Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

CO2 signal prediction for energy demand-side management: a Service-Oriented Architecture.

Mohamed Elmougi

Course of Study:	Computer Science
Examiner:	Prof. Dr. Marco Aiello
Supervisor:	Laura Fiorini, M.Sc
Commenced:	December 3, 2018
Completed:	June 3, 2019

Abstract

Carbon dioxide is an essential component of the Earth's atmosphere. Due to high energy consumption, residential and office buildings are responsible for one-third of the total CO_2 emissions in the European Union [EU19]. The energy industry has the highest contribution of the Greenhouse Gases (GHGs) emissions in Germany [uba19]. Carbon dioxide is considered the most harmful greenhouse gas in the atmosphere regarding its effect on global warming and climate change issues [NASA19]. The amount of CO_2 emitted to produce electricity (i.e., CO_2-equivalent intensity factor) can greatly vary in time, depending on the sources used to generate it. CO_2-efficient rescheduling of electricity consumption and integration of energy carriers may enable up to 40% emission reductions [FA18]. The thesis proposes an implementation of a service-oriented architecture that provides one-day ahead forecasts for the CO_2-equivalent intensity factor in Germany using electricity generation forecasting data from ENTSO-E API [ENTSO19] and Weather forecasting data from Darksky API [darksky19]. This work proposes an alternative approach to choose and engineer the required independent variables for the forecasting process. Ordinary Least Squares, Ridge Regression, Polynomial Regression, Decision Trees, Random Forest, and KNNs are applied and evaluated to model the forecasting problem. The results show that the Random Forest model has better evaluation performance among the six models with a R^2 value of 0.94 on a separated test dataset. Therefore, the implemented service-oriented architecture uses the Random Forest model to forecast the CO_2-equivalent intensity factor in Germany. The implemented architecture provides the forecasts using a REST API based architecture through an HTTP GET request with a JSON response.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

Many scientists agree that the main reason for global warming is the emission of Greenhouse gases (GHGs) due to human activities [NASA19]. Greenhouse gas is any gas that is able to absorb the infrared radiation (net heat energy) from the earth emissions and re-radiate it back to Earth's surface. Water vapour, Carbon dioxide, and methane are all types of greenhouse gases and probably the most important greenhouse gases in terms of effect. Carbon dioxide is the most important greenhouse gas in the atmosphere in terms of its effect on global warming and climate change issues.

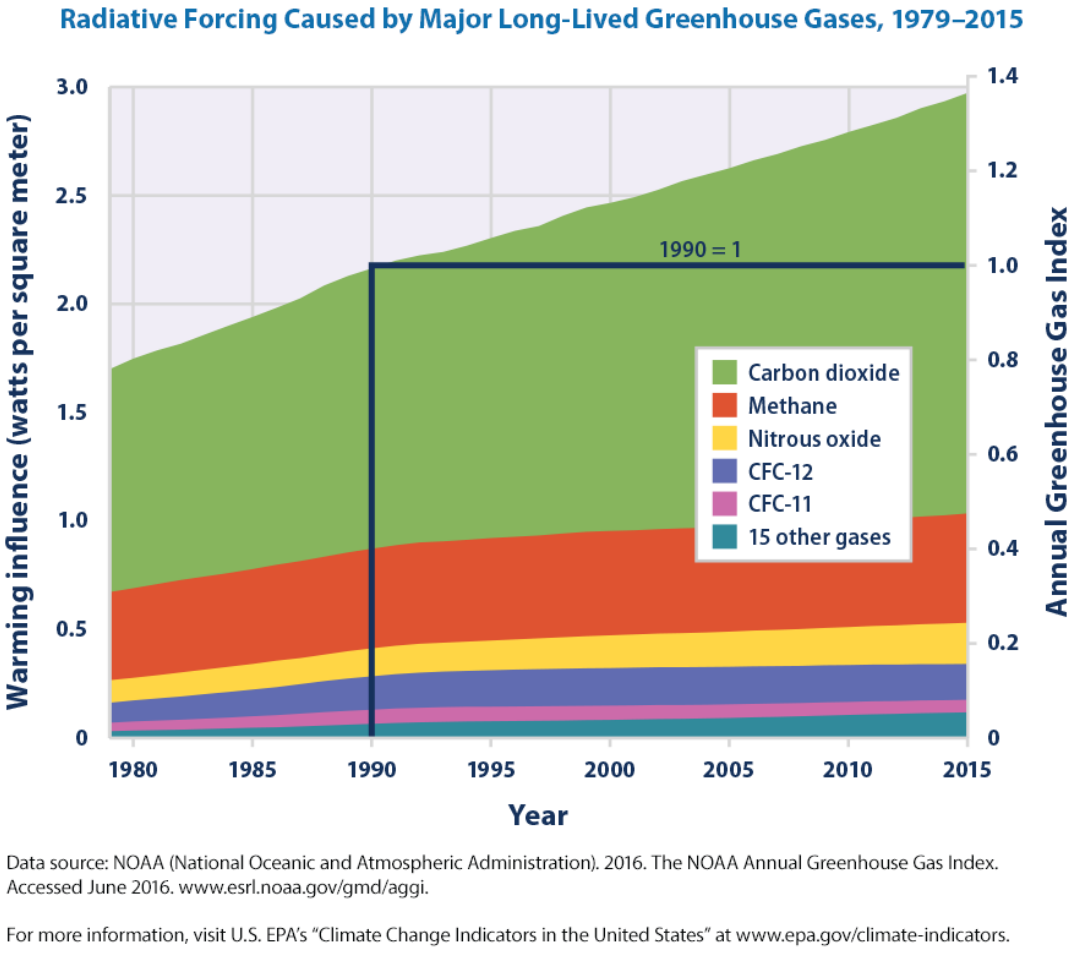

Figure 1.1: Radiative Forcing Caused by Major Long-Lived Greenhouse Gases, 1979-2015

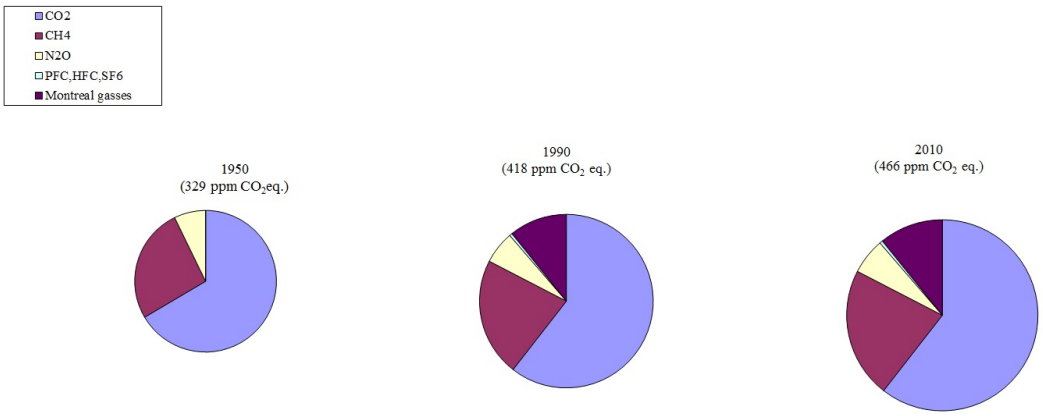

Figure 1.2: Contribution of different GHGs to the overall GHGs concentration in 1950- 1990- 2010 [eea19].

The United States Environmental Protection Agency (EPA) has done research to measure the "radiative forcing" or heating effect caused by greenhouse gases in the atmosphere [epa17]. Figure 1.1 shows the amount of radiative forcing caused by various greenhouse gases, based on the change in concentration of these gases in the Earth's atmosphere since 1750. Radiative forcing is calculated in watts per square meter, which represents the size of the energy imbalance in the atmosphere. On the right side of the graph, radiative forcing is converted to the Annual Greenhouse Gas Index, which is set to a value of 1.0 for 1990 [epa17]. It is clear from Figure 1.1 that the Carbon dioxide (CO_2) plays the biggest role in the warming influence. It can be concluded from Figure 1.1 and Figure 1.2, there is a high contribution to the most warming influencing gas in the atmosphere (Carbon dioxide). Such conclusion is supported also by the European Environmental Agency [eea19], and its research on the contribution of different GHGs in the European Union in 1950, 1990, and 2010, shown in Figure 1.2. CO_2 can be emitted from natural sources (e.g., volcanoes and respiration) as well as from human activities (e.g., Energy production, Transport, and Industry).

1.1 Problem Statement

Finding a solution to reduce the CO_2 emissions is an urgent task all over the world. In Germany and according to the German Environment Agency (Umwelt Bundesamt), the energy industry has the highest contribution of the GHGs emissions between all the human activities that affect emissions [uba19]. According to the European Commission and due to the high energy consumption, residential and office buildings are responsible for one-third of total CO_2 emissions in the European Union [EU19]. There are 13 energy production types in Germany [ENTSO19]. Each energy production type has a different environmental effect and can affect differently in emitting CO_2 [FA18]. The emission intensity factor associated with kWh in a windy and sunny day will significantly be lower when most of the electricity generated by plants burning fossil fuels [FA18].

Rescheduling and controlling the electricity consumption in residential and office buildings can help in reducing the amount of the CO_2 emissions. If the residential and office buildings know the amount of the Carbon dioxide emissions Intensity in advance, a CO_2 reducing demand rescheduling can be done in terms of the timing of the electricity consumption and switching between the energy carriers. In [FA18] a novel CO_2-efficient energy management approach to schedule household appliances in Germany is done assuming that the CO_2 signals are known at the beginning of the simulated day. The scheduling is modeled as a mixed-integer linear programming (MILP) problem considering six common appliances and seasonal thermal loads, which include the hot water and space heating demands. The research has concluded that switching energy carriers can successfully enable up to 40% emissions reductions while indicating that shifting loads in time has little impact.

Therefore, if there is a precise prediction of the CO_2 Intensity factor, emissions can be reduced greatly using different techniques of scheduling. The problem of energy-related CO_2 emissions forecasting (CEF) is not widely discussed in researches. Nevertheless, CEF can help in reducing a big amount of the emissions which leads to a huge positive impact on environmental issues as global warming and climate change. In this research, we are applying different algorithms to predict the energy-related CO_2 emissions in Germany using the available data of the energy production from [ENTSO19].

1.2 Research Questions

In this thesis, a service-oriented architecture is developed to forecast the CO_2-equivalent intensity factor that is related to the electricity generation in Germany hourly. The forecasting is done using the available data about electricity generation from [ENTSO19] and the weather condition data from [darksky19] in Germany. The main research questions we investigate in this thesis are the following:

- How can a CO_2 emissions forecasting be done using machine learning algorithms?
- How are the machine learning algorithms evaluated to provide the best emissions forecasting?
- How are the independent variables evaluated and handled further than electricity generation that are needed for the emissions forecasting to provide a precise forecasting?

- For how long should the forecasting be done and in which time manner e.g, hourly, daily, monthly or yearly?
- How will the forecasted emissions data be provided as a service using a service oriented architecture?

These questions will be answered through the next chapters and as a result of the thesis a service-oriented architecture to provide forecasting of CO_2 emissions will be produced.

1.3 Methodology

The task of forecasting the CO_2 emissions using electricity generation data is considered as supervised learning in the field of machine learning. In supervised learning, the input value (X) and the output value (Y) are known, and one of the algorithms will be used to find the map function between the input (X) and the output (Y) [Machle19]. Furthermore, after approximating the mapping function a new input data (X) can be used to predict the new output data (Y) using the produced mapping function. In particular, it is a regression problem since the mapping function will be used to predict an output of real value; the CO_2 emissions. The supervised learning can be explained in a simple equation according to [Ste11]:

$$Y = F(X)$$

In this equation, Y is the output variable, in our case, the CO_2 emission values and X represents the input values, in our case, the electricity generation values and new other input variables will be declared during the thesis. The goal is to find F which is the mapping function between the input and output variables. To create a prediction model, historical data are needed to train the model and to test it. Therefore, providing historical data of the input and output data should be the first step after declaring the input variables. The provided input variables should pass some constraints as declared in Chapter 3 to reach the goal of forecasting the CO_2-equivalent intensity factor hourly. For instance, the input variable should be available in advance for the needed forecasting period; e.g, tomorrow, two days from today or a week ahead. After providing the historical data, different algorithms of machine learning are applied and evaluated to produce a precise model. Finally, a service-oriented architecture is implemented to provide the forecasted data as a service.

1.4 Thesis Structure

The thesis starts with background research in Chapter 2. The research is done in the topics of electricity consumption forecasting, electricity price forecasting, carbon dioxide emissions intensity forecasting, and the effect carbon dioxide on global warming. After that, the thesis proposes a data engineering pipeline for handling the needed data for forecasting the CO_2-equivalent intensity factor in Chapter 3. Then, six regression models (Ordinary Least Squares, Ridge Regression, Polynomial Regression, Decision Trees, Random Forest, and KNNs) are applied and evaluated in Chapter 4 and Chapter 5. The model with the best evaluation "Random Forest" is used to forecast the CO_2-equivalent intensity factor in Germany in hourly based for one day ahead. The forecasts are provided through a REST API based architecture as explained in Chapter 6. Chapter 7 summarizes the work that is done through the thesis and shows a future work plan that adds some enhancements to different levels.

2 Background

Forecasting refers to predicting what will happen in the future through the collection and analysis of past and current data. Lately, the researchers in a lot of fields are using forecasting to change a specific behavior in the future or at least trying to control it. To the best of the author's knowledge, that is since there are not much CO_2 emissions forecasting researches and the CO_2 emissions have a tight relationship with the electricity consumption, and sometimes price, forecasting. The study of electricity consumption and electricity price forecasting practices can help in building a picture of the tools, algorithms, ideas, and circumstances that might be faced during CO_2 emissions forecasting. In the first subsection, general knowledge about the CO_2 emissions and how it can affect global warming will be discussed. In the second subsection, some papers in the field of electricity consumption and price forecasting, as well as some papers regarding CO_2 emissions forecasting that are written in different universities will be discussed.

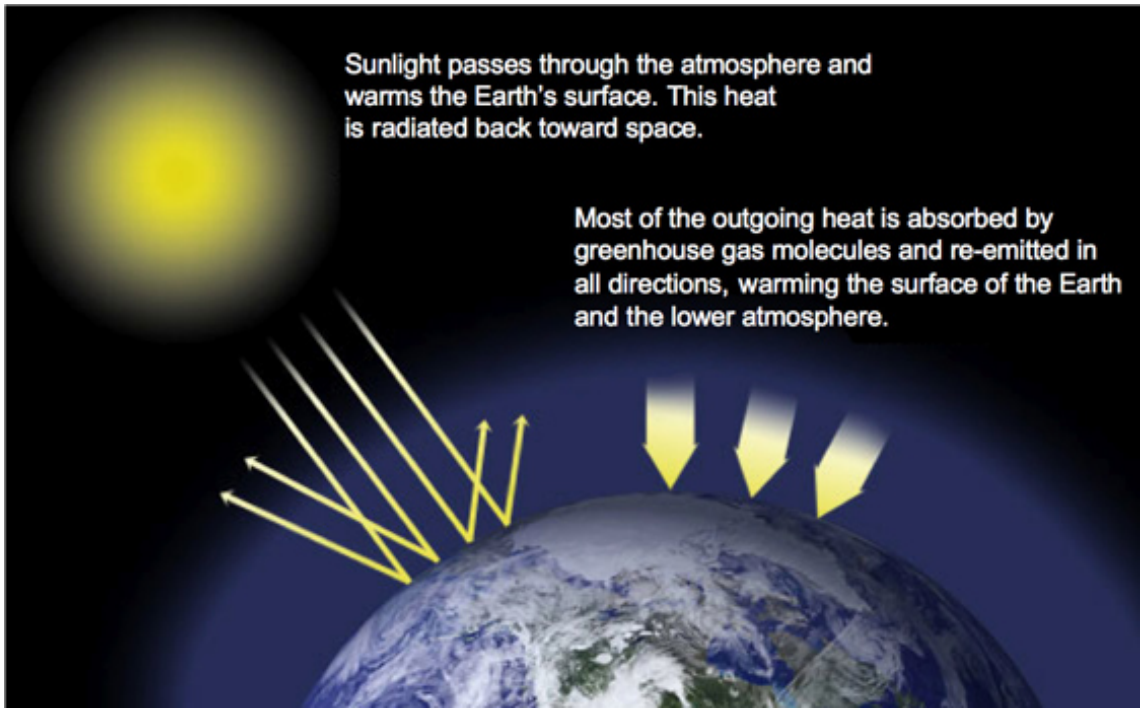

Figure 2.1: A blanket around the Earth [NASA19]

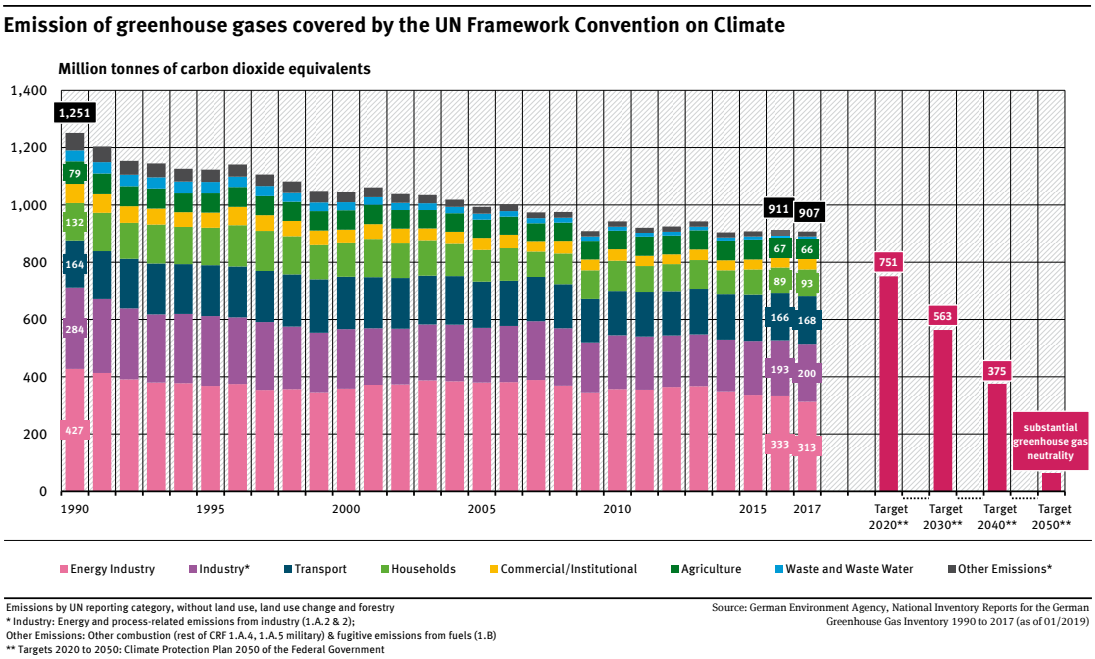

Figure 2.2: Greenhouse gases emissions [uba19]

2.1 Carbon Dioxide

According to climate scientists, Greenhouse gases (GHGs) are the main reason for global warming. Hence, they warm the Earth because they play the role of a ”blanket” for insulating the Earth and absorbing the energy which leads to slowing the rate at which the energy escapes to space. Greenhouse gases (GHGs) absorb that heat and release it gradually over time. The GHGs differ in terms of the ability to absorb energy which is known scientifically as ”radiative efficiency” and differ also in the period of time that gas can stay in the atmosphere which is known scientifically as ”lifetime”. Some of the gases do not respond physically or chemically to changes in temperature which are scientifically called ”Forcing”. And the gases that respond physically or chemically to changes in temperature like the water vapor are called ”feedbacks” [NASA19]. In Figure 2.1, the NASA global climate explains the effect of the GHGs on Earth and how the GHGs act like a blanket. Among GHGs, CO_2 is the main responsible for global warming. CO_2 is released either through natural processes such as a volcano or through human activities such as electricity generation. Humans increased the concentration of CO_2 in the atmosphere to more than third the usual since the Industrial Revolution began [NASA19].

In Germany, the German Environment Agency (Umwelt Bundesamt) [uba19] declares that the GHGs emissions declined by around 28 % between 1990 and 2017. In Figure 2.2 the German Environment Agency has provided an indicator for the GHGs emissions in Germany for the period between 1990 to 2017. The figure also shows the German Governmental target plan for up to the year 2050. The study shows that the energy industry and the industry responsible for around 57% of the emissions in total, where the industry emissions here refer to the energy and process-related emissions from industry. Changing the amount of the CO_2 emissions caused by the energy industry would affect a large scale of the total amount of the emissions.

2.2 Forecasting

Being able to predict what could happen in the future is a great challenge in many fields. After the improvements happening for machine learning in the last decade, forecasting became a main topic in the industry [AAGE10]. Machine Learning Forecasting is certainly the method to achieve high forecast accuracy and move forward. With Machine Learning Forecasting, processors learn from mining loads of big data without human interference to deliver unparalleled customer demand insights. Applying machine learning forecasting for the CO_2 emissions can promote more sustainable electricity consumption and reduce the environmental impact of CO_2. Demand-wise most of the forecasting related to the energy were aiming to forecast the consumption or the price. Some of the forecasting models will be discussed in the next sub-sections.

2.2.1 Electricity Consumption Forecasting

In the last 15 years, due to the increase in electricity consumption, many papers have focused on electricity consumption forecasting to avoid having an electricity shortage and to guarantee adequate infrastructures by using different techniques.

In [BMN09], an electricity consumption forecasting in Italy is done using linear regression models. The forecasting is done using a historical electricity consumption data from 1970 to 2007, gross domestic product (GDP), gross domestic product per capita (GDP per capita), and population. The model uses annual historical electricity consumption data and aims at forecasting annual electricity consumption up to 2030. The paper differentiates between domestic and non-domestic consumption in the models. Therefore, the paper expresses more than a linear regression model in linear logarithmic form linking the quantity of annual domestic electricity consumption to electricity price and GDP per capita for both domestic and non-domestic consumption. After generating comparisons and analysis to the produced model, the paper concludes that there is no need to consider electricity price as an explaining variable in forecasting models for Italian electricity consumption and those pricing policies cannot be used to promote the efficient use of electricity in Italy.

In [AAA97], a monthly domestic electricity consumption forecasting in Eastern Saudi Arabia is done using AIM (Abductory Induction Mechanism). Key weather parameters and demographic and economic indicators are the independent variables for the monthly forecasting. The produced model can forecast up to 1 year by using 5 years of historical data to train the model. The increase in electricity consumption in Eastern Saudi Arabia is reflecting the overall increase in electricity consumption all over the world. The paper claims that an AIM model which uses only the mean relative humidity and air temperature gives an average forecasting error of about 5–6% over the year.

In [KTCH15], least squares support vector machines (LS-SVMs) is used to forecast the electricity consumption in Turkey while considering the traditional regression analysis and artificial neural networks (ANNs). Gross electricity generation, installed capacity, total subscription and population using historical data from 1970 to 2009 are used as the independent variables to train the model in this paper. Furthermore, the paper applies a multilinear regression (MLR) to forecast the electricity consumption in Turkey and the regression coefficients are obtained by computer-aided solution of the regression equation. The third model to be applied in the study is ANN, a multilayer feed-forward

	Trainning			Test		
Performance criteria	**MLR**	**ANN**	**LS-SVM**	**MLR**	**ANN**	**LS-SVM**
MAPE (%)	4.01	4.86	0.876	3.34	1.19	1.004
MaxError	7.62	3.03	1.05	8.25	5.92	4.4
MSE	6.06	3.08	0.1699	6.45	3.3	2.06
RMSE	2.46	1.76	0.412	2.54	1.82	1.435
SSE	163.69	3.19	4.59	83.79	42.85	26.782
R-squared (%)	99.72	99.89	99.99	99.76	99.88	99.98

Table 2.1: Comparison between the MLR, ANN and LS-SVM models used for forecasting electricity consumption in Turkey [KTCH15].

back-propagation neural network model is used for the estimation of net electricity consumption in Turkey as well. The MATLAB [MAT19] neutral network toolbox is used to train the data for this model. MATLAB [MAT19] is used to implement, train and the analysis of the algorithm. Several performance criteria are used to compare the models and taking the decision of the suitable model for the case. In Table 2.1 the performance evaluation is done to show that the LS-SVM can be a good Model for electricity consumption. Therefore, the paper claims that the proposed LS-SVM model is an accurate and quick prediction method.

In [GT04], an analysis for the Italian electricity demand is carried out. A modification of the EDM (Energy Demand Model) is used to forecast the national energy consumption in industry, households, and services [GT04]. The paper uses some equations in a form of the simultaneous system indicated as LES (Linear Expenditure System) to allocate the energy products. The forecasting for the Italian consumption of energy in this paper is up to 2020.

2.2.2 Electricity Price Forecasting

The need for electricity price forecasting is raised in the last decades due to the ongoing liberalization of electricity markets. The substantial dependence of companies and private households on the price of the electricity has increased the need for modeling the electricity prices in the energy researches [ZSH15].

In [ZSH15], an hourly based econometric model for the electricity prices of the European Power Exchange (EPEX) is done using an efficient iterative re-weighted lasso approach. The data of the period 28.09.2010 up to 01.05.2014 is used to train the model. In addition, the paper has further approaches rather than modeling, such as providing insights for the structure of the leverage effect and providing the effect of the wind and solar energy to the prices of the electricity. Specific holidays effect and daylight are taken into account for the wind and solar generation. As a result of the study, the paper provides evidence for the effect of renewable energy on the reduction of the price of electricity. The study shows that an increase in combined wind and solar power of 1 GWh leads to a decrease in the price of 2.03 EUR/MWh.

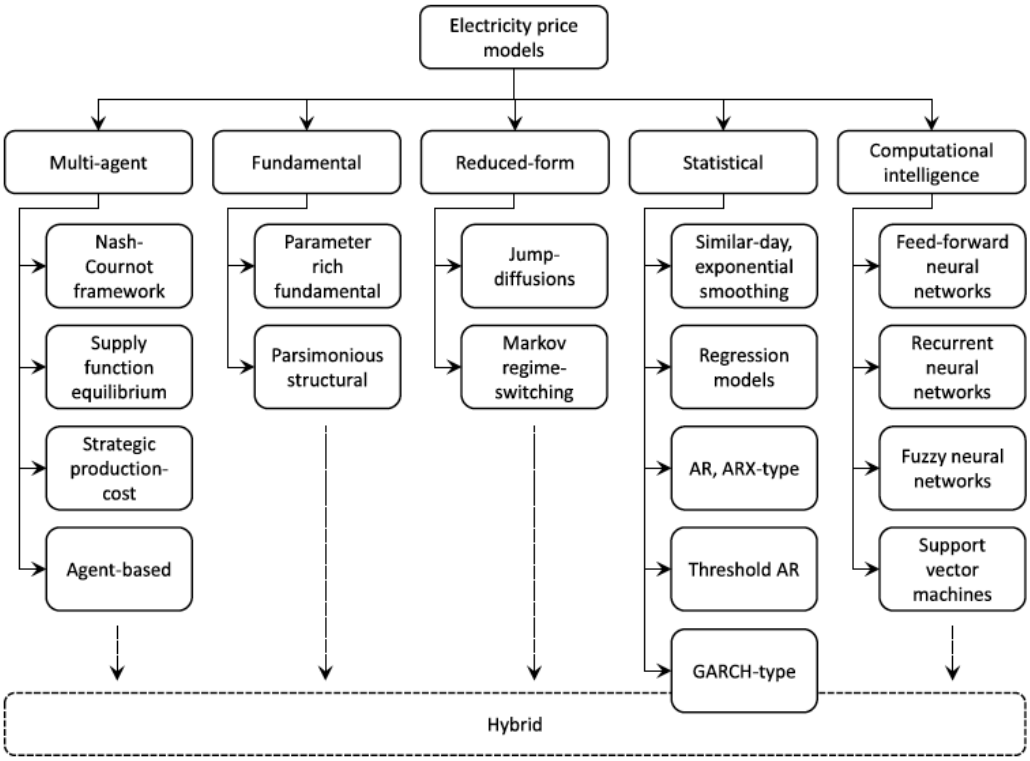

Figure 2.3: A taxonomy of Electricity Spot Price Modeling approaches [Wer14].

In [Wer14], Rafal Weron proposes a review for the electricity price forecasting in terms of the complexity of the available methods, their strength, and weaknesses. The paper explains in details the electricity price spots and how is it done practically in terms of day-ahead auctions. The paper tries to make it easier for the researchers and encourage them to work on electricity price forecasting. In Figure 2.3 the paper shows the approaches that are used in the paper to forecast electricity spot prices are shown.

Another review paper for the electricity price forecasting is done in [ASK09]. Moreover, stochastic time series, causal models and artificial intelligence based models are the main discussed techniques in the paper. A comparison between different models is done as a result of the evaluation of the paper. The factors that can affect the electricity price forecasting are discussed in details in the paper and mainly are divided into 5 main categories: market characteristics, nonstrategic uncertainties, other stochastic uncertainties, behavior indices, and temporal effects.

2.2.3 Carbon Dioxide Emissions Intensity Forecasting

The CO_2 Emissions Forecasting (CEF) can play as a key factor in most of the environmental related disciplines. Unfortunately, little research has been done on CO_2 emissions forecasting (CEF). In particular, for the caused CO_2 emissions by the electricity generation.

In [KP17], a CEF is done in Zambia using Waikato Environment for Knowledge Analysis (WEKA) which is a data mining tool. Sequential minimal optimization regression (SMOreg) is used in the analysis for the data because it supports linear and nonlinear regression models. The CO_2 emissions forecasting is done in a yearly manner and up to 2021. The paper uses historical data from 1964 to 2016. The study divides the data into 5 different sectors according to the source of emissions, which are, transport, manufacturing, and construction, electricity and heat production, residential, commercial and public services sectors. The paper discusses some global policies for CO_2 emissions and local policies in Zambia. A conclusion of the paper would be that, the researchers show the need for implementing new policies to control the CO_2 emissions in Zambia, especially the emissions from the transport sector.

Another CEF and modeling is done in [ST08]. The United States is emitting approximately 5.9 billion metric tons of carbon dioxide in the atmosphere in 2008 which is ranking the United States as the second most CO_2 emitting country in the world after China in the first place [ST08]. The paper develops two different statistical models to predict carbon dioxide emissions and another model to predict the carbon dioxide in the atmosphere. The two models use a monthly historical data from 1981 to 2003 for the carbon dioxide emissions and another monthly set of data to predict the CO_2 concentrations in the atmosphere from 1965 to 2004. The developed models are multiplicative seasonal auto-regressive integrated moving average, ARIMA models. The prediction is done for 12 months and the researches admit that the result is encouraging.

According to [MN11], the CO_2 emissions from fossil fuel combustion accounts for about 60% of the global total. Therefore, the authors of the paper propose three models for CO_2 emissions from fossil fuel combustion using the logistic equation with three different algorithms. The researchers noticed that except for a few countries, the main figures of CO_2 emission from fossil fuel combustion in other countries are S-shaped curves. The S-shaped curve is a curve that shows an initial rapid growth (exponential growth) and then slows down as the carrying capacity is reached. Hence, three different algorithms of the logistic equation are used to model the emissions considering different emission characteristics of different industries. The researches claimed that multiple linear regression models (e.g., ordinary least square and partial least square) are not suitable to be used for predicting since the independent variables of the emissions are not clear which makes the time the only available independent variable. Hence, the researchers prefer to use the logistic equations which makes the prediction only valid for the countries with S-shaped curves for the emissions. As an evaluation, the researchers used the Mean Absolute Percentage Error (MAPE) to evaluate the models. As a conclusion, the researchers propose a comparison between a linear regression model that is developed in [KB10] and the results were in the side of logistic equations as shown in Table 2.2. From Table 2.2, we conclude that the model of the logistic equation has better results and precision than the linear regression model in all of the given years.

	Real emissions	Logistic equation	Relative error (%)	Linear regression	Relative error (%)
1998	10.392	10.11	-2.71	8.317	-19.97
1999	0.422	10.576	1.48	9.623	-7.67
2000	10.586	11.116	5.01	10.929	3.24
2001	10.791	11.748	8.87	12.235	13.38
2002	11.604	12.496	7.69	13.542	16.7
2003	13.556	13.395	-1.19	14.848	9.53
2004	15.941	14.494	-9.08	16.154	1.34
2005	17.743	15.870	-10.56	17.46	-1.59
2006	19.693	17.637	-10.44	18.766	-4.71
2007	21.219	19.991	-5.79	20.073	-5.4

Table 2.2: Comparison of different methods in forecasting the CO_2 emissions from fossil fuel combustion [MN11].

Another hybrid model for forecasting energy-related CO_2 emissions is proposed in [14] and it is suitable for developing countries whose CO_2 emissions have not reached the inflection point of the long-term S-shaped curve and usually present short-term linear or approximately exponential trends [14]. The researchers combine the exponential equation and a linear equation to form a hybrid forecasting equation combined and used data from 1992 to 2011 in China to apply the model. As a conclusion, the researchers claim that the hybrid model can respond more quickly to changes in emission trends than the two models because can of the specialized equation structure.

In [WLL+15], a novel multi-variable grey model is used for modeling and forecasting of CO_2 emissions in the BRICS (Brazil, Russia, India, China, and South Africa) countries. The model uses electricity consumption and economic growth as independent variables. The study discusses the relationship between energy consumption, urban population, economic growth and CO_2 emissions in all of the five countries. The relationship between two variables is not always the same in all of the countries e.g, the economic growth has a decreasing effect on the CO_2 emissions in Brazil and Russia and has an increasing effect in India, China, and South Africa, while the urban population and energy consumption have an increasing effect on the CO_2 emissions in all of the countries. The researchers have used short-term data. Hence, grey prediction modeling was their choice since it requires a limited amount of data to acquire the behaviors of unknown systems.

3 Data Engineering

Nowadays data play center stage. On the one hand, development in automation and technologies allows for the collection and storing of a large amount of data. On the other hand, there is the need for elaborating, analyzing, and filtering those data. Computer science, mathematics, and statistics are the main fields involved in the process of manipulation and understanding of data, with machine learning being the main technology. Historical data are old collected data about a specific subject in a specified period of time. Historical data are important in recognizing any patterns in a series of events. In the context of machine learning, historical data are the trained data that the model needs to discover the patterns. Hence, historical data play an important role in our process of implementing a model to predict the CO_2-equivalent intensity factor. Therefore, to be able to predict the emissions hourly in Germany, we need to clean historical data about the emissions in Germany. That pushes to apply the whole process of data engineering pipeline as shown in Figure 3.1 before starting the creation of the model to make sure that clean integrated data are obtained. The typical data engineering pipeline in Figure 3.1 was not fitting the targeted process perfectly. Therefore, some changes to the data engineering pipeline were applied to cope with the target of the task. As shown in Figure 3.2 the data cleaning step is done twice since some

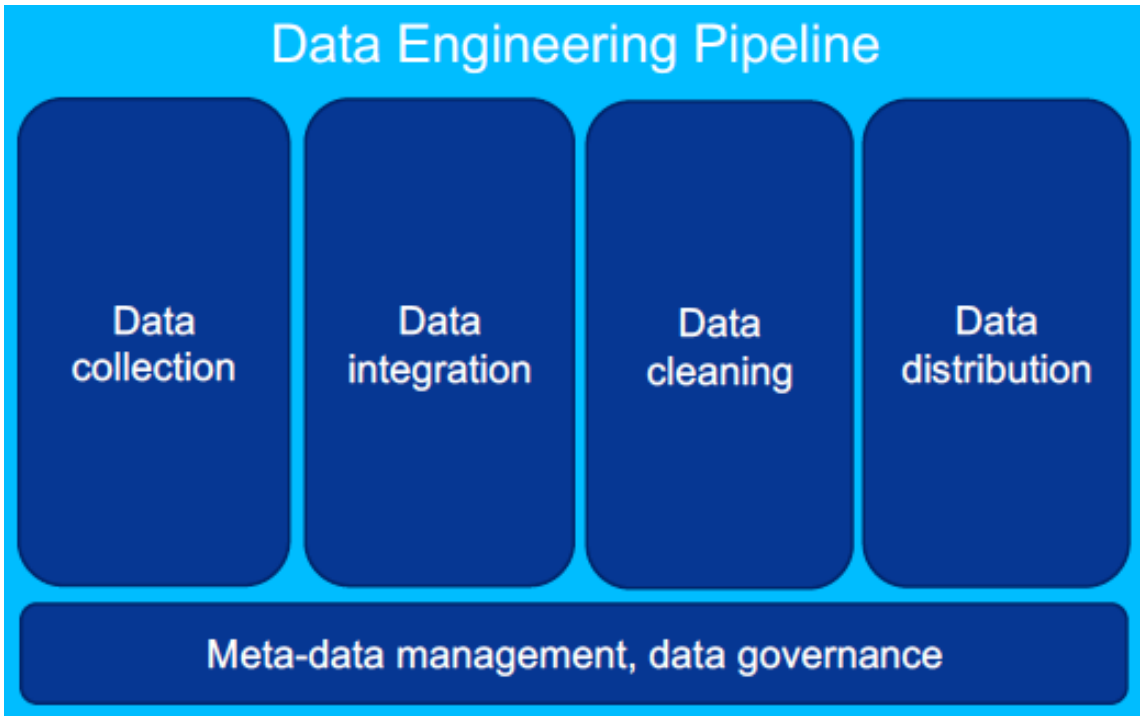

Figure 3.1: The Data Engineering Pipeline. The source: Professor, Melanie Herschel | Data Engineering | summer term 2017 | University of Stuttgart.

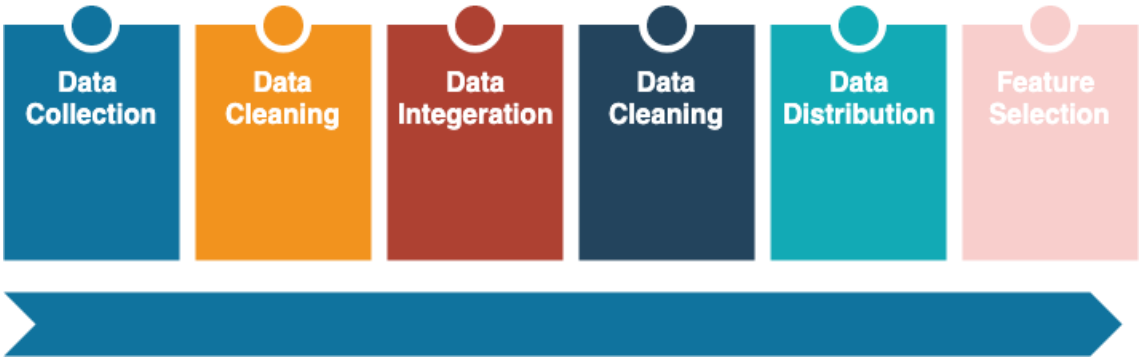

Figure 3.2: The New Data Engineering Pipeline.

of the data need to be cleaned directly after collecting it and another data cleaning process is done after the integration. A new step is added which is the feature selection. In the feature selection step, the final set of the independent variables (features) will be produced according to the designed feature selection criteria. This step is important to improve the performance of the model. More information about feature selection criteria is explained in Section 3.6. The chapter starts with a setup section (Section 3.1) that explains all the used technologies through the implementation procedure, not only the used technologies in the data engineering procedure but also the used technologies in Chapter 4, Chapter 6, and, Chapter 5.

Technology	Use
Python 3.6	The main programming language and used for scripting (data engineering), modeling, forecasting, visualization and creating the service-oriented architecture.
scikit-learn	Python library used for data engineering tasks and modeling.
Django Rest-Framework	Python rest framework used for building the service-oriented architecture.
Pandas	Python library used for data engineering tasks and modeling.
Numpy	Python library used for data engineering tasks and modeling.
seaborn	Python library used for data engineering tasks and modeling.
math	Python library used for mathematical calculations.
matplotlib.pyplot	Python library used for data visualization.
Jupyter Notebook	Web Application used as IDE to write Python code and supports visualization.
Latex	Used to write the document of the thesis.
EntsoePandasClient	Python wrapper for the ENTSO-E API.
forecastio	Python wrapper for the Darksky API.
Overleaf	Web application to use latex in writing the thesis.
Postman	API Development Environment used API client to send APIs requests.

Table 3.1: Technologies and their main use

3.1 Set up

This section declares the technologies, frameworks and applications that are used to produce the service oriented architecture that forecasts the CO_2-equivalent intensity factor in Germany. Table 3.1 shows them with an explanation for their use.

3.2 Data Collection

Data collection is the first step of the data engineering pipeline in Figure 3.2. As a result of this section, historical data of dependent and independent variables will be collected and ready for the data cleaning step in the data engineering pipeline. Before starting the process of the data collection, the dependent and independent variables should be declared. First of all, the dependent variable in the machine learning context is the variable that needs to be found (forecasted) which, in this case, is the CO_2-equivalent intensity factor. The dependent variable depends on other variables in order to be forecasted, these variables are called independent variables or features. Since the CO_2-equivalent intensity factor that is emitted by the electricity generation is being forecasted, then it is clear that the electricity generation is the main independent variable in this case. Furthermore, other independent variables can affect the model and increase the precision of the model. For instance, there may be a relationship between the wind speed in Germany and the CO_2-equivalent intensity factor. To make it clear and by substituting the main explained regression equation Equation (1.1). Y is the output of the equation which is the dependent variable (CO_2-equivalent intensity factor), while X is the input which are the independent variables (electricity generation, wind speed, temperature, etc.). F is the regression function that specifies the relation between the dependent and the independent variables. The ultimate goal of the function F is to predict Y with the maximum accuracy for a given new input X. Different regression models are used as regression functions and explained in Chapter 4.

3.2.1 Dependent Variable collection

As mentioned previously, the dependent variable is the CO_2-equivalent intensity factor. There is no data source that has hourly based historical data regarding CO_2-equivalent intensity factors in Germany. Therefore, it needs to be computed using the available data of the electricity generation in Germany in [ENTSO19]. The ENTSO-E platform provides an API to request the electricity generation data per production type. Knowing the generated electricity from each production type, the suggested formula in [FA18] can be followed to calculate the CO_2-equivalent intensity factor:

$$EF_{e,t} = \frac{\sum_{pp} E_{t,pp} \cdot EF_{t,pp}}{\sum_{pp} E_{t,pp}}$$

where $E_{t,pp}$, is the energy produced by power plant pp in time step t, and $EF_{t,pp}$ is the lifecycle GHG emission factor of power plant pp as shown in Table 3.2.

The historical data of the electricity generation per power plant for Germany are available starting from 2015 in [ENTSO19]. The ENTSO-E transparency platform provides a RESTful API to retrieve the data. The available data for the generation per production type is in an every-15-minutes manner and expressed in terms of MW data as shown in their GUI (see Figure 3.3). Noticeably, ENTSO-E shows 16 different production types for Germany from the year 2015 through the end of 2016, then it adds fossil oil as the 17_{th} production type in 2018 through May-2019. The production types in Germany are shown as well in Table 3.2. The data are retrieved using Python and converted into dataframes using the library pandas for the ease of use. The applied data cleaning step to the retrieved data is explained in Section 3.3 and as a result, the step produces hourly based values in kWh for each production type. By substituting the generation values per production type in

Biomass	Solar PV	Wind onshore	Wind offshore	Geothermal
71	43	8	9	45
Nuclear	**Lignite**	**Coal**	**Coal-derived gas**	**Gas**
11	820	800	800	400
Run-of-the-river	**Reservoir**	**Other renewable**	**Waste**	**Other**
4	9	71	690	247
Pumped-Storage	**Oil**			
43	520			

Table 3.2: Life-cycle GHG emission for power plants, expressed in gCO_2eq/kWh from the source [FA18].

MTU	Biomass	Fossil Brown coal/Lignite	Fossil Coal-derived gas	Fossil Gas	Fossil Hard coal	Fossil Oil
	Actual Aggregated	Actual Aggregated	Actual Aggregated	Actual Aggregated	Actual Aggregated	Actual Aggregated
	[MW]	[MW]	[MW]	[MW]	[MW]	[MW]
	D	D	D	D	D	D
00:00 - 00:15	3937	14664	0	328	14152	N/A
00:15 - 00:30	3924	14604	0	325	13803	N/A
00:30 - 00:45	3912	14606	0	324	13129	N/A
00:45 - 01:00	3917	14608	0	323	12498	N/A
01:00 - 01:15	3925	14649	0	324	11882	N/A
01:15 - 01:30	3920	14625	0	324	11361	N/A
01:30 - 01:45	3921	14617	0	323	11331	N/A

Figure 3.3: The energy per production GUI in ENTSO-E Transparency Platform [ENTSO19].

kWh in the given equation after being cleaned and converted, the CO_2-equivalent intensity factor is calculated in gCO_2eq/kWh. Noticeably, the CO_2-equivalent intensity factor in Germany varies between 111.7 to 552.2 gCO_2eq/kWh between the years 2015 to 2018 (see Figure 3.4).

3.2.2 Independent Variables collection

In this subsection the process of collecting the data of the independent variables is explained and as a result of this section a potential set of independent variables data will be produced. Choosing further independent variables rather than electricity generation variables is done through a criteria. The goal of this criteria is to make sure that the chosen independent variables can be used to improve the forecasting of the dependent variable(CO_2-equivalent intensity factor). There are some general constraints that any independent variable, including the electricity generation, should pass before the procedure of the data collection starts. These constraints are:

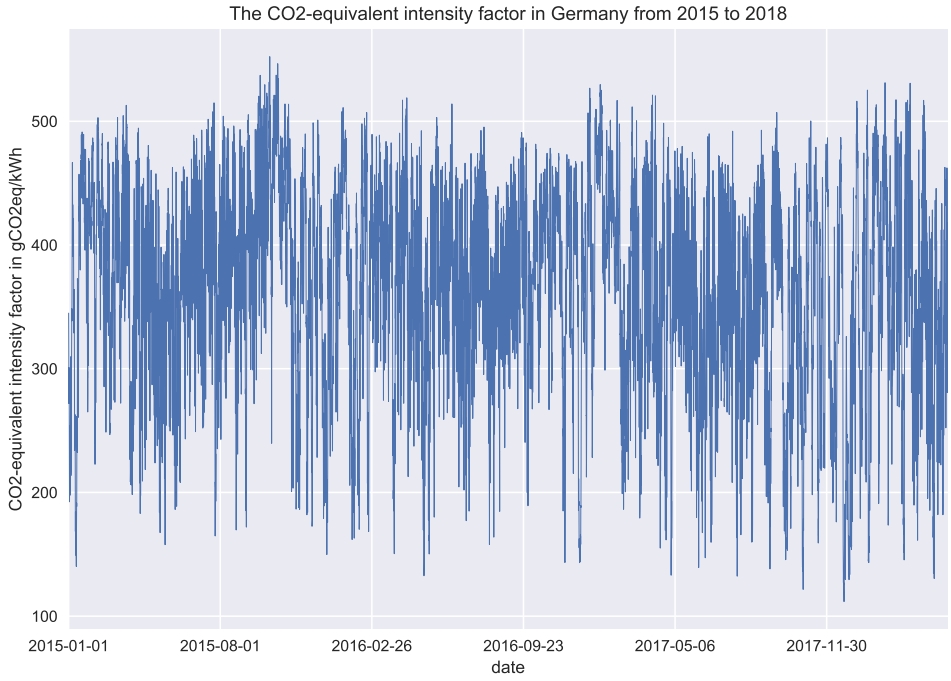

Figure 3.4: The CO_2-equivalent intensity factor in Germany from 2015 to 2018.

1. The independent variable should have available historical values that can be used to train the model.
2. The independent variable should have future forecasted values at least for one day in advance. Clearly, if forecasting the CO_2-equivalent intensity factor in the future is needed, then each variable that is used in training the model as input should be available as an input for the future at least for one day.
3. The independent variable's historical and forecasted values should be available in an hourly-based manner or less than hourly (e.g., every-half-an-hour, minutely or every-15-minutes) to be able to forecast the CO_2-equivalent intensity factor in an hourly-based manner.

These constraints should be kept in mind before starting collecting any independent variables.

The electricity generation in Germany is the first potential independent variable. By applying the constraints to the electricity generation independent variable with respect to the order and answering with a pass or fail for each constraint, the result is:

1. Pass. The historical data are available on ENTSO-E API
2. Pass. The forecasting data are available on ENTSO-E API, keeping in mind that the forecasted data are for the total electricity generation and not by production type as the historical data.
3. Pass. The historical data are available in an every-15-minute manner. The forecasted data are hourly based.

Listing 3.1 Python code using ENTSO-E API to collect data for electricty generation per production type in Germany from 1-1-2015 to 31-12-2017.

```
from entsoe import EntsoePandasClient
import pandas as pd
# query the data from 1-1-2015 to 31-12-2017
  client = EntsoePandasClient(api_key='your-api-key')
  start = pd.Timestamp('20150101', tz='Europe/Berlin')
  end = pd.Timestamp('20180101', tz='Europe/Berlin')
  country_code = 'DE'
  df = client.query_generation(country_code, start=start, end=end, psr_type=None)
```

The electricity generation variable passed all the constraints. The available forecasted data from ENTSO-E platform are for the total electricity generation while the historical data are divided per generation type. Therefore, the historical data are summed and used as a total electricity generation independent variable to train the model. The historical data are in an every-15-minute manner while the forecasted data are in an hourly manner. Therefore, the historical data are cleaned and converted to hourly based data in the cleaning step Section 3.3. The ENTSO-E API provides forecasting data for the electricity generation in Germany for one day ahead. Therefore, According to the available data, the service to provide forecasted data is limited to forecast the independent variable (CO_2-equivalent intensity factor) for one day ahead. Collecting the data is done using a wrapper for the available get requests that ENTSO-E API provides. In the request, the type of requested data, the needed period of time, the country, time zone, and the API key should be specified. Listing 3.1 is an example of how to collect data using the ENTSO-E API.

The ENTSO-E provides one-day-ahead forecasting for the electricity generation for the wind Onshore, wind Offshore and solar production types. Clearly, using the historical data of the electricity generation from these production types can be helpful for the model. By applying the constraints to the new independent variables (electricity generation from wind Onshore, wind Offshore and solar) with respect to the order and answering with a pass or fail for each constraint, the result is:

1. Pass. The historical data are available on ENTSO-E API
2. Pass. The forecasting data are available on the ENTSO-E API.
3. Pass. The historical data are available in an every-15-minute manner. The forecasted data are every-15-minutes.

Therefore, three new independent variables (electricity generation from wind Onshore, wind Offshore and solar) are added to the potential set of independent variables. The data collection is done the same way explained in Listing 3.1.

The availability of historical and forecasted data for the weather conditions and the time increases the chance of using both of them as independent variables. Timing can be represented in different forms (e.g, year, month, week, day or hour). Weather conditions can be expressed in terms of temperature, humidity, wind speed or wind direction. Therefore, finding an API that provides hourly historical and forecasted data will lead to passing all the discussed general constraints. Therefore, Darksky API is used to provide different independent variable in terms of weather conditions and its

timing [darksky19]. Darksky provides a restful API to request the weather historical and forecasting data. The date, longitude, latitude and the API key should be specified in the request. The German longitude and latitude are provided from [GPS19] as 51.1657 North for latitude and 10.018343 East for longitude. A date range function is used to give the ability to request data in a period of time. Since the available data of the electricity generation are only starting from 2015 and till the end of 2018, the weather data should be requested for the same period. A sample of used Python scripts to query the weather data is shown in Listing 3.2.

The new weather conditions independent variables from Darksky are added to the potential list of independent variables and as a result, the potential set of the independent variables that are collected in the step of data collection are shown in Table 3.3

Listing 3.2 Python code to query weather historical data of Germany from the year 1/1/2015 till 31/12/2016.

```
  # importing the needed libraries and adding the values of the const Variables
import forecastio as fc
import pandas as pd
import json
from pandas.io.json import json_normalize
from datetime import datetime
from datetime import timedelta, date
api_key = "your-api-key"
lat = 51.1657
lng = 10.018343


# function to go through a range of dates so i can use it to go in loop
# because for the darksky Api you can only request one day per each request
# and up to 1000 requests per day
# The requestes have been done in two days to get four years
def daterange(start_date, end_date):
    for n in range(int ((end_date - start_date).days)):
        yield start_date + timedelta(n)


#2015 and 2016
#specifying the range of the dates to get the weather data
#the response is json and its not only the hourly weather data
#there is more than data block and data points in the response
#we are interested in the hourly response which is in the datablock hourly.
start_date = datetime(2015,1, 1)
end_date = datetime(2017,1,1)
daydf = pd.DataFrame()
for single_date in daterange(start_date, end_date):
    forecast = fc.load_forecast(api_key, lat, lng,time = single_date)
    byHour = forecast.hourly()
    x = 0
    for hourlyData in byHour.data:
        # creating a new dataframe for each day and then combine them with the daydf which
            is the main one
        # adding the hour and the date to the dataframe because they were not added
        df = pd.DataFrame.from_dict(hourlyData.d, orient='index')
        df = df.T
        df['date'] = '' + str(single_date.date())
        df['hour'] = x
        daydf = daydf.append(df,ignore_index=True)
        x = x+1
daydf = daydf.set_index(['date', 'hour'])
daydf
```

Independent variable	Explanation
Generation	The total electricity generation in Germany.
Solar	The electricity generation from the Solar power plants in Germany.
Wind Offshore	The electricity generation from the Wind Offshore power plants in Germany.
Wind Onshore	The electricity generation from the Wind Onshore power plants in Germany.
weekDay	The weekday starting from 0 for Monday to 6 for Sunday.
monthDay	The day of the month from the calendar.
month	The present month from 1 to 12.
apparentTemperature	The apparent (or "feels like") temperature in Celsius.
dewPoint	The dew point in degrees Fahrenheit.
humidity	The relative humidity, between 0 and 1, inclusive.
precipAccumulation	The amount of snowfall accumulation expected to occur, in inches. (If no snowfall is expected, this property will not be defined.)
precipIntensity	The intensity (in inches of liquid water per hour) of precipitation occurring at the given time. This value is conditional on probability (that is, assuming any precipitation occurs at all).
precipProbability	The probability of precipitation occurring, between 0 and 1, inclusive.
temperature	The air temperature in Celsius
windBearing	The direction that the wind is coming from in degrees, with true north at 0° and progressing clockwise. (If windSpeed is zero, then this value will not be defined.)
windGust	The wind gust speed in kilometer per hour.
windSpeed	The wind speed in kilometer per hour.
hour	The hour is starting from 0 to 24.
uvIndex	The UV index.
precipType	The type of precipitation occurring at the given time. If defined, this property will have one of the following values:'rain', 'snow', or 'sleet' (which refers to each of freezing rain, ice pellets, and "wintery mix"). (If precipIntensity is zero, then this property will not be defined. Additionally, due to the lack of data in our sources, historical precipType information is usually estimated, rather than observed.)

Table 3.3: The potential set of the chosen independent variables after the step of data collection

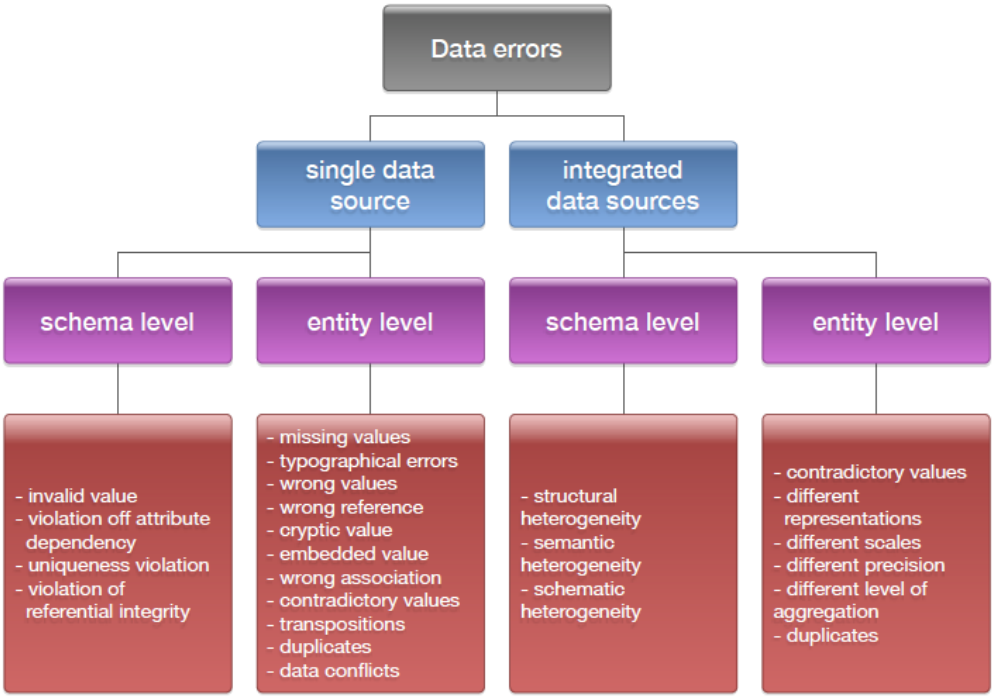

Figure 3.5: The levels of data cleaning. The source: Melanie Herschel | Data Engineering | Summer term 2017

3.3 Data Cleaning

Following the introduced data engineering pipeline, the data cleaning step is directly the second step after the data collection step and furthermore, another data cleaning step is done in the fourth step after data integration (see Figure 3.2). Data cleaning is an important step to improve the quality of the data after collecting it from the data sources. The need to apply data cleaning directly to the collected data is the reason to apply this step two times in the introduced data engineering pipeline. Data cleaning is the solution that handles all types of data errors as shown in Figure 3.5. The data errors are divided according to the type of the data source into two types (single data source and integrated data sources). Single data source expresses the fact that the data are accumulated from one data source, while integrated data sources express the accumulation of data from different sources and integrating them. Both single data source and integrated data sources can have schema level data errors and entity level data errors. Schema level data errors occur when there is a violation of the data structure or order. For instance, having a value of ”hot” for the temperature is a schema error, if the temperature is expressed in Celsius as a real number in the data structure. Entity-level data errors occur when there is a violation of the business logic (e.g., missing values or wrong values). In the first data cleaning step the single data source errors are handled, while in the second data cleaning step the integrated data sources errors are handled. In this section, the data cleaning procedure is explained as a solution for all the levels of data errors.

3.3.1 Single Data Source Cleaning

In this subsection, the data cleaning level is completed in a single data source level. The data are collected from two different data sources (ENTSO-E APY and Darksky API). Therefore, the single data source cleaning is divided according to these two different data sources.

3.3.1.1 Electricity Generation - ENTSO-E API Data Source

In this subsection, the data cleaning step is applied on the collected data from the ENTSO-E API including the dependent variable (CO_2-equivalent intensity factor) since it is calculated from the electricity generation data that are provided by ENTSO-E API.

As explained in the data collection step Section 3.2, the CO_2-equivalent intensity factor is calculated and not collected using the provided Equation (3.1). To substitute in Equation (3.1), the collected electricity generation per production type data should be converted from MW to kWh in an hourly manner. Therefore, a data cleaning step is needed to compute the CO_2-equivalent intensity factor in gCO_2eq/kWh. Figure 3.6 shows an example to simplify the steps of calculating the CO_2-equivalent intensity factor. Pandas dataframes are used for this task from Python. After this step, the historical data for electricity generation per production type is converted to kWh in an hourly manner and the dependent variable is calculated.

After calculating the dependent variable, some independent variables need to be created as well (e.g., weekDay, monthDay, hour and Month). Therefore, using pandas functions, the weekDay, monthDay, hour and Month are added as independent variables and, furthermore, multiple indexing for the collected data is done using the date and the hour. The multiple indexes help to index each row exactly and nevertheless, plays a big role in the data integration step.

As shown in Table 3.3, the independent variable generation is the total electricity generation from all the production type, therefore, a summation for all the production types is done and as a result, a total electricity generation in kWh is produced. The electricity generation from solar, wind Onshore and wind Offshore are used as independent variables as well and, therefore, all the other production types are dropped as they are not needed for the next steps.

The missing values of the electricity generation per production type can be handled in different ways. One of which is using the mean value of the production type, or using the maximum value. In this case, the number of rows that have missing values is 400 out of 34000 approximately. The percentage of the missing values is not high and, therefore, for the quality of the model, the rows that have the missing values are dropped.

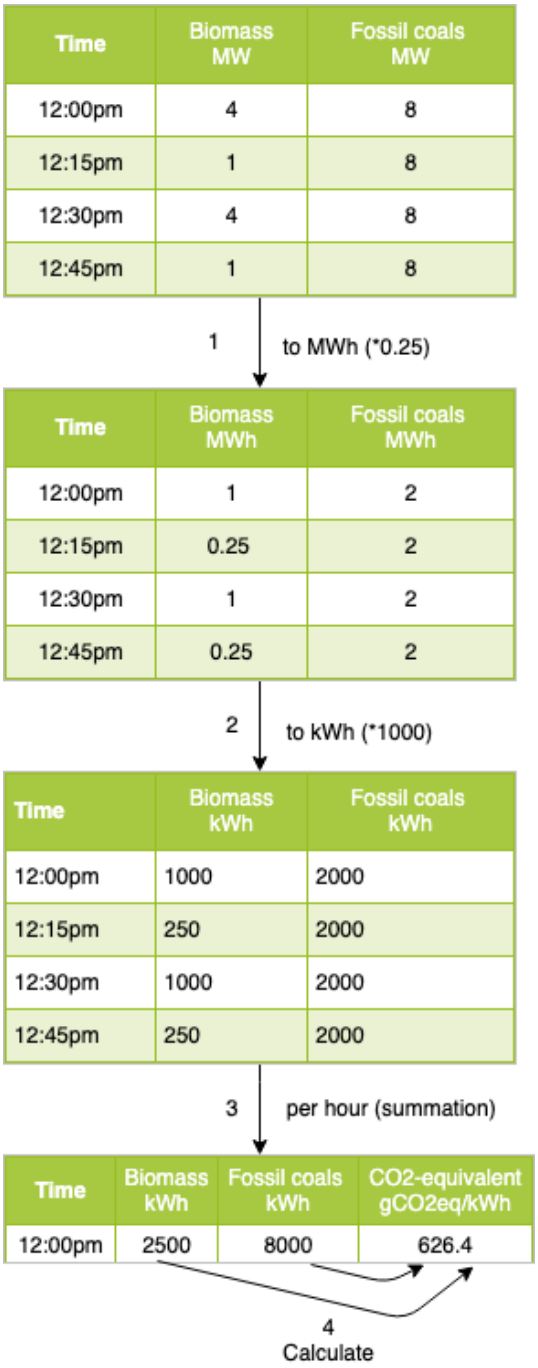

Figure 3.6: Steps of calculating CO_2-equivalent intensity factor after data collection step.

3.3.1.2 Weather Conditions - Darksky API Data Source

For the weather condition independent variables, multiple indexing using the date and hour is applied. The missing data in the weather condition data are handled either by a logical substituting or by dropping. The independent variables uvIndex and precipType are completely dropped because they are almost always not specified from the API between the period of 2014 to 2018. As the API declares, when the values for windBearing variable is not specified (missed), it means that there is no direction for the wind (no angle). Therefore, putting the value -1 for the missing values is an optimal solution to express the ”no angle wind” case knowing that the windBearing value is between 0 to 365. The Darksky API declares that if the value of precipAccumulation is missed, that means that there is no snowfall accumulation occurring. Therefore, setting the value of the missing values for precipAccumulation to 0 is an optimal solution to express no snowfall accumulation. The other rows with missing values are dropped to ensure the quality of the used data.

3.3.2 Integrated Data Sources Cleaning

After data integration, the entity level data error (different level of aggregation) is produced. Since the rows that are dropped in both of generation data and weather condition data are different, after integrating them some of the rows will still have non-specified (missing) values in either of them. Therefore, these rows are dropped to keep the final potential set with no missing values. After applying the data cleaning steps, the final size of the integrated data is reduced from 34000 to 28000 rows.

Listing 3.3 Integrating two different datasets from two different data sources using Python.

```
import pandas as pd
# Integerating both of the data from weather data source and electricty generation data
    source
#finalco2 is the final csv file of the cleaned data for electricity generation
#including the dependent variable (CO2)
gendf = pd.read_csv('finalCo2.csv')
#indexing the dataframe using multiple index (date,hour)
gendf = gendf.groupby(['date','hour']).sum()
#weather_final is the final csv of the cleaned data for weather conditions
weather_data = pd.read_csv('weather_final.csv')
#indexing the dataframe using multiple index (date,hour)
weather_data = weather_data.groupby(['date','hour']).sum()
result = pd.concat([gendf,weather_data], axis=1)
result
```

		Solar	Wind Offshore	Wind Onshore	weekDay	monthDay	month	Generation	Co2_emission	apparentTemperature
date	hour									
	0	0.0	516500.0	8128000.0	3.0	1.0	1.0	50247755.0	344.936967	2.67
	1	0.0	516250.0	8297500.0	3.0	1.0	1.0	48578755.0	332.111556	2.56
	2	0.0	514000.0	8540000.0	3.0	1.0	1.0	47864505.0	326.103077	2.34
	3	0.0	517750.0	8552000.0	3.0	1.0	1.0	46835005.0	322.628211	2.45
	4	0.0	519750.0	8643500.0	3.0	1.0	1.0	47032505.0	322.830702	2.34
	5	0.0	520000.0	8711750.0	3.0	1.0	1.0	46367255.0	312.030997	2.12
	6	0.0	521500.0	9167250.0	3.0	1.0	1.0	45306755.0	302.548202	2.12
	7	0.0	520250.0	9811000.0	3.0	1.0	1.0	45656255.0	296.505796	1.56
	8	53000.0	525250.0	9683000.0	3.0	1.0	1.0	46540005.0	298.498974	1.56

Figure 3.7: Subset of the final result after integrating all the independent and dependent variables using multiple index.

3.4 Data Integration

The data integration is done as the third step in the data engineering pipeline (see Figure 3.2). Since the data are collected from two different sources (ENTSO-E and Darksky), these data should be integrated. As a preparing step for data integration, in the data cleaning section, all the collected data are indexed using multiple indexing with the date and hour. This pre-step made the integration easier using Python for processing data integration. Listing 3.3 shows how pandas library from Python can help in integrating two different dataframes having the same indexing using axis 1. As a result of this step, a dataframe with all the independent variables and the dependent variable is created and shown in Figure 3.7. The data integration step produces a "different level of aggregation"data error. As shown in Figure 3.5, this data error is an entity level for integrated data sources. Therefore, there is another cleaning step after integration as explained in Section 3.3.2.

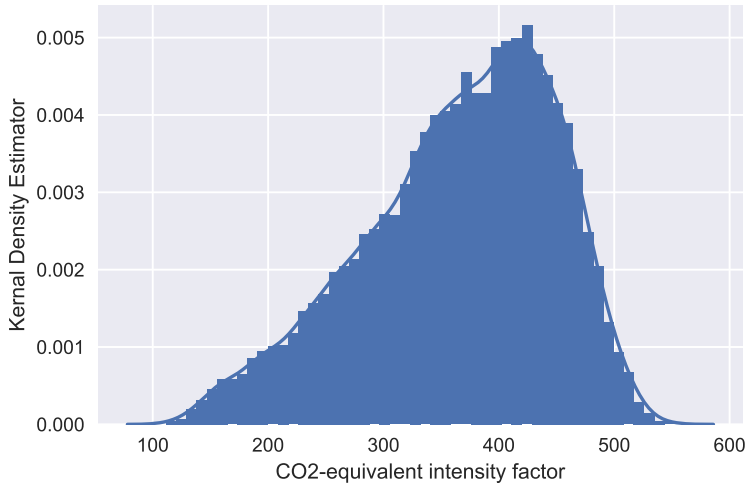

Figure 3.8: Univariate distribution for CO_2-equivalent intensity factor (2015-2018, Germany).

3.5 Data Distribution

The data distribution process is a pre-step for the feature selection process in the introduced data engineering pipeline Figure 3.2. Data distribution helps in getting an overview of the numerical data that are obtained. Since data distribution functions are used in modeling and forecasting as well as in the feature selection process, the data distribution step is defined as a continuous process in this project. One of the most well-known distributions is called the normal distribution, also known as the bell-shaped curve [Rum16]. The normal distribution is based on numerical data that are continuous. As an application of normal distribution for the dependent variable, a univariate distribution by drawing a histogram and fitting a kernel density estimate (KDE) as shown in Figure 3.8. In statistics, 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, using KDE diagrams for the CO_2-equivalent intensity factor values in Germany helps in getting an overview of the frequent intervals of values. Figure 3.8 shows high density of the CO_2-equivalent intensity factor values between 300 and 500 gCO_2eq/kWh for the historical data. Another data distribution type that is mainly used in the feature selection process is the correlation matrix. Figure 3.9 shows the correlations between the independent variables. In statistics, correlation measures the degree to which two variables move in relation to each other. Darker blue or red shows a positive and negative high correlation between two independent variables. More information about correlations between independent variables is discussed in Section 3.6.

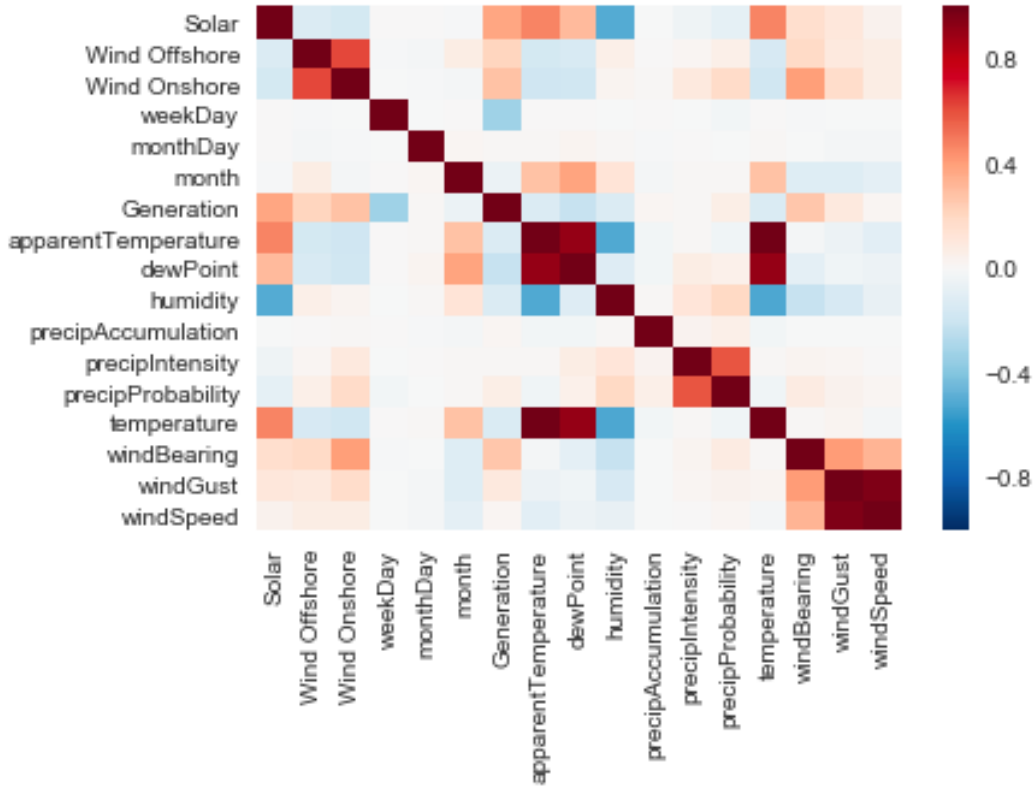

Figure 3.9: Correlation matrix for the potential set of the independent variables .

3.6 Feature Selection Criteria

The goal of these criteria is to make sure that the introduced independent variables will help our model to have a higher performance while keeping in mind avoiding the overfitting of the model. More information about the overfitting problem will be discussed in Chapter 4. In this section, the feature selection criteria are introduced and, afterwards, an application of these criteria in the potential independent variable set is done. As a result of this section, a final set of the independent variables (features) that will be used for modeling and forecasting is created.

3.6.1 Criteria

In machine learning and multiple regression, this procedure is known as feature selecting and sometimes dimensional reduction. Feature selecting is an important step in machine learning as it has a large direct impact on the performance of the model. There are four benefits of feature selecting for the model:

- Reduces overfitting.
- Improves Accuracy.
- Reduces training time.

- Reduces the complexity of a model and makes it easier to interpret.

In [elitedata19], deleting the redundant features is one of the solutions to reduce overfitting. Redundancy in this context expresses the features that are correlated to each other. Correlation is a statistical measure that indicates the extent to which two or more variables fluctuate together. A positive correlation indicates the extent to which those variables increase or decrease in parallel. A negative correlation indicates the extent to which one variable increases as the other decreases. Furthermore, it is clear that the accuracy of the model will improve if the misleading features (independent variables) are removed, which will reduce the training time as the used independent variables are less than the full potential set. Therefore, our feature selecting procedure is independent of any machine learning algorithm and can be summarized in the next steps:

- Removing the redundancy between the features (independent variables).
- Univariate feature selection.
- Recursive Feature elimination.

The problem of redundancy of features is called multicollinearity, which occurs when some of the features (independent variables) are correlated with each other. In another term, some of the features are linear combinations of others (or nearly so). The correlation matrix is the proposed solution to detect the multicollinearity as shown in Figure 3.9. Pearson correlation coefficient is used to build the correlation matrix in this project. Pearson correlation coefficient is s a measure of the linear correlation between two variables x and y using the following equation provided by [BCHC09]:

$$\frac{n\left(\sum xy\right) - \left(\sum x\right)\left(\sum y\right)}{\sqrt{\left(n\sum x^2 - \left(\sum x\right)^2\right)\left(n\sum y^2 - \left(\sum y\right)^2\right)}}$$

Where n is the sample size, x is the first independent variable and y is the second independent variable. The coefficient value can range between -1.00 and 1.00. If the coefficient value is in the negative range, then that means the relationship between the variables is negatively correlated, or as one value increases, the other decreases. If the value is in the positive range, then that means the relationship between the variables is positively correlated, or both values increase or decrease together [BCHC09].

The Univariate feature selection works by selecting the best features based on univariate statistical tests [PVG+11]. The univariate statistical test is a test that is done statistically individually between one of the independent variables and the dependent variable to find a relationship between them without considering the other independent variables. There is more than an algorithm that can apply that but, SelectKBest class from scikit-learn library in Python is used to rank the best features [PVG+11]. The used algorithm to rank the best features is f-regression. F-regression is testing to find the linear correlation individually between each feature (independent variable) and the target (dependent variable). The scikit-learn algorithm works in two steps. Firstly computing the correlation between each feature and the target using the following equation:

$$\frac{(X - mean(X)) \cdot (Y - mean(Y))}{std(X) \cdot std(Y)}$$

Solar	Wind Offshore	Wind Onshore	weekDay
month	Generation	apparentTemperature	dewPoint
humidity	precipAccumulation	monthDay	precipIntensity
precipProbability	temperature	windSpeed	windGust
windBearing			

Table 3.4: The final potential set of features (independent variables) before applying the feature selection step.

Where X is the feature and Y is the target. The second step is to convert the correlation into F-score and then into p-value [PVG+11]. Using the scoring of each feature, the most relevant features for the task can be found.

The last step in our feature selecting criteria is recursive feature elimination (RFE). A greedy search is performed in recursive feature elimination to find the best performing feature subset. It works iteratively by creating models with the full set of the features in the beginning and then determines the best or the worst performing feature. At each step, it removes the worst attribute remaining in the set. It then ranks the features based on the order of their elimination. RFE can perform in the worst case a greedy search for 2^N while N is the number of features[Daca18]. In the development, RFE class from the library scikit-learn is used to apply the algorithm.

3.6.2 Feature Selection Application

Before starting the application of feature selection, Table 3.4 shows the set of the features that are still available after applying the first 4 steps of the data engineering pipeline Figure 3.2.

Applying the discussed criteria starts by removing the redundancy between the features. Pearson correlation coefficient is used and the correlation matrix is computed as shown in Figure 3.9. Removing the features that have a high correlation coefficient is the goal of these steps. Specifying the threshold coefficient that the correlations should exceed depends on the data. [COdess19] specifies that a high degree of correlation is when the coefficient value lies between ± 0.50 and ± 1, then it is said to be a strong correlation. A moderate degree is when the value lies between ± 0.30 and ± 0.49, then it is said to be a medium correlation. A low degree is when the value lies below + .29, then it is said to be a small correlation. While in [BJ] a value of ± 0.5 is used as a correlation threshold and any correlation coefficient greater than + 0.5 or less than - 0.5 is considered as strong correlation. Some other projects used values of ± 0.9 as the correlation threshold [toSci19]. Therefore, the correlation coefficient threshold differs between ± 0.1 and ± 0.9. To determine the correlation threshold in this project, the application of the feature selection criteria with a threshold of ± 0.3 is employed and a linear regression model with the remaining features is created to be used as an evaluation method. The final set of the features after applying the criteria is (Solar, Wind Offshore, weekDay, Month, windSpeed, and PrecipIntensity). The created model has an R^2 of 0.3. While the correlation threshold increases from ± 0.3 to ± 0.4 and reapplying the same steps. The model will have an R^2 of 0.8. Furthermore, the Generation feature (total electricity generation) is added to the final features set after having a threshold of ± 0.4 and adding it to the set improved the performance of the model from having R^2 of 0.3 to 0.8. Therefore, a correlation threshold of ± 0.4 is used in the used feature selection criteria.

	Specs	Score
2	Wind Onshore	19191.503214
1	Wind Offshore	5012.718325
14	windBearing	2371.381445
0	Solar	1534.180597
9	humidity	797.753462
13	temperature	613.632463
7	apparentTemperature	548.878921
15	windGust	527.001858
3	weekDay	430.468267
8	dewPoint	262.147613
6	Generation	150.100441
12	precipProbability	142.808940
16	windSpeed	99.897046
5	month	48.944702
11	precipIntensity	48.414637
4	monthDay	1.144232
10	precipAccumulation	0.983864

Figure 3.10: The scoring of the potential set of features using scikit-learn library from Python.

The feature selection criteria are iteratively done, such that the three steps are not done only once but several times. For instance, Figure 3.9 shows a high correlation coefficient (0.9) between apparent temperature and dew point feature which is over the chosen threshold (± 0.4). Therefore, one of the features is to be removed. Choosing which feature of them to be removed is done using the second step in our feature selection criteria. Applying the univariate feature selection to the features to have scoring that indicates the significance of the features. Figure 3.10 shows an ordered list according to the ranking of each feature in our potential list. Since the apparent temperature has a higher scoring than the dew point, dew point feature is removed. The scoring differs in each set of features and it is not fixed. The same two steps are applied to all the features that have correlation coefficients more than the introduced threshold. Since the recursive feature elimination has an expensive calculation time that may last for days, its score was considered only for the last 7 features. After removing all the high correlations from the feature, a set of 7 features remains. Figure 3.11 shows the scoring of the remaining features. The last step in this criteria is to apply univariate feature selection and recursive feature elimination to find the top 6 features.

	Specs	Score
1	Wind Onshore	19191.503214
0	Solar	1534.180597
2	weekDay	430.468267
5	Generation	150.100441
6	windSpeed	99.897046
4	month	48.944702
3	monthDay	1.144232

Figure 3.11: The scoring of the no correlation set of features using scikit-learn library from Python.

	Solar	Wind Onshore	weekDay	month	Generation	windSpeed
Solar	1	-0.1602	0.00462446	-0.00953355	0.379625	0.0403435
Wind Onshore	-0.1602	1	-0.00301944	-0.0178085	0.284703	0.0777052
weekDay	0.00462446	-0.00301944	1	0.00429394	-0.327336	-0.00456519
month	-0.00953355	-0.0178085	0.00429394	1	-0.0572209	-0.0888884
Generation	0.379625	0.284703	-0.327336	-0.0572209	1	0.0192662
windSpeed	0.0403435	0.0777052	-0.00456519	-0.0888884	0.0192662	1

Figure 3.12: The correlation matrix of the final feature set before modelling.

As a final result of the introduced data engineering pipeline, the following set of features are considered, with a correlation matrix shown in Figure 3.12, to be used for modeling and forecasting:

- Solar.
- Wind Onshore.
- weekDay.
- month.
- Generation.
- windSpeed.

4 Modeling and Forecasting

In this chapter, different regression models are used to forecast the CO_2-equivalent intensity factor. The produced final set of independent variables from Chapter 3 as shown in Figure 3.12 is used as an input for the regression models to find the output (CO_2-equivalent intensity factor). The collected historical data for the independent and dependent variables are needed to train and evaluate the models. Therefore, the historical data are divided randomly into two datasets as shown in Figure 4.1.

Each dataset is divided vertically into inputs and target. The inputs are the columns of the independent variables data (Solar, Wind Onshore, weekDay, month, Generation, and, windSpeed), while the target is the column of the dependent variable data. The training dataset is the dataset that is used for fitting the models. Fitting the model in machine learning is the process of finding the model coefficients that minimize the loss function. The testing dataset is used to evaluate the models by using its input data to predict the output (dependent variable) then comparing the results with the test target data (dependent variable). The training dataset is used to tune the hyperparameters of the

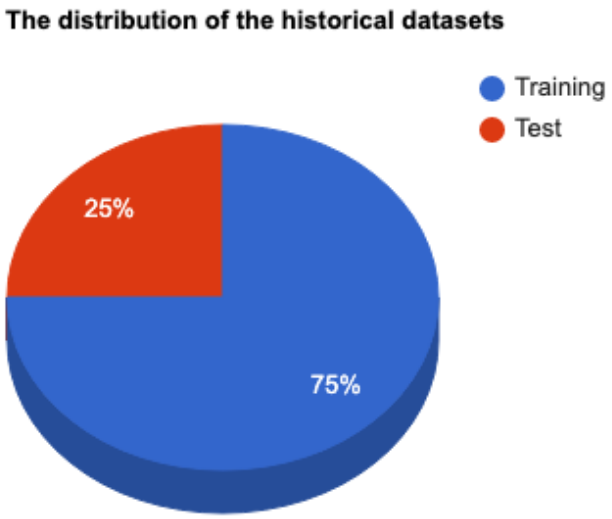

Figure 4.1: The distribution of the historical data.

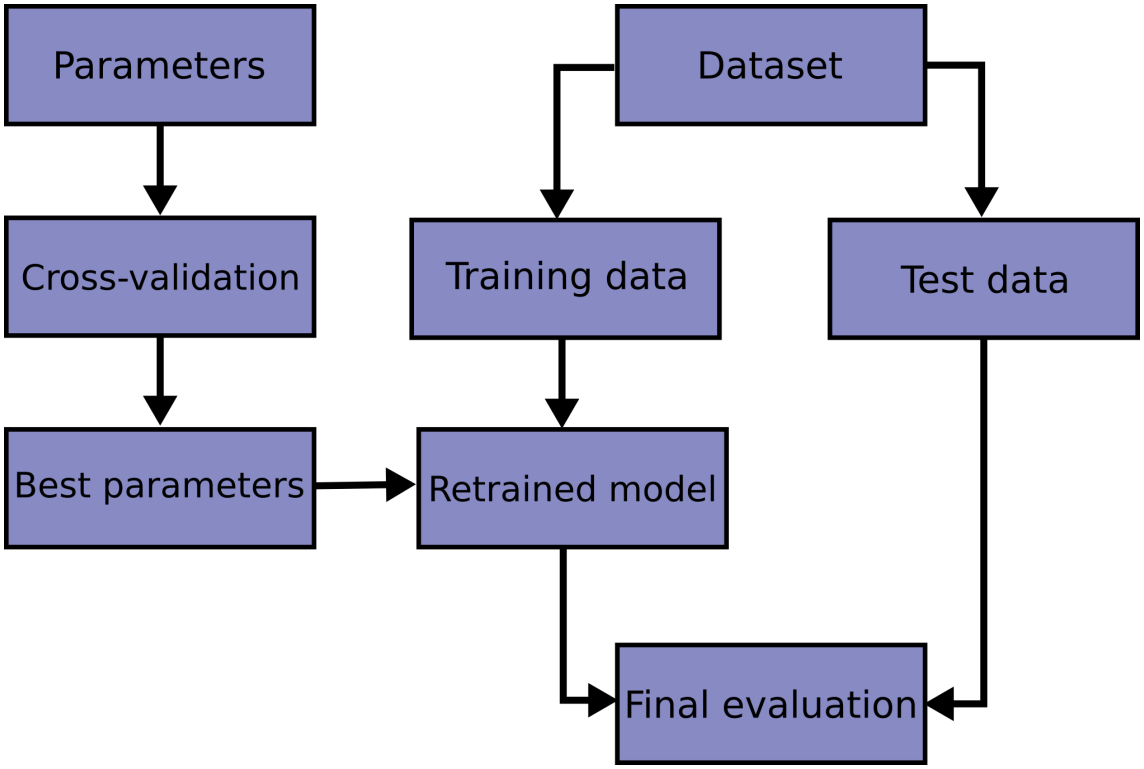

Figure 4.2: The flow chart shows a full procedure of modelling [PVG+11]

models using the cross-validation technique. Cross-validation technique is explained in details in Section 4.1.2.1. Hyperparameters are structural choices for each model family. Logistic regressions, Decision Trees, SVMs, and neural networks all are in fact different families of models. Each model family has a set of structural choices that must be made before actually fitting the model parameters [elitedata19]. For example, within the logistic regression family, separate models can be built using either L1 or L2 regularization penalties. Within the decision tree family, there are different models with different structural choices such as the depth of the tree, pruning thresholds, or even the splitting criteria. Hyperparameters are "higher-level"parameters that cannot be learned directly from the data using gradient descent or other optimization algorithms. They describe structural information about a model that must be decided before fitting model parameters. Cross-validation is the chosen technique to tune the hyperparameters as shown in Figure 4.2.

Generally, the steps of modeling and forecasting independently from the model family are the same. The result of each step may be different from model to model, but, the order of the steps is fixed in all the modeling techniques. The applied steps are:

1. Splitting the data randomly into training and test datasets.
2. Tuning the hyperparameters of the chosen model if needed.
3. Fitting the model with the training data.
4. Forecasting the output using the input of the test data.
5. Evaluating the model using different techniques as shown in Chapter 5.

Before applying the discussed steps, there is a main obstacle in modeling that should be considered and avoided in this process. This obstacle is known as overfitting. The next section discusses the problem of overfitting and how is this problem handled during and before modeling. Furthermore, the applied models are discussed individually in the following sections. Six regression models are used to forecast the CO_2-equivalent intensity factor. The models are chosen in a way that covers different regression families with different techniques for modeling. Models for Generalized linear models, Decision Trees, ensembles and nearest neighbors families are all applied in the modeling procedure. The modeling is done using scikit-learn library from Python [PVG+11]. Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

4.1 Overfitting

One of the most important fundamental notions of data science is that of overfitting and generalization [PF13]. In modeling, the interest is in finding patterns that generalize well for instances that have not yet been observed; which is the concept of generalization. Generalization refers to how well the concepts learned by a model of machine learning applied to specific examples that the model did not see when learning[Machle19]. Overfitting happens when the model tends to find chance occurrences in data that look like interesting patterns, but do not generalize for all the instances. In overfitting, the model learns the detail and noise from the training data to the level that can negatively impact the model's performance new data. This means that the model picks up and learns the noise or random fluctuations in the training data as concepts. The problem is that these concepts do not apply to new data and negatively impact the models' ability to generalize [Roh14]. Formally, overfitting is the tendency of data mining procedures to tailor models to the training data, at the expense of generalization to unseen data points [PF13].

4.1.1 Detecting Overfitting

A key challenge with overfitting, and with machine learning in general, is that it cannot be known how well the model will perform on new data until it is actually tested. Therefore, one of the best practices in detecting the overfitting is splitting the data into training and test datasets and keeping the test dataset untouched till the very end of modeling [PF13]. Having a holdout dataset gives the opportunity to test the model in a new dataset and by comparing the performance of the model with the training dataset and the new (test) dataset, the overfitting can be detected. For example, if model saw 99% accuracy on the training set but only 55% accuracy on the test set, it would be a big red flag. Training data evaluation does not provide an evaluation of how well the model generalizes to unseen cases [PF13]. While splitting the datasets is an appropriate way to discover overfitting, but, it is just a single estimate, it could have been just one lucky (or unlucky) selection of training and test data. It is therefore not always an easy task to detect the overfitting.

The more complex the model is, the more subject it is for overfitting [PF13]. Model complexity is a bit subjective and can be characterized by many things. Model complexity often refers in machine learning to the number of features or terms included in a given predictive model, and whether the selected model is linear, nonlinear, etc. It can also refer to the complexity of algorithmic

learning or computational complexity. Therefore, trying models with different complexity is often recommended to detect the overfitting. If two models have comparable performances, then usually the simpler one should be chosen [elitedata19].

4.1.2 Preventing Overfitting

Detecting overfitting is useful, but it does not solve the problem. There are different available options to avoid overfitting. As previously explained in Section 3.6, one of the goals of the introduced feature selection criteria is to avoid overfitting. Selecting the proper features for modeling is a key step in removing the noise and make it clear for the model to focus on finding the interesting patterns. Hence, the feature selection criteria are the first solution to avoid overfitting. Moreover, the criteria in avoiding overfitting rely heavily on cross-validation technique. Therefore, the next subsection explains our criteria based on cross-validation to avoid overfitting.

4.1.2.1 K-fold Cross-validation

Cross-validation plays more than a role in the modeling procedure, not only to avoid overfitting but also to tune the hyperparameters is the cross-validation used. Cross-validation is a powerful preventative measure against overfitting. Briefly, in k-fold cross-validation, the training dataset is partitioned into k subsets, called folds. Then, iteratively train the algorithm on k-1 folds while using the remaining fold as the test set (called the “holdout fold”). The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. Applying the same technique for all sets of hyperparameters, the set with the best performance is chosen. This approach can be computationally expensive but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a major advantage in problems such as inverse inference where the number of samples is very small [PVG+11]. Using this technique is allowing us to use the training dataset to tune the hyperparameters while keeping the testing dataset truly unseen for selecting the final model. Hyperparameters have an influence on the complexity of the model. As previously explained, the more complex the model is, the more subject it is for overfitting [PF13]. Models will be better if they fit the data better, but they will also be better if they are simpler. This general methodology is called regularization, a term that is heard often in data science discussions [PF13]. Using cross-validation helps in finding the hyperparameters set with improved performance as well as making the model simpler by finding the best hyperparameters set. Figure 4.3 shows the flow chart of the grid search technique (GridSearchCV) that the library scikit-learn provides to apply the cross-validation iteratively with different sets of hyperparameters to find the best performance set. In Figure 4.3, 5-fold cross-validation is used by dividing the training data into 5 folds.

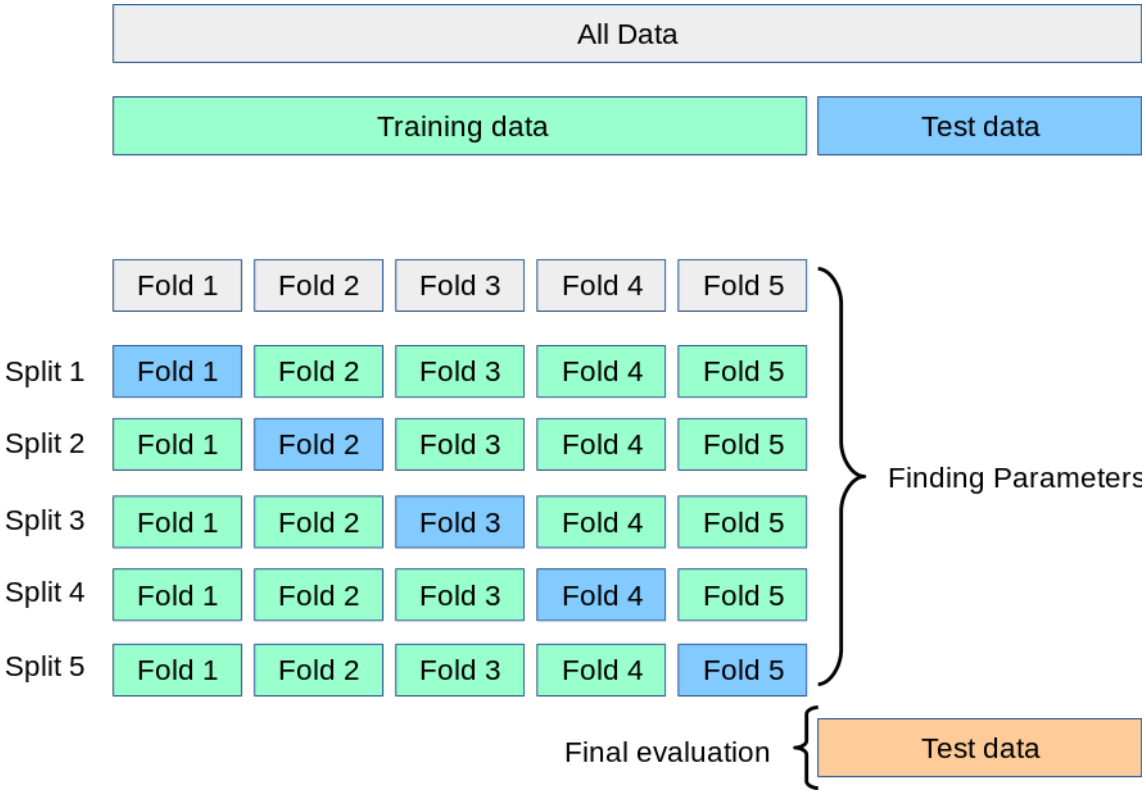

Figure 4.3: The flow chart of the process of tuning the hyperparameters using k-fold cross-validation [PVG+11]

4.2 Ordinary Least Squares Regression (OLS)

Ordinary Least Squares belongs to the category of generalized linear models [PVG+11]. The generalized linear models family is a regression family that is intended for regression in which the target value is expected to be a linear combination of the input variables [PVG+11]. In mathematical notion, if $\hat{y}$ is the predicted value:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + ... + w_p x_p$$

where the vector $w = (w_1,...,w_p)$ includes the coefficient ,the vector x is the input variables and w_0 is the intercept of the model. Ordinary Least Squares regression (OLS) is more commonly named linear regression (simple or multiple depending on the number of explanatory variables). OLS fits a linear model with coefficients $w = (w_1,...,w_p)$ to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation [PVG+11]. Mathematically it solves a problem of the form:

$$\min_w \|Xw - y\|_2^2$$

Listing 4.1 Using scikit-learn OLS modeling to follow our introduced procedure in modeling.

```
#Importing the needed packages
from sklearn import linear_model
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

# Upload the csv that has the data for the chosen independent variables and the dependent
    variable

model = pd.read_csv('finalModel.csv')


# Split into Target (y) and inputs (X)
y = model['Co2_emission']
X = model11
# Split the historical data randomly into test dataset and train dataset for both of the
    target and inputs
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# In OLS there is no hyperparameters that should be tuned, therefore we fit directly the
    model


# Create linear regression object
regr = linear_model.LinearRegression()
# Train the model using the training sets
regr.fit(X_train, y_train)

# Evaluate the model on the test dataset and on the training dataset as well
# Evaluating is done by Computing the R square
print(regr.score(X_train ,y_train))
print(regr.score(X_test , y_test))
```

As previously explained, we are following the same steps in modeling independently from the type of the model. Listing 4.1 shows an application of our modeling steps while using the provided OLS algorithm from scikit-learn. The evaluation of the models is discussed in Chapter 5 and according to the evaluation, the model with the best evaluation is chosen to be used in our Service-oriented Architecture (SOA).

4.3 Ridge Regression (RR)

Ridge Regression is from the same regression category as OLS "generalized linear models". Ridge Regression gives an estimate which minimizes the sum of square error like OLS as well, but, it imposes a penalty on the size of coefficients. Some features cause a very small influence on the dependent variable and some features may have a big influence. In OLS, all the features are treated

the same, while Ridge Regression penalizes the estimates [PVG+11]. Briefly, the ridge coefficients minimize a penalized residual sum of squares. Mathematically, it is described by the following equation:

$$\min_{w} \|Xw - y\|_2{}^2 + \alpha \|w\|_2{}^2$$

here $\alpha \geq 0$, is a complexity parameter that controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity [PVG+11]. Ridge Regression can improve predictions made from new data (reduce overfitting) by making the prediction less sensitive to the training data and this is done by adding the Ridge Regression penalty to the feature that must be minimized. Finding the α is a process of tuning the hyperparameters as explained previously. Therefore, cross-validation is used to find the hyperparameters before fitting the model. Listing 4.2 shows sample code for applying our criteria in modeling using Ridge Regression algorithm from scikit-learn library.

4.4 Polynomial Regression (PR)

Polynomial Regression is from the same regression category as OLS and ridge "generalized linear models". In machine learning, one common pattern is the use of linear models trained on nonlinear data functions. This approach maintains the generally fast performance of linear methods while allowing them to fit a much wider range of data [PVG+11]. This is the case in Polynomial Regression by constructing polynomial features from the coefficients. For instance, the equation of two-dimensional data model (two features) is described as following:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + w_2 x_2$$

If we want to fit a paraboloid to the data instead of a plane, we can combine the features in second-order polynomials, so that the model looks like this:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2$$

If we imagine creating the a new variable of $z = [x_1, x_2, x_1 x_2, x_1^2, x_2^2]$ then by substituting in the previous equation the result will be:

$$\hat{y}(w, x) = w_0 + w_1 z_1 + w_2 z_2 + w_3 z_3 + w_4 z_4 + w_5 z_5$$

Furthermore, the resulting model represents a generalized linear model, therefore, any techniques for solving linear models can be used. The model has the flexibility to fit a much wider range of data by considering linear fits within a higher-dimensional space built with these basis functions. [PVG+11]. Tuning the hyperparameters in Polynomial Regression was basically using cross-validation to find the optimal polynomial degree. After finding the optimal polynomial degree, OLS, ridge, etc. can be used for modeling. Complexity-wise, OLS is used after finding the optimal polynomial degree in our implementation.

Listing 4.2 Using scikit-learn Ridge modeling to follow our introduced procedure in modeling

```
#Importing the needed packages
from sklearn.linear_model import LinearRegression, Ridge
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

# Upload the csv that has the data for the chosen independent variables and the dependent
    variable
model = pd.read_csv('finalModel.csv')

# Split into Target (y) and inputs (X)
y = model['Co2_emission']
X = model11
# Split the historical data randomly into test dataset and train dataset for both of the
    target and inputs
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Tuning the hyperparameters using cross-validation with 5 folds of the traning data
parameters1= [{'alpha': [1,0.1,0.01,0.001,0.0001,0,10,100],'normalize':[True,False]}]
ridge_gsv = GridSearchCV(Ridge(), parameters1,cv=5, n_jobs=-1)
ridge_gsv.fit(X_train, y_train)
ridge_gsv.best_params_
# RESULT: {'alpha': 100, 'normalize': False}
# Using the Ridge algorithm with the optimal hyperparameters to fit the model
model_ridge = Ridge(alpha = 100, normalize = False)
model_ridge.fit(X_train, y_train)

# Use the produced model to predict using the test dataset and train dataset
ypred_train = model_ridge.predict(X_train)
ypred_test = model_ridge.predict(X_test)

# Evaluate the model on the test dataset and on the training dataset as well
# Evaluating is done using RMSE and R square, more information will be discussed in
    evaluation chapter
error_train = sqrt(mean_squared_error(y_train,ypred_train))
error_test = sqrt(mean_squared_error(y_test,ypred_test))
print('R-squared:')
print('Train:', model_ridge.score(X_train, y_train))
print('Test:', model_ridge.score(X_test, y_test))
print('RMSE:')
print('Train: ',error_train)
print('Test: ',error_test)

# RESULT: R-squared:
# Train: 0.89394936794
# Test: 0.897965610945
# RMSE:
# Train:  26.991548321031786
# Test:  26.475425650120528
```

4.5 Decision Trees (DTs)

Decision Trees (DTs) are non-parametric supervised learning methods used for classification and regression [PVG+11]. Decision Trees learn hierarchically by repeatedly splitting the dataset into separate branches to maximize each split's information gain. This branching structure allows regression trees to naturally learn non-linear relationships [elitedata19]. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features [PVG+11]. The deeper the tree, the more complex the rules of the decision, and the more fit the model. The end result is a tree with nodes of decision and nodes of a leaf. There are two or more branches in a decision node. Leaf node represents a classification or decision. In a tree, the root node is the top decision node and it corresponds to the best predictor. Decision Trees can handle both categorical and numerical data [DT18]. In the implementation, the DecisionTreeRegressor class is used from scikit-learn in our procedure of modeling. Furthermore, the process of tuning the hyperparameters for the Decision Trees algorithm requires to find the max depth of the Decision Trees.

4.6 Random Forest (RS)

Random Forest is considered as an ensemble method and specifically as a bagging method by[PVG+11]. The aim of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm to improve over a single estimator generalizability-robustness. Bagging methods form a class of algorithms that build up several instances of a black box estimator on random subsets of the original training set and then aggregate their individual predictions to form a final prediction [PVG+11]. The bagging methods are used to reduce the overfitting that can happen from a base regression model (e.g., a decision tree) by introducing randomization into its construction procedure and then making an ensemble out of it. Random Forest is specially designed for using Decision Trees as the randomized estimator by [PVG+11]. Each tree in the ensemble is built in Random Forest from a sample drawn from the training set with replacement (i.e., a bootstrap sample). Moreover, when a node is split during tree construction, the selected split is no longer the best split between all features. Rather, the selected split is the best split between a random subset of features[PVG+11]. Furthermore, the process of tuning the hyperparameters for the Random Forest algorithm requires finding the optimal number of randomized estimators as well as the max depth of the Decision Trees.

4.7 K-Nearest Neighbors Regression (KNNs)

The principle behind closest neighboring methods is to predict a label by finding a predefined number of training samples that are neighbors regarding the distance to the new point and using these samples to predict. The number of samples can be a user-defined constant (k-nearest neighbor learning) or vary based on the local density of points (radius-based neighbor learning) [PVG+11]. In the implementation, K-nearest neighbors regression is used and the number of samples (K) is a part of our procedure of tuning hyperparameters. Another hyperparameter to be tuned is the weights hyperparameter. The basic nearest neighbors regression uses uniform weights: that is, each point in the local neighborhood uniformly contributes to a query point classification. Under certain circumstances, weight points such that nearby points contribute more to regression than faraway points can be advantageous. This can be accomplished through the weights keyword. The default value, weights = ”uniform”, assigns equal weights to all points. Weights = ”distance” assigns weights proportional to the inverse of the distance from the query point [PVG+11].

5 Evaluation and Results

Evaluation is the last step in the procedure of modeling as described previously in Chapter 4. In this chapter, all the produced models from Chapter 4 are evaluated. Both of the training dataset and the test dataset are evaluated for each model. Three different evaluating techniques are used to evaluate the models (R^2, RMSE and plotting). The use of different techniques in the evaluation process enriches the evaluation process by emphasizing the performance of the models in different techniques. [PVG+11] uses R^2 to evaluate the models through a provided method called score. In [KTCH15], R^2 and RMSE are used in evaluating different regression models. Therefore and according to best practices, R^2 and RMSE are common techniques in evaluating regression models. The residual plotting is a visualization method that is used to evaluate the models by looking at the visualized performance of each model. The residual plotting helps in assessing the full picture of the model performance such that, it assesses whether the observed error (residuals) is consistent with stochastic error. Therefore, R^2, RMSE and residual plotting are all used to evaluate the regression models in this work.

R-Squared (R^2) is the proportion of the variance in the dependent variable that is predictable from the independent variable(s) [PF13]. The coefficient R^2 is defined as $(1 - \frac{SSE}{SSTO})$, where SSE is the sum of squared errors of prediction and quantifies how much the data points vary around the estimated regression line and it is computed in implementation as $((y_t - y_p)^2).sum()$ where y_p is the predicted value from the model using some set of inputs and y_t is the real value from the dataset with the same inputs [PVG+11]. $SSTE$ is the total sum of squares and quantifies how much the data points vary around their mean and it is computed in implementation as $((y_t - y_t.mean())^2).sum()$. The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0 [PVG+11].

The second used technique for evaluating the models is Root Mean Square Error (RMSE). RMSE is used as a measure of the differences between values predicted by a model and the true values. In other words, it indicates how concentrated the data are around the line of best fit [HK06]. RMSE is calculated as $\sqrt{MSE}$ where MSE is the mean square error. If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the mean squared error (MSE) estimated over n_{samples} is defined as [PVG+11]:

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

RMSE is always non-negative, and a value of 0 (almost never achieved in practice) would indicate a perfect fit to the data. In general, a lower RMSE is better than a higher one [HK06]. Table 5.1 shows the evaluation by R^2 and RMSE of all the developed models using our training and testing

Model	Training		Test	
	R-Squared	RMSE	R-Squared	RMSE
OLS	0.893	26.99	0.897	26.47
Ridge Regression	0.893	26.99	0.897	26.47
Polynomial Regression	0.915	24.08	0.918	23.60
Decision Trees	0.949	18.69	0.912	24.54
Random Forest	0.991	7.75	0.941	20.10
KNNs	0.930	21.82	0.923	22.88

Table 5.1: Evaluation of the six introduced models using R^2 and RMSE on both of training and testing datasets.

datasets. Evaluating the training dataset gives an idea about the performance of the model, but, the interest is more focused on the evaluation of the testing dataset since it is an unseen dataset for the model and was not used for training the models.

The third used evaluation technique is plotting. Residual plotting is used to show the performance of the models on the test data. A residual is the difference between the observed value of the dependent variable and the predicted value (Residual = Observed – Predicted) [elitedata19]. The most useful way to plot the residuals, though, is with predicted values on the x-axis, and the residuals on the y-axis. Positive values for the residual (on the y-axis) mean the prediction was too low, and negative values mean the prediction was too high; 0 means the guess was exactly correct. Figure 5.1 shows the residual plot for all of the six models using the test dataset.

5.1 Results

Table 5.1 shows the score of each model in R^2 and RMSE using both of the training and testing datasets. Having the same scores for both of the Ridge and OLS models leads to the fact that the avoidance of overfitting is the same in OLS and Ridge Regression. As explained in Section 4.3, the only difference between OLS and Ridge Regression (RR) is that Ridge Regression imposes a penalty on the size of the coefficients which makes the dependent variable less sensitive to some features. Treating the features differently in Ridge Regression helps in reducing overfitting by making the model less sensitive to the features the causes the overfitting while in OLS all the features are treated the same. Thus, having the same evaluation performance for the OLS and RR indicates that all the features are treated the same way in both of OLS and RR models and this indicates the success of the introduced feature selection criteria as in Section 3.6 for ending up with features that are not needed to be treated differently to avoid overfitting. Overall, the scores of the six models in both testing and training datasets are close to each other for each model. In other words, the models perform in a new dataset (testing dataset) at a very close level as they perform in an old dataset (training dataset) which indicates that the overfitting is avoided to a good extent as explained in Section 4.1. Moreover. the evaluation points out that the Random Forest has better results in both of the techniques (R^2 and RMSE). Not only in training dataset but also in testing dataset performs the Random Forest model better than the rest. R^2 value of 0.94 and RMSE value of 20.10 on the testing dataset for the Random Forest indicate a great performance level of the model since the best possible score for R^2 is 1.0 and for RMSE is 0. To the best of the author's knowledge, there are no related researches can be found to be compared with the produced results, but, according to the available range of the results of R^2 and RMSE, the introduced models are performing in a high-performance level overall such that the R^2 values range from 0.89 to 0.94 for all the models. Considering the residual plots in Figure 5.1, overall, all the models performed in a good manner, but, Random Forest shows an overall better range of residuals than the rest of the models. KNN and Random Forest show an overall better range of residuals between -50 to 100 which is less than the other models' ranges. Finally, Random Forest has a better evaluation of our evaluation procedure. Therefore, Random Forest is the chosen model to be used in our SOA.

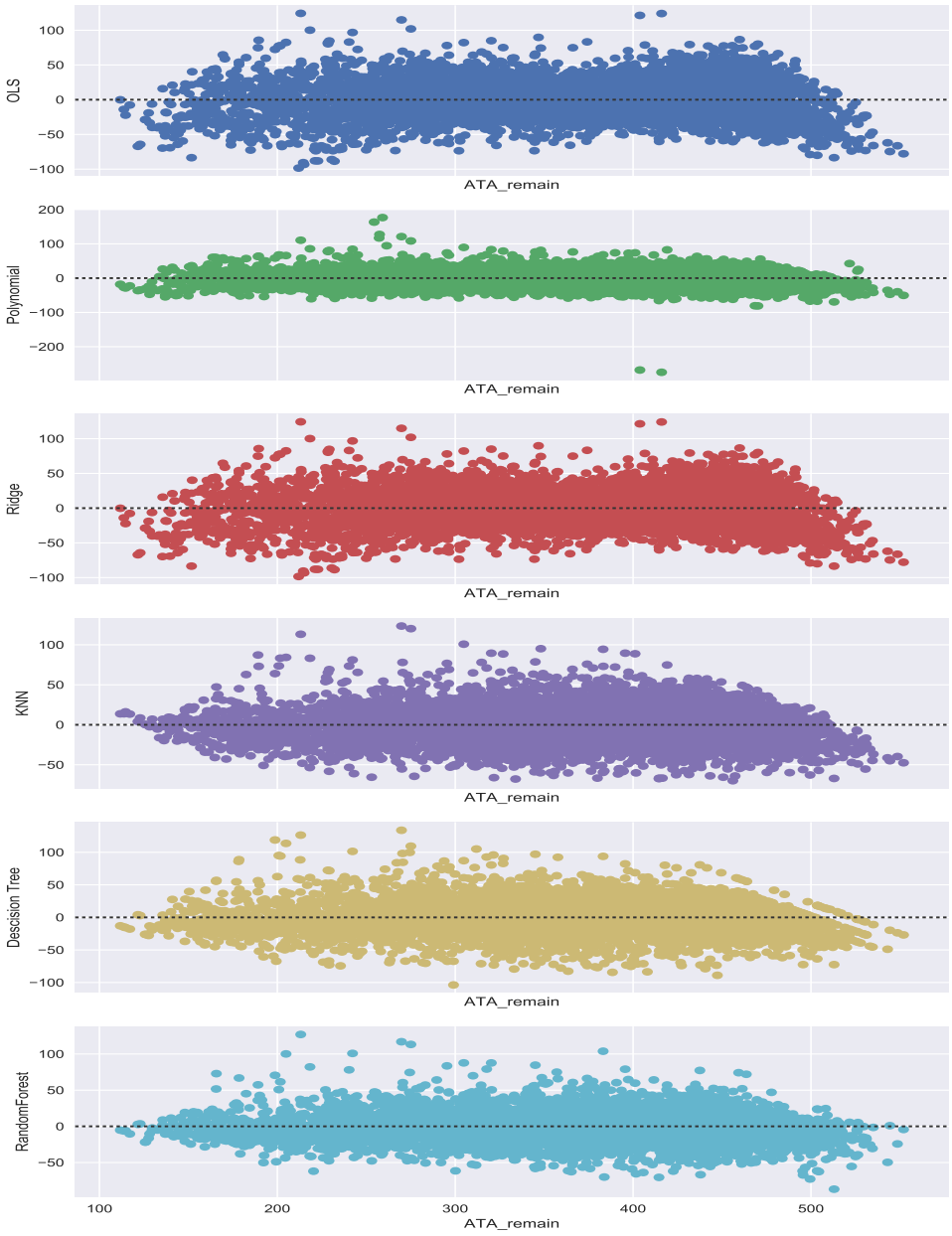

Figure 5.1: The residual plot for the produced six models.

6 Service-oriented Architecture SOA

Service-oriented architecture (SOA) is a software design style in which services are provided by application components to the other components via a network communication protocol [soa16]. The paper [Kom06] claims the main benefits of using SOA as following:

- System can be easily modified by replacement of services.
- Rapid and low-cost system development by combination of services is realized (re-use, separated system development, usage of standard).
- Total system quality can be enhanced and homogenized by using valid or proven services. And, fault isolation is easy.
- Operation and maintenance cost can be reduced by realizing the simple ease-to-understand system architecture.

There are several principles that are underlying premises in the application of SOA to information technology (e.g., loose coupling, location transparency, and parallel development) [soa16]. Therefore, using SOA to provide our CO_2-equivalent intensity factor forecasts will enable increased business agility, improved business workflows, extensible architecture, enhanced reuse, and a longer life span of our application. Many techniques and standards are used for realizing SOA.

6.1 REST APIs

Web Services are, for most organizations, the simplest approach for implementing a SOA [soa16]. A RESTful API ' ' also known as a RESTful web service " is based on Representational State Transfer (REST) technology, an architectural style and communication approach frequently used in the development of web services.[Res14]. REST as an architectural style has two main key points [SOC lecture| University of Stuttgart | Frank Leymann]:

- Stateless interactions: message meaning does not depend on the recipient's state or former messages.
- Uniform interface: set of generic actions.

Speed, scalability, simplicity, data independence, interaction with URI addressable resource, Standard data format and the fact that all context is fully understandable from the message are all goals that can be achieved by using REST as an architectural style. A RESTful API explicitly takes advantage of the RFC 2616 protocol defined HTTP methodologies. Use GET to get a resource; PUT to change or update a resource that can be an object, file or block; POST to create that resource and DELETE to remove it [Res14]. Therefore, Implementing a RESTful API for providing the CO_2-equivalent intensity factor forecasts through an HTTP GET request will achieve the stated

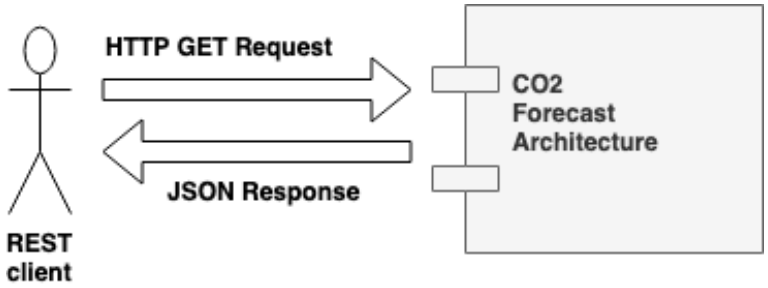

Figure 6.1: The CO_2 Forecast Architecture (CFA) provides the CO_2 forecasts through a REST API

goals of using SOA and REST. The CO_2 Forecast Architecture (CFA) responses with a JavaScript Object Notation (JSON) file format of the forecasts for one-day ahead. Figure 6.1 shows a level 0 architecture of the CO_2 Forecast Architecture (CFA).

6.2 Implementation

REST APIs can be developed in different techniques and frameworks. Since the model is implemented in Python, implementing a REST API in Python would ease the integration process between the model and the architecture. Therefore, Django REST framework (DRF) is used to create the REST API [dja19]. Django REST framework is a powerful and flexible toolkit for building Web APIs [dja19]. DRF is a full-stack web framework for Python and comes with an integrated template engine that allows developers to define the user-facing layer of a web application without additional time and effort. DRF is used and trusted by internationally recognized companies including Mozilla, Red Hat, Heroku, and Eventbrite [dja19].

As previously explained in Chapter 3 and Chapter 4, forecasting CO_2-equivalent intensity factor for one day ahead needs the forecasts of the six introduced independent variables for one day ahead to be used as inputs to the regression model. Therefore, the CO_2 Forecast Architecture (CFA) should interact with the ENTSO-E API and the weather API to get the forecasts of electricity generation and the weather to be used as inputs to the regression model. Figure 6.2 shows a level 1 architecture of the implemented CFA. For clarification, We are building a REST API- based architecture (CFA) that uses two other REST APIs to get the inputs to forecast the CO_2-equivalent intensity factor (see Figure 6.2). The produced regression model from Chapter 4 (Random Forest model) is serialized after been trained and is de-serialized in CFA to be used for forecasting. The process of serializing and de-serializing the regression model is known as pickling and un-pickling by Python community [PYT19]. Pickling is the process whereby a Python object hierarchy is converted into a byte stream, and un-pickling is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy [PYT19]. The CFA uses the pickled model as a service and does not contribute in the modeling procedure, which gives more flexibility in using different models in the future and improving the forecasts by just replacing the used model.

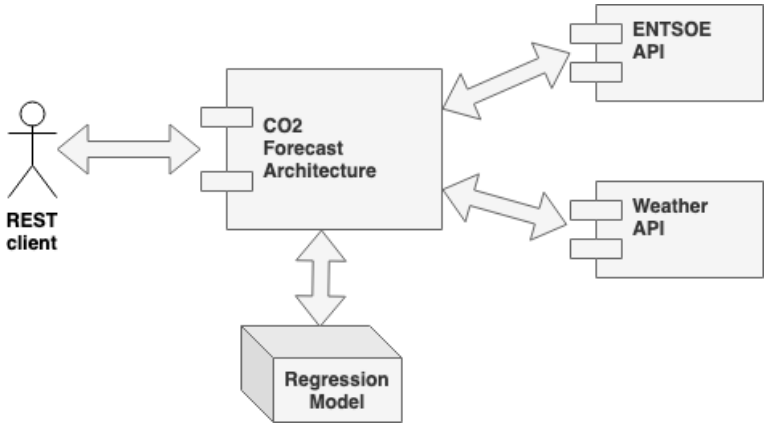

Figure 6.2: The CO_2 Forecast Architecture interacts with ENTSO-E and Weather API to retrieve the needed inputs for the regression model

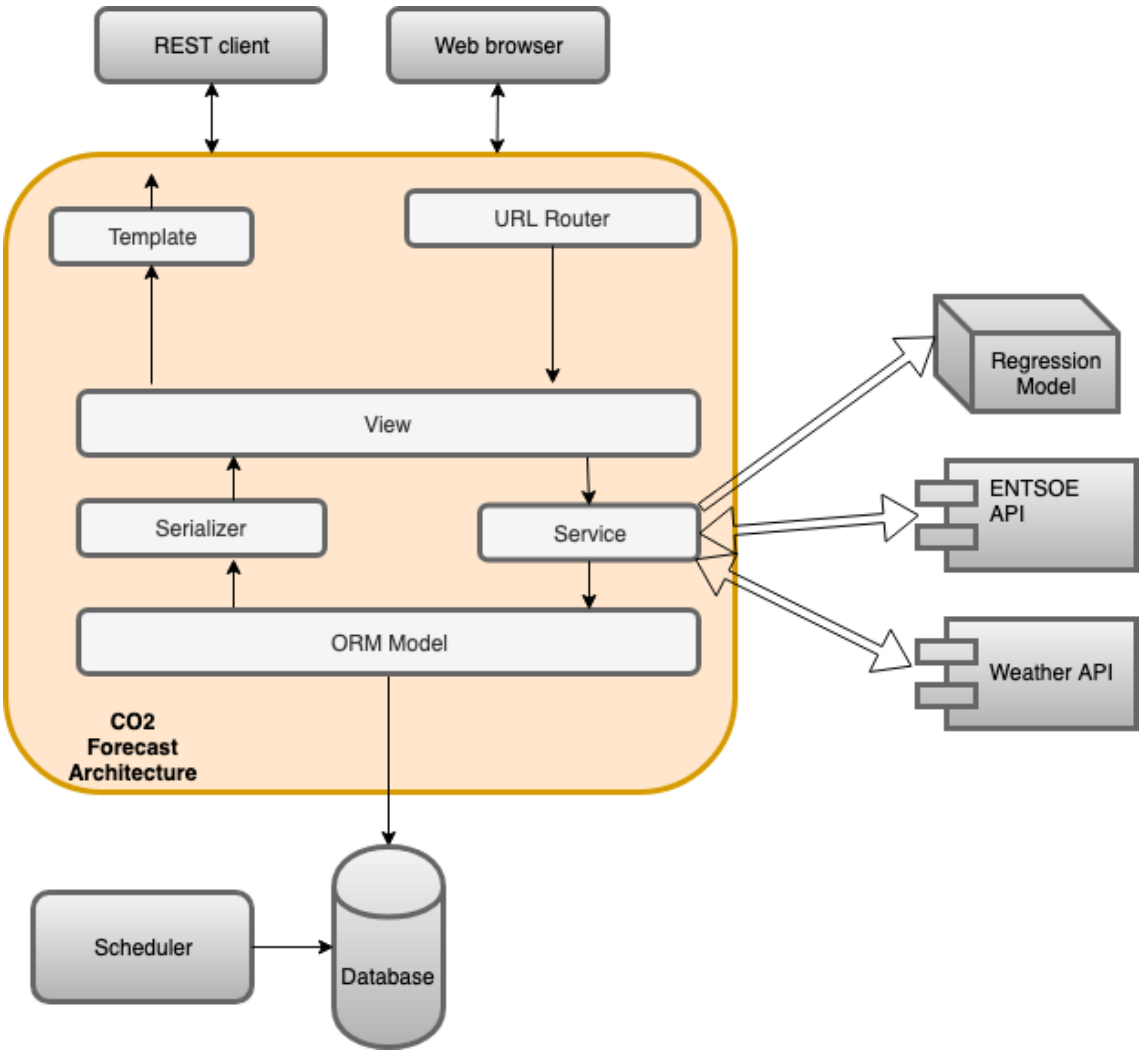

Figure 6.3: The CO_2 Forecast Architecture (CFA)

6.2.1 Architecture

Like other web frameworks, Django uses an architecture similar to MVC (Model, View, Controller), in this case, MVT (Model, View, Template) such that the controller is called view and the view is called Template in Django. The CO_2 Forecast Architecture (CFA) is following an adapted MVC architecture by introducing other layers to the main architecture. Figure 6.3 shows a level 2 architecture of our application (CFA) after adapting it to fit the business logic of the project. The following list provides an explanation of each layer in Figure 6.3:

- **REST client** is any component that hits an HTTP request and expects a JSON response from the CFA.
- Since Django provides a web-based API such that it provides a default template for the developed API, the **Web Browser** is working as a REST client in this sense for hitting the API request and showing the response in a template.
- **URL Router** is responsible for routing the requests to the right view using the URL.
- In Django, **view** is responsible for taking the requests from the client and for returning the response. In CFA there is only one view for providing the forecasts. The CFA provides only one REST API for retrieving the one-day-ahead forecasts through an HTTP GET request.
- **Service** layer is an introduced layer in our architecture such that it provides the business logic. In CFA, the main task of the service layer is the forecasting procedure and updating the database with the forecasts.
- In Django, **ORM model** is the object that is mapped to the database. The Object-relational mapping (ORM) is a mechanism that makes it possible to address, access and manipulate objects without having to consider how those objects relate to their data sources. In CFA, there is one ORM model that consists of two fields (hour and CO_2), the hour field is an integer field that represents the hour of the forecasting and the CO_2 field is a float field that represents the CO_2-equivalent intensity factor at this hour.
- In CFA, **Database** is where the forecasts are stored. In CFA, the SQLite database is used for keeping the forecasts.
- In CFA, **Scheduler** is a background scheduler that empties the forecasts table in the database from the old forecasts every day at 12:00 am.
- In Django, **serializer** allows complex data such as querysets and model instances to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types. In CFA, there is one serializer for the queryset of the forecasts that converts it into Python datatype and then to JSON.
- In Django, **template** is simply a text document or a Python string marked-up using the Django template language. In CFA, the default template is used to show the response of the API request through a web browser.

Figure 6.4 shows an activity diagram of the business logic of the CO_2 forecasts architecture (CFA) which starts with receiving a GET request and ends with the JSON response. The table forecasts that is in the database is responsible for having the updated forecasts for one day ahead while the scheduler is responsible to empty this table when the forecasts are not valid (every day at 12:00 am).

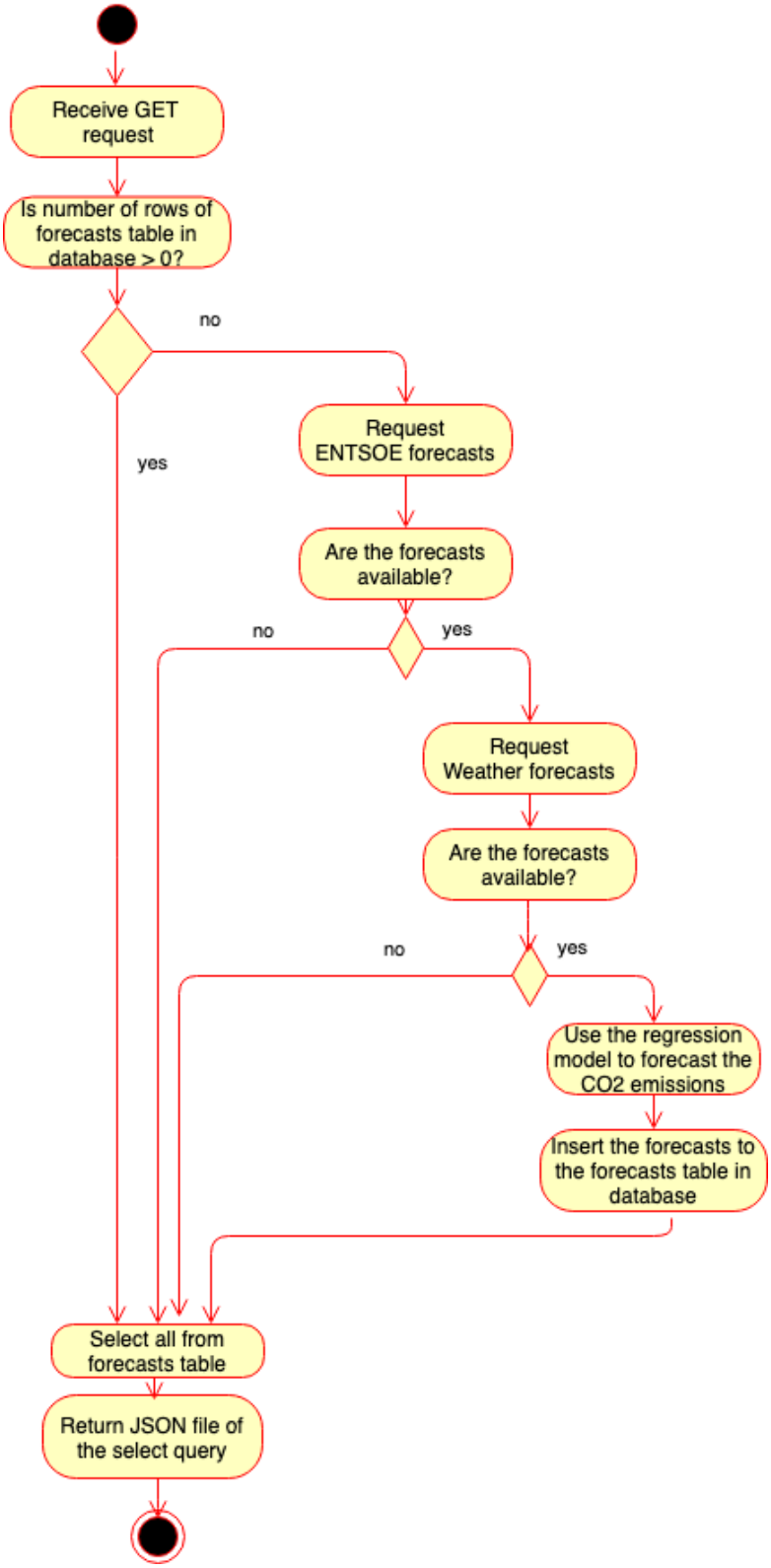

Figure 6.4: Activity diagram of the business logic of the CO_2 forecasts architecture (CFA).

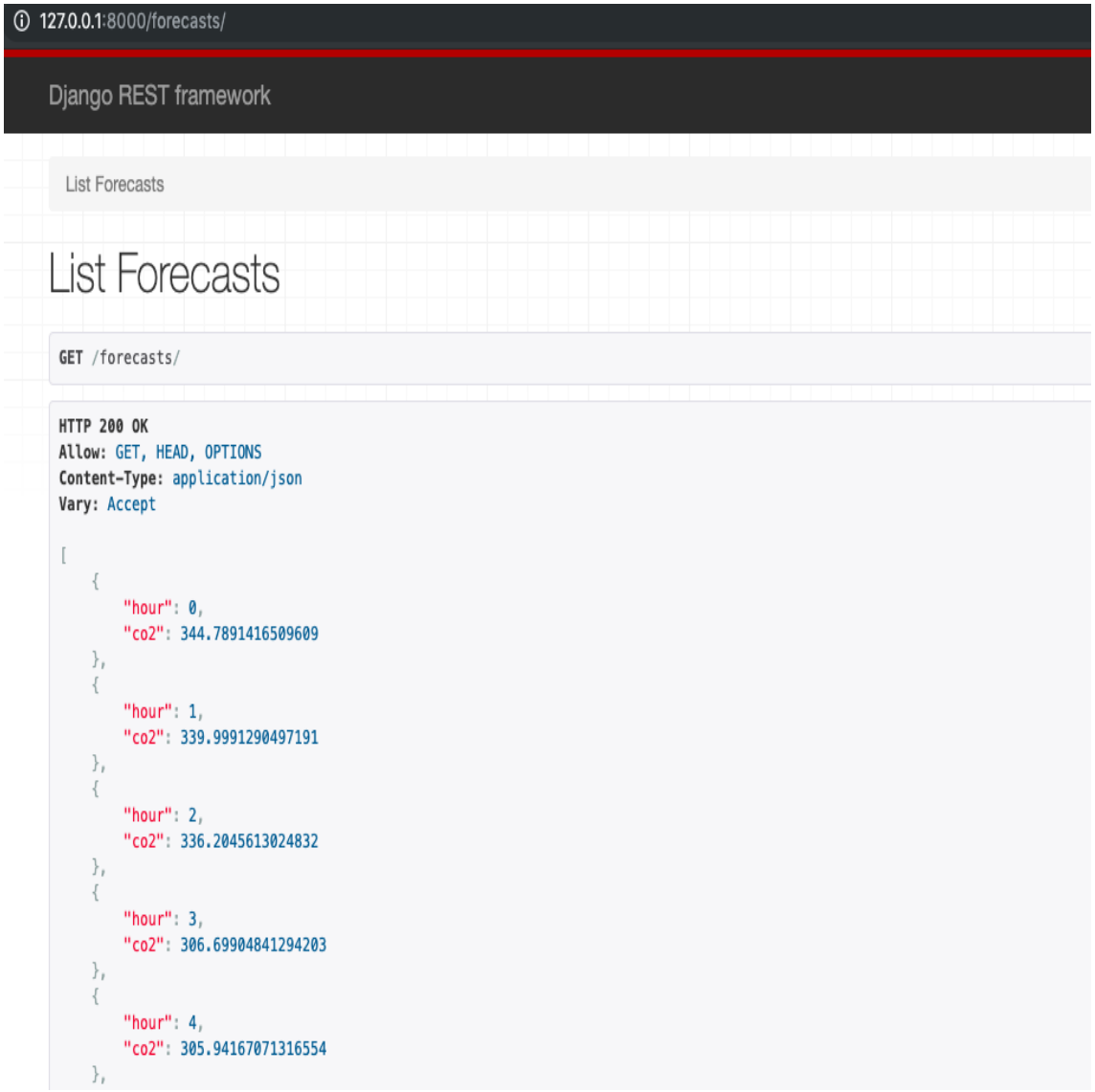

Figure 6.5: The response of the CFA as a web-based API.

The data engineering pipeline as shown in Figure 3.2 except the feature selection step is a part of our business logic such that the electricity generation and weather forecasts are collected, cleaned and integrated before being used by the regression model (Random Forest model) to forecast the CO_2-equivalent intensity factor. Figure 6.5 shows the response of the GET request of the day of 18_{th} of May 2019 using the default template of the Django REST framework. In case of one of the inputs being not available, the CFA will response with an empty JSON file.

7 Conclusion and Future Work

The energy industry has the highest contribution of the Greenhouse Gases (GHGs) emissions in Germany [uba19]. Carbon dioxide is the most harmful greenhouse gas in the atmosphere regarding its effect on global warming and climate change issues [NASA19]. Rescheduling the electricity consumption and switching the energy carriers by knowing in advance the CO_2-equivalent intensity factor may lead to 40% emissions reduction in Germany [FA18]. The thesis proposes an implementation of a service-oriented architecture that provides one-day ahead forecasts for the CO_2-equivalent intensity factor in Germany. The forecasting problem is formulated as a supervised learning (regression) problem in the machine learning context, where the dependent variable is the CO_2-equivalent intensity factor and there are six independent variables used as inputs. The historical data of total electricity generation, solar electricity generation, wind onshore electricity generation, wind speed, month and day of the week in Germany are used as independent variables to train the produced model, while the historical data of CO_2-equivalent intensity factor is calculated using the electricity generation per power plant data by Equation (3.1). Choosing and engineering the data of the six independent variables is done through a data engineering pipeline (see Figure 3.2) by starting with a potential list of 21 independent variables and ending with 6. The data engineering pipeline ends with a feature selection procedure based on correlations reductions between the independent variables and statistical tests between the independent and dependent variables, and it aims at producing a final set of independent variables that avoids overfitting, improves accuracy, reduce training time and reduces the complexity of the model. Ordinary Least Squares, Ridge Regression, Polynomial Regression, Decision Trees, Random Forest, and KNNs are all used for modeling the problem. The implementation of the models is done using the Python library scikit-learn. The models are evaluated following an evaluation procedure based on R^2, RMSE and Residual Plots. According to best practices, R^2 and RMSE are the common techniques for evaluation regression models. The used library for modeling (scikit-learn) provides a scoring method for evaluating the models with the use of R^2 in the background for evaluation. Residual plotting is a different technique that depends on comparisons between graphs and provides a wider overview of the model's performance. To the best of the author's knowledge, there are no related researches can be found to be compared with the produced results, but, according to the available range of the used techniques of, the introduced models are performing in a high-performance level overall. The R^2 values range from 0.89 to 0.94 in the evaluation procedure which indicates an overall high performance for all of the models considering that 1.0 is the best possible R^2 value. The OLS and Ridge Regression (RR) models produced exactly the same results in the evaluation procedure. Ridge Regression modeling is based on giving the ability to treat the features differently to reduce the influence of the features that can lead to overfitting while in OLS all the features are all treated the same way. Thus, having the same evaluation performance for the OLS and RR emphasizes the importance of the introduced feature selection criteria in producing features with a high influence on the dependent variable and in avoiding overfitting to a good extent. The Random Forest model

is used in the produced architecture after showing great results in the evaluation procedure as explained in Chapter 5. The service-oriented architecture is a REST API based architecture and provides one-day ahead forecasts as a JSON file through an HTTP GET request.

7.1 Future Work

The future work can be done in three different levels:

- Geographical improvement.
- Regression model improvement.
- CO_2 Forecast Architecture (CFA) improvement.

Geographical improvement can be done by including different countries other than Germany in the CO_2 Forecast Architecture (CFA). The ENTSO-E API and Weather API provide historical data and forecasts for different European countries other than Germany. The provided data can be used as inputs for the regression model to forecast for other countries. **Regression model improvement** can be done through different enhancements such as using more historical data to train the model can improve the performance of the model, applying different regression models and techniques that may lead to finding another regression model with a better performance than Random Forest, and, finding other independent variables that satisfy the independent variables constraints can improve the performance of the model as well. **CFA improvement** can be done by providing another API for the historical data of the CO_2-equivalent intensity factor and by providing the historical forecasts as well.

Bibliography

[14] "A small-sample hybrid model for forecasting energy-related CO2 emissions". In: *Energy* 64 (2014), pp. 673–677. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2013.10.017. URL: http://www.sciencedirect.com/science/article/pii/S0360544213008669 (cit. on p. 25).

[AAA97] R. Abdel-Aal, A. Al-Garni, Y. Al-Nassar. "Modelling and forecasting monthly electric energy consumption in eastern Saudi Arabia using abductive networks". In: *Energy* 22.9 (1997), pp. 911–921. ISSN: 0360-5442. DOI: https://doi.org/10.1016/S0360-5442(97)00019-4. URL: http://www.sciencedirect.com/science/article/pii/S0360544297000194 (cit. on p. 21).

[AAGE10] N. K. Ahmed, A. F. Atiya, N. E. Gayar, H. El-Shishiny. "An Empirical Comparison of Machine Learning Models for Time Series Forecasting". In: *Econometric Reviews* 29.5-6 (2010), pp. 594–621. DOI: 10.1080/07474938.2010.481556. eprint: https://doi.org/10.1080/07474938.2010.481556. URL: https://doi.org/10.1080/07474938.2010.481556 (cit. on p. 21).

[ASK09] S. K. Aggarwal, L. M. Saini, A. Kumar. "Electricity price forecasting in deregulated markets: A review and evaluation". In: *International Journal of Electrical Power Energy Systems* 31.1 (2009), pp. 13–22. ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2008.09.003. URL: http://www.sciencedirect.com/science/article/pii/S0142061508000884 (cit. on p. 23).

[BCHC09] J. Benesty, J. Chen, Y. Huang, I. Cohen. *Pearson Correlation Coefficient*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–4. ISBN: 978-3-642-00296-0. DOI: 10.1007/978-3-642-00296-0_5. URL: https://doi.org/10.1007/978-3-642-00296-0_5 (cit. on p. 44).

[BJ] S.-d. Bolboaca, L. Jantschi. "Pearson versus Spearman, Kendall's Tau Correlation Analysis on Structure-Activity Relationships of Biologic Active Compounds". In: *Leonardo Journal of Sciences* (), p. 2006 (cit. on p. 45).

[BMN09] V. Bianco, O. Manca, S. Nardini. "Electricity consumption forecasting in Italy using linear regression models". In: *Energy* 34.9 (2009), pp. 1413–1421. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2009.06.034. URL: http://www.sciencedirect.com/science/article/pii/S0360544209002539 (cit. on p. 21).

[COdess19] statisticssolutions. *complete dissertation*. 2019. URL: https://www.statisticssolutions.com/pearsons-correlation-coefficient/ (cit. on p. 45).

[Daca18] Umwelt Bundesamt. *feature selecting*. 2018. URL: https://www.datacamp.com/community/tutorials/feature-selection-python (cit. on p. 45).

[darksky19] Darksky. *weahter historical and forcasting data*. 2019. URL: https://darksky.net/dev (cit. on pp. 3, 15, 34).

[dja19] django. *Django REST framework*. 2019. URL: https://www.django-rest-framework.org/ (cit. on p. 64).

[DT18] Brandon Rohrer. *How decision trees work*. 2018. URL: https://brohrer.github.io/how_decision_trees_work.html (cit. on p. 57).

[eea19] European Environmental Agency. *Total greenhouse gas emission trends and projections*. 2019. URL: https://www.eea.europa.eu/ (cit. on p. 14).

[elitedata19] EliteDataScience. *Overfitting in Machine Learning: What It Is and How to Prevent It*. 2019. URL: https://elitedatascience.com/ (cit. on pp. 44, 50, 52, 57, 60).

[ENTSO19] European Network of Transmission System Operators for Electricity. *Transparency Platform*. 2019. URL: https://transparency.entsoe.eu/ (cit. on pp. 3, 15, 30, 31).

[epa17] United States Environmental Protection Agency. *Understanding Global Warming Potentials*. 2017. URL: https://www.epa.gov/ghgemissions/understanding-global-warming-potentials (cit. on p. 14).

[EU19] European Commissions. *Buildings*. 2019. URL: https://ec.europa.eu//energy/en/topics/energy-efficiency/buildings (cit. on pp. 3, 15).

[FA18] L. Fiorini, M. Aiello. "Household CO2-efficient energy management". In: *Energy Informatics* 1.1 (Oct. 2018), p. 29. ISSN: 2520-8942. DOI: 10.1186/s42162-018-0021-7. URL: https://doi.org/10.1186/s42162-018-0021-7 (cit. on pp. 3, 15, 30, 31, 69).

[GPS19] GPS coordinates. *GPS coordinates*. 2019. URL: https://gps-coordinates.org/germany-latitude.php (cit. on p. 34).

[GT04] F. Gori, C. Takanen. "Forecast of energy consumption of industry and household services in Italy". In: *International Journal of Heat and Technology* 22 (Jan. 2004), pp. 115–121 (cit. on p. 22).

[HK06] R. J. Hyndman, A. B. Koehler. "Another look at measures of forecast accuracy". In: *International Journal of Forecasting* (2006), pp. 679–688 (cit. on p. 59).

[KB10] A. C. Koene, T. Bueke. "Forecasting of CO2 emissions from fuel combustion using trend analysis". In: *Renewable and Sustainable Energy Reviews* 14.9 (2010), pp. 2906–2915. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2010.06.006. URL: http://www.sciencedirect.com/science/article/pii/S136403211000153X (cit. on p. 24).

[Kom06] N. Komoda. "Service Oriented Architecture (SOA) in Industrial Systems". In: *2006 4th IEEE International Conference on Industrial Informatics*. Aug. 2006, pp. 1–5. DOI: 10.1109/INDIN.2006.275708 (cit. on p. 63).

[KP17] D. Kunda, H. Phiri. "An Approach for Predicting CO2 Emissions using Data Mining Techniques". In: *International Journal of Computer Applications* 172 (Aug. 2017), pp. 7–10. DOI: 10.5120/ijca2017915098 (cit. on p. 24).

[KTCH15] F. Kaytez, M. C. Taplamacioglu, E. Cam, F. Hardalac. "Forecasting electricity consumption: A comparison of regression analysis, neural networks and least squares support vector machines". In: *International Journal of Electrical Power Energy Systems* 67 (2015), pp. 431–438. ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2014.12.036. URL: http://www.sciencedirect.com/science/article/pii/S0142061514007637 (cit. on pp. 21, 22, 59).

[Machle19] Machine Learning Mastery. *Supervised and Unsupervised Machine Learning Algorithms*. 2019. URL: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ (cit. on pp. 16, 51).

[MAT19] MathWorks. *MATLAB*. 2019. URL: https://www.mathworks.com/products/matlab.html (cit. on p. 22).

[MN11] M. Meng, D. Niu. "Modeling CO2 emissions from fossil fuel combustion using the logistic equation". In: *Energy* 36.5 (2011), pp. 3355–3359. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2011.03.032. URL: http://www.sciencedirect.com/science/article/pii/S0360544211001988 (cit. on pp. 24, 25).

[NASA19] the Earth Science Communications Team. *Global Climate Change*. 2019. URL: https://climate.nasa.gov/causes/ (cit. on pp. 3, 13, 19, 20, 69).

[PF13] F. Provost, T. Fawcett. *Data Science for Business: What You Need to Know About Data Mining and Data-analytic Thinking*. 1st. O'Reilly Media, Inc., 2013. ISBN: 1449361323, 9781449361327 (cit. on pp. 51, 52, 59).

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. 44, 45, 50–53, 55, 57–59).

[PYT19] Python Software Foundation. *pickle - Python object serialization*. 2019. URL: https://docs.python.org/3/library/pickle.html (cit. on p. 64).

[Res14] Maxine Giza. *Guide to SOA and the cloud*. 2014. URL: https://searchmicroservices.techtarget.com/definition/RESTful-API (cit. on p. 63).

[Roh14] S. Roh. "The Signal and the Noise: Why So Many Predictions Fail—But Some Don't by Nate Silver, New York Penguin Press. 2012". In: *Risk Analysis* 34.2 (2014), pp. 396–398. DOI: 10.1111/risa.12177. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/risa.12177. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/risa.12177 (cit. on p. 51).

[Rum16] D. J. Rumsey. *Statistics For Dummies*. 2nd ed. Paperback, 2016. ISBN: 978-1-119-29352-1 (cit. on p. 42).

[soa16] David Linthicum. *Service Oriented Architecture (SOA)*. 2016. URL: https://web.archive.org/web/20160206132542/https://msdn.microsoft.com/en-us/library/bb833022.aspx (cit. on p. 63).

[ST08] S. H. Shih, C. P. Tsokos. "Prediction Models for Carbon Dioxide Emissions and the Atmosphere". In: *Neural, Parallel Sci. Comput.* 16.1 (Jan. 2008), pp. 165–178. ISSN: 1061-5369. URL: http://dl.acm.org/citation.cfm?id=1466627.1466640 (cit. on p. 24).

[Ste11] J. Stewart. *Calculus*. Cengage Learning, 2011. ISBN: 9780538497817. URL: https://books.google.de/books?id=AavjDHGwGpIC (cit. on p. 16).

[toSci19] Towards science. *Feature selection—Correlation and P-value*. 2019. URL: https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf (cit. on p. 45).

[uba19] Umwelt Bundesamt. *Greenhouse gas emissions*. 2019. URL: https://www.umweltbundesamt.de/en/indicator-greenhouse-gas-emissions (cit. on pp. 3, 15, 20, 69).

[Wer14] R. Weron. "Electricity price forecasting: A review of the state-of-the-art with a look into the future". In: *International Journal of Forecasting* 30.4 (2014), pp. 1030–1081. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2014.08.008. URL: http://www.sciencedirect.com/science/article/pii/S0169207014001083 (cit. on p. 23).

[WLL+15] L. Wu, S. Liu, D. Liu, Z. Fang, H. Xu. "Modelling and forecasting CO2 emissions in the BRICS (Brazil, Russia, India, China, and South Africa) countries using a novel multi-variable grey model". In: *Energy* 79 (2015), pp. 489–495. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2014.11.052. URL: http://www.sciencedirect.com/science/article/pii/S0360544214013073 (cit. on p. 25).

[ZSH15] F. Ziel, R. Steinert, S. Husmann. "Efficient modeling and forecasting of electricity spot prices". In: *Energy Economics* 47 (2015), pp. 98–111. ISSN: 0140-9883. DOI: https://doi.org/10.1016/j.eneco.2014.10.012. URL: http://www.sciencedirect.com/science/article/pii/S0140988314002576 (cit. on p. 22).

All links were last followed on May 14, 2019.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature