, which will address data engineering needs and enable the design of novel services or mechanisms with minimum technical or financial risk.

Now, data collected from smart grid ecosystems as well as from electric vehicles can be used for both academic and industrial purposes. Increased data input has been a major concern in the energy informatics sector over the last decade [15]. Previous studies of different EV datasets include statistical analysis of data collected in the Netherlands by ElaadNL [16, 17] and analysis of EV energy consumption based on data collected by the US Department of Energy [18]. However, studies require reliable data for understanding behaviors, exploring flexibility, and extrapolating results for other similar cases or sites, where data collection equipment is not yet available. The lack of reliable data to help advance research is a known problem [19]. Even where efforts have been made to make datasets publicly available [20, 21], some limitations still arise.

A first concern is the limited range and size of the available public datasets, which prevents the application of sophisticated machine learning models that typically require huge amounts of data for training. In addition, the data available may still be subject to copyright and may not be freely shared for academic or public use. For example, electricity distribution system data obtained from Advanced Metering Infrastructure (AMI) contains Personally Identifiable Information (PII), and sophisticated algorithms are required for anonymization [22] according to the legal framework [23].

In summary, although we live in the age of Big Data, in the typical case, most of the data contains private and sensitive information that cannot be made available to the

public. Even the few publicly available data is not enough for research in the domain and as a result most of them is overly used. To overcome this obstacle, we design a dataset generator for the domain of EVs charging management in Smart Grid settings. The generator takes as input anonymized- possibly inconsistent and overused-data, describing different energy generation and demand types, as well as charging profiles of EVs and corresponding trip and type information and it generates new synthetic data, which nevertheless originates from models with relationships that sufficiently capture the properties of the actual real-world datasets.

1.1 Contributions

It therefore becomes apparent that the lack of availability and difficulty in accessing datasets that can be used to analyse smart grid ecosystems is a major obstacle to further research in the field. Datasets related to EV driving and charging habits, in particular, are notoriously hard to obtain, given also the current relatively small EVs penetration rate in the automobile market. This is notwithstanding the fact that such data is of utmost importance for EVs integration in the smart grid [24, 25]. As such, the need for dataset generators like the one presented in this thesis is imperative.

Against this background, we address these difficulties by studying several anonymized publicly available datasets and their properties, and subsequently creating a generation mechanism that can produce new, synthetic data, which are however governed by the same principles as the originals and which are available to the public at zero cost. More specifically, our contributions are as follows:

- We combine several publicly available datasets to create novel ones to be used in the data generation process.
- We design and implement a novel dataset generator that trains on available real world data to create new datasets, different from the originals yet adhering to their statistics and principles.
- We provide four different synthetic data generation methods, appropriate for different types of input data; and enhance these with a data smoothing ability.

- Our generator comes complete with data summarization and visualization capabilities, which we use in order to offer a thorough evaluation of the various generation methods.
- Our dataset generator is provided as a free-to-use web service for further evaluation and use by the community.

1.2 Outline

The rest of this thesis is structured as follows: Chapter 2 contains all the necessary theoretical background which constitutes the foundations of our approach. We present the basic concept of the *Smart Grid* and we briefly describe the consepts of the *Histograms*, *Kernel Density Estimation*, *Generative Adversarial Networks* and *Frequency Tables*. Furthermore we summarize related work in the domain. In Chapter 3, we provide an overview of the proposed dataset generation methods. At first, we analyze collected data and afterwards we present the pre-processing steps we took to guarantee data consistency. Subsequently, we describe the different data generation methods we adopt and then, we briefly describe the web-based User Interface. Chapter 4 consists of the experimental setup and the experimental evaluation of our approach. Finally, Chapter 5 concludes this thesis and outlines directions for future work.

Chapter 2

Background & Related Work

2.1 Background

In this section we present the background required to read this thesis. We begin with an introduction to the smart grid and we continue with the necessary theoretical background related to the synthetic data generation techniques that we apply. Finally, we briefly describe the main obstacles encountered in our approach.

2.1.1 Smart Grid Overview

Electricity is a vital commodity for modern society. Communities that lack electricity, even for short periods of time, endanger public health as well as the economic prosperity of their citizens [26]. Electricity networks, which are the means of transport and distribution of energy, were sufficient for our uses a few decades ago, however, as more sophisticated patterns of electricity generation and consumption emerge, their management becomes more complex. Therefore, the need to renovate them becomes imperative, so that their operation remains efficient and reliable [6].

Nowadays, the electricity produced does not come only from thermal power plants that burn fossil fuels, but much of the production comes from decentralized energy producers that use renewable energy sources such as sunlight and wind. The targets set by 196 countries under the Paris Agreement [27] to reduce greenhouse gas emissions indicate that the use of renewable energy sources will increase significantly in the near future. However, due to their dependence on weather conditions, RES are by definition very

unreliable, and in combination with the fact that storing electricity is quite an expensive issue, their large-scale integration and efficient use of the modern electricity grid energy becomes quite complex and difficult. Therefore, the amount of electricity produced must be consumed at the time of its production. If the electricity balance (i.e. the difference between production and consumption) is not maintained for long periods, the grid could collapse with extremely serious consequences, as entire areas could be without electricity for many hours [28, 6].

In addition to the evolution of electricity production, electricity consumption is also changing rapidly. The increasing use of information technology and consumer electronics has reduced tolerance for interruptions, voltage fluctuations and other power quality disturbances [26]. However, the mass introduction of computers in every aspect of our daily lives is reshaping the landscape of the energy sector (eg smart meters, electric vehicles) and creating opportunities to overcome the difficulties of the past. Smart grid technology that incorporates bidirectional data flow allows customers to adjust power consumption using real-time information on electricity production, consumption and prices. This control over the use of electricity is called demand side management (DSM).

The aforementioned descriptions could be summarized in the following definition of Smart Grid which was given by the U.S. Department of Energy [26]: ***Smart Grid*** *is a fully automated power delivery network that monitors and controls every customer and node, ensuring a two-way flow of electricity and information between the power plant and the appliance, and all points in between. Its distributed intelligence, coupled with broadband communications and automated control systems, enables real-time market transactions and seamless interfaces among people, buildings, industrial plants, generation facilities, and the electric network.*

A Smart Grid is essentially a collection of smaller interconnected networks. These networks are called microgrids [29] and vary in size, from countries and large cities to small villages and neighborhoods. More specifically, a microgrid (Figure 2.1) is an integrated autonomous energy system that contains distributed energy sources as previously described, along with a number of electrical loads as well as storage units, such as electric vehicles, batteries, etc.

In accordance with the above, but also with what we briefly described in the introduction, it is obvious that EVs are directly related to the Smart Grid. On the one hand, by charging their batteries using mainly energy from renewable sources, operating in

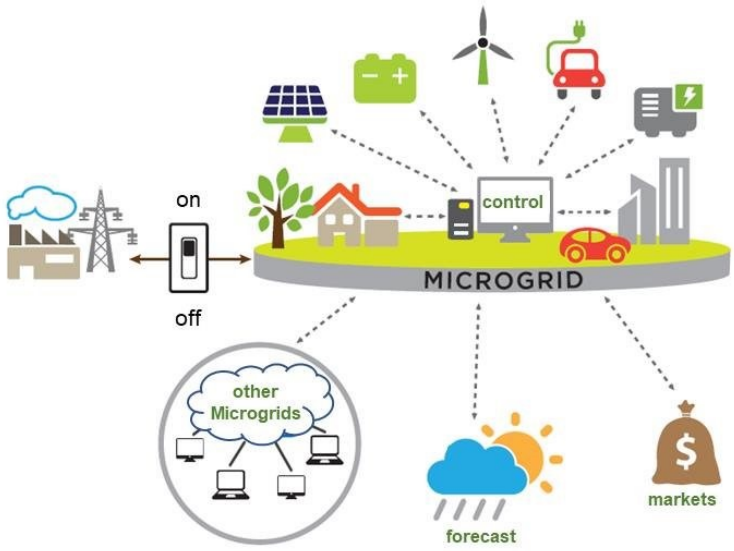


Figure 2.1: Schematic overview of a Microgrid

Grid-to-Vehicle (G2V) mode[2]. On the other hand, by remaining connected to the grid and discharging their batteries, in this way operating as temporal storage devices[3] in a Vehicle-to-Grid (V2G) mode[4].

Therefore, the Smart Grid is a wide field of application of artificial intelligence methods, and action of multi-agent systems. However, most AI approaches require a lot of data, which as we have already mentioned is difficult to find freely available as most of them belong to large companies and/or are subject to third party copyrights. It therefore becomes apparent that the lack of availability and difficulty in accessing datasets that can be used to analyse smart grid ecosystems is a major obstacle to further research in the field. Against this background, we create a generation mechanism that can produce new, synthetic data, which are however governed by the same principles as the originals and which are freely available to the public.

However, there were some constraints that we encountered during the implementation of this thesis and which led us to make specific assumptions in order to approach as best as possible the idea of the synthetic data generator. These constraints are due to the different sources from which the original data comes and especially due to the nature of the data themselves, as there are many differences in their structure and properties. The various constraints are described in detail later in the present thesis, where necessary.

2.1.2 Data Representation & Characterization Methods

2.1.2.1 Histrogram

The histogram is an approximate representation of the distribution of numerical data. It was first introduced by Karl Pearson [30] and constitutes a graphical representation that organizes a group of data points into ranges, specified by the user. The histogram condenses data series into an easily interpreted visual by taking many data points and grouping them into certain ranges or bins. In order to construct a histogram, the first step is to 'bin' the range of values—that is, divide the entire range of values into a series of intervals—and then count how many values fall into each interval. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The following is an example of a histogram and the raw data from which it was made.

Suppose we have the data shown in Table 2.1, which represent the age of 20 people. The histogram of the specified data set is shown in Figure 2.2.

Table 2.1: Dataset of the age of 20 people

28	56	33	69	36	38	42	92	47	56
49	23	56	58	52	61	33	65	73	45

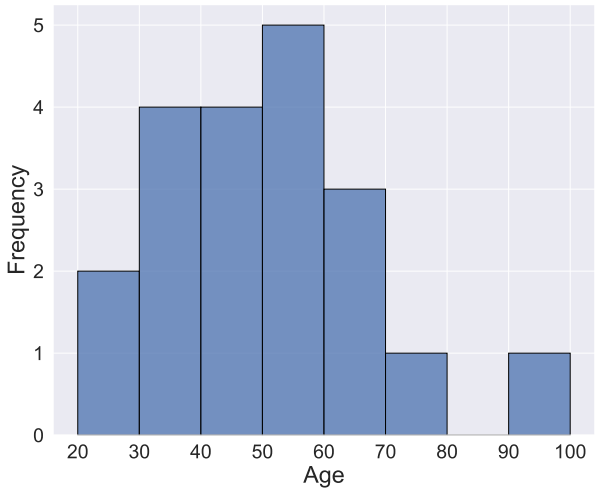


Figure 2.2: Histogram of Table 2.1

As mentioned earlier, to construct a histogram from a continuous variable we first need to split the data bins. In the example above, age has been split into bins, with each bin representing a 10-year period starting at 20 years. Each bin contains the number of occurrences of scores in the data set that are contained within that bin. Table 2.2 records the frequencies in each bin have been tabulated along with the scores (ages) that contributed to the frequency in each bin.

Table 2.2: Analysis of the histogram in Figure 2.2

Bin	Frequency	Scores (Age) included in Bin
20-30	2	28, 23
30-40	4	33, 33, 36, 38
40-50	4	42, 45, 47, 49
50-60	5	56, 56, 56, 58, 52
60-70	3	61, 69, 65
70-80	1	73
80-90	0	-
90-100	1	92

Therefore, given an unknown probability distribution and sufficient data sampled from it, it is possible to sample additional data approximating the unknown distribution with a histogram. In subsection 3.3.1 we describe in detail how we use histogram sampling technique, to generate synthetic data using multiple subsets of the original datasets.

2.1.2.2 Kernel Density Estimation

Kernel Density Estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable. Kernel density estimation is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample. Therefore given a sample of n independent, identically distributed (i.i.d) observations $(x_1, x_2, \ldots, x_n)$ of a random variable from an unknown distribution at any

given point x, the kernel density estimate is given by:

$$p(x) = \frac{1}{nh}\sum_{j=1}^{n} K\left(\frac{x - x_j}{h}\right) \tag{2.1}$$

where $K(x;h)$ is the kernel function and h is the smoothing parameter, also called the bandwidth. There is a wide range of kernel functions, such as Gaussian, Top Hat, Exponential, Linear, Cosine, Epanechnikov and others.

In order to provide an into dive explanation of the KDE method, we will compare it with the most common density estimation technique, to which we referred earlier, the histogram. As we mentioned, a histogram is a simple visualization of data where bins are defined, and the number of data points within each bin is tallied. A major problem with histograms, however, is that the choice of binning can have a disproportionate effect on the resulting visualization. For example, suppose we have created a data set of 20 points, that are drawn from two regular distributions. Figure 2.3 shows two histograms for the specific data set, with different binning for each one of them. On the left, the histogram makes clear that this is a bimodal distribution. By contrast, on the right, we see a unimodal distribution with a long tail. Comparing the two histograms we probably could not guess that they were built from the same data.

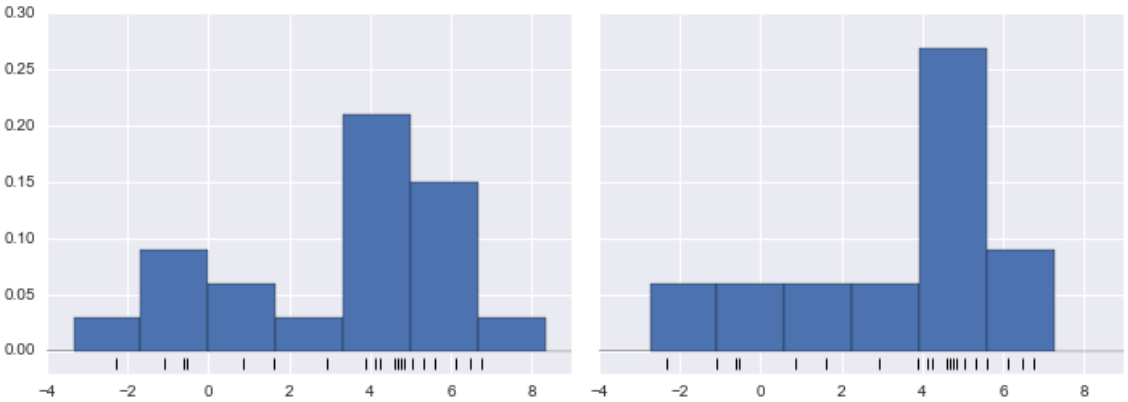


Figure 2.3: Histograms of a 20-point dataset (7 bins-left, 6 bins-right)

Therefore it is obvious that histograms depend on end points of bins. We can think of a histogram as a stack of blocks, one block per point (Figure 2.4). By stacking the blocks in the appropriate grid space, we can recover the histogram. The problem with the two binnings (Figure 2.3) stems from the fact that the height of the block stack often depends not on the actual density of points nearby, but on coincidences of how the bins align with the data points. This mis-alignment between points and their blocks is a potential cause of the poor histogram results seen in this example.

KDE attempts to remove the dependence on the end points of the bins. Instead of stacking the blocks aligned with the bins, each of the blocks is centered at each data point it represents. Therefore the blocks will not be aligned, but the sum of their contributions at each position along the x-axis gives the desired result. This is the basic idea behind kernel density estimation, in one dimension.

The result of the above procedure is shown in Figure 2.5, which is a much more robust reflection of the actual data characteristics than is the standard histogram. However, the rough edges still do not best reflect the true properties of the data. In order to smooth them out, we could replace the blocks at each location with a smooth function–like a Gaussian–as shown in Figure 2.6. This smoothed-out plot, with a Gaussian distribution contributed at the location of each input point, gives a much more accurate idea of the shape of the data distribution, and is the one that has the least variance (i.e., changes much less in response to differences in sampling).

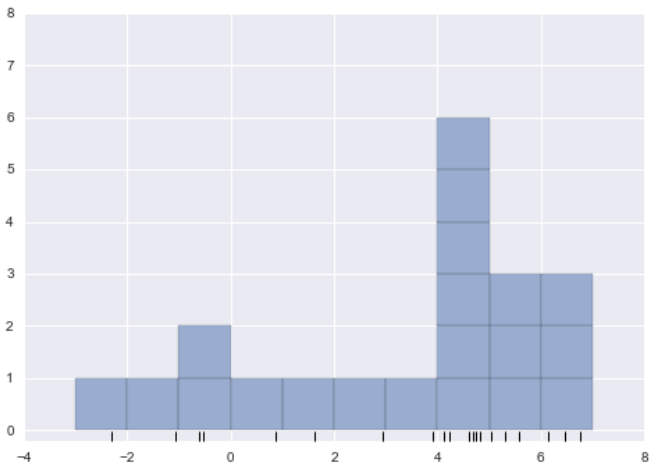


Figure 2.4: 10 bins Histogram of the 20-point dataset shown in Figure 2.3

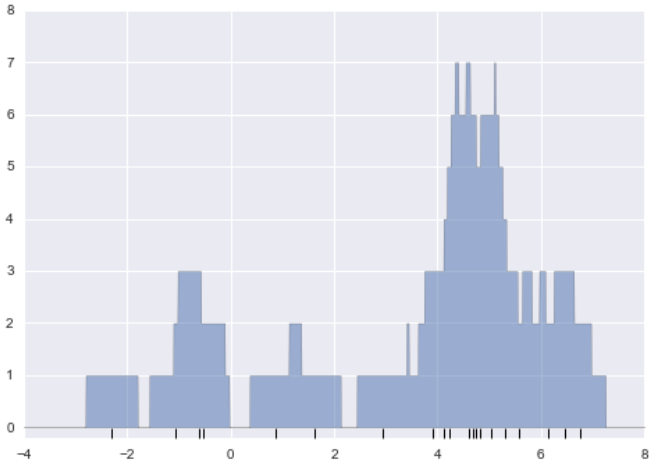


Figure 2.5: KDE of the 20-point dataset shown in Figure 2.3-TopHat kernel

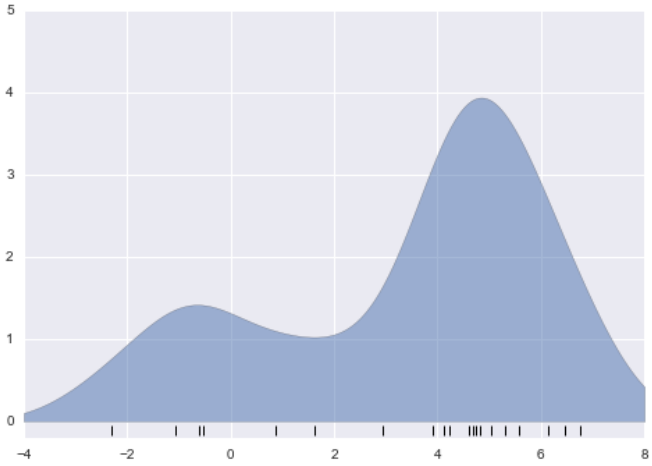


Figure 2.6: KDE of the 20-point dataset shown in Figure 2.3-Gaussian kernel

2.1.2.3 Generative Adversarial Networks

A *Generative Adversarial Network (GAN)* [31] is a class of machine learning frameworks that constitute an approach to generative modeling using deep learning methods, such as convolutional neural networks. Generative modeling is an unsupervised learning task that involves automatically discovering and learning the regularities or patterns in input data, in such a way that the model can be used to generate or output new examples that could have plausibly been drawn from the original dataset.

The GAN model architecture involves two sub-models: a generator model for generating new examples and a discriminator model for classifying whether generated examples are real (from the domain), or fake (generated by the generator model). The two models are trained together. Initially the generator generates a batch of samples, and these, along with real examples from the domain, are provided to the discriminator and classified as real or fake. The discriminator is then updated to get better at discriminating real and fake samples in the next round, and importantly, the generator is updated based on how well, or not, the generated samples 'fooled' the discriminator.

In this way, the two models are competing against each other, they are adversarial in the game theory sense, and are playing a zero-sum game,[1] until, ideally, the discriminator model is 'fooled' about half the time, meaning the generator model is generating plausible examples.

Time-series Generative Adversarial Networks

Time-series Generative Adversarial Networks (TimeGANs) [33], are GAN based models, able to generate time-series data that preserve temporal dynamics, in the sense that new sequences respect the original relationships between variables across time. Unlike other GAN architectures where unsupervised adversarial loss on both real and synthetic data is implemented, TimeGAN introduces the concept of supervised loss: the model is encouraged to capture time conditional distribution within the data by using the original data as a supervision. In contrast to GAN, which as we mentioned consists of two sub-models/networks, TimeGAN synthesizes sequential data composed by 4 networks that play distinct roles in the data modelling process: the generator, the discriminator, and recovery and embeddings models. In subsection 3.3.3 we describe in more detail the TimeGan's structure and how we use them in order to generate synthetic data.

[1] In game theory and economic theory, a zero-sum game is a mathematical representation of a situation in which an advantage that is won by one of two sides is lost by the other. If the total gains of the participants are added up, and the total losses are subtracted, they will sum to zero. [32]

2.1.2.4 Frequency Tables

A frequency table is a chart that shows the popularity or mode of a certain type of data. When examining the frequency, we count the number of times an event occurs within a given scenario. A relative frequency table is a chart that shows the popularity or mode of a certain type of data based on the population sampled. Therefore, when referring to relative frequency, we refer to the number of times a specific event occurs compared to the total number of events.

In other words, a relative frequency table shows the "popularity" of a particular value, based on the input data. In order to find the relative frequency of each value, we count how often a data value (or a range of values) appears in the original data, and then we scale it by dividing it with the total number of counts across all values.

In order to understand how a Relative Frequency Table works in practice, let us take a look at a simple example. Suppose we have a total of 500 electric vehicles, of specific models. The first column of Table 2.3 lists all the available models, the second column lists the frequency of their occurrence, while in the third column we have calculated their relative frequency. Essentially, the relative frequencies give us the probability of getting an observation from the corresponding category in a random draw. To confirm the above, the sum of the 2nd column is equal to 500, while the sum of all relative frequencies is equal to one. Finally Figure 2.7 visualizes the relative frequency distribution of Table 2.3.

Table 2.3: Relative Frequency Table of 500 EVs

EV models	**Frequency**	**Relative Frequency**
Tesla model S	115	0.23
Peugeot e-208	70	0.14
Nissan Leaf	105	0.21
BMW i3	55	0.11
Tesla model X	80	0.16
Opel Corsa-e	45	0.09
Honda e	30	0.06

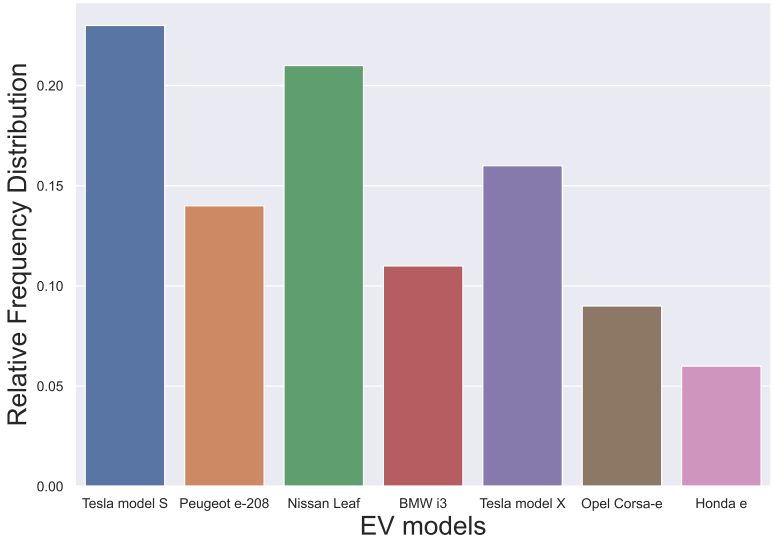


Figure 2.7: Bar-plot of Table 2.3

2.2 Related Work

To address the limitations we discussed in the introduction, various researchers have turned to creating synthetic datasets. The resulting data can be considered equivalent, as it is generated from probabilistic models that are trained on the actual data, while maintaining the confidentiality of the original data set.

In [34] the authors focus on the problem of how expensive and time-consuming the creation of labeled training datasets is, when incorporating machine learning techniques. To tackle these shortcomings, they propose a paradigm for the computational creation of training sets, known as data programming. In data programming, users express weak supervision strategies as noisy and perhaps conflicting programs (labeling functions) that label subsets of the data. Data programming is considered a way for creating machine learning models by non-specialists, especially in cases where training data is limited or unavailable. Data programming is a technique attracting interest in various fields. For example [35] applies data programming to the problem of cross-modal weak supervision in medicine, where weak labels derived from text are used to train models over a different target modality (i.e., images).

In a more relevant work to ours, [36] generates smart meter data with relatively small

datasets as input, by training autoregressive time series models. However, this approach requires hand-crafted features such as time series "deseasonalization". Accurate modeling of the underlying causes is a process that requires several assumptions to be made (e.g., the Markov property), which are not necessarily true, thus affecting the reliability of the generated data. In [37], the authors develop a probabilistic time series model for the creation of synthetic data for smart grid ecosystems by learning the probability distribution of real time-series data using a Deep Generative Adversarial Network (GAN) model. They then use this model to create synthetic data of end-user energy consumption.

Flammini et al. [38] used beta mixture models to represent multi-modal probability distributions and to analyze EV-related variables. Based on the derived results, the authors draw interesting results and suggestions, without, however, including a process of producing and evaluating samples from the models they used. In addition, the use of probability density functions based on Gaussian Mixture Models (GMMs) to represent key charging characteristics of EVs (e.g., arrival times) can be used to sample random arrivals. This technique was applied in [16], where real data from 221 EVs of the largest test in the United Kingdom and Europe (My Electric Avenue[1]) were used. Another method involves the use of a stochastic simulation methodology to create a program of daily travel and charging profiles for a population of electric vehicles [39].

Authors in [40] introduce ACN-Data, a dynamically populated dataset of EVs charging in workplaces, which includes over 30,000 sessions. Gaussian mixture models are incorporated to learn and predict user behavior, and, combined with historical data, the size of on-site solar generation is adjusted to minimize charging costs. Although ACN-Data seems to be a useful and reliable tool, it only includes workplace EV charging data and does not provide any EVs trip related or production and consumption data. This is also the case for the work of [41], where a review of available EV charging data is included.

To the best of our knowledge, only two cases of EV-specific data generation have been reported in the literature so far. The first one, RAMP-mobility [42], relies on user input that indicates specific properties of the data, e.g., population characteristics, peak intervals duration, etc., and generates mobility and demand data for a number of EVs, following a stochastic model. The second one, emobpy [43], which is closer to the approach that we adopt, receives in the input relative frequencies calculated using real data, to

[1] https://www.ssen.co.uk/myelectricavenue/

generate time series of vehicle mobility, driving energy consumption, grid connectivity, and electricity demand of charging tasks.

Based on the previously presented works, we conclude that despite the efforts to produce synthetic data and make them available to the public free of charge, the lack of data on smart grids and especially regarding EVs, remains a major obstacle for further research in the field. The limited volume of the available data is a crucial problem, while at the same time most of it may still be subject to copyright and, thus, not freely available. In any case, it is quite difficult to gain access, let alone datasets that combine different types of data. Our work enables everyone to have free access to a range of synthetic data, which is produced according to the end-users preferences. The next section presents the data generator in detail. At first, we analyze collected data and afterwards we present the pre-processing steps we took to guarantee data consistency. Subsequently, we describe the different data generation methods we adopt and then, we briefly describe the web-based User Interface.

In our approach, apart from offering three different data generation methods to choose from, we also include energy supply and demand time series of configurable intervals, that describe sources other than EVs. This is crucial information to consider, if we take into account that the Smart Grid is a MAS setting with different types of stakeholders, each pursuing different goals. For example, if the goal is to maximize the utilization of renewable sources, the levels of specific types of production must be examined, in parallel with the aggregate demand, which does not result by EVs alone; or to balance the aggregate demand and supply, G2V/V2G schemes must analyze and exploit the flexibility of other consuming tasks as well. Moreover, the models are fitted automatically, asking the user only for the minimum configuration input (e.g., the time horizon, the population of EVs, etc.), nevertheless allowing for more customizations if preferable, such as changing the input dataset used for training, or selecting a different data generation method. This is important because, as we argued earlier, it is often very difficult to obtain and analyze real world data from different sites and different users, in order to calculate the input required by other approaches. Furthermore, it is offered as a freely available service to be used by the community.

Chapter 3

Our Approach on Synthetic Data Generation

In this chapter we present our approach on synthetic data generation. At first we give an overview of workflow components in our system and we decribe must the required configuration parameters to be set by the user in order to generate data. Furthermore, we present the input and output data of our system, as well as the preprocessing to which the first ones must be submitted in order to be properly entered into the dataset generator. Next, we describe in detail the synthetic data generation techniques we apply, and then, we briefly present the web-based user interface of our system. Finally we present the limitations we encountered and the assumptions we were called upon to make in order to create a tool that produces synthetic data, which are close to the originals and at the same time are as realistic as possible.

3.1 Method Overview

This section provides a description of the input data, and an overview of our approach. To begin, the original data collected, and consequently the resulting generated data, from our approach, is divided into two main categories:

- Energy Production and Consumption Data
- EVs' and Drivers' Behavior Data

The first category includes data related to the total consumption of both renewable and non-renewable energy for a given region's infrastructure and the equivalent energy production. The second category consists of data related to EVs and their characteristics, as well as the specifications of the available EV chargers. Also, the input dataset consists of data related to drivers' driving behavior, in terms of the number and time of day they choose to charge, as well as the trips they make. At this point it is worth mentioning that since it is difficult to find consistent charging and trip events, one has to enforce particular constraints, in order to adhere to realistic situations, as we detail in subsection 3.2.2.

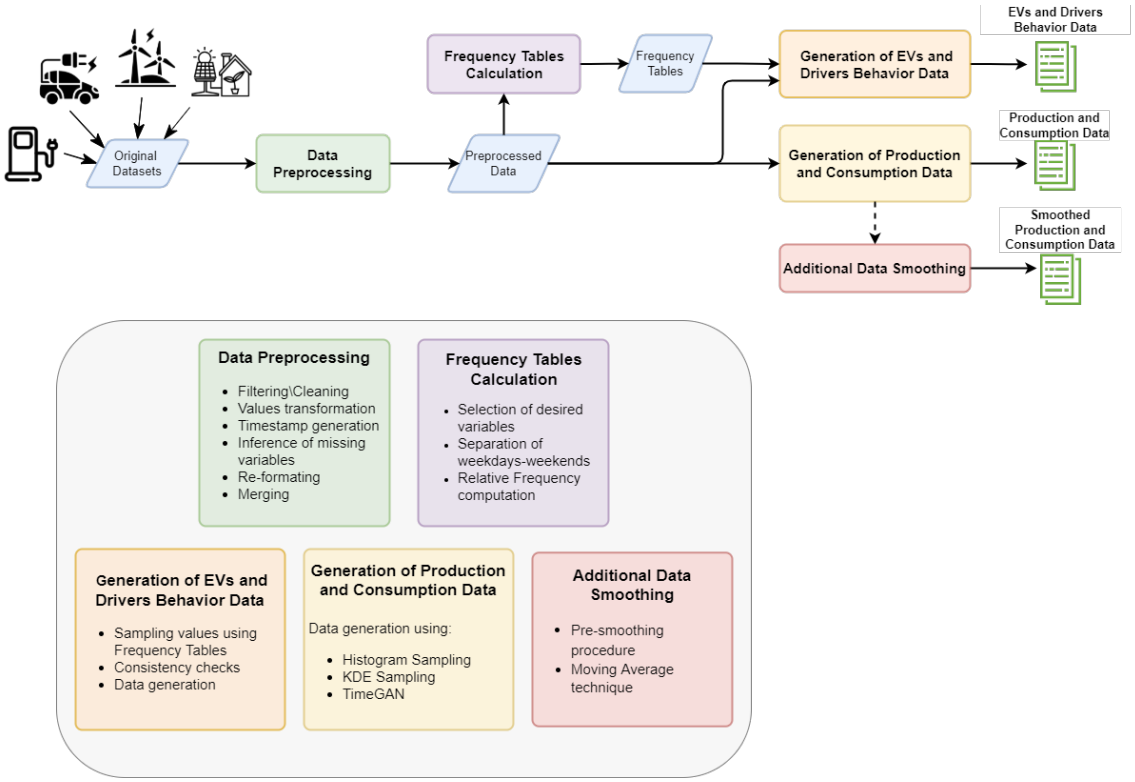


Figure 3.1: Schematic overview of workflow components in our system

Figure 3.1 shows the workflow and the components of our system, starting with the original input data, and resulting to the final output data, i.e., the generated dataset. First, data is preprocessed to align values, generate compatible timestamps, and perform the required transformations and merging (see, Section. 3.2). Then the possibility of additional smoothing is given, which is described in detail in subsection 3.3.5 For the case of energy production and consumption data, the preprocessed dataset is fed directly

to the generation methods of Sec. 3.3 and we obtain the results in the output. For the case of EVs' and drivers' behavior data, an additional step is required, that is to compute the frequency tables of specific variables of interest (e.g., travel speed, time of connection, etc.). Finally, the calculated frequencies, along with specific constraint checks, are used to generate the corresponding dataset. Detailed information about the generation methods used, is provided in Section 3.3. Furthermore, our dataset generator is available for download via a github repository.[1]

3.1.1 User Configurations of the Dataset Generation

To generate data, the user can either *(a)* use a web-based GUI that helps with configuring the generator more easily, (described in Section. 3.4); or *(b)* by using generation script templates and setting values to a few required configuration parameters via JSON files and the command line.

For the latter, the user must specify: the number of EVs; the length of the time horizon; the desired data generation technique among the available ones; the categories of EVs to be included in the generated dataset; the types of EV chargers to be considered.

For the first way, after filling in the appropriate variables, such as:

- The number of EVs
- The length of the time horizon
- The desired data generation technique among the available ones
- The categories of EVs to be included in the generated dataset
- The types of EV chargers to be considered
- The additional smoothing parameters, if desired

This information contains all required configurations for initiating the generation process. When everything is set, the user can begin the data generation process by executing the "MyMain.py" script. The results, i.e. the generated dataset in the form of '.csv' files, and the summarizing figures, are provided in the corresponding folders.

[1] the repository will be available soon

Furthermore, any number of original datasets—different from the default, public one that we include—can also be incorporated for the training of the generation models. This is particularly helpful for organizations that need to protect private data, but are still in need to share anonymized information with third parties. In what follows, we give a detailed description of the input and output data, as well as of the various constraints we impose in order to ensure the soundness of the produced results.

3.2 Data Acquisition and Preprocessing

Most of the data related to EVs and Smart Grid is collected by utilitiy or multinational companies. However, these companies have a legal obligation not to share it publicly. Nevertheless, as we have already mentioned, the availability of such data is necessary for further research in the domain. There are several ways that in combination with the tool we present, could make realistic data available to the public.

For example, companies could anonymize a small portion of their customers data, perhaps a few hundred, to make anonymous data publicly available, and then anybody could use the data generator to produce data representing thousands of customers. Additionally, in case that the company does not want to publish real data, even if it is anonymous, it could publish a representative sample of them. For example, a company holds data for thousands of consumers. It could employ the K-means algorithm [44] in order to group the data into clusters and then provide any researchers requestings data with the centers of the clusters, rather than the actual measurements. In addition a researcher could work in a way similar to ours. That is, to collect various data from different sources, and then combine it through the data generator in order to produce new data.

This section describes our datasets, and the functionality of the 'Data PreProcessing' module, where all data is pre-processed. The original data comes from different sources and has different formats. Therefore, we need to apply specific transformations, and then combine it properly in order to end up with the final input files. Table 3.1 provides links to the original data sources as well as a brief description of the data sets, while Table 3.2 provides an overview of the form of input and output data (see Figure 3.1).

Table 3.1: Original datasets used

Company/Platform	Dataset Description
My Electric Avenue https://eatechnology.com/resources/projects/my-electric-avenue/	EV driving behavior and charging data from up to 200 EVs, monitored on low voltage distribution networks
ENTSOE transparency platform https://transparency.entsoe.eu/dashboard/show	Measurements concerning the Actual Generation per Production Type and the Actual Total Load for all European regions for the recent years
Mendeley Data platform https://data.mendeley.com/datasets/tb9yrptydn/2	Electric passenger cars with their specifications
Tesla https://www.tesla.com/en_EU/support/european-union-energy-label	Technical features of Tesla models
Spirit Energy https://www.spiritenergy.co.uk/kb-ev-understanding-electric-car-charging	Specifications of EV's chargers

Table 3.2: Description of input and output data

Data Category	*Input of 'Data Preprocessing' Component*	*Output of Dataset Generator*
Production and Consumption Data	**Production**: DayTimeOfMeasurement (start-end), Lignite, FossilGas, Solar, Wind **Consumption**: DayTimeOfMeasurement (start-end), ActualTotalLoad *irrelevant or/and incomplete elements are omitted (e.g. Area, FossilOil etc.)	Year, MonthOfYear, DayOfWeek, TimeOfDay, LigniteGen, FossilGasGen, SolarGen,WindGen, TotalLoad, ImportExport
EVs and Drivers Behavior Data	**EVs specs**: CarModel, BatteryCapacity, MaximumDCChargingPower, MeanEnergyConsumption,Category **Chargers specs**: ChargerType, ChargerRating, ACDC, RatedPower **Drivers Behavior-Charging events**: ParticipantID BatteryChargeStartDate, BatteryChargeStopDate, StartingSoC, EndingSoC **Drivers Behavior-Trip events**: ParticipantID, TripStartDateTime, TripStopDateTime, TripDistance, PowerConsumption *irrelevant elements are omitted (e.g. Length, Wheelbase etc.)	Date, DriverID, Year, MonthOfYear, DayOfWeek, TimeOfDay, EVModel, BatteryCapacity, Category, MeanEnergyConsumption, MaxDCPower, Status,StartingSoC, EndingSoC, ChargerType, ChargerPower,ChargingTime, ChargingConsumption, TripTime, TripDistance, TripConsumption

3.2.1 Energy Production and Consumption Data

In this category, we have time-series data following the structure and format of the ENTSOE transparency platform (see Table 3.1). This includes real measurements concerning the Actual Generation per Production Type and the Actual Total Load for all European regions for the recent years. This information is available online, free of charge. Our main input consists of '.csv' files, having $n \in \mathbb{N}$ rows and $c \in \mathbb{N}$ columns. Each row corresponds to a series of measurements for a configurable time interval, e.g. an hour of the day, while each column contains information about, (a) the particular date and time, (b) the measurements of the actual generation per production type (e.g. Fossil Gas, Solar, Wind), and (c) the total load consumption. All measurements refer to kilowatt-hours

(kWh). The output file has a similar format (see Table 3.2). For example, given hourly measurements and a time horizon of 30 days, the number of rows would be $30 \cdot 24 = 720$, while the columns correspond to the particular date (i.e., year, month, day of week, time of day) and the different production/consumption measurements from each type. More information about energy production and consumption data is given in table A.1.

3.2.2 EVs and Drivers Behavior Data

As we see in Table 3.1, the data related to electric vehicles comes from many different sources and differs significantly from the previous case, with the main difference being that in this category we have a non time-series format. For the characteristics of electric vehicles, the input dataset comes mainly from the online, publicly available Mendeley Data platform, while this data can easily be augmented with additional information from other sources. For example, we also add some technical features of Tesla models available from the official website of Tesla in Europe, which were not included in the initial Mendeley dataset. All the above information is publicly available online. The combination of these two data sources provides information regarding EV technical characteristics, such as the equipped battery specification and the overall energy consumption of the EV.

A different input '.csv' file (see Table 3.3), produced during preprocessing, is used to dictate the specifications of the available chargers. This is based on information from the website of "Spirit Energy" (Table 3.1), and it contains the different types of chargers (AC or DC, Regular socket, Single phase, or 3 phases), their charging speed and their rated power. Finally, the data related to the drivers' driving behavior comes from the 'My Electric Avenue' project (see Tables 3.1 and 3.1). There, participants monitored the usage of over 200 EVs, a number of low voltage networks, and the switching of the EVs to support those networks. The trial participants' charging and driving behavior was recorded using the EVs' telematics systems. We utilize two files from that project: the first one contains information about the times each EV charged, as well as the state of charge for each time an EV is connected and charged; and the second one, contains distance, times, and power consumption information for each of the recorded drivers' trips.

3.2.3 Data Preprocessing

As already mentioned the original data has different formats, therefore, we need to apply specific transformations, and then combine it properly in order to end up with the final input files. In Table 3.3 we present the original and preprocessed data and we give a brief description of the preprocessing as well.

Table 3.3: Data files before and after preprocessing

Original Data	Source	Preprocessing	Preprocessed Data
EVChargeData.csv	My Electric Avenue (see Table 3.1)	Remove incomplete elements Normalize state of charge values (Starting_SoC & Ending_SoC) to the [0, 1] interval Add column SoC_Diff Add column ChargeTime	EVChargeData_PreProcessed.csv (see Table A.4 in the Appendix A)
EVTripData.csv	My Electric Avenue (see Table 3.1)	Remove incomplete elements Add column TripTime Add column Velocity	EVTripData_PreProcessed.csv (see Table A.5 in the Appendix A)
Actual_Generation_2017.csv Actual_Generation_2018.csv Actual_Generation_2019.csv Actual_Generation_2020.csv Total_load_2017.csv Total_Load_2018.csv Total_Load_2019.csv Total_Load_2020.csv	ENTSOE transparency platform (see Table 3.1)	Remove irrelevant or/and incomplete elements Transform the information about the day and time to a time series format (e.g. Date, Year, MonthOfYear, DayOfWeek, TimeOfDay) Merge Actual_Generation and Total_Load files according to year & add column ImportExport (Creation of Entsoe_2017.csv, Entsoe_2018.csv, Entsoe_2019.csv, Entsoe_2020.csv) Merge all files	Entsoe_PreProcessed.csv (see Table A.1 in the Appendix A)
FEV-data-Excel.xls	Mendeley data platform (see Table 3.1)	Remove irrelevant elements (e.g. Length, Wheelbase etc.) Fill in missing information about technical features of some Tesla models (source Tesla company, see Table 3.1) Add column Category	EV_specs_PreProcessed.csv (see Table A.2 in the Appendix A)
	Spirit Energy (see Table 3.1)	Create the .csv file with chargers specifications	Charging_Specs.csv (see Table A.3 in the Appendix A)

As a first step of the generation process, we preprocess the acquired data files and assign them appropriate timestamps. That is, we transform the information about the day and time in a format that is consistent across all generated files, i.e. to a time series format (e.g.*Date, Year, MonthOfYear, DayOfWeek, TimeOfDay*).

Next, we focus on EV characteristics. For this, we incorporate the data from the Mendeley platform in its original format, and fill in some missing information about the technical features of some EV models which is nevertheless available from other online sources, as described in the previous section. In addition, we include the new column 'Category', that signifies different EV types according to battery capacity. All

aforementioned data is combined into a single file, and another one holds the information of the EV chargers (see Table 3.3). Tables A.2 and A.3 in the Appendix A contain detailed information on the aforementioned files.

As for the data related to the charging events, the corresponding file (see Table 3.3) contains the exact dates and times that charging events start and end, as well as the EV battery status at the beginning and at the end of a charging event, noted as *Starting_SoC* and *Ending_SoC*, respectively. Again, here we need to assign compatible timestamps to each event, and also to normalize state of charge (SoC) values to the $[0, 1]$ interval. The normalization was based on the equation:

$$New\_value = \frac{x - min}{max - min} \tag{3.1}$$

where x is the old value, min and max are the minimum and maximum values of the original set, i.e. 0 and 12 respectively. Each row of the file represents a charging event/action, and each column includes data related to the charging events, i.e. the exact time of charging events, duration of charges, and the status of the battery. Finally, we create two new columns: *SoC_Diff*, which contains the difference '*Ending_SoC* - *Starting_SoC*', indicating what percentage of the battery was charged during a charging event; and *ChargingTime*, expressing the duration (in minutes) of each charge, as calculated from its *start* and *end* timestamps. Additional information about preprocessed charging data files can be found in Table A.4 in the Appendix A.

Regarding the data related to the trip events, the file (see Table 3.3) has the same format as that of the charging events: each row represents a trip event/action and each column, the exact time of a trip event, the duration of the trip, the trip distance and the power consumed during each trip respectively. By extension, we followed similar techniques in order to pre-process the data. So we converted the start and end dates of the trip events into the appropriate format and from them we calculated the duration of each event (in minutes), in a new column called *TripTime*. Lastly, we calculate the velocity of the vehicles through a simple time-distance relationship, assuming that the vehicle performs linear motion with constant speed. Table A.5 in the Appendix A, contains additional information about preprocessed trip data files.

In terms of the generated synthetic data output files (*Ev's and Drivers Behavior Data* in Table 3.2), their number is proportional to the number of drivers that the user has

specified as a configuration. For each one, the number of rows corresponds to measurements for a particular time interval, and each column represents the characteristics of each driver's EV, detailed information about charging events (i.e., exact time of charging event, duration of charge and status of the battery), and the characteristics of the charger with which a charging event takes place (i.e., AC or DC, single-phase or 3-phase, charger's charging speed and rated power). Finally, detailed information on trip events is provided (i.e., exact time of a trip event, the duration of a trip, the distance travelled and the power consumed during each trip). In addition, since the columns of the output files refer to both charging and trip events, in case we have a charging event, the columns related to the trip event are empty and vice versa. If neither event takes place, all aforementioned columns are left empty. Detailed information about drivers behavior output files can be found in Table A.7 in the Appendix A. In this table we give the example of a specific driver, in the 3 possible status modes. (idle mode, charge event, trip event).

The other data category, energy production and total consumption, originates from the ENTSOE platform and comes in the form of 8 '.csv' files, 4 for actual generation and 4 for total load, one for each year, from 2017 to 2020 (see Table 3.3). Our preprocessing is performed on a yearly basis. First we convert the timestamp to the appropriate format, i.e. in the following 4 columns: *Year*, *Month_of_Year*, *Day_of_Week* and *Time_of_Day*. We then combine the production and consumption data into a common data-frame and remove all incomplete rows (if any). Then, we insert a new column that holds the imbalance between the aggregates of consumption and production. If the resulting number is positive, then this indicates that energy is imported from neighbouring regions, otherwise, it is exported. Table A.1 in the Appendix A contains additional information about production and consumption preprocessed and output files.

3.3 Data Generation Techniques

In this section, we present our dataset generation process, which uses different statistical methods to produce synthetic data. There are two desired types of data to generate: 1) energy consumption and production, 2) EVs and drivers behavior data. For the former, we adopt three different generation methods that can be used interchangeably:

- Histogram Sampling
- KDE Sampling
- TimeGAN

At this point we should mention that regarding the Production and Consumption Data, the user has the ability to apply additional smoothing on them. More information on the data smoothing techniques we use is given in Subsection 3.3.5, while Section 4.1 provides a detailed presentation of the results obtained with and without additional smoothing.

Since the drivers' behavior data do not come in a time series format, the application of the aforementioned methods (*Histogram Sampling, KDE Sampling* and *TimeGAN*) is not meaningful. Therefore, we also put forward a *Frequency Tables Method*, which requires no such assumption for the input. Algorithm 1 gives the general picture of the operation of our system, while algorithms 2 and 3 describe the dataset generation process for the production and consumption data and the drivers' behavior data respectively. In what follows, we give detailed descriptions of each method.

Algorithm 1 Dataset Generation

1: Read the input *json* file with user's options
2: Create the output folder ▷ All generated files stored here.
3: Generate Energy Production & Consumption Data, Algorithm 2
4: Generate Drivers' Behavior Data, Algorithm 3

3.3.1 Histogram Sampling

In the Histogram Sampling case, we create a dataframe with size respective to the time period given by the user, and then fill in all the columns related to the date and time variables. Next, we define what measurements of the original data we need to consider for each column/variable. By analyzing and plotting the time-series of each variable for intervals of one week, we sought to find peaks, periodicity and irregularities within this time period. Therefore after visual inspection, we recorded how variables are related to the month, day and time, and so we group each column based on the time dependence it was shown to have by this analysis. For the total load consumption we take into account the month, day and hour, while for Solar and Fossil gas generation we keep the month and time of day; and finally, for the Lignite and Wind generation, for which no pattern was observed for the day and time, we take into account only the corresponding month of the year. Therefore, every time we want to generate new data we take into account the specific time dependencies of the variables, without having to repeat the above process.

Subsequently, having a series of measurements in each group, we create a histogram with a default number of bins. In general, we can construct a histogram with an arbitrary number. The more the bins, the more precise the sampling will be, but at the same time, most of the bins should contain a sufficient number of counts to be statistically significant. By conducting exploratory experiments, we converged to an empirically "optimal" number of bins such that they have an average population of at least 5 elements.

To generate values using a histogram, we first calculate the corresponding cumulative distribution, and then sample a random number from a uniform distribution between 0 and 1. This random sample indicates a point in the x-axis of the CDF, from which we find the respective y-axis value, which constitutes our final synthetic dataset value. In this way, for each row of data we sample a new value for the selected column. The process is repeated for every column of the preprocessed file (see Table 3.3).

In cases where the values that we sample refer to *Solar_Gen*, we perform an additional check as values are expected to be equal to zero during night time and low during cloudy days. However, mainly due to faulty measurements, there are rare cases where the original values are quite close to zero, but different than zero, thus forming non-zero probability mass for a positive value to come up, even during night hours. Thus, to avoid large inconsistencies with non-zero values during hours when it is not plausible (e.g.,

during night hours), in cases where the sample has a value less than a minimum level (as this is inferred by the original datasets), we treat it as zero. As we saw in practice, this intervention helps towards a better fit between the real data in the input and the generated data in the output, as shown in our experimental results in Chapter 4.

3.3.2 KDE Sampling

In this case we calculate the kernel density estimates of the original Production and Consumption data of the whole time horizon, using Python's machine learning library scikit-learn.

Therefore given a sample of n independent, identically distributed (i.i.d) observations $(x_1, x_2, \ldots, x_n)$ of a random variable from an unknown distribution at any given point x, the kernel density estimate is given by:

$$p(x) = \frac{1}{nh} \sum_{j=1}^{n} K\left(\frac{x - x_j}{h}\right) \tag{3.2}$$

where $K(x; h)$ is the kernel function and h is the bandwidth. In our approach we use Gaussian kernel:

$$K(x; h) \propto exp(-\frac{x^2}{2h^2}), \tag{3.3}$$

since it is one of the most widely used kernel functions. Having set a specific range in bandwidth, we tune the bandwidth parameter via cross-validation and select the parameter value that maximizes the log-likelihood of data. Therefore, similarly to Histogram Sampling, after selecting the corresponding subset of data $(x_1, x_2, \ldots x_n)$, where n is the the subset's length), we sample a new value for the selected column and the process is repeated for all columns. Also, as described earlier, in the case of *Solar_Gen* we perform the corresponding checks.

3.3.3 Time-series GANs

As we have already mentioned, Time-series Generative Adversarial Networks (TimeGANs) are GAN based models, that synthesize sequential data composed by 4 networks that play distinct roles in the data modelling process: the generator, the discriminator, and recovery and embedding models.[1] In terms of losses, three types are considered:

- *Reconstruction loss*, which refers to embeddings and the comparison between the original and reconstructed (generated) data.
- *Supervised loss*, which is responsible for the approximation of the time dimension in the latent space.
- *Unsupervised loss*, which reflects the relationship between generator and discrimination networks (min-max game)

Given the network architecture and the losses defined above, there are 3 training phases: first, we train the autoencoder on the provided sequential data to optimally reconstruct it; then, the supervisor is trained using real data sequencies to capture the historic temporal relationships; and finally, the combined training of the four networks, while minimizing all three loss functions.

Note that, in the case of TimeGAN, the time dimension of the input data matches that of the output. Thus, in this case, the input data must be further pre-processed. The preprocessing includes the following steps:

1) Normalization of time-series data to the range [0,1]. The normalization parameters are stored as a separate model to be used at the end of the synthesizing process to recover the generated data values.
2) Rolling windows creation - following the original paper recommendations [33], we create rolling windows with overlapping sequences of 24 data points.
3) Shuffling the observations in order for them to meet the independent and identically distributed (i.i.d.) property.

[1] To implement TimeGAN we used part of the code publicly available in the github repository ydata-synthetic, which develops various GANs using Tensorflow 2.0 [45].

At this point we must mention that due to the structure of the model, the generated data is the same size as the input data. Therefore, since the input data we use is data of 1 to 4 years, this method can be used to generate 1, 2, 3 or 4 years of data respectively. However, input data with a different time horizon can be used, as long as it has the same format as ours.

Algorithm 2 Generate Production & Consumption Data

```
Create dataframe df according user's time horizon
if method = HIST then                          ▷ The method is chosen by the user.
    Apply HIST method                          ▷ Histogram Sampling, see Section 3.3.1
else if method = KDE then
    Apply KDE method                           ▷ KDE Sampling, see Section 3.3.2
else if method = TimeGAN then
    Apply TimeGAN method                       ▷ TimeGAN, see Section 3.3.3
Save df to .csv file
if AdditionalSmoothing = True then             ▷ User set parameter.
    Take the smoothing parameters (rolling window) for each variable
    Apply smoothing on df and save smoothed data to new .csv file ▷ Data Smoothing, see Section 3.3.5
    Create figures and metrics for Smoothed Generated Data
Create figures and metrics for Generated Data
```

3.3.4 Frequency Tables Method

We now discuss the Frequency Tables method, used to generate the drivers' behavior data. Here, the data is generated based on Frequency tables and files including information regarding the EV characteristics. The Frequency tables are used to determine the probability of an event happening at a particular time instant, by counting the original dataset values. To construct them, we read the files related to the charging and trip events, and create 2 for each of the following 5 variables:

- Charge Start Hour: The hour that a charging event begins
- Daily Charges: The number of daily charges an EV makes
- Starting SoC: The EV's SoC when a charging event begins
- Ending SoC: The SoC when a charging event ends
- Trip Velocity: The average speed of the EV during a trip

More specifically, the two files we create for each of the above variables (see *Frequency Tables* in Figure 3.1) correspond to a relative frequency table for weekdays and another one for weekends. As described in detail in subsection 2.1.2.4, the relative frequency indicates the number of times a specific value (or range of values) has been observed, compared to the total number of measurements. This, each of the relative frequencies correspond to the weight of a specific value, which we use later during the sampling process. The larger the weight, the greater the probability the specific value will appear in our synthetic data.

We now proceed to the description of the main process for this part of our system. As outlined in Alg. 3, it is repeated for each EV, until the number of EVs/drivers chosen by the user is completed. First, we create a dataframe according to the time period given by the user and then we fill in all the columns related to the date and time variables, just as described earlier. Then we calculate the number of days that correspond to the time period entered by the user and for each one of them, we sample the number of daily charges from the corresponding frequency table. Then we sample charging events from the respective frequency tables, and fill in the cells of the dataframe. A charging event $chEv_i$, where $i = 1, ...N$, is the number of the total charging events for a specific EV, is represented by a triple $chEV_i = (h_i, stS_i, enS_i)$ in $\mathbb{R}^3$, where h_i denotes the *Charge Start Hour*, stS_i denotes the *Starting SoC* and enS_i denotes the *Ending SoC*. We must also point out that some data (DriverID and the specs of the EV) is sampled only once for each driver and remains unchanged throughout the file.

Algorithm 3 Generate Drivers' Behavior Data

for each EV **do** ▷ Number of EVs is chosen by the user
  Create dataframe *df* according user's time horizon
  Fill in the columns of *df* related to date and time
  Create an empty dataframe *df_s* ▷ To store variables that will be sampled only once (DriverID & EV specs)
  Calculate $numOfDays$ ▷ The number of days corresponds to user's time horizon
  Sample the num of daily charges for every $numOfDays$
  Create Sample Storage for weekdays, Algorithm 4 $Sampling\_Charge\_Data(0, numOfDays * 4, 0, df\_s, 0, 2, 0)$
  Store DriverID & EV specs to *df_s*
  Create Sample Storage for weekends: Algorithm 4 $Sampling\_Charge\_Data(0, numOfDays * 4, 0, df\_s, 0, 2, 1)$
  if $checksMode = Strict$ **then** ▷ choosing a checksMode
    $StrictConstraintsDataGeneration()$, Alg. 5 ▷ See Section 3.3.4
  else if $checksMode = Looser$ **then**
    Apply Looser Constraints mode ▷ See Section 3.3.4
  else if $checksMode = Min$ **then**
    Apply Absolutely Necessary Constraints mode ▷ See Section 3.3.4
  for the whole *df* **do**
    Find 2 consecutive charges
    if $Starting\_SoC of 2nd < Ending\_SoC of 1st$ **then**
      $CreateTripEvent()$, Algorithm 6
  Save *df* to *.csv* file
Merge all *.csv* files to 1 file
Create figures and metrics for Generated Data

Each time a new charging event is sampled, we check if the corresponding starting time is later than the end of the previous charging event's end. Also, the *Starting_SoC* should be less than the *Ending_SoC* of the preceding event's. However, in order to avoid constant data sampling from various Frequency Tables, every time we need data related to charging events, we create a storage of samples. Through a method (*Sampling_Charge_Data()*) we sample the maximum number of charging data that correspond to the number of

days we calculated earlier (i.e. $sampling\_data(n) = charging\_data\_for\_1\_charge \times 4charges\_a\_day \times Ddays$).

Algorithm 4 Sampling_Charge_Data()

procedure SAMPLING_CHARGE_DATA($check, samples$ $minHour, df\_s, checkSoC, preEndingSoC, isWeekend$) ▷
When we use this function to create the data storage, the input parameters are defined as follows: $(0, numOfDays * 4, 0, df\_s, 0, 2, 0/1)$
  if $isWeekend = 0$ **then**
    Read data from Weekdays Frequency Tables
  else
    Read data from Weekend Frequency Tables
  Sample the Charge Start Hour for *samples*
  for row in dataframe **do** ▷ means for each *sample*
    if checkSoC = 1 **then** ▷ when func used in alorithm 2
      Sample $Starting\_SoC < preEndingSoC$
    else ▷ when use func for data storage
      Sample Starting_SoC
    Sample $Ending\_SoC < Starting\_SoC$
  if check = 0 **then** ▷ when use func for data storage
    Sample EV specs with ▷ 1 sample
    (Model,BatteryCap,Category,EnergyCons,MaxDCPower)
  else ▷ When func used inside alorithm 2
    Use specs from df_s
  Sample charger specs (ChargerType,ChargerPower) for *samples*
  Calculate $ChargingHours$:
  $chargeRate \leftarrow ChargerPower$
  if $ChargerPower \geq 50$ **then**
    $chargeRate \leftarrow min(chargePower, MaxDCPower)$
  $batteryLoad \leftarrow Ending\_SoC - Starting\_SoC$
  $ChargingHours \leftarrow (batCapacity * batteryLoad)\ /chargeRate$

Algorithm 4 shows how *Sampling_Charge_Data()* works in case of the initial creation of the aforementioned storage of samples. Therefore, if the conditions are met, we randomly pick a sample from the storage, otherwise we try to pick a new sample that meets our requirements. If there is no such sample, we call the *Sampling_Charge_Data()* method but with different arguments. The method works as in the previous case with the difference that it returns only 1 new charging sample that meets our requirements.

With the respective configuration parameter, the user can choose among the different dataset generation algorithms. Now, since the generation process is based on random sampling, it is probable that the output might not adhere to real-world constraints in some cases, e.g. an EV discharging more that it would be possible between charging events, or a charge event appearing before the previous being completed. For this reason, to better control the level of constraints that will be enforced throughout the generation process, we provide three different modes:

- *Strict constraints*, where the time intervals that the EVs charge are finely dispersed throughout the day
- *Looser constraints*, where charges must be completed within the same day
- *Absolutely necessary constraints*, i.e. there are no time restrictions regarding when an EV begins charging.

In all these modes, we operate based on the number of daily charges. We now give a brief description of the *Strict constraints* mode (shown in Algorithm 5). Initially, if the number of daily charges is equal to 1 then no special action is required, we just fill in the dataframe, according to the sample we have chosen. If the number of daily charges is 2, then we divide the day into two 12-hours and carry out the 1st charge. If the end of the charge is within the first 12-hours, then we take a new sample and carry out the 2nd charge. In case the number of charges is equal to 3, we first divide the day into three 8-hour periods, and carry out the 1st charge. If the end of the charge is within the first 8-hour period, we take a new sample and perform the 2nd charge. Similarly, we check if the end of the current charge is within the second period and we perform the 3rd charge. However, in case the 1st charge is not completed within the first period, we check if it is completed within the second, so that then at least the 2nd charge can be performed. The same rationale applies for 4 or more daily charges, where we perform similar steps.

For the looser constraints, the only check that is performed before sampling the next charging event, is that the previous charge should have been completed before 23:00 of the same day. Finally in the absolutely necessary constraints, there is no extra rule, apart from that the *Starting_SoC* during a charging event must be lower than the *Ending_SoC* of the preceding one. In all 3 modes we sample different data for the weekdays and different for the weekends, by using the corresponding Frequency tables as input.

After filling the basic dataframe with the charging events, it is then rescanned in order to fill in the necessary trip events between the successive charging events (see Algorithm 6). The rationale behind this part is that if there is power consumption between two consecutive charges, a trip event must have intervened. The only variable that is sampled and is related to a trip event, is the velocity of the EV. The trip information is thus filled in by sampling the velocity and by calculating the remaining parameters related to EVs (i.e., *TripTime*, *TripConsumption* and *Ending_SoC*).

Algorithm 5 StrictConstraintsDataGeneration()

if $DailyCharges = 1$ **then**
    $Case\_Charges\_1$ ▷ fills our dataframe, based on the charging sample we have chosen.
else if $DailyCharges = 2$ **then**
    Divide the 24-hours into 2 12-hours.
    $Case\_Charges\_1$ ▷ Do 1st Charge
    if endOfCharge is in 1st 12-hour **then**
        Take *sample* from the storage. ▷ Start **STEP 1**
        if *sample* NOT meets our requirements **then**
            $Sampling\_Charge\_Data(1, 1, prevHour,$
            $df\_s, 1, prevEndSoC, 0/1)$
            $Case\_Charges\_1$ ▷ End **STEP 1**

```
12: else if DailyCharges = 3 then
13:     Divide the 24-hours into 3 8-hours.
14:     Case_Charges_1                                   ▷ Do 1st Charge
15:     if endOfCharge is in 1st 8-hour then
16:         Repeat STEP 1                                ▷ Do 2nd Charge
17:         if endOfCharge is in 2nd 8-hour then
18:             Repeat STEP 1                            ▷ Do 3rd Charge
19:     if endOfCharge is in 2nd 8-hour then
20:         Repeat STEP 1                                ▷ Do 2nd Charge
21: else if DailyCharges = 4 then
22:     Divide the 24-hours into 4 6-hours.
23:     Case_Charges_1                                   ▷ Do 1st Charge
24:     if endOfCharge is in 1st 6-hour then
25:         Repeat STEP 1                                ▷ Do 2nd Charge
26:         if endOfCharge is in 2nd 6-hour then
27:             Repeat STEP 1                            ▷ Do 3rd Charge
28:             if endOfCharge is in 3rd 6-hour then
29:                 Repeat STEP 1                        ▷ Do 4th Charge
30:         if endOfCharge is in 3rd 6-hour then
31:             Repeat STEP 1                            ▷ Do 3rd Charge
32:     if endOfCharge is in 2nd 6-hour then
33:         Repeat STEP 1                                ▷ Do 2nd Charge
34:         if endOfCharge is in 3rd 6-hour then
35:             Repeat STEP 1                            ▷ Do 3rd Charge
36:     if endOfCharge is in 3rd 6-hour then
37:         Repeat STEP 1                                ▷ Do 2nd Charge
```

Algorithm 6 CreateTripEvent()

Calculate $battCons$ ▷ Battery Consumption during trip event
Calculate $kmDriven$ ▷ Km driven during trip event
if $weekday$ **then**
    Sample $velocity$ from corresponding Frequency Table for weekdays
else
    Sample $velocity$ from corresponding Frequency Table for weekend
$tripTime \leftarrow kmDriven/velocity$
Calculate $availTime$ ▷ Time between 2 consecutive charges
if $tripTime < availTime$ **then**
    Calculate $reqVelocity$ ▷ The velocity required to complete the trip
    try
        if $weekday$ **then**
            Sample $velocity > reqVelocity$ from corresponding Frequency Table for weekdays
        else
            Sample $velocity > reqVelocity$ from corresponding Frequency Table for weekend
    catch ValueError
        $velocity \leftarrow reqVelocity$
    end try
For the time intervals of the trip event, fill in the columns of dataframe related to trip event, i.e. $Status$, $Starting\_Soc$, $Ending\_Soc$, $TripTime$, $TripDistance$, $TripConsumption$

3.3.5 Data Smoothing & Curve Matching

As mentioned earlier, regarding the Production and Consumption Data, the user has the ability to apply additional smoothing on them. The main smoothing method we use is the moving average technique.

The moving average, also known as rolling mean, is commonly used with time series to smooth random short-term variations and to highlight other components (trend, season, or cycle) present in the data. There are many types of moving averages, such as, the simple moving average (SMA), the cumulative moving average (CMA), and the exponential moving average (EMA). In our case we use the SMA. The simple moving average is the unweighted mean of M data points. The selection of M (sliding window) depends on the amount of smoothing desired since increasing the value of M improves the smoothing at the expense of accuracy. For a sequence of values, we calculate the simple moving average at time period t as follows:

$$SMA_t = \frac{x_t + x_{t-1} + x_{t-2} + ... + x_{M-(t-1)}}{M} \tag{3.4}$$

where x is the variable (e.g. Wind_Gen in our dataset) and t is the current time period (e.g. the time horizon of our dataset). We calculate the simple moving average by using the "pandas.Series.rolling" method. This method provides rolling windows over the data. On the resulting windows, we can perform calculations using a statistical function (in this case the mean). The size of the window (number of periods) is specified by the user, as described in section 3.4.

In addition, to produce data even closer to the originals, we perform a pre-smoothing procedure, which is described below. First we check the CSV file of the generated data (*Production and Consumption Data* in Figure 3.1 and Table A.1 in the Appendix A). For each value of a specific variable (e.g. Wind_Gen) we calculate the mean of the values for a time horizon of 24 hours, 12 hours before and 11 hours after the specific time interval. Then we randomly select a value from the original data set, which corresponds to the month, day and time we are in, and we calculate in the same way the corresponding mean. Afterwards, we compare the two means, having already calculated their difference. If the mean of the original data is greater than that of the generated ones, we add the difference to the value we have; otherwise we subtract it. The process is applied to the entire file, for each variable. Thus, by applying these two techniques-processes the time series curves of the generated data seem to be more in line with those of the original's.

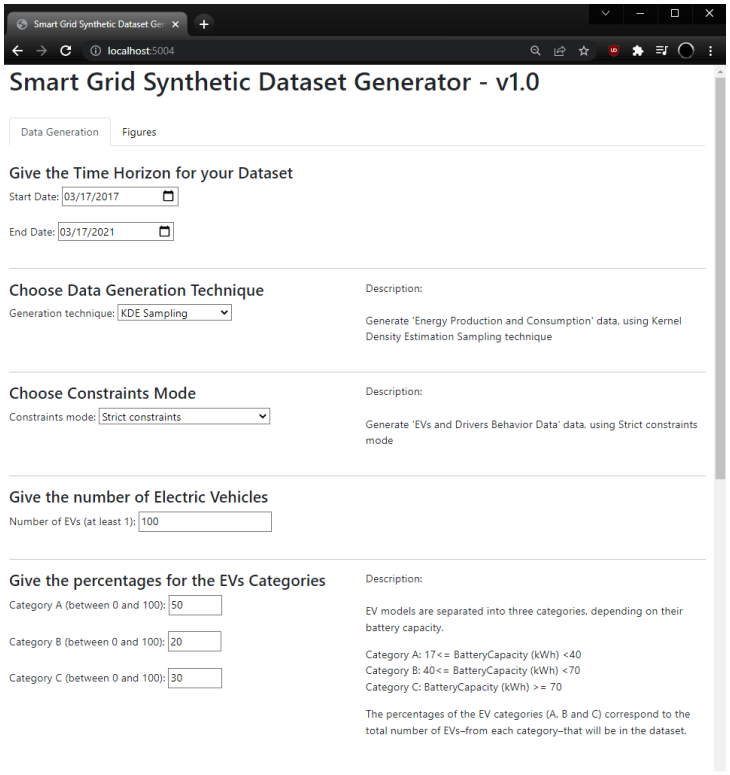
3.4 Web-Based User Interface

The previous sections presented in detail the functions of our system, as well as the techniques are used to produce synthetic data. In addition to the basic functions described, we have created a web-based user interface (UI), which helps the user to easily handle our Dataset Generator. The UI uses a backend web server, implemented in Flask[1]–a Python micro web framework– while the frontend part was implemented using mainly *HTML* and *JavaScript*. Essentially the UI runs locally and specifically on port 5004. So the user firstly runs the *Flask_App.py* file and then opens a new tab of a web browser, at 'localhost: 5004'. This is the URL of the UI, which can be used to perform all the procedures described in section 3.3.

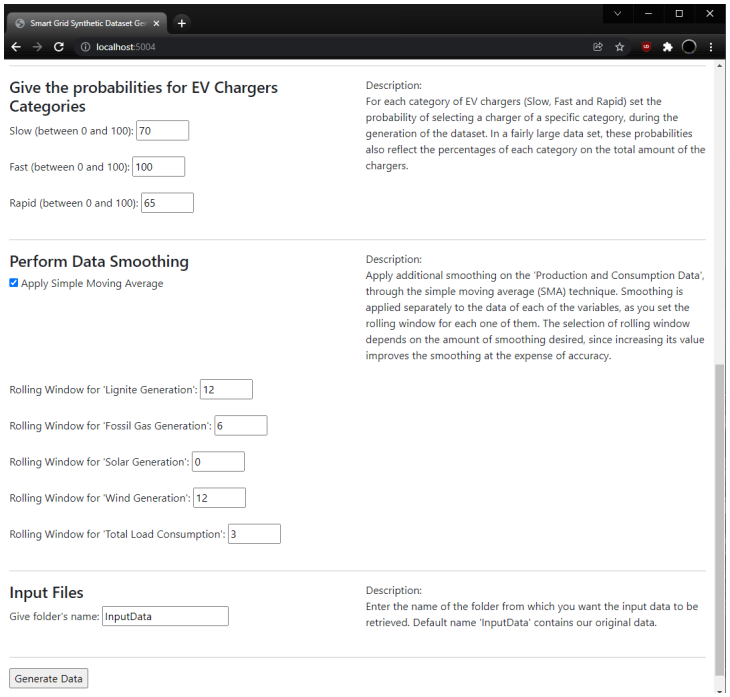
More specifically, the user first selects the time horizon for the data set (start and end date) and one of the 3 available techniques for the generation of Production and Consumption Data. He then enters the number of EVs, as well as the percentages of the EV categories (A, B and C), which correspond to the number of EVs–from each category– that will be in the dataset. Afterwards, the user, for each category of EV chargers (Slow, Fast and Rapid) sets the probability of selecting a charger of a specific category, during the generation of the dataset. In a fairly large data set, these probabilities also reflect the percentages of each category on the total amount of the chargers. After that the user can select to apply additional smoothing to the generated Production and Consumption Data, by setting the rolling window for each variable. Subsequently the user enters the name of the folder from which he wants the input data to be retrieved. The default name is 'InputData', and it contains our original data. Finally, the user selects the 'Generate Data' button and once all the necessary values have been filled in correctly, the generation of the synthetic data and the corresponding figures and statistics begins.

In addition to the above, the user can generate figures and statistics for data he has already created. On the second page of the UI, the user just have to select the folders from which the original and generated data will be retrieved and by pressing the corresponding submit button, the figures and statistics in which the selected data are compared are automatically created.

[1] https://flask.palletsprojects.com/en/2.0.x/



(a)



(b)

Figure 3.2: Screenshots of the web-based user interface

3.5 Assumptions and Limitations

Through the various methods we use, we managed to create a tool that produces synthetic data, which are close to the originals and at the same time are as realistic as possible. In our attempt to maintain this balance we have encountered various limitations and we had to make specific assumptions to deal with many of them. However, these limitations do not prevent us from generating realistic synthetic data sucessfully. The main limitations of our approach are described below:

- The calculated velocity values were too many and different from each other, so the resulting 'FrequencyTable_TripVelocity.csv' file was quite large and difficult to use. Therefore we rounded up and grouped them by 5 units. Thus without any significant deviation from the original data, the velocity values ended up being from 20 to 100 and rounded every 5 km/h.

- The frequency table of Starting_SoC includes all the possible values less than 1. This is because it does not make sense for someone to charge his EV if the battery is fully charged.

- 3rd category EVs are not allowed to use slow chargers, as the low rated power of chargers in combination with the large battery capacity of EVs results in unrealistic charging time (depending on the case, it can exceed 20 hours).

- In case the time interval between 2 consecutive charges is not enough to accomplish a trip event with some of the available velocity values, then the minimum speed required for the trip completion is calculated and is assigned as the EV's velocity in that trip. Therefore, in some cases the velocity of a trip event might exceed 100 km/h, which is the maximum velocity of a car in the original dataset.

- The start time of the generated datasets is defined as the time interval from 00:00 to 01:00 of the start date entered by the user. Respectively, the end time of the generated datasets is defined as the time interval from 23:00 to 23:59 of the previous date from the end date entered by the user. For example, if the user enters the time period '01/01/2021- 01/02/2021' the files he will receive will contain data that have been generated until 31/01/2021, 23:59. However, because there is a case for

a trip or charging event to start on 31/01/2021 and end on 01/02/2021, the last date included in the generated datasets is set at 01/02/2021, 23:59.

Chapter 4

Experimental Evaluation

In this chapter we present the results from our experimental evaluation. To evaluate the performance of our approach and illustrate its validity, we compare the distributions of the original data considered as input, to those of the synthetic resulting from the generation process. Intuitively, the difference should not to be too large, so as to imply different underlying models. To obtain a picture of this, first, we rely on visual inspections, where the distributions (in the case of energy production and consumption data), and the difference between relative frequencies-weights (in the case of EVs and drivers behavior data) are plotted against each other for each measurement type. This way, the user can detect large deviations and act accordingly to what each application setting dictates, i.e. either ignore irrelevant differences, or apply more constraints.

Our experimental evaluation spans on a maximum of 4 years time horizon, including 100 EVs. HIST refers to the histogram approach, KDE to the kernel density estimation, and TimeGAN to the generative adversarial network approach. For the case of production and consumption data, the fitting of the models and the generation of the synthetic datasets takes 7 minutes for the HIST method, while KDE requires 3 to 4 hours, and TimeGAN less than a minute. However, in the case of the TimeGAN method, whenever data generation is done for the first time and there is no training model, the production time–including model training–is around 6 hours. For the EV and Drivers Behavior dataset, the Frequency Tables method required 30 minutes. Experiments were performed on an Intel Core i7-5500U CPU 2.40GHz processor, with 8GB RAM. Subsequently, we describe in detail the experimental evaluation for each of the two data categories.

4.1 Experimental Evaluation - Energy Production and Consumption Data

To quantify a 'distance' between the distributions under comparison, we employ the Kolmogorov - Smirnov D statistic [46]. The Kolmogorov – Smirnov statistic quantifies a distance between the empirical distribution functions (CDF) of two samples, in our case between original and generated data. Consequently apart from testing statistical significance in difference, it can also be used for measuring the effect size of the generation process. The results are shown in Table 4.1. In terms of the K-S statistic, we observe that the overall performance of each method depends on the size of the output data. As is expected in probabilistic methods, a larger sample size leads to closer fits and better results in general.

Initially, in both cases, when we use one year and four year of input data respectively, the KDE and Histogram sampling methods have almost the same results, and both outperform TimeGAN by a relatively large margin. More specifically, in the cases of KDE and HIST, the value of K-S statistic for all columns is really small, meaning that the data generated is very close to the original, something that is also clearly reflected in the histograms.

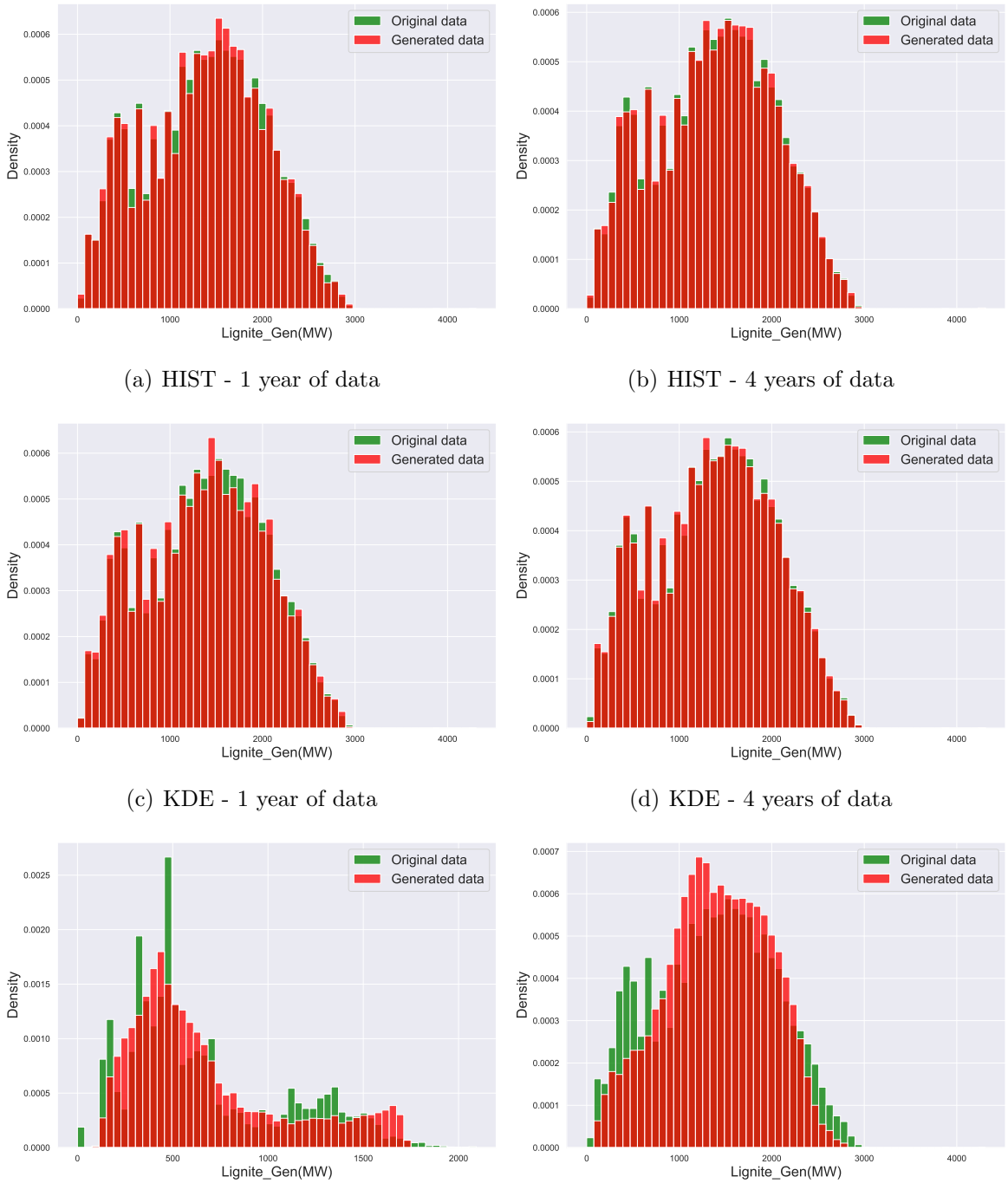
In case we use only one year of data as input, for the HIST and KDE, we could sample any number of measurements, so we use the original 4 year data, while we generate data via sampling only for 1 year. So, it is somewhat expected that with fewer samples we will achieve slightly worse distribution fit. As for the TimeGAN method, it seems that there is only a small effect due to the size of the original data, since increasing it does not improve performance significantly. So one-year data seems to be enough, in order to learn - as far as it can - the original relationships between variables across time.

The aforementioned observations are also reflected in the histograms presented in the figures 4.1 - 4.5. For example, in Figure 4.2 we observe that in all 3 cases, the results when producing 4-year data (Figures 4.2(b), 4.2(d) , 4.2(f)) are better, compared to when producing 1-year data (Figures 4.2(a), 4.2(c) , 4.2(e)). However in the case of HIST and KDE, the generated data is very close to the original, while the distributions of the data produced by the TimeGAN method are quite different from those of the originals. At this point it is worth mentioning in particular the case of the *Solar_Gen*. Fig. 4.3 and Table 4.1 make apparent the big difference in the behavior of the TimeGAN method

Table 4.1: K–S statistic of Production and Consumption Data. Best performance results indicated in bold.

Output Data time-horizon	*Method*	*Lignite_Gen*	*FossilGas_Gen*	*Solar_Gen*	*Wind_Gen*	*Total_Load*	*Average*
4 years of data	HIST	0.005	0.005	**0.03**	**0.004**	0.005	**0.01** (**1%**)
	KDE	**0.004**	**0.004**	0.036	**0.004**	**0.004**	**0.01** (**1%**)
	TimeGAN	0.08	0.038	0.459	0.06	0.035	0.134 (13.4%)
1 year of data	HIST	0.011	0.011	**0.031**	**0.009**	0.009	**0.014** (**1.4%**)
	KDE	**0.001**	**0.006**	0.038	0.01	**0.007**	**0.014** (**1.4%**)
	TimeGAN	0.052	0.072	0.464	0.068	0.044	0.14 (14%)

compared to the other two methods, in terms of solar generation. *Solar_Gen*, produced by TimeGAN, has the largest discrepancy between original and generated data, both compared to the other two methods and to the other columns of the same method. This is probably due to the difficulty of simulating solar activity, an important issue which we partially solved with our intervention in the KDE and HIST methods, as it is described in subsection 3.3.1.



(a) HIST - 1 year of data

(b) HIST - 4 years of data

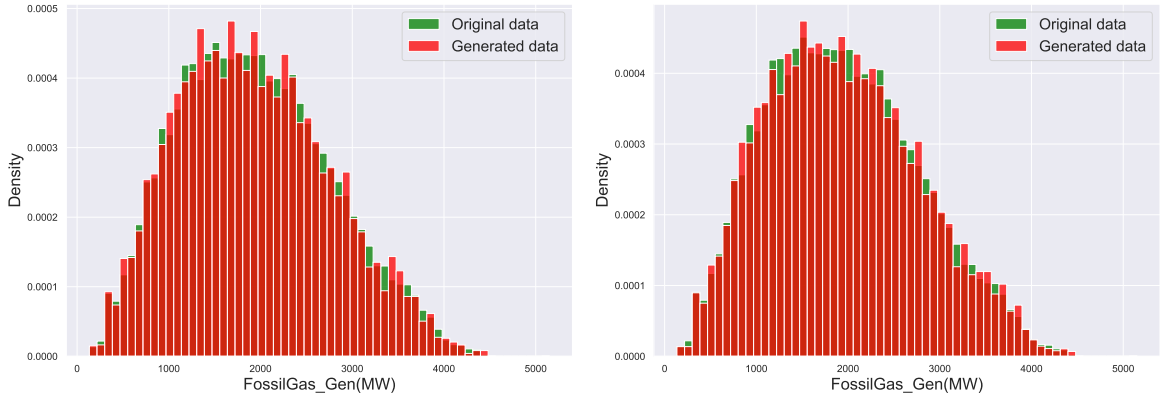
(c) KDE - 1 year of data

(d) KDE - 4 years of data

(e) TimeGAN - 1 year of data

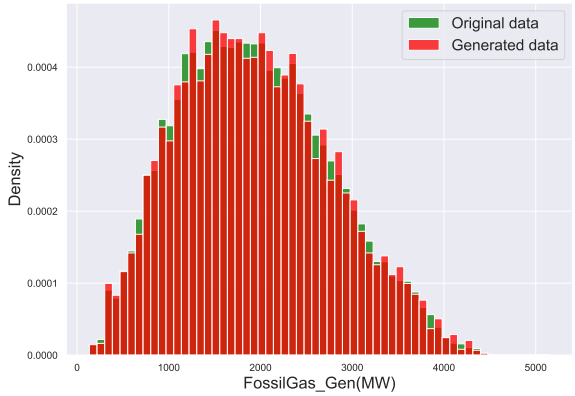
(f) TimeGAN - 4 years of data

Figure 4.1: Histograms of column *Lignite_Gen*

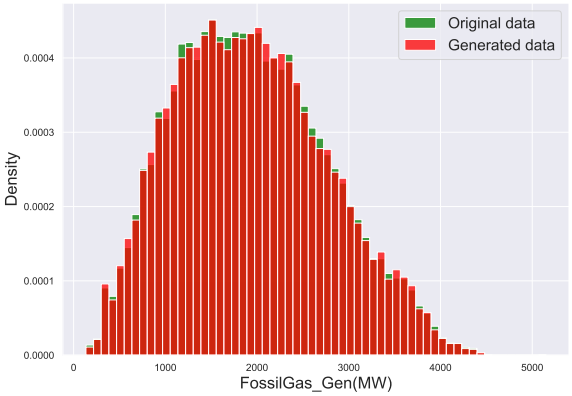


(a) HIST - 1 year of data

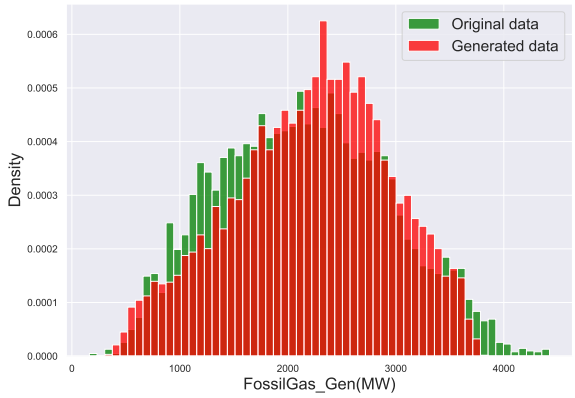
(b) HIST - 4 years of data



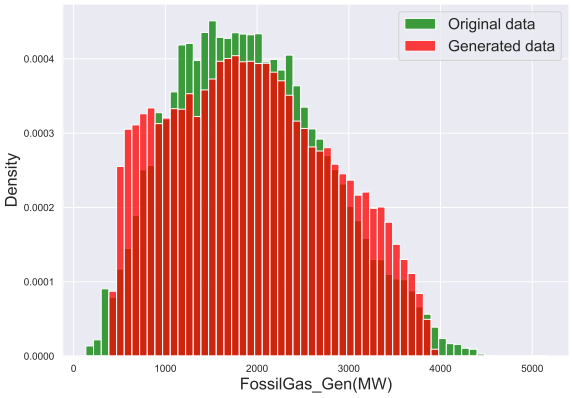
(c) KDE - 1 year of data



(d) KDE - 4 years of data

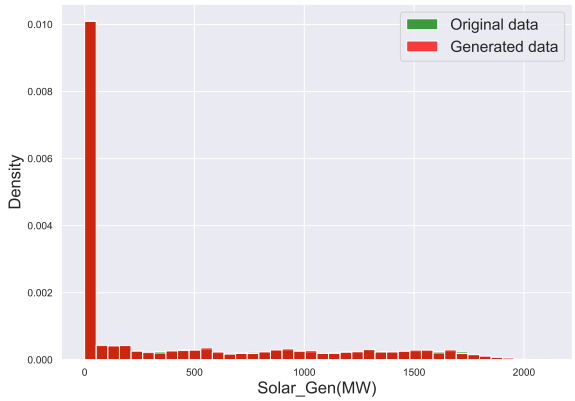


(e) TimeGAN - 1 year of data

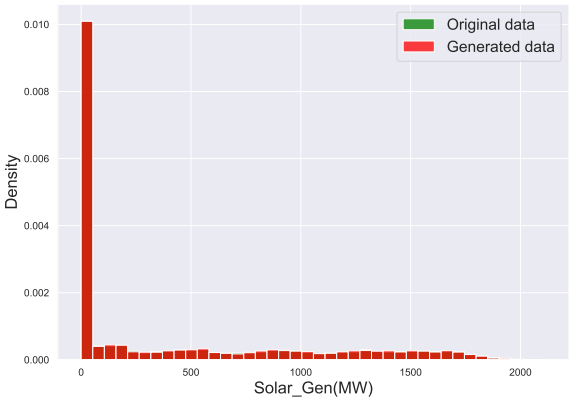


(f) TimeGAN - 4 years of data

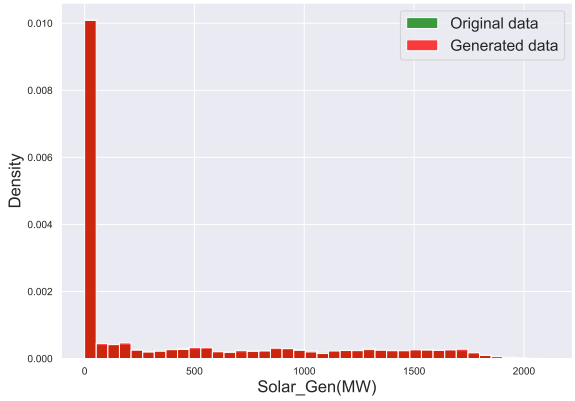
Figure 4.2: Histograms of column *FossilGas_Gen*



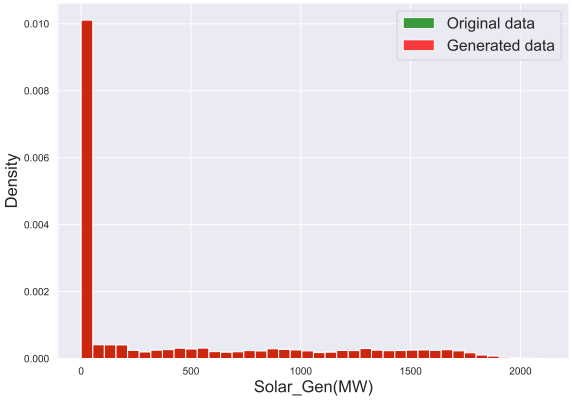
(a) HIST - 1 year of data



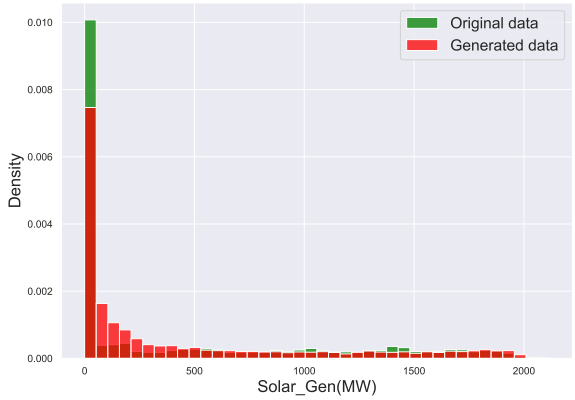
(b) HIST - 4 years of data



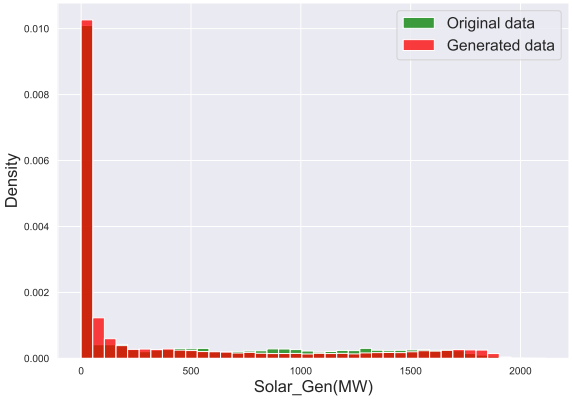
(c) KDE - 1 year of data



(d) KDE - 4 years of data



(e) TimeGAN - 1 year of data

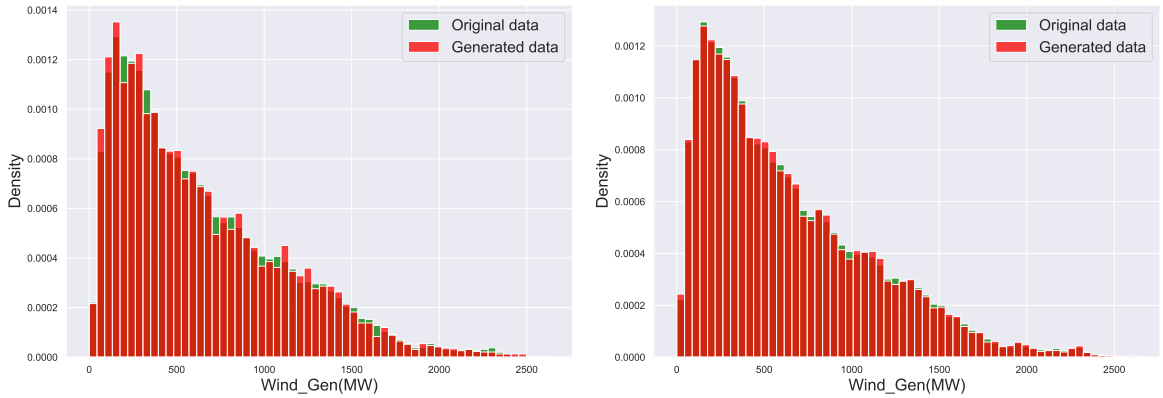


(f) TimeGAN - 4 years of data

Figure 4.3: Histograms of column *Solar_Gen*

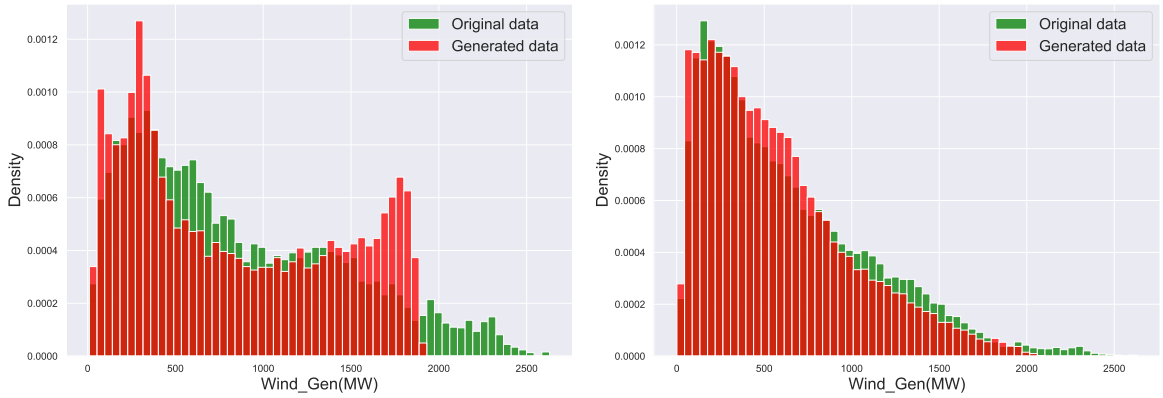
(a) HIST - 1 year of data

(b) HIST - 4 years of data



(c) KDE - 1 year of data

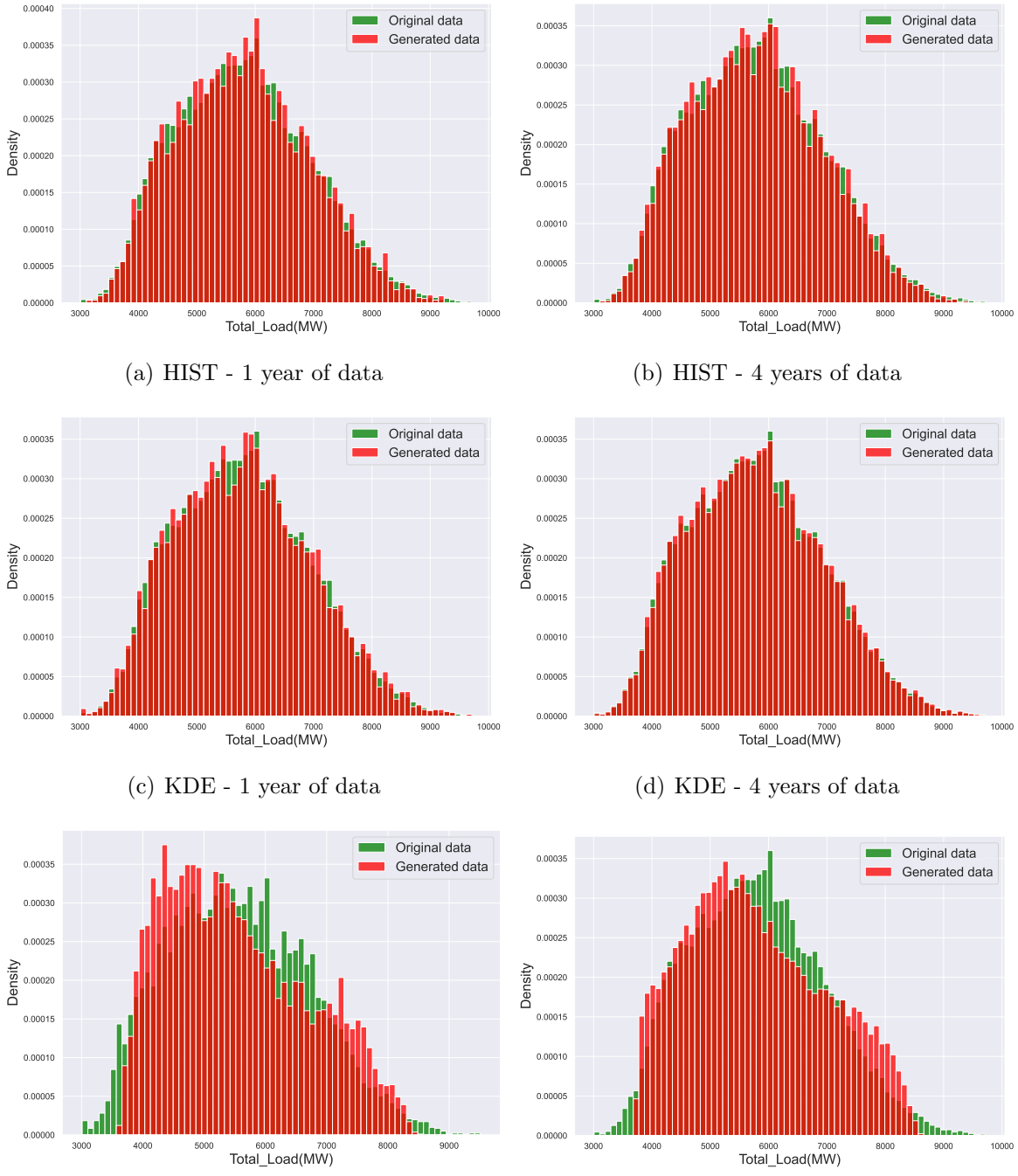
(d) KDE - 4 years of data



(e) TimeGAN - 1 year of data

(f) TimeGAN - 4 years of data

Figure 4.4: Histograms of column *Wind_Gen*



(a) HIST - 1 year of data

(b) HIST - 4 years of data

(c) KDE - 1 year of data

(d) KDE - 4 years of data

(e) TimeGAN - 1 year of data

(f) TimeGAN - 4 years of data

Figure 4.5: Histograms of column *TotalLoad_Gen*

4.1.1 Time-Series Evaluation - Data Smoothing

As we explained earlier in Section 4.1, the distributions of the original and generated the data are very close, according to the histograms and statistical measurements we perform. In the cases of Histogram and KDE Sampling in particular, they are almost identical. However if we observe the time series of the variables for shorter intervals, e.g. for a time horizon of 1 week (Figure 4.6), in some cases we observe relatively large deviations. For this reason we give the user the ability to apply additional smoothing to the generated data (Section 3.3.5), so that the time series curves of the generated data becomes more in line with those of the original's.

In what follows, we present the time series of data generated by the Histogram Sampling technique. Afterwards, we compare the histograms and the time series of some of the most characteristic variables before and after the application of rolling average, for different values in the rolling window, as well as after the application of the overall smoothing method. We chose HIST method, as as we have already mentioned it is the most efficient (along with the KDE) and the fastest of our methods.

To begin, Figure 4.6 presents the time series of energy production and consumption data for each variable. More specifically in Figures 4.6(a) and 4.6(b), where a periodicity is observed, the curves of the generated data are very close to those of the original. Generated data also perform relatively well in the case of *Fossil_Gen* (see Figure 4.6(c)). In contrast, in the cases of *Wind* and *Lignite* generation (see Figure 4.6(d) and 4.6(e) respectively), which do not show any periodicity, the results are much worse.

From the histograms of the Fossil Gas Generation (Figures 4.7, 4.8), we observe that the larger the rolling window, the greater the discrepancy between the distributions of the original and the generated data[1]. In Figure 4.7(d), where we have a rolling window of 24 intervals, the data have been overly smoothed resulting in a large degree of distortion. In Figure 4.8(b) we have obviously better results, while in Figure 4.8(d), where the Pre-Smoothing technique has also been used, the results seem to be even better.

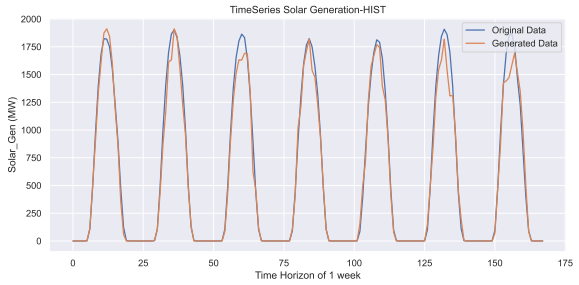
In Figure 4.9 we can see the time series for Wind Generation, which as we have mentioned does not follow a time pattern. Therefore the results after smoothing are obviously improved, since there are no more continuous and abrupt fluctuations in values, however

[1]Note that increasing the value of rolling window improves the smoothing at the expense of accuracy (see Subsection 3.3.5)

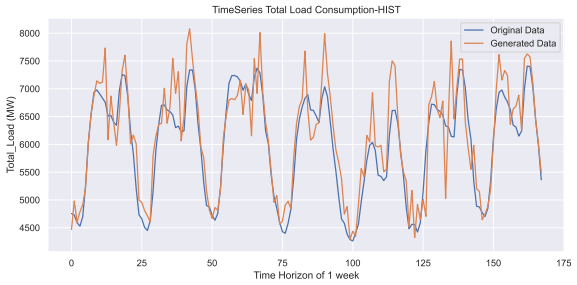
the curve of the generated data does not seem to follow the pattern of the originals. Total Load Consumption (Figure 4.10) and Solar Generation are the two variables that are best simulated, since as we have already mentioned, there is a periodicity. In Solar_Gen it is clearer, and for this reason we did not consider it necessary to do additional smoothing and present it in our results. As for Total Load, there is a partial improvement on the time series after the smoothing, however in the histograms we observe what we have analyzed before, that is, a greater deviation between the distributions.

Therefore we conclude that the choice of additional data smoothing has the corresponding price, which is the heterogeneity between the distributions of generated-original data. If the user wants to have a more general good picture in his data, that is, to be evenly distributed in relation to the originals, then he should avoid smoothing. On the contrary, if the user is not particularly interested in the uniformly distributed data, but wants to focus on more realistic time series, therefore the best hourly results, then additional smoothing will have a better performance.

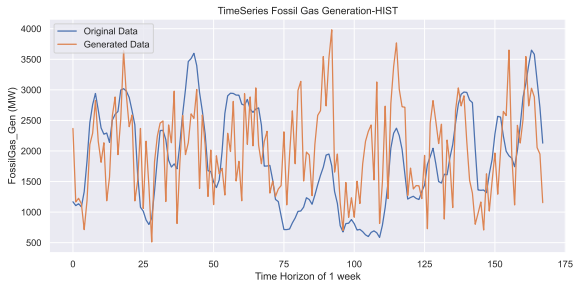
At this point it should be mentioned that for time series curves, different results may occur during their creation, as the time horizon of 1 week is randomly selected from the total time horizon of the data set. Therefore the values of variables such as Lignite and Wind, in which no periodicity is observed, may vary considerably between weeks, even between days.



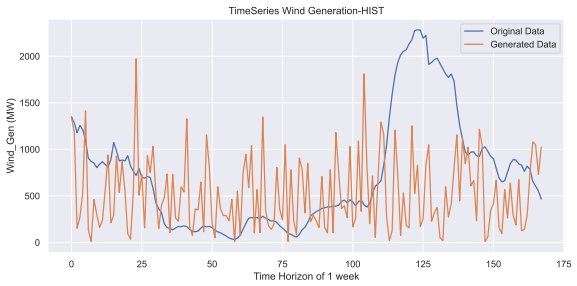
(a) Solar Generation



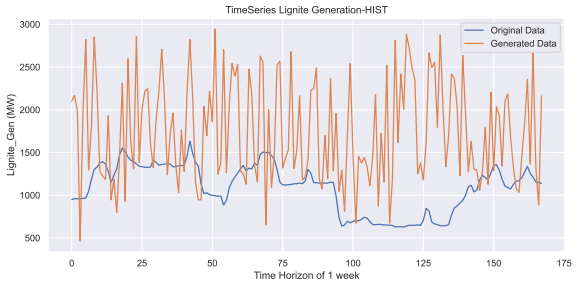
(b) Total Load Consumption



(c) Fossil Gas Generation



(d) Wind Generation



(e) Lignite Generation

Figure 4.6: Time-Series of Production and Consumption Data, HIST method

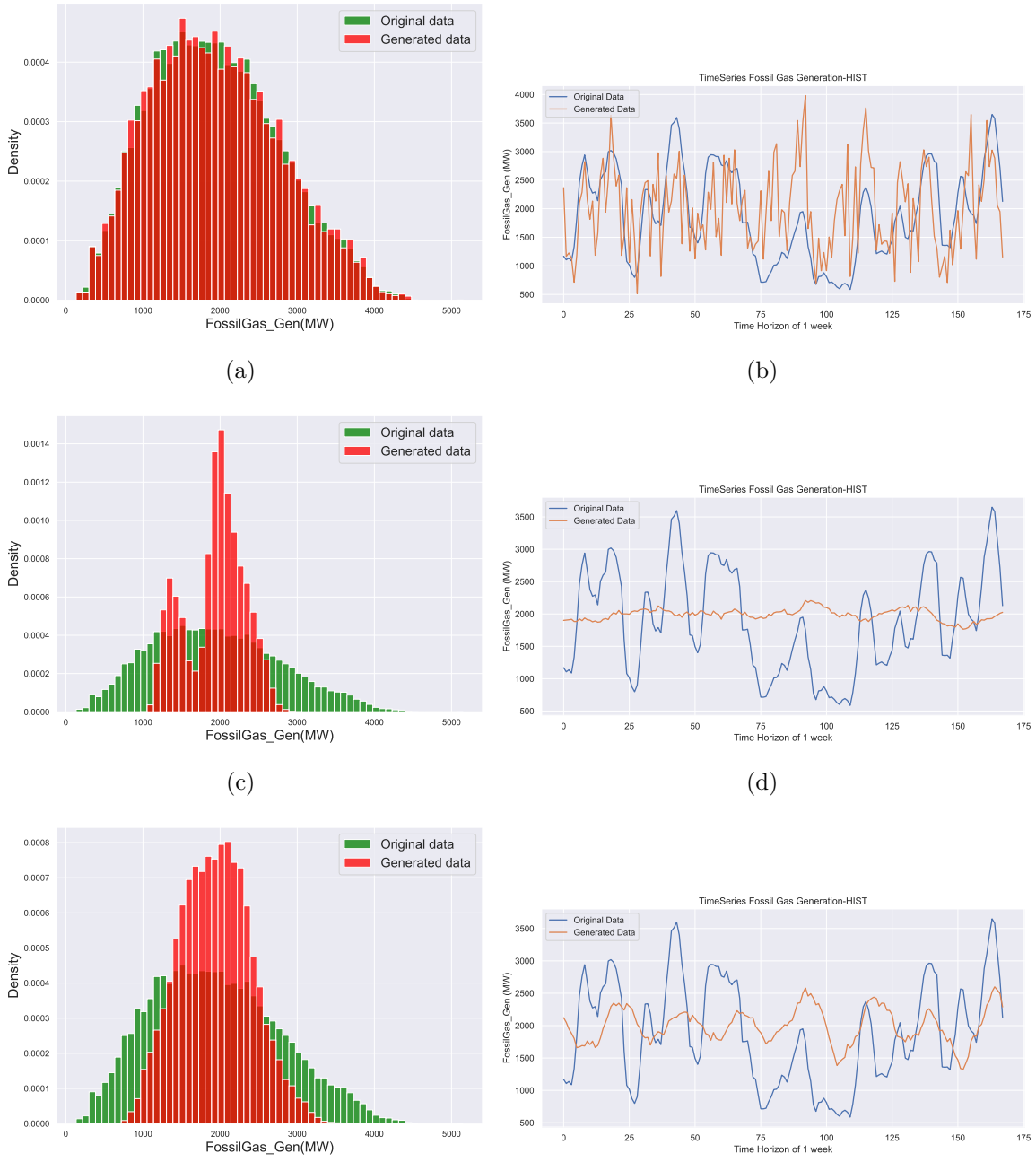


Figure 4.7: Fossil Gas Generation. (a)(b) Raw Data, (c)(d) Smoothed Data,rolling=24, (e)(f) Smoothed Data,rolling=12

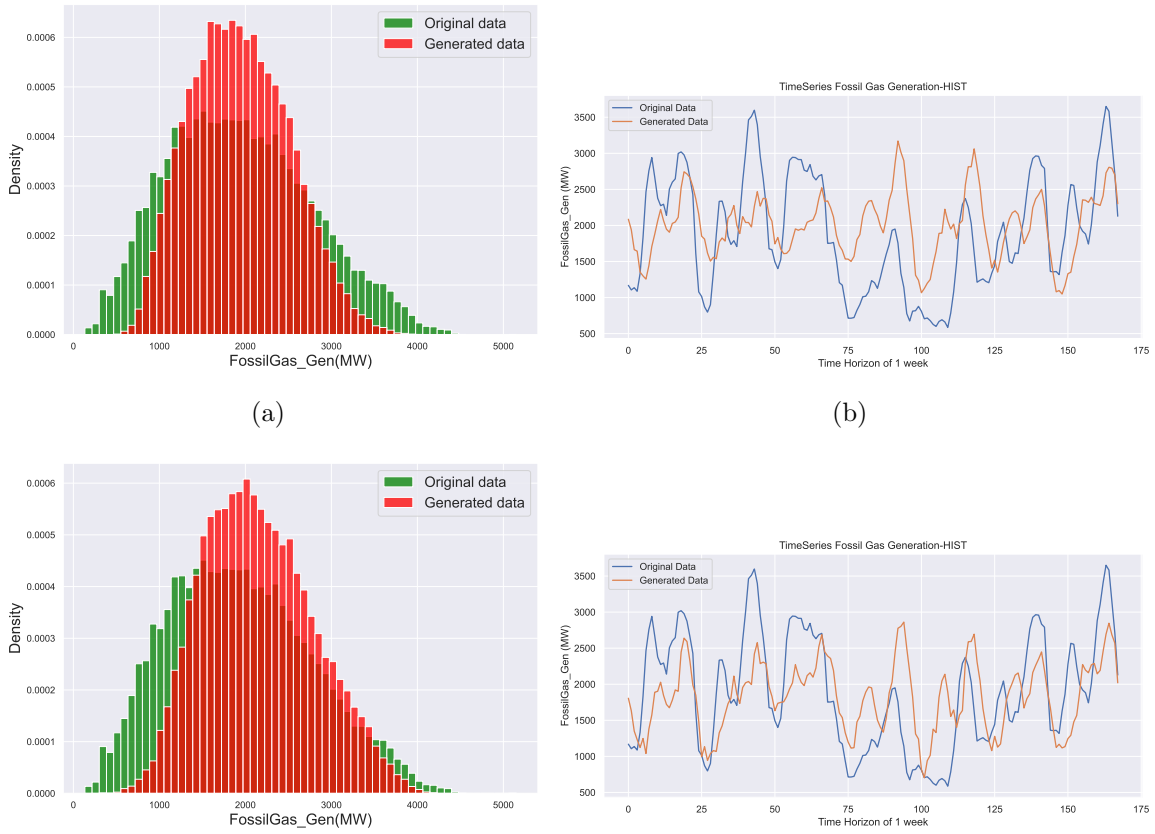
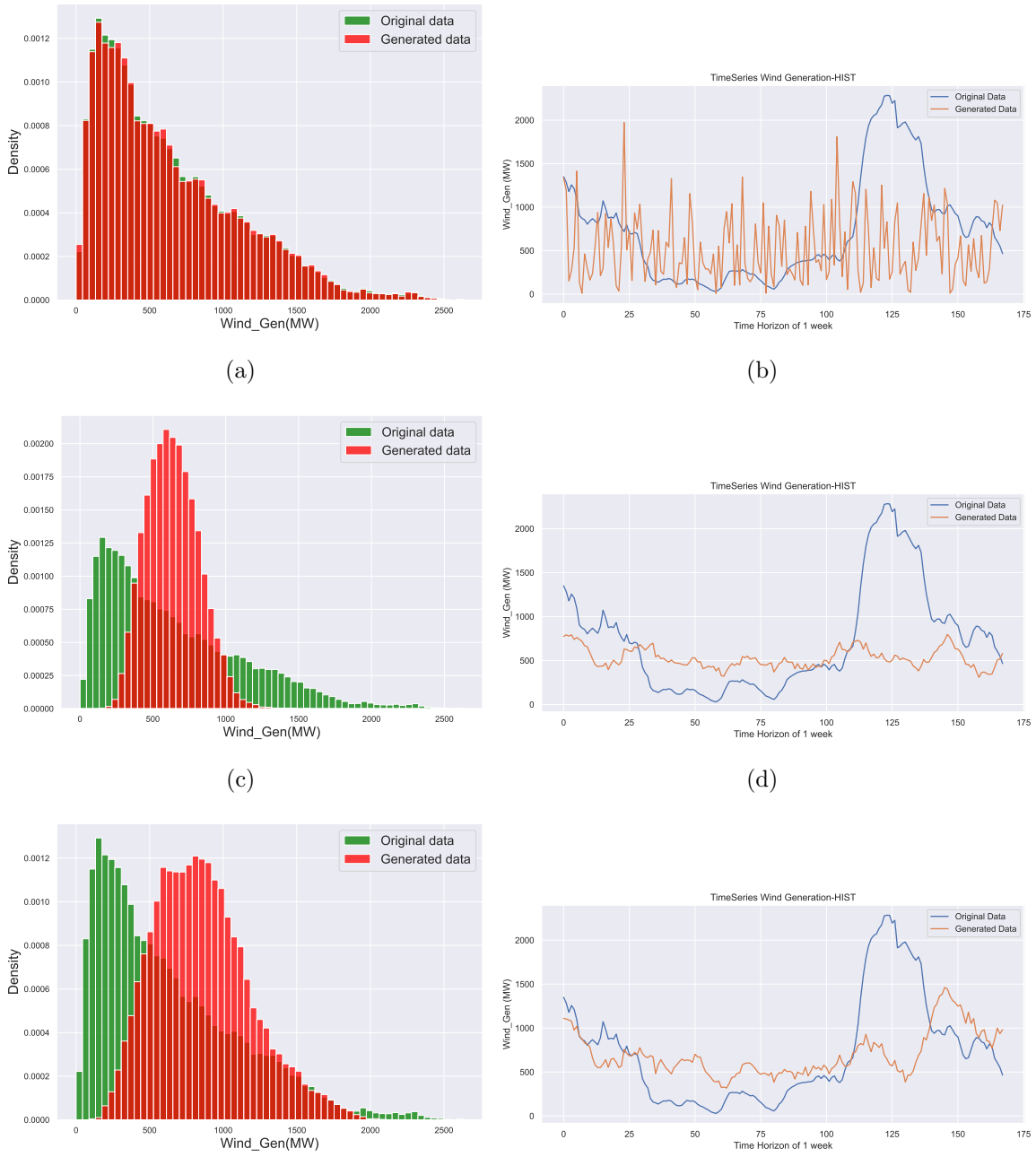


Figure 4.8: Fossil Gas Generation. (a)(b) Smoothed Data,rolling=6, (c)(d) Smoothed Data,rolling=6,Pre-Smoothing



(a) (b)

(c) (d)

(e) (f)

Figure 4.9: Wind Generation. (a)(b) Raw Data, (c)(d) Smoothed Data,rolling=12, (e)(f) Smoothed Data,rolling=12,Pre-Smoothing

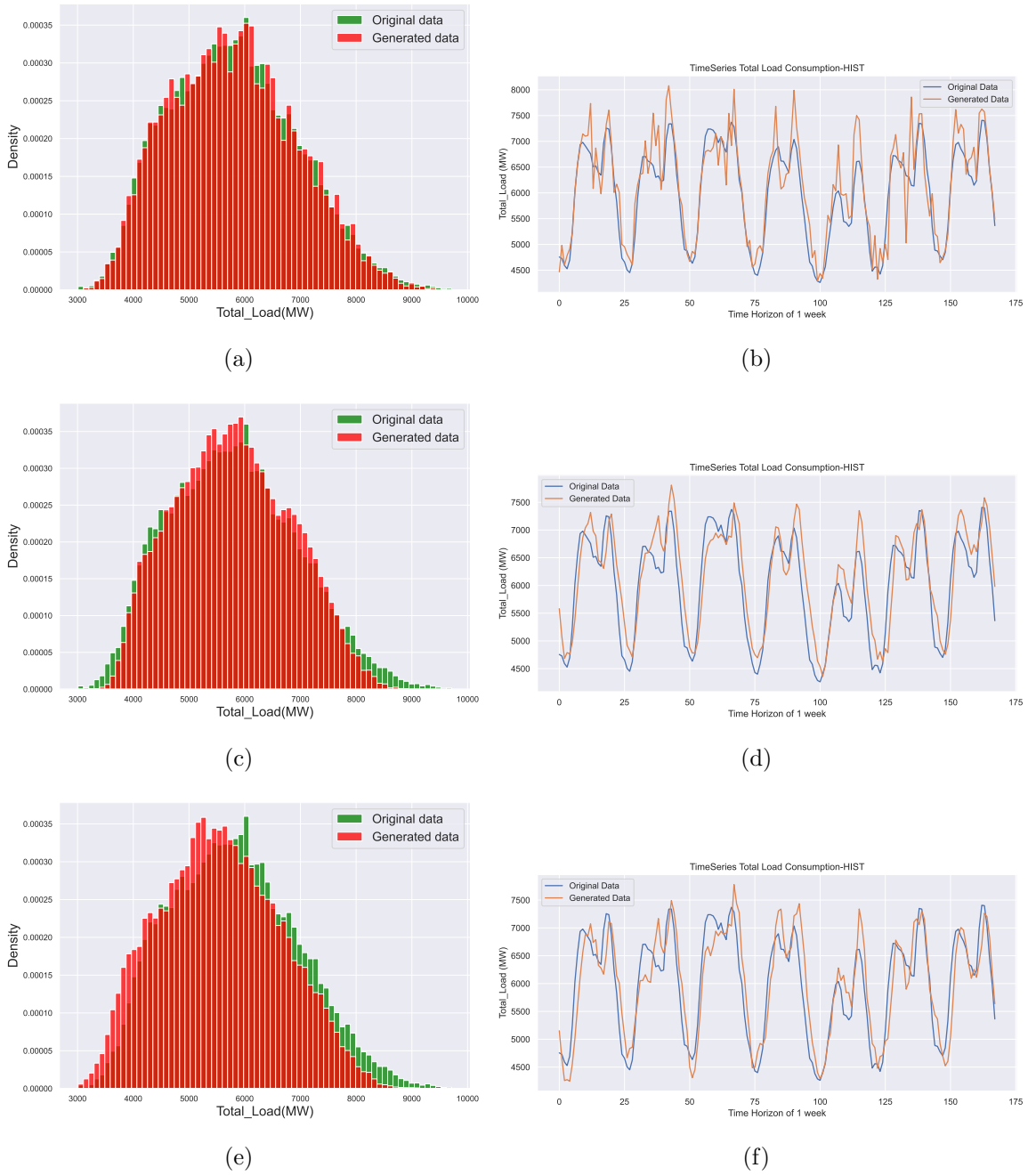


Figure 4.10: Total Load Consumption. (a)(b) Raw Data, (c)(d) Smoothed Data,rolling=3, (e)(f) Smoothed Data,rolling=3,Pre-Smoothing

4.2 Experimental Evaluation - EVs and Drivers Behavior Data

For the case of EVs and drivers behavior, where the generated data has undergone a major transformation and therefore differs greatly in their format, we compare in terms of the absolute difference between their relative frequencies-weights, depicted by bar-plots. The results are presented in Table 4.2 and they are calculated from experiments with 100 EVs in a time horizon of 1 year. *StrictChecks* refers to the Strict constraints mode, *LooserChecks* to the Looser constraints, and *MinChecks* to the absolutely necessary constraints mode.

As for the differences between the bar-plots of the original and generated data, at first glance the overall performance of *MinChecks* mode is better than the other two modes. However, by observing each variable/column individually, we realize that the results intepretation is a bit more intricate. First, the column *DailyCharges* has the largest difference between the original-generated data for the *StrictChecks* method and the smallest for the *MinChecks* method. This is also illustrated in Fig. 4.11: in the *StrictChecks* mode we have mainly 1 to 2 charges, in rare cases 3 and very rarely 4 charges, while for *LooserChecks* and *MinChecks* the number of total daily charges is distributed similarly to the original data.

Table 4.2: Difference of Drivers Behavior Data Frequency Tables. Best performance results indicated in bold.

Method	*DailyCharges*	*ChargeStartHour*	*Starting_SoC*	*Ending_SoC*	*DailyTrips*	*TripStartHour*	*Average*
StrictChecks	0.419	**0.057**	0.079	0.231	0.821	**0.345**	0.325
LooserChecks	0.095	0.232	0.077	0.227	0.619	0.465	0.286
MinChecks	**0.072**	0.173	**0.067**	**0.221**	**0.057**	0.453	**0.26**

The specific results are meaningful, considering that in the *StrictChecks* mode, we first visit a number of constraints regarding the number of charges. As mentioned in Section 3.3.4, in order for the next charge to take place, the previous one must have been completed at a certain time point during the day. Conversely, in the *LooserChecks* mode, the only check that is done before attempting the next charge is that the previous one has to be completed before 23:00 on the same day, while in *MinChecks* no check is

performed. Consequently, the more constraints we apply *wrt* charges–i.e., the narrower the time period during which a charge can be made–the fewer charges will be performed daily. At the same time, however, the strict constraints of the first method give us better control over the variables *ChargeStartHour* and *TripStartHour* and therefore optimize the results for the respective columns, while all three modes have similar performance with respect to the *StartingSoC* and the *EndingSoC* variables.

(a) Daily Charges

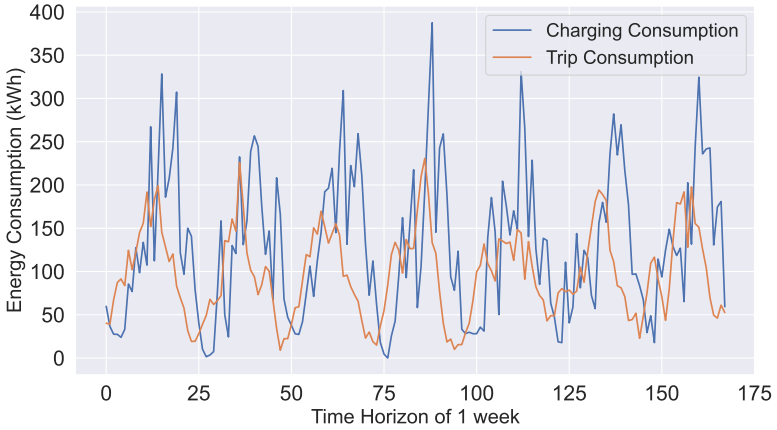
(b) Charging Start Hour

(c) Starting SoC

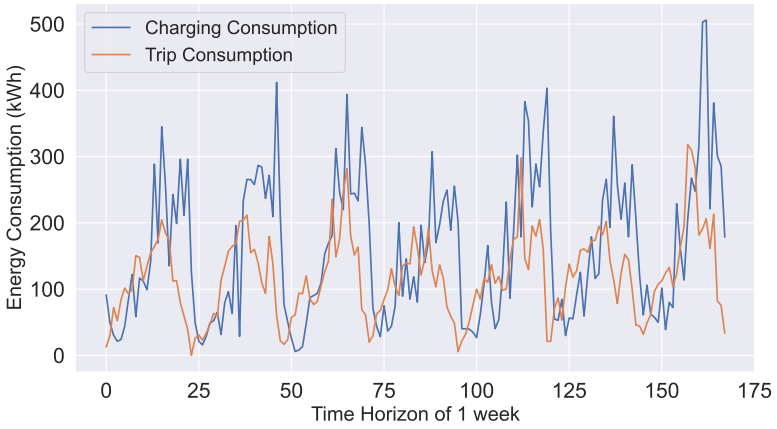
(d) Ending SoC

Figure 4.11: Barplots of Drivers Behavior Frequency Tables

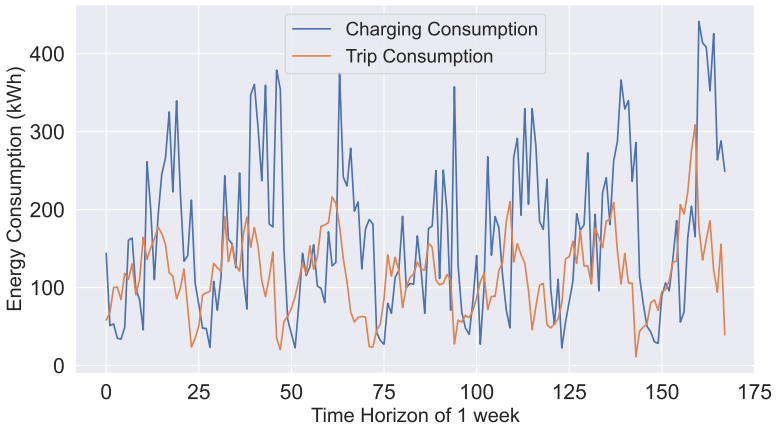
Finally Figure 4.12 shows the total energy consumption during charging and trip events of 100 EVs for a time horizon of 1 week, for each one of the 3 modes. Initially we observe a phase difference between Charging and Trip Consumption, which makes sense, as a vehicle can not charge its battery at a charging station and travel at the same time. In addition, we notice that there are some energy losses. Considering that in our data (both in the original and in the generated), the majority of drivers, fully charge their EV battery at each charge (Figure 4.11(d)), then according to [47] if charging continues beyond 80% of SoC, losses are almost double compared to the suggested SoC area (i.e. 20%–80%). Moreover, battery losses occur due to a number of additional factors. Namely, the electrical power conversion from the AC supply to the DC Liion battery for which losses can reach 5% for an AC power of 11 kW [48], wiring losses, several undesired internal electrochemical reactions, an inadequate operation of the BMS and cell warming due to internal resistance [49]. At this point we can not omit the regenerative braking factor. Regenerative braking is a process where during the trip an amount of energy is going into the battery and therefore we have partial energy recovery [50].



(a) StrictChecks



(b) LooserChecks



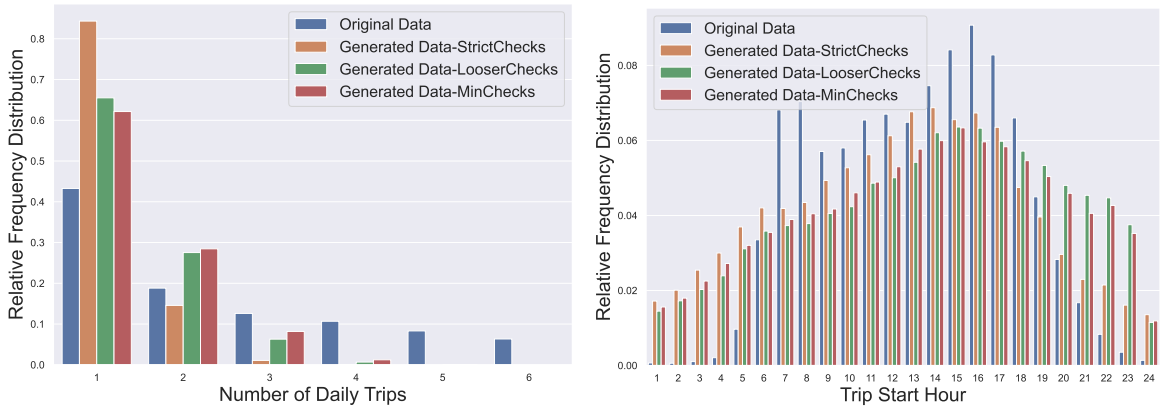
(c) MinChecks

Figure 4.12: Time-Series of EVs Consumption-Generated Data

Trip-Related Data

The present work does not focus on the mobility part of the data, but inevitably, in the context of generating data related to EVs we also export the relevant information. However focusing on the charging part (which is why we provide 3 different constraint modes) the data related to trip events is not so consistent. For example, due to the direct dependence of daily trips with daily charges,[1] the number of the former as well as the start time of a trip, differ significantly between generated and original data.

In particular, the smaller the number of daily charges, the fewer daily trips will be performed. Considering that in the generated data we mainly produce 1-2 daily charges, sometimes 3 and fewer times 4, then, respectively, only 1 or 2 daily trips will be generated, fewer times 3 and very rarely 4 or more. Due to this, the daily trips generated distributions that are quite different than the original one (see Table 4.2 and Figure 4.13(a)). In Figure 4.13(b) we also observe large discrepancies, due to the aforementioned correlation between charging and trip events. More information on this topic, as well as ideas for future work are given in Chapter 5.



(a) Daily Trips

(b) Trip Start Hour

Figure 4.13: Barplots of Drivers Behavior Frequency Tables related to Trips

[1]Recall that we first sample the charging events (e.g. number of daily charges), and then fill the trip number and details, as inferred from the sampled charges.

4.3 Discussion

Our results above show that in cases where we create a relatively large number of Production and Consumption data using the KDE and Histogram sampling methods, the overall picture of the generated data is very close to the original. However, when observing the time-series of specific variables, e.g. in the cases of Lignite and Wind generation, that exhibit no periodicity, we see that the methods tested are less effective in producing similar time series patterns as the original data.[1]. In addition, we report that the TimeGAN method, despite its positive evaluation in the literature [33] which was the motivating factor for including it in our current implementation, generates time series whose patterns are quite different than the ones in the original data for all generated variable types.

We note that our dataset generator can be used for the verification of multiagent systems and other G2V/V2G approaches. The heterogeneity of such models and related environments implies dynamic changes and fluctuations in the various measurements and exchanged values. Thus, to make sure that an experimental approach is robust and resilient in various application sites, it is crucial to test it using a number of different datasets, with varying, yet realistic behaviors, which is often very hard to obtain. Also, datasets that have a different form with respect to their time series patterns, but follow similar models, are also of interest, since they can be used to examine attack patterns and mitigations, among other security issues. To the best of our knowledge, not much related research exists, and we believe that the proposed dataset generator constitutes a valuable tool for this domain of research too.

[1]Of course, it has to be noted that a perfect match is in many cases not desirable, due to privacy-related concerns.

Chapter 5

Conclusion and Future Work

The EVs sector has been evolving dramatically the last decade and the existance of efficient algorithms that will be able to manage different EV activities is crucial. However, the development and evaluation of such algorithms frequently depends on the availability of large datasets. Such datasets, though, are usually either not large enough, or not freely available. This thesis addresses a pressing problem and need in the Smart Grid, via developing a dataset generator for EVs charging management. Our generator takes as input several publicly available datasets which contain data that describe different energy generation and demand types, as well as charging profiles of EVs and corresponding trip and type information. It then fits Histograms, Kernel Density Estimates, Generative Adversarial Networks, and Frequency Tables using the predifned data as training sets. Finally, it generates new synthetic data, which is not identical to the input, but adheres to the same principles, and relationships. The generator also accompanies the output datasets with corresponding summarizations, useful for detecting consistency issues. Our dataset generator will be made freely available for further evaluation and use by the community.

In terms of future work, we aim to evaluate the application of additional data analysis techniques for datasets that exhibit some periodicity in their values; and at the same time, to explore ways to more accurately simulate data that does not appear to be periodic, such as wind generation. Since in this thesis we focus on the charging and energy demand aspects, there is still room for improvement with respect to trip and mobility related data generation, towards more realistic and complete datasets. In this regard, our goal is to integrate in the existing work positioning and trajectories data of EVs and to use variable

speed (instead of the constant we use now) for our calculations. In addition, we could balance the difference between trips-charging events, by giving the user the ability to choose in which order the events will take place, i.e. the trip events first and then the charging ones or vice versa. So the user will be able to focus on either energy or mobility domain.

Additionally, as far as the web-based part is concerned, we aim to attach our project to a server, so that it is easily accessible by anyone and can be used without having to be downloaded. At the same time, by entering some personal information, such as an email address, the user will be able to receive a link to download the desired datasets, as soon as they are generated. There is also the possibility of improving some of the features of the GUI, such as adding an interactive representation of the results.For example, the generated plots could be displayed and also take advantage of mouse over events to display the statistics directly, e.g. the differences between the distributions.

Furthermore, apart from its main function, our dataset generator could also serve as a tool towards the verifiability of various multiagent systems, as explained in our discussion in Section 4.3. Finally, we intend to extend our dataset generator to cover the V2G mode of EV operation, which is crucial for the efficient integration of renewables into the Grid.

References

[1] Newell, R., Raimi, D., Aldana, G.: Global energy outlook 2019: the next generation of energy. Resources for the Future (2019) 8–19 1

[2] Valogianni, K., Ketter, W., Collins, J.: A multiagent approach to variable-rate electric vehicle charging coordination. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. (2015) 1131–1139 1, 9

[3] Lund, H., Kempton, W.: Integration of renewable energy into the transport and electricity sectors through v2g. Energy policy **36**(9) (2008) 3578–3587 1, 9

[4] Kempton, W., Tomić, J.: Vehicle-to-grid power implementation: From stabilizing the grid to supporting large-scale renewable energy. Journal of power sources **144**(1) (2005) 280–294 1, 9

[5] Rigas, E.S., Ramchurn, S.D., Bassiliades, N.: Managing electric vehicles in the smart grid using artificial intelligence: A survey. IEEE Transactions on Intelligent Transportation Systems **16**(4) (2015) 1619–1635 1

[6] Ramchurn, S.D., Vytelingum, P., Rogers, A., Jennings, N.R.: Putting the 'smarts' into the smart grid: a grand challenge for artificial intelligence. Communications of the ACM **55**(4) (2012) 86–97 1, 2, 7, 8

[7] Hayakawa, K., Gerding, E.H., Stein, S., Shiga, T.: Online mechanisms for charging electric vehicles in settings with varying marginal electricity costs. In: Twenty-Fourth International Joint Conference on Artificial Intelligence. (2015) 1

[8] Gerding, E.H., Robu, V., Stein, S., Parkes, D.C., Rogers, A., Jennings, N.R.: Online mechanism design for electric vehicle charging. In: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. (2011) 811–818 1

[9] Koufakis, A.M., Rigas, E.S., Bassiliades, N., Ramchurn, S.D.: Offline and online electric vehicle charging scheduling with v2v energy transfer. IEEE Transactions on Intelligent Transportation Systems **21**(5) (2019) 2128–2138 1

[10] Flammini, M.G., Prettico, G., Fulli, G., Bompard, E., Chicco, G.: Interaction of consumers, photovoltaic systems and electric vehicle energy demand in a reference network model. In: 2017 International Conference of Electrical and Electronic Technologies for Automotive, IEEE (2017) 1–5 2

[11] Xiong, Y., An, B., Kraus, S.: Electric vehicle charging strategy study and the application on charging station placement. Autonomous Agents and Multi-Agent Systems **35**(1) (2021) 1–19 2

[12] Steward, D.M.: Critical elements of vehicle-to-grid (v2g) economics. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States) (2017) 2

[13] Kamboj, S., Kempton, W., Decker, K.S.: Deploying power grid-integrated electric vehicles as a multi-agent system. In: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. (2011) 13–20 2

[14] Tao, M., Ota, K., Dong, M.: Foud: Integrating fog and cloud for 5g-enabled v2g networks. IEEE Network **31**(2) (2017) 8–13 2

[15] Watson, R.T., Boudreau, M.C., Chen, A.J.: Information systems and environmentally sustainable development: energy informatics and new directions for the is community. MIS quarterly (2010) 23–38 2

[16] Develder, C., Sadeghianpourhamami, N., Strobbe, M., Refa, N.: Quantifying flexibility in ev charging as dr potential: Analysis of two real-world data sets. In: 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), IEEE (2016) 600–605 2, 18

[17] Quirós-Tortós, J., Navarro-Espinosa, A., Ochoa, L.F., Butler, T.: Statistical representation of ev charging: Real data analysis and applications. In: 2018 Power Systems Computation Conference (PSCC), IEEE (2018) 1–7 2

[18] Islam, E., Moawad, A., Kim, N., Rousseau, A.: An extensive study on sizing, energy consumption, and cost of advanced vehicle technologies. Contract ANL/ESD-17/17 (2018) 2

[19] Pevec, D., Babic, J., Podobnik, V.: Electric vehicles: A data science perspective review. Electronics **8**(10) (2019) 1190 2

[20] Barker, S., Mishra, A., Irwin, D., Cecchet, E., Shenoy, P., Albrecht, J., et al.: Smart*: An open data set and tools for enabling research in sustainable homes. SustKDD, August **111**(112) (2012) 108 2

[21] Pecan Street: Pecan Street Dataport. https://www.pecanstreet.org/dataport/ Online; accessed 24 February 2021. 2

[22] Nalmpantis, C., Vrakas, D.: Machine learning approaches for non-intrusive load monitoring: from qualitative to quantitative comparation. Artificial Intelligence Review **52**(1) (2019) 217–243 2

[23] Asghar, M.R., Dán, G., Miorandi, D., Chlamtac, I.: Smart meter data privacy: A survey. IEEE Communications Surveys & Tutorials **19**(4) (2017) 2820–2835 2

[24] Li, B., Kisacikoglu, M.C., Liu, C., Singh, N., Erol-Kantarci, M.: Big data analytics for electric vehicle integration in green smart cities. IEEE Communications Magazine **55**(11) (2017) 19–25 3

[25] Lee, Z.J., Johansson, D., Low, S.H.: Acn-sim: An open-source simulator for data-driven electric vehicle charging research. In: 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), IEEE (2019) 1–6 3

[26] Energy, U.: Grid 2030-a national vision for electricity's second 100 years. US Department Energy, Tech. Rep. (2003) 7, 8

[27] Agreement, P.: Paris agreement. In: Report of the Conference of the Parties to the United Nations Framework Convention on Climate Change (21st Session, 2015: Paris). Retrived December. Volume 4., HeinOnline (2015) 2017 7

[28] Grid, S.: European technology platform for electricity networks of the future. European Commission (2005) 8

[29] Asmus, P.: Microgrids, virtual power plants and our distributed energy future. The Electricity Journal **23**(10) (2010) 72–82 8

[30] Pearson, K.: X. contributions to the mathematical theory of evolution.—ii. skew variation in homogeneous material. Philosophical Transactions of the Royal Society of London.(A.) (186) (1895) 343–414 10

[31] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**(11) (2020) 139–144 14

[32] Osborne, M.J., et al.: An introduction to game theory. Volume 3. Oxford university press New York (2004) 15

[33] Yoon, J., Jarrett, D., van der Schaar, M.: Time-series generative adversarial networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., eds.: Advances in Neural Information Processing Systems. Volume 32., Curran Associates, Inc. (2019) 15, 33, 67

[34] Ratner, A.J., De Sa, C.M., Wu, S., Selsam, D., Ré, C.: Data programming: Creating large training sets, quickly. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R., eds.: Advances in Neural Information Processing Systems. Volume 29., Curran Associates, Inc. (2016) 17

[35] Dunnmon, J.A., Ratner, A.J., Saab, K., Khandwala, N., Markert, M., Sagreiya, H., Goldman, R., Lee-Messer, C., Lungren, M.P., Rubin, D.L., Ré, C.: Cross-modal data programming enables rapid medical machine learning. Patterns **1**(2) (2020) 100019 17

[36] Iftikhar, N., Liu, X., Danalachi, S., Nordbjerg, F.E., Vollesen, J.H.: A scalable smart meter data generator using spark. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer (2017) 21–36 17

[37] Zhang, C., Kuppannagari, S.R., Kannan, R., Prasanna, V.K.: Generative adversarial network for synthetic time series data generation in smart grids. In: 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), IEEE (2018) 1–6 18

[38] Flammini, M.G., Prettico, G., Julea, A., Fulli, G., Mazza, A., Chicco, G.: Statistical characterisation of the real transaction data gathered from electric vehicle charging stations. Electric Power Systems Research **166** (2019) 136–150 18

[39] Brady, J., O'Mahony, M.: Modelling charging profiles of electric vehicles based on real-world electric vehicle charging data. Sustainable Cities and Society **26** (2016) 203–216 18

[40] Lee, Z.J., Li, T., Low, S.H.: Acn-data: Analysis and applications of an open ev charging dataset. In: Proceedings of the Tenth ACM International Conference on Future Energy Systems. (2019) 139–149 18

[41] Amara-Ouali, Y., Massart, P., Poggi, J.M., Goude, Y., Yan, H.: A review of electric vehicle charging session open data: Poster. In: Proceedings of the Twelfth ACM International Conference on Future Energy Systems. (2021) 278–279 18

[42] Mangipinto, A., Lombardi, F., Sanvito, F., Quoilin, S., Pavičević, M., Colombo, E.: Rap-mobility: A novel application of the ramp main engine for generating bottom-up stochastic electric vehicles load profiles (2020) 18

[43] Gaete-Morales, C., Kramer, H., Schill, W.P., Zerrahn, A.: An open tool for creating battery-electric vehicle time series from empirical data, emobpy. Scientific data **8**(1) (2021) 1–18 18

[44] Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. Pattern recognition **36**(2) (2003) 451–461 24

[45] YData Company: ydata-synthetic. https://github.com/ydataai/ydata-synthetic (2021) Online; accessed 10 Septempber 2021. 33

[46] Massey Jr, F.J.: The kolmogorov-smirnov test for goodness of fit. Journal of the American statistical Association **46**(253) (1951) 68–78 48

[47] Kostopoulos, E.D., Spyropoulos, G.C., Kaldellis, J.K.: Real-world study for the optimal charging of electric vehicles. Energy Reports **6** (2020) 418–426 64

[48] Apostolaki-Iosifidou, E., Codani, P., Kempton, W.: Measurement of power loss during electric vehicle charging and discharging. Energy **127** (2017) 730–742 64

[49] Dai, H., Zhang, X., Wei, X., Sun, Z., Wang, J., Hu, F.: Cell-bms validation with a hardware-in-the-loop simulation of lithium-ion battery cells for electric vehicles. International Journal of Electrical Power & Energy Systems **52** (2013) 174–184 64

[50] De Cauwer, C., Maarten, M., Heyvaert, S., Coosemans, T., Van Mierlo, J., et al.: Electric vehicle use and energy consumption based on realworld electric vehicle fleet trip and charge data and its impact on existing ev research models. World Electric Vehicle Journal **7**(3) (2015) 436–446 64

Appendix A

Data Decription Tables

Table A.1: Energy Production and Consumption Data (CSV File Example)

Column Name	*Description*	*Example*
Date	The date in appropriate format (yyyy-mm-dd)	2021-01-01
Year	The year corresponding to each date	2021
Month_of_Year	The month corresponding to each date	1
Day_of_Week	The number corresponding to the day of the week (1 to 7, Monday to Sunday)	5
Time_of_Day	The number corresponding to the one-hour period of the day (1-24, 1 is for 00:00 to 01:00 and so on)	5
Lignite_Gen(MW)	Energy produced from Fossil Brown coal/Lignite (in megawatts)	1798.0
FossilGas_Gen(MW)	Energy produced from Fossil Gas (in megawatts)	1944.0
Solar_Gen(MW)	Energy produced from Sun (in megawatts)	0
Wind_Gen(MW)	Energy produced from Wind (in megawatts)	246.0
Total_Load(MW)	Actual total Load (in megawatts)	5036.0
Import_Export(MW)	The sum of the produced energy, from all its sources, must be equal to the total energy load. This column lists the difference: TotalLoad - (LigniteGen + FossilGasGen + SolarGen + WindGen). If the sign is positive then it means that energy / load is imported in micro grid or regional levels. Otherwise, energy-load is exported.	1048.0

Table A.2: EV's Technical Characteristics (CSV File Example)

Column Name	*Description*	*Example*
CarModel	The model of each EV	Nissan Leaf
BatteryCapacity (kWh)	The battery capacity of the EV (in kilowatt hours)	40.0
Maximum_DC_Charging_Power (kW)	The maximum charging capacity of the EV, with direct current, (in kilowatts). Maximum charging capacity cannot exceed this value, even if the charging station allows this.	50
MeanEnergyConsumption (kWh/100km)	The mean energy consumption (in kilowatt hours per 100 kilometers)	18.5
Category	Letters A, B or C. EV models are separated into three categories, dependingon their battery capacity. Category A:17<= BatteryCapacity<40 Category B:40<= BatteryCapacity<70 Category C:BatteryCapacity >=70	2

Table A.3: Charger's Specifications (CSV File Example)

Column Name	*Description*	*Example*
ChargerType	The type of the charger. (Regular socket, Single phase or 3 phase)	Single Phase 32A
ChargerRating	Rating depending on charging speed (Slow, Fast, Rapid)	Fast
AC_DC	Type of charger depending on current type (AC or DC)	AC
Rated_Power(kW)	Rated power of the charger (in kilowatts) TimeToCharge = BatteryCapacity(kWh)/ Rated_Power(kW)	7.4

Table A.4: EV's Preprocessed Charging Data (CSV File Example)

Column Name	*Description*	*Example*
ParticipantID	Driver's ID as given from "My Electric Avenue" data	GC10
BatteryChargeStartMonth	The month of year that the charging event started (1-12)	2
BatteryChargeStartDay	The date of month that the charging event started (1-31)	16
BatteryChargeStartHour	The hour of the day that the charging event started (1-24)	18
BatteryChargeStopMonth	The month of year that the charging event stopped (1-12)	2
BatteryChargeStopDay	The date of month that the charging event stopped (1-31)	16
BatteryChargeStopHour	The hour of the day that the charging event stopped (1-24)	19
ChargingTime(min)	The total duration of the charging event (in minutes)	40.0
Starting_SoC	The initial State of Charge of the battery, i.e. when the car was connected to the charger. (0 to 1, 0 corresponds to empty battery and 1 to full).	0.17
Ending_SoC	The final State of Charge of the battery, i.e. when the car was disconnected to the charger. (0 to 1, 0 corresponds to empty battery and 1 to full).	0.92
SoC_Diff	The difference: Ending_SoC- Starting_SoC, essentially the specific value shows us what percentage of the battery was charged during charging event.	0.75

Table A.5: EV's Preprocessed Trip Data (CSV File Example)

Column Name	*Description*	*Example*
ParticipantID	Driver's ID as given from "My Electric Avenue" data	GC10
TripStartMonth	The month of year that the trip event started (1-12)	11
TripStartDay	The date of month that the trip event started (1-31)	6
TripStartHour	The hour of the day that the trip event started (1-24)	10
TripStopMonth	The month of year that the trip event stopped (1-12)	11
TripStopDay	The date of month that the trip event stopped (1-31)	6
TripStopHour	The hour of the day that the trip event stopped (1-24)	12
TripTime(min)	The total duration of the trip event (in minutes)	4.0
TripDistance(m)	The distance (in meters) was covered during the trip event	1177.0
PowerConsumption(Wh)	The power consumed by the EV during the trip event (in Watthours)	366.0
Velocity(km/h)	The velocity of the vehicle during the trip (in kilometers per hour) considering that it ideally performs linear motion with constant speed u=dx/dt. Values are from 20 to 100.	18.0

Table A.6: Frequency Tables (CSV Files Example)

Type/Variable (examples)	*Column Names*		
	Values	*Frequency*	*Weights*
Daily Charges (1,30381,0.66...1)	The different values that correspond to the number of the daily charges that a driver can make. (1 to 4).	The frequency of occurrence of the corresponding variable, i.e. how many times it appears on the original dataset.	The weight corresponds to the respective value. The weight is essentially the same as the relative frequency of the value. The higher the weight, the greater the probability that the specific value will appear in the produced dataset and vice versa.
Charge Start Hour (17,6903,0.09...5)	The different values that correspond to the start time of a charging event. (1 to 24).		
Starting SoC (0.58,9149,0.12...4)	The different values that correspond to the initial State of Charge of the battery (0 to 1).		
Ending SoC (1.0,50720,0.66...5)	The different values that correspond to the final State of Charge of the battery (0 to 1).		
Trip Velocity (30,50780,0.16...7)	The different values that correspond to the velocity of the vehicle during a trip event (20 to 100, and rounded every 5 km/h).		

Table A.7: Behavior Data of a given Driver in all 3 different Status Modes (CSV File Example)

Column Name	*Description*	*Example 1*	*Example 2*	*Example 3*
Date	The respective date	2021-01-01	2021-01-01	2021-01-01
DriverID	The driver's ID, in form "DRx", where x is 1 for the 1st driver, 2 for the 2nd and so on.	DR1	DR1	DR1
Year	The year corresponds to each date	2021	2021	2021
Month_of_Year	The month corresponds to each date	1	1	1
Day_of_Week	The number corresponds to the day of the week (1 to 7, Monday to Sunday)	5	5	5
Time_of_Day	The number corresponds to the one-hour period of the day (1-24, 1 is for 00:00 to 01:00 and so on)	1	1	1
EV_Model	The model of each EV	Jaguar I-Pace	Jaguar I-Pace	Jaguar I-Pace
Battery_Capacity(kWh)	The battery capacity of the EV (in kilowatt hours)	90.0	90.0	90.0
Category	Letters A, B or C. EV models are separated into three categories, dependingon their battery capacity. Category A:17<= BatteryCapacity<40 Category B:40<= BatteryCapacity<70 Category C:BatteryCapacity >=70	3	3	3
Mean_Energy_Consumption(kWh/100km)	The mean energy consumption (in kilowatt hours per 100 kilometers)	21.2	21.2	21.2
Max_DC_Power(kW)	The maximum charging capacity of the EV, with direct current, (in kilowatts). Maximum charging capacity cannot exceed this value, even if the charging station allows this.	100	100	100
Status	Values 0 to 2, 0: inactivity state 1: charging event happens 2: trip event happens	0	1	2
Starting_SoC	The initial State of Charge of the battery when the one hour interval begins (0 to 1, 0 corresponds to empty battery and 1 to full).	N/A	0.67	1.0
Ending_SoC	The final State of Charge of the battery when the one hour interval ends (0 to 1, 0 corresponds to empty battery and 1 to full).	N/A	1.0	0.95
Charger_Type	The type of the charger. (Regular socket, Single phase or 3 phase, AC or DC)	N/A	3 phase-DC	N/A
Charger_Power(kW)	Rated power of the charger (in kilowatts) TimeToCharge = BatteryCapacity(kWh)/ Rated_Power(kW)	N/A	120.0	N/A
Charging_Time(mins)	The total duration of the charging event (in minutes)	N/A	18	N/A
Consumption_Charging(kW)	The energy consumed by the EV (in kilowatts) during the corresponding charging period, i.e. on each row	N/A	36.0	N/A
Trip_Time(mins)	The duration of the trip event (in minutes) which corresponds to the interval of each one hour, i.e. on each row.	N/A	N/A	60
Trip_Distance(km)	The distance covered by the EV (in kilometers) which corresponds to the interval of each one hour, i.e. on each row.	N/A	N/A	40
Trip_Consumption(kwh)	The power consumed by the EV, corresponds to the interval of each one hour, i.e. on each row.	N/A	N/A	8.48