Università degli Studi di Padova

Scuola di Ingegneria

Dipartimento di igegneria Industriale

Corso di Laurea Magistrale in Ingegneria dell'Energia Elettrica

Modelling variable loading through machine learning prediction model for an impact assessment concerning distribution networks

Relatore:
Prof. Roberto Turri

Correlatore:
Dr. Keith Sunderland

Laureando:
Samuele Faccioni
Matricola 1156897

Anno accademico 2018-2019

Alla mia famiglia e a Claudia

Contents

List of Figures

List of Tables

Aknowledgments

0.1 Aknowledgments

At the end of this academic pathway I would like to extend my gratitude to Prof. Roberto Turri, for giving me the opportunity to make this international experience at the Tecnological University Dublin and to Dr. Keith Sunderland (Assistent Head of School Electrical and Electronic Engineering at TU Dublin) for the help he gave me and because he allowed me to work on a topic projected into the future.

A big thank you is for my parents Renzo and IOle who have allowed me to reach this wonderful goal, supporting me in all my choices and moments of difficulties and to my sister Georgia who is always able to listen and ready to help me. Special thanks are for Claudia, you have been able to understand me and support me in difficult moments.

Many thank at all the guys of the room 13 of the TU Dublin of Kevin street for the beautiful moments and for the fantastic working ambient which they made possible.

0.2 Ringraziamenti

Alla fine di questo percorso di studi desidero porgere la mia gratitudine al Prof. Roberto Turri, per avermi dato la possibilità di fare questa esperienza internazionale presso la Tecnological University Dublin e al Dr. Keith Sunderland (capo dipartimento di ingegneria elettrica del DIT) per l'aiuto che mi ha dato e poiché mi ha permesso di lavorare su un argomento proiettato al futuro.

Un grande ringraziamento va ai miei genitori Renzo e Iole che mi hanno permesso di raggiungere questo meraviglioso traguardo, appoggiandomi e sostenendomi in ogni mia scelta e nei momenti di difficoltà e a mia sorella Georgia per la pazienza nell'ascoltarmi e per essere sempre pronta ad aiutarmi.

Un ringraziamento speciale va a Claudia per l'aiuto e il sostegno nei momenti difficili, lungo questa avventura.

Abstract

In this thesis we will deal with the application of new machine learning technologies to an Irish power grid model. The project involves the collection, preparation, cleaning and employement of historical load data of 4000 Irish customers to train the SVR and RFR algorithms. The impact of forecasting associated with the application of two models to and example of Irish distribuition Network and the implementation on system losses, is analysed. The results deriving from the two forecasting models are then compared, inserting them in a Simulink and matlab environment that proceeds to perform a network power flow. It can be seen that the SVR (Support Vector Machine) and RFR (Random Forest Regressor) models cannot achieve an optimal error to be used without the help of training optimization algorithms and the parameters that manage training sets. However, this technology is becoming more and more important in the field of electricity grid management, helping the control center both to manage the power peaks of a line and to increase the protection of sensitive data obtained through the smart grid.

Abbreviations

Abbreviation	Meaning
Adeline	ADAptive LInear NEuron
AI	Artificial Intelligence
AMI	Advanced Metering Infrastructure
ANN	Artificial Neural Network
CER	Commision for Energy Regulation
CRU	Commission for Regulation of Utilities
GA	genetic Algorithm
HV	High Voltage
IG	Information Gain
LV	Low Voltage
ML	Machine Learning
MSE	Mean Square Error
MV	Medium Voltage
NEAT	NeuroEvolution of Augmenting Topologies
PSO	particle Swarm Optimization
pu	Per Unit
PV	Photovoltaic
RBF	Radial Basis Function
RFR	Random Forest Regressor
SSE	Sum of Squared Errors
SST	Total Squared Sum
SVM	Support Vector Machine
SVR	Support Vector Regressor

Chapter 1

Introduction

Due to the deeper and deeper penetration of renewable energy systems in our power grid, and as consequence of their intermittent and non-programmable energy sources, power flow will be highly variable, adding more difficult to predit them. Right now, the final user,thanks to the solar power generation, could be a producer and not just a consumer; so the system needs to be able to manage the energy and power flows between the generation's systems and the loads. This option is available with a smart-grid, which facilitates the control and protection in real time even in the case of sudden non-programmable events. Also having the possibility to collect real time data about the state of the grid, it is possible to do a prediction of the dayly utilized power. A prediction of the dayly power flux, letting us to manage better our Battery Energy Storage Systems (BESS), reducing their cost, by minimizing the capacity of the BESS or by optimizing it's operation [1] or, as in Australia, is usefull to choose the optimal allocation of generators[2]. The main strenght of the smart grid is the communication and is also its weakness: smart meters are used to collect real time data and ,after measurement, send it to the control station which can take the proper decision, but this chain-system is weak to cyber-attacks. The data can be tampered or attacked to mislead the decision-making of the control station [3]. To try to defend to this attack, ML-algorithms, as support vector machine (SVM), can be usefull to detect the false input data injectred in the system.

The thesis work is divided into 8 chapters:

- chapter 1 : introduction to machine learning
- chapter 2 : machine learning features
- chapter 3 : case of study and data profile
- chapter 4 : pre-processing data and SVR, RFR models

- chapter 5 : simulink model
- chapter 6 : matlab model
- chapter 7 : results
- chapter 8 : conclusion and futures works

1.1 Machine Learning

Machine learning is a branch of Artificial Intelligence, which studies and develops learning algorithms to make forecast prediction or to identify rules inside data and build a model to explain this intrinsic feature of the dataset. Without a human presence to detect structures and logic connections inside a dataset, the machine learning is able to detect rules and to create models to analyse data and to allow us to make a better idea in order to take a better future decision. Nowadays Machine Learning is getting more and more important also in our daylife, because it is a common tool which we use every day, as spam mail filter, voice or text recognition software, in the internet reasearch motor and also is already built in car's autopilot [4].

The first to theorize machine learning algorithms was Arthur Samuel in 1959 [5]. At the end of the 50s researchers like Arthur Samuel, Marvin Minsky and Frank Rosenblatt, tried to create machines that were able to learn from data, using, as an approach, both various formal methods and the first embryos of neural networks [6].

Also in the 50s, Alan Turing proposed the idea of a learning machine, that is able to learn and therefore become intelligent. Turing's specific proposal anticipates genetic algorithms [7].

For problems in both theoretical and practical data acquisition, the development of machine learning, which was based mainly on probabilistic systems, slowed down. Only towards the end of the 90s when his objective changed from obtaining artificial intelligence, therefore a machine capable of autonomous thinking, to tackle solvable problems of a practical nature and headed towards methods and models borrowed from statistics and theory of probability [8].

Nowadays the fields of use of these learning algorithms are various: they cover from text speech recognition, to automatic driving of vehicles, to use as automatic classifiers in various fields such as medical, astronomical or software. In astronomy, machine learning algorithms are used to classify and identify images of space celestial bodies, just as in medicine they are used to aid in medical diagnosis, for example by helping to recognize potential cancer cells from healthy cells [9] [10] [11].

1.2 Three main types of machine learning

Machine learning tasks are typically classified into three broad categories, depending on the nature of the "signal" used for learning or the "feedback" available to the learning system. There three categories are: Supervised and Semi-supervised learning, Unsupervised learning, Reinforcement learning.

In the supervised learning, the model is given examples, in the form of possible inputs and the respective desired outputs, and the goal is to extract a general rule that associates the input to the correct output.

In the unsupervised learning, the model has the purpose of finding a structure in the inputs provided, without the inputs being labeled in any way.

In the reinforcement learning, the model interacts with a dynamic environment in which it tries to reach a goal (for example driving a vehicle), having a teacher who tells him only if he has achieved the goal. Another example is to learn to play a game by playing against an opponent.

Halfway between supervised and unsupervised learning is semi-supervised learning, in which the "teacher" provides an incomplete training dataset, that is, a set of training data among which there are data without the respective desired output.

Another categorization of machine learning tasks is detected when considering the desired output of the machine learning system.

In classification, the outputs are divided into two or more classes or labels and the learning system must produce a model that assigns inputs not yet seen to one or more of them. This is usually dealt with in a supervised manner. Anti-spam filtering is an example of classification, where the inputs are emails and the classes are "spam" and "not spam".

In the regression, which is also a supervised problem, the output and model used are continuous. An example of regression is the determination of the amount of oil present in an oil pipeline, having the measurements of the attenuation of gamma rays passing through the conduit. Another example is the prediction of the value of the exchange rate of a currency in the future, given its values in recent times [12].

In clustering a set of inputs is divided into groups. Unlike in the case of classification, groups are not known before, typically making it an unsupervised task. The cluster analysis, or clustering, is the assignment of a set of observations in subgroups (clusters) so that the observations in the same cluster are similar in certain characteristics. It is a common technique for statistical data analysis.

1.2.1 Supervised learning

The main goal of the supervised learning algorithm is to extrapolate a model from a set of labeled training data, which allows us carry out a prediction with future data. The 1.1 is an exmplame of the logic behind the process.

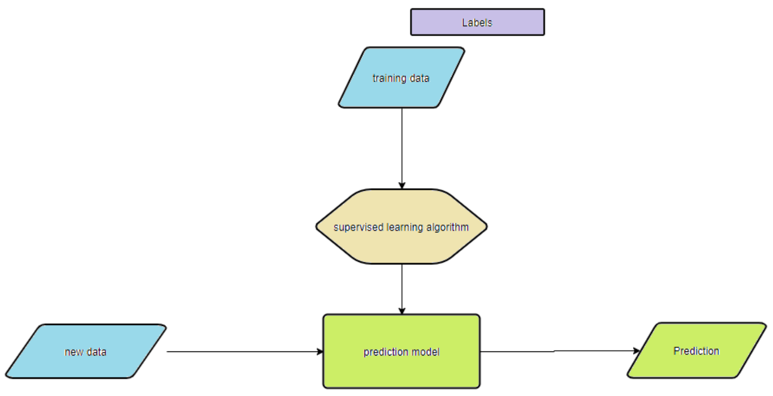


Figure 1.1: Flow chaarft of supervised learning algorithm

The name supervised comes from the fact that we already know the output of the training dataset, so we know the labels. After the training we have a model which can be used to classy new data or we can chose to predict new data output from new data input.

1.2.2 Unsupervised learning

With this type of algorithm we have not labeled data, or even unknown structure, but the learning process is able to give us back important information about the structure of our data without a fixed reward or a given output. As shown before there are different technique of unsupervised learning as clustering methods or multi dimensional data reduction, for clearing the data from rumors.

1.2.3 Reinforcement learning

In the reinforcement learning the main purpose is to create a system which can improve perfomance thanks to interactions with the enviroment. This can be understood with an example: think about the chess game. Our enviroment will be the chessboard. Our algorithm have to learn a series of action to try to maximaze its reward, which in this case is the victory of the game, using a trial-and-error approach. So in the reinforcement learning the main purpose of the learning algorithm will be try to get the reward with the maximum efficiency.

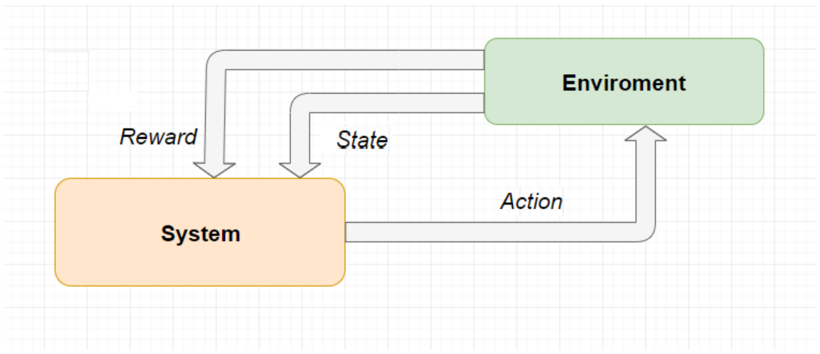


Figure 1.2
Flow chart, Reinforced Learning mechanisms [4]

1.2.4 Classifier

In the classification the main purpose is to give the correct label to the new dataset input thanks to the past observation that it has done in the training session. The labels are discrete value, no ordered and we can have a binary classification or multiclass classification. The second one is used to recognise the written text, where we have multiple labels for each alphabet letters, so we have a better precision in the predicted classification of new written text from different people. Although, it's important to remind which the system cannot recognise the numbers if these aren't inside our training dataset.

In this figure is possible to see how a binary classifier works on a bidimensional dataset where each data can be assigned at one color, red or green. With a supervised learning algorithm we can obtain a rules to divide our data (which is represented by the linear boundary, the dashed line) and to be able to divide also the new input data thanks their x and y values.

1.2.5 Predictor

The second type of supervised learning algorithms are the regressor analysis algorithm, in which we have some predictive variables, our features, and one continuos target variable, our result. The main purpose is to find a correlation between features and target. There are different type of regression,

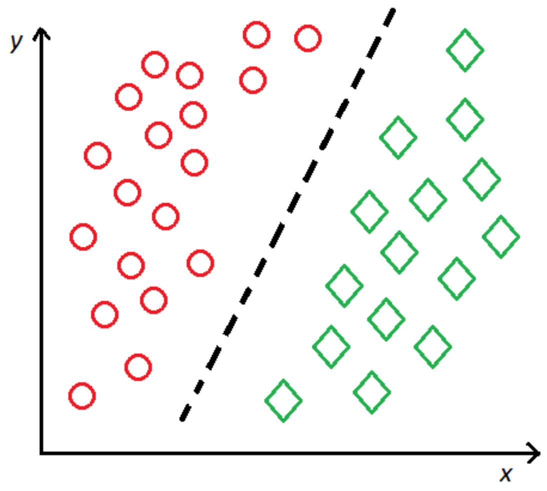


Figure 1.3: Example of Binary Classifier

most common: linear-regressor, SVR, RFR. In the next chapter they will be explained.

Chapter 2

Machine learning features

2.1 Introduction

In recent years a lot of different machine learning process are developed for different problems.It is important to remark which every machine learning algorithm has some advantage and some disadvantage respec to the others [13][14].The problem now is : how to choose the best ML algorithm.
To compare the different algorithms between themselves is needed a refrence metrics; usually is used the accuracy as reference. Another important step is to tune the algorithm parameters, an operation which is different for every different problem we are going to analyse. To do it, is possibile to use a optimization programs.

2.1.1 Quality prediction parameters: MSE and R^2

The MSE (Mean Squared Error) is the mean value of the minimizing SSE cost function.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - {y_p}^{(i)})^2$$

It is very usefull to confront different regression model, to optimize their parametres.

Sometimes is better use a coefficient of determination (R^2) that could be saw as a standard version of the MSE error and could give us a better model's performance.
R^2 is defined as:

$$R^2 = 1 - \frac{SSE}{SST}$$

where SST is the total square sum $SST = \sum_{i=1}^{n}(y^{(i)} - \mu_y)^2$ which is the variance of the response. For the training dataset R^2 is bounded between 0 and 1, where $R^2 = 1$ means that the model colud perfectly rappresent the dataset and MSE will be 0. The coefficient of determination could be negative for the test set, in this case our model can't undestand the relationship between the variables.

2.2 Data pre-processing

It is rare that data is already optimally shaped for our learning alogrithm, so most of the time pre-processing work is required to have the maximum efficiency from our learner. Most of the ML algorithms need also that all the input data must be standardized in a range from 0 to 1 or in the standard normal distribuition with a mean value of zero and a variance of one. Some features of the dataset could have a high correlation bewtween themselves, so it maybe possible to cancel some of them with out changing the final result or even improving the total efficiency of the program [4]. Moreover the reduction of dimensionality of the input dataset reduces the necessary memory to store the dataset and reduces the training time. It is also important split our input dataset into a training group and test group of data. The training group is used to let our algorithm to make the prediction model, instead the test group is used at the end of model building process, to check the accuracy of the model when new data are given as input. In chapter 4 this topic will be studied in depth and shown.

2.3 Adeline

Studying the principles of the functioning of the human brain to apply them to artificial intelligence, Warren McCullock and Walter Pitts published in 1943 a first simplified brain cell scheme, the so-called McCullock-Pitts neuron. They then tried to create a logic gate that functioned like a neuron of the human brain, which acquires multiple input signals and when they exceed a certain threshold begins to transmit with the next cell [15] .

A few years later, Frank Rosenblatt published the first concept of the perceptron learning rule based on the McCullock-Pitts neuron model [16].

The author proposed an algorithm that would automatically learn the optimal weight coefficients to be multiplied with the input characteristics so as to be able to make the decision on whether a neuron is activated or not.

It can therefore be traced back to a binary classification problem where there are two classes, one 1 (positive class) and -1 (negative class).

We can then define an activation function $\Phi(*)$ is a piecewise-defined function, which is sometimes called the Heaviside step function:

$$\Phi(z) = \begin{cases} 1 & if\ z \geq \theta \\ -1 & else \end{cases}$$

it takes a linear combination of certain input values "**x**" and a corresponding weight vector "ω", where "**z**" is the so-called network input; if the activation of a specific sample "x" is greater than a certain threshold "Θ", it is possible to make a prediction if the sample will fall into the positive or negative class and then label it.

z can be written in a compact form like:

$$z = \omega^T x$$

It is thus seen in the figure 2.1 that the input z is reduced in a binary output by the activation function and how it can be used to discriminate between two classes that can be separated in a linear way.

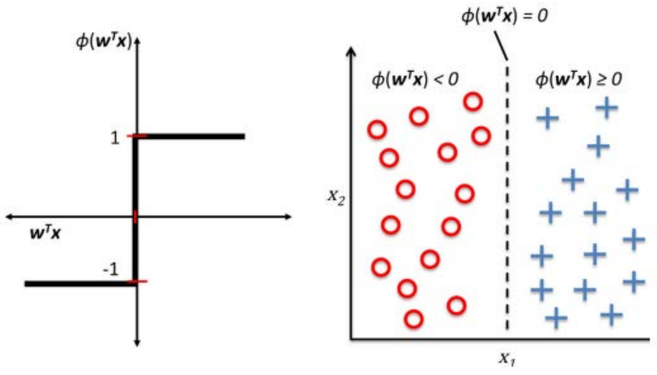


Figure 2.1
[4]

From the preception's theory one of the first and oldest mono neural network born: Adeline, ADAptive LInear NEuron, it was published in the 1960 by Bernard Widrow and Ted. it is a early single-layer artificial neural network which introduces the concept of definition and minimization of a cost function J, which will be the bases of all the later machine learning algorithms as SVM or ANN [17]. The next section will explain J.
The main feature of ADALINE, respect of the preceptron, is that in the learning phase, the weights are adjusteed according to the weighted sum of the inputs (the net).

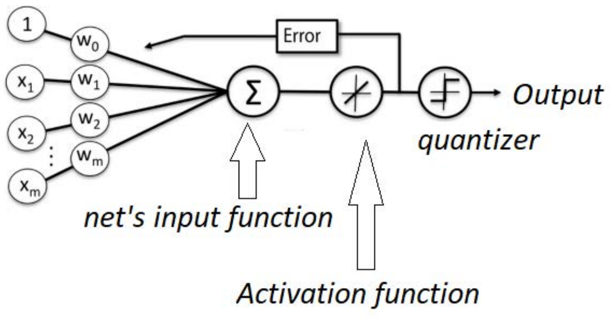


Figure 2.2: Learning inside a single layer ADALINE

In the figure 2.2 we can see how a single layer Adaline works:

- INPUT:
  1. x is the input vector
  2. ω is the weight vector
  3. η is the number of inputs
  4. Θ some constant
- OUTPUT: $y = \sum_{j=1}^{n} x_j w_j + \theta$ which is the output of the model.

Here we can use the continuos output given by the linear activation function to compute the model's error and to correct the weights instead of using the labels of a class.

In Adaline the linear activation function is used for learning weights and a quantizer is used to predict the classes of labels, such as the unit step function previously seen in the preceptron.

2.3.1 Optimization for a cost function

One of the main features of supervised machine learning algorithms is to define an objective function which will be optimized along the training process. Usually this objective function is a cost function; for example using Adaline, it is possible to define the cost fuinction as J to learn the weights as SSE (Sum of the Squared Errors) betweeen the true values and the calculated values.

$$J = \frac{1}{2}\Sigma_i(y^{(i)} - \phi(z^{(i)}))^2$$

The main advantage of this function is which it is differential and convex, so we can apply an optimization algorithm (as the gradient discend) to find the weights which minimize the cost function.

2.4 Linear regression

The main goal of a linear regression model is try to undestand and to replicate the relation between one characteristic (ξ) and a continuos response (variable target y) [18]. The y is defined as:

$$y = w_0 + w_1 x$$

- ω_0 is the y-intercept
- ω_1 is the coefficient of the descriptive variable

If we know weights to describe the relation,correlation between the input and the output of the variables, it will be able to predict also the response of new descriptive variables, which aren't in the initial dataset (training dataset). We cans ay that the wea re looking for the best linear function to interpolate the sampling points as in the figure 2.3.

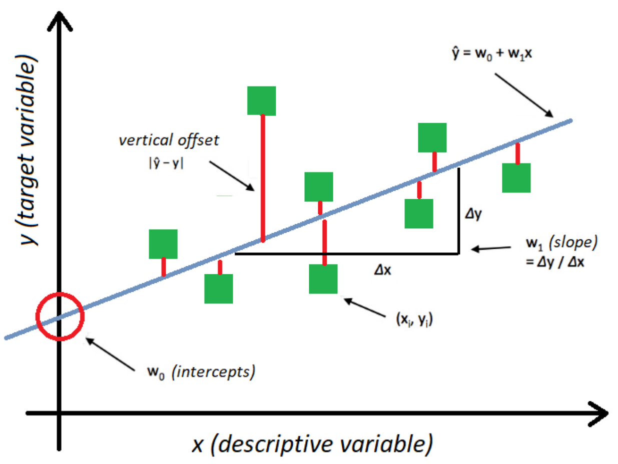


Figure 2.3: Linear regression example

This is an example of simple linear regression with only one variable, but is possible to extend this type or regression at a multiple linear regression, where multiple correlated dependedent variables are predicted [19].

2.5 SVM and SVR

Another exstention of the old preceptron is the SVM, Support Vector Machine, which is a supervised learning model that could be used to analyse data for classification and regression analysis. With Adaline our main goal was minimizing the cost function, now with the SVM the main purpose is to maximize the margin defined by the hyperplane and the training set.
The hyperplane is essentially a linear subspace of a dimension smaller than one (n - 1) with respect to the space in which it is contained (n). If space has dimension 3, its hyperplanes are the plans.

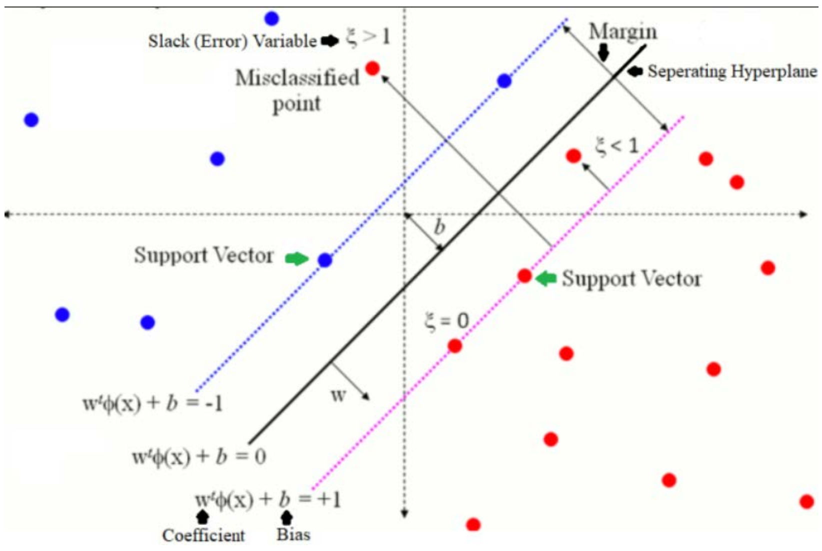


Figure 2.4: Linear SVM, Margin and hyoerplane [20]

An exmaple of margin for a linear SVM is in figure 2.4 where is possible to see also the Support Vectors, which are the nearest training points to the hyperplane. In this type of algorithms is important to maximize the margin, to reduce the overall error and avoid the overfitting effect.

An SVM model is a representation of the input data (examples) as points in a space, mapped so that the input data, which are belonging at different categories, are divided by a clear gap (margin) that we want as wide as possible. New examples are then added into the same space and predicted to belong to a certain category according to which side of the margin the fall.
SVM are able to do linear and non-linear prediction by using the kernel transform, basiccaly mapping their inputs into a high-dimensional feature spaces [21].
For the linear case is possible to write $f(x) = \omega \cdot x + b$; to reach the *flatness*

of the function is way easier to minimize the norm ($\|w\|^2$) [21].
So this is a convex optimization problem where we have to:

$$minimize\ \frac{1}{2}\|w\|^2$$

subject to:

- $y_i - \omega \cdot x_i - b \leq \xi$
- $\omega \cdot x_i + b - y_i \leq \xi$

Sometimes the convex optimization problem is not feasible or we want to allow some errors inside it, so similar to the "Soft margin"loss function, can be introduce the salck variables ξ_i and $\xi_i{}^*$ to cope with otherwise infeasible constraints of the problem. So we use the Vapnik formulation (1995):

$$minimize\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i{}^*)$$

subject to:

- $y_i - \omega \cdot x_i - b \leq \xi + \xi_i$
- $\omega \cdot x_i + b - y_i \leq \xi \xi_i{}^*$
- $\xi_i, \xi_i{}^* \geq 0$

So now the problem is feasible with a quadratic programs (for more compelte mathematical analysis see [21]).
The variabile ξ was introduced becasue the linearity constraints must be mitigated for non-linear data, to allow the convergence of the system in the case of bad classificated data.

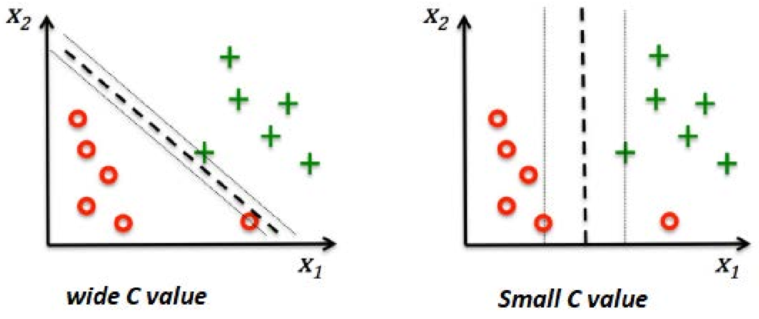


Figure 2.5: Effects of C on out prediction

The figure 2.5 shows as modifying C, we obtain different compromise between Bias and variance. The variance measures the coherence (or variability) of the prediction of the model for a given instance of the sample if we repeat the training of the model several times, for example, on different sets of the training dataset. We can say that the model is sensitive to the randomness of training data. On the contrary, the bias, the discrepancy, measures how far the forecasts are compared to the corrected values in general, if we reconstruct the model several times on different training datasets; the bias measures the systematic error that is not related to randomness [4].

The C variable lets us to control the penalty for a wrong classification, therefore the contant $C > 0$ determines the trade-off bewteen the flatness of the function and the amount up to which deviations larger than ξ are tolerated. So larger C value means greater penalty of the errror.

Non-linear problems, Kernel solution

A problem can be defined in a space of finite dimensions, but often it happens that the sets to be distinguished are not linearly separable in that space. For this reason it was proposed that the original space of finite size be mapped into a space with a larger number of dimensions, presumably making it easier to find a separation in this new space.
To resolve non-linear problems the utilization of the kernel method is required. The kernel method's ideas consist of creating a linear combination of the original features to map them into a higher dimensional space using a function $\Phi(\cdot)$, where the data are linearly separable. The figure 2.7 is a graphical rappresentation of how a kernel method works: at the begging

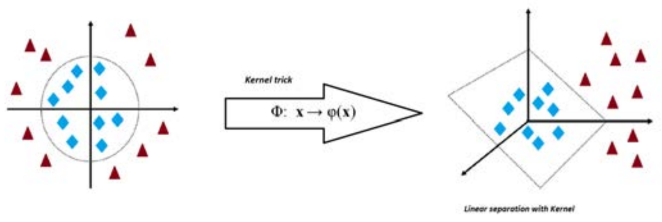

Kernel trick
$\Phi: x \rightarrow \varphi(x)$
Linear separation with Kernel

Figure 2.6: Kernel transformed

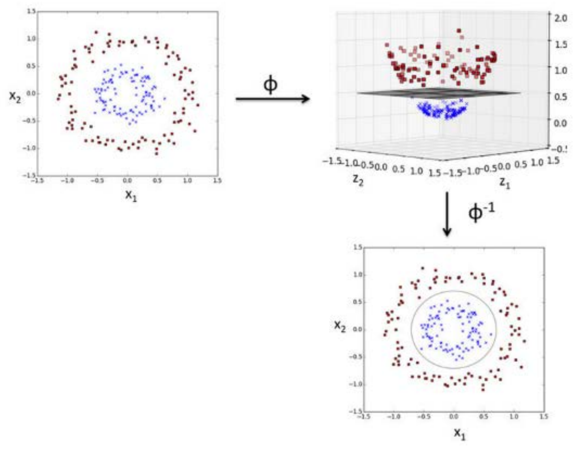


Figure 2.7: Step of kernel method

we have a bidimentional space, which becomes a tridimentional space where the two classes are linear separable through a projection:

$$\Phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, {x_1}^2 + {x_2}^2)$$

This trick lets us to use a linear hyperplane to divide the two classes, even if our original data aren't linearly separable. The main problem of this method is that more dimensions we have, the more the computational cost raises. To avoid it is better change the product $x^{(i)T}x^{(j)}$ into $\Phi(x^{(i)})^T\Phi(x^{(j)})$. So to avoid the computational cost of explicit calculation, we define the Kernel function:

$$k(x^{(i)}, x^{(j)}) = \Phi(x^{(i)})^T\Phi(x^{(j)})$$

This process is essential to allow the SVM or SVR to work: these algorithms work by creating a linear separation between two input classes, but if the input space is not linearly separable the algorithms would not reach a result. It is therefore necessary to use the kernel method to be able to divide them into two groups.

One of the most used kernel function is the Radial Basis Function (kernel RBF), which are defined as:

$$k(x^{(i)}, x^{(j)}) = exp(-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{2\sigma^2})$$

It is possible rewrite the exquation as:

$$k(x^{(i)}, x^{(j)}) = exp(-\gamma \left\|x^{(i)} - x^{(j)}\right\|^2)$$

with $\gamma = \frac{1}{2\sigma^2}$ which is a free parameter that can be optimizated. If we increase the γ value, we are using a decisional boundary softer. It is important to not choose a γ value too big, to avoid to incur into overfitting of the data.

2.6 Decision tree and RFR

2.6.1 Structure of a decision tree

A decision tree is a flow-chart structure which take decisions based on the response of a series of question as show in figure 2.8. The same structure is used for the numerical cases.

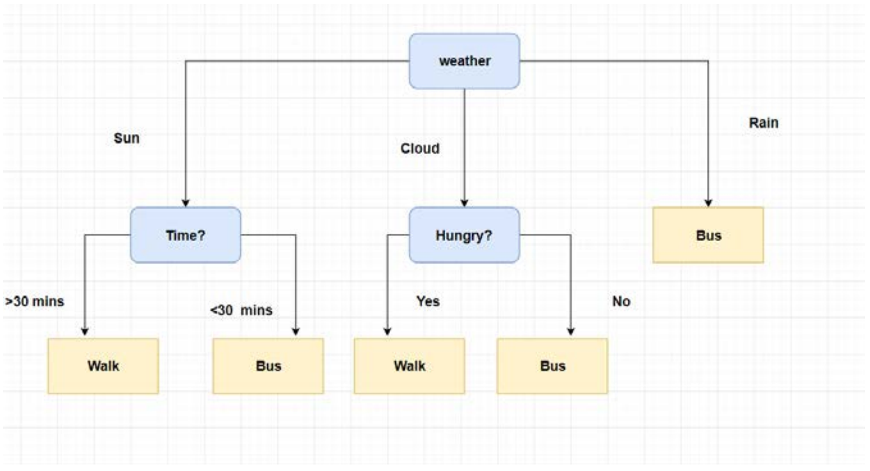


Figure 2.8: Example of decisional tree

Based on the training dataset features, the decision tree model learns a series of question to define (by inference) the labels of the classes. The working schedule is easy, we give at the tree's root the input data and it will follow the path to the leaf which gives the maximum information gain, called "IG". In the pratical case is important to limit the depth of the tree to avoid the overfitting phenomenon, which cause a wrong classification of the new input data.

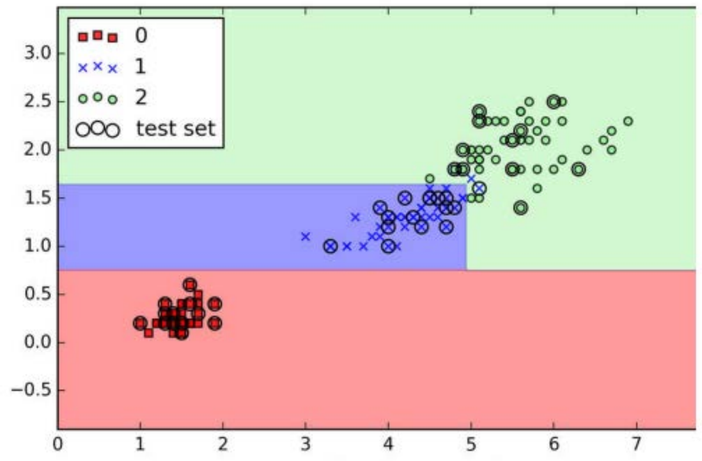


Figure 2.9: Example of decisional tree classificaion [4]

The information gain is a synonym for Kullback-Leibler divergence, which is a measure of how one probability distribution is different from a second reference probability distribution [22].
So IG is define as:

$$IG(D_f, s) = I(D_f) - \sum_{j=1}^{m} \frac{N_s}{N_f} I(D_s)$$

where:

- D_f and D_S are father's and son's dataset
- I is a measure of impurity
- N_f is the total number of samples of the father node
- N_s is the total number of samples of the j-son node

Therefore the Information gain is the difference between father's impurity and the summ of the son's impurity: so lesser impurity in the son's node means higher IG. In the binary case, there are only two sons from one father (right and left son) so the formula is easier to compute:

$$IG(D_f, s) = I(D_f) - \frac{N_{left}}{N_f} I(D_{left}) - \frac{N_{right}}{N_f} I(D_{right})$$

Usually are used three impurity criteria: Gini's impurity, entropy or classification error.

The index of heterogeneity, or impurity, of Gini is an index of heterogeneity for qualitative variables. It offers a measure of the heterogeneity (homogeneity) of a statistical distribution starting from the relative frequency values associated with the κ mode of a generic variable χ [23].
This means that if the data are distributed heterogeneously on all the κ modes of χ (that is, if the modes have similar numbers or, in the case of maximum heterogeneity, the same), the Gini index is high , vice versa, in the case of a uniform frequency distribution, the index will be (low percentage).
The Gini index is defined as follows:

$$I = 1 - \sum_{i=1}^{k} {f_i}^2$$

where the f_i are the relative frequencies of κ mode of χ [23]. So the impurity of Gini can be considered as a criterion to minimize the probability of a wrong classification.

In the entropy criterion, the impurity is 0 if all the samples of a node belong to the same class and it is maximum if we have a uniform distribution in the classes. So the entropy criterion seeks to maximize mutual information within a tree [4].
In practice, both the entropy criterion and the Gini criterion provide similar results.

The classification error is defined as:

$$I_E = 1 - maxp(i|t)$$

where $p(i|t)$ are all non-empty classes.
This criterion is less sensitive to changes in the probability that nodes belong to certain classes, so it is recommended for pruning branches, but not for growing trees.

2.6.2 Random Forest Regressor

The random forest is made by a decisional trees group, in which every tree gives a prediction of the output label, and at the end the most voted label for a specific input are assigned to it. Basically the random forest technique is a combination of weak learning processes to build a stronger and more robust final model, which gives a less general error and is less susceptible to the overfitting problem.
The main advantage of the Random forest model is more resistant to input's background rumors than the decision tree, so they need less tuning on the parameters.
The only parameters which must be check is the number of the tree which we are using, more trees means more accuracy, but more computational costs.

In the next chapters an SVR model and an RFR model will then be used to try to make short-term power predictions and compare the results.
The SVR algorithm was chosen as a test, since it responds well to jobs with short-term forecasts and can be better adapted to power forecasts, as it manages to manage the high dimensionality of the input data, thanks to the Kernel method. For the same reason, that is the high dimensionality of the data, a comparison RFR is used, since it is a model that manages to manage well even large dimensional spaces, at the cost of a higher computational request, compared to the SVR.

Chapter 3

Case of study and Data profile

3.1 Introduction

In these last years the European Union has been a move towards smarter electricity networks where Advanced Metering Infrastructure (AMI) has been installed. Smart metering is an important tool to achieve the EU energy policy goal of a modern, competitive and climate-neutral economy by 2050. These rules define three main goals: 20% cut in greenhouse gas emissions (compared to 1990 levels),20% of the energy needs to be obtained from renewable sources and a 20% improvement in the energy efficiency. These objectives were set in 2007 by EU leaders and they are the main goals to a sustainable and inclusive European growth for the 2050. The electronic meters are positioned between the users and the energy distribution grids to give back usefull information to distributors and sellers, improving the service offered by suppliers and the management of the power and enrgy flow in the network. The smart meters allow us to create a two-way communication channel to transport data from the users to the data collection and management center. So with this tecnology is possible to better monitoring and control the network parameters and also give back in real time some important data to the users as electricity tariffs and electrical parameters [24]. This technology is particulary involved in the residential sector, because thank to smart metering is possible to extrapulate a lot of detailed data about electricity consumption, which can be used to forecast maximum dayly, weekly or monthly peaks of power deamnd for example.

The smart metering method was carried out by Commission for Energy Regulation (CER) between 2009 and 2010 (over the period 1st July to 31st December); they installed smart meters in over 4000 residential houses in Ireland.

3.1.1 Smart metering method and data collection

The Commission for Regulation of Utilities (ex CER, founded in the 1999, changed name in the 2017) is the control and regulation center for the electricity and natural gas sectors in Ireland, it is a organization which works within the framework of national and EU energy policy which aim to create a single European electricity market that best meets the needs of Europe's energy consumers. The energy data were recorded with a frequency of half hour and a series of questions are summited to the end users to define their habits. All the data are recorded as anonymus data, to protecct the personal customer informations.

The next table report an example of how the data were collected: in the first column there is the date of the year, in the second the energy value at the corrispondent time and at the third column there is the ID of the customer. The date is stored with a particular resolution: the first three numbers are the day of the year and the last two digit are the day interval in which the data were acquired. So a relevation done at the midnight of the first January will be reported as : 001(first day of the year) 01(first interval of the day). The data from the 00.31 till 01.00 of the same day (first of January) will be stored as 00102 and so on.

time	energy value	ID
19501	0.958	1005
19502	0.885	1005
19501	0.999	1111
19502	0.850	1111

Table 3.1: Example of data

Therefore the data start from the 19501 of the 2012 and finish at the 73048. So the first data are from the midnight of the 14 July of the 2012 and the last one is from 23.30 of the 31 December 2013. At these data will be joined the ambient and the PV production data of the 2014 in Ireland.

3.2 Network considered

The generic urban model used for the simulations is shown in 3.1 and in the Table 3.2 are reported the data used.

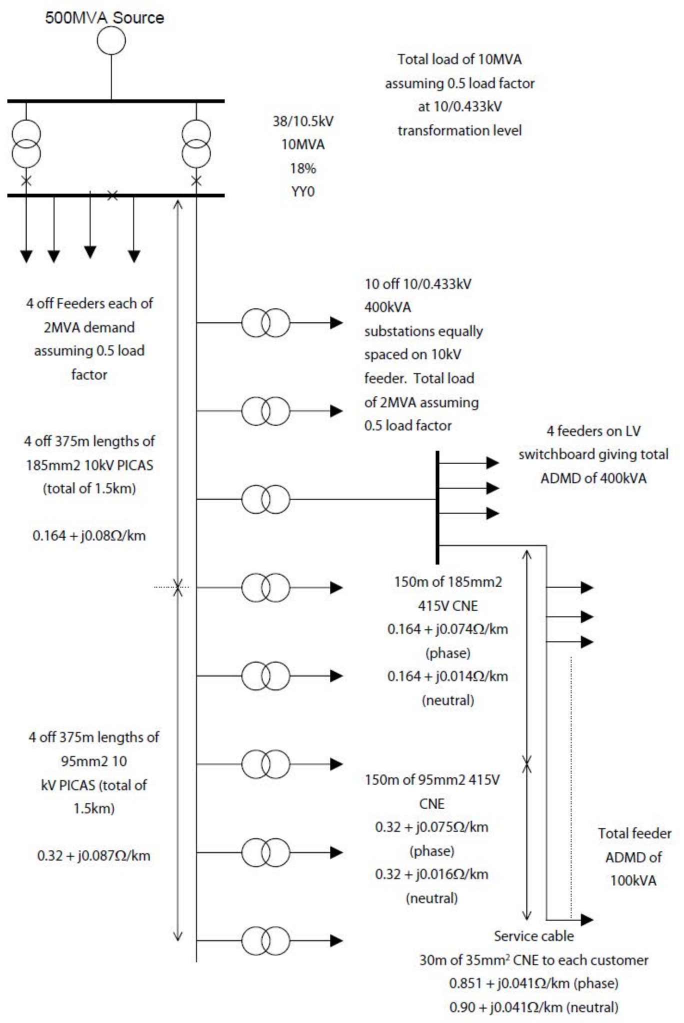

500MVA Source
Total load of 10MVA assuming 0.5 load factor at 10/0.433kV transformation level
38/10.5kV 10MVA 18% YY0
4 off Feeders each of 2MVA demand assuming 0.5 load factor
10 off 10/0.433kV 400kVA substations equally spaced on 10kV feeder. Total load of 2MVA assuming 0.5 load factor
4 off 375m lengths of 185mm2 10kV PICAS (total of 1.5km)
0.164 + j0.08Ω/km
4 feeders on LV switchboard giving total ADMD of 400kVA
150m of 185mm2 415V CNE 0.164 + j0.074Ω/km (phase) 0.164 + j0.014Ω/km (neutral)
4 off 375m lengths of 95mm2 10 kV PICAS (total of 1.5km)
0.32 + j0.087Ω/km
150m of 95mm2 415V CNE 0.32 + j0.075Ω/km (phase) 0.32 + j0.016Ω/km (neutral)
Total feeder ADMD of 100kVA
Service cable 30m of 35mm² CNE to each customer 0.851 + j0.041Ω/km (phase) 0.90 + j0.041Ω/km (neutral)

Figure 3.1: Network used in the simulation [24] [25] [26]

For the detailed description of the network, please refer to the thesis work of the engineer Beccaria [24] [25] [26]. Below is a brief summary of the original network layout taken from [24] [25] [26]:

1. Four of the feeders are modelled as simple lumped loads whilst the fifth feeder is represented in full detail.
2. There is a total of 144 domestic single-phase house loads, distributed equally between the feeder cables.
3. Three-phase and balanced loads will be considered. Neutral wire will therefore not be considered in the model.
4. Three of the 400V feeders are represented as simple lumped loads with only the fourth being represented in detail.

Component	**Description**	**Values**
10kV detailed Feeder Circuit	• Five feeder circuit comprising 8 x 400kVA substations. • Feeder cable comprises 1.5km of 185mm^2 3 core PICAS plus 1.5km of 95mm^2 3 core PICAS • 400KVA substations distributed equally along 3km feeder	• 185mm^2 Cable parameters: $0.164 + j0.080\Omega/km$ • 95mm^2 Cable parameters: $0.32+j0.087\Omega/km$
10/0.433kV Substation	• Comprises one 400kVA transformer • Four outgoing 400V 3 phase feeders • ADMD of each feeder is 100kVA • 144 customers supplied	
10/0.433kV Transformer	• 400kVA • 5% impedance • Dy11 windings • X/R ratio of 15 • Taps set at 0% on HV side • Off load ratio of 10/0.433kV	
400V Detailed Feeder	• Feeder comprises two segments of cable, 150 m of 185 mm^2 CNE and 150 m of 95 mm^2 CNE cable • 36 customers distributed evenly along each feeder • Customers are distributed evenly across three phases. Overall balanced load is considered	• 185mm2 Cable parameters: $0.164 + j0.074\Omega/km$ (phase) $0.164 + j0.014\Omega/km$ (neutral) • 95mm^2 Cable parameters: $0.32+j0.075\Omega/km$ (phase) $0.32 + j0.016\Omega/km$ (neutral)
Individual customers	• Power factor equal to 1.0 • 30 m of service cable, 35 mm^2 CNE	• Cable parameters: - $0.851 + j0.041\Omega/km$ (phase) $0.9 + j0.041\Omega/km$ (neutral)

Table 3.2: Table of network values [24] [25] [26]

Chapter 4

Pre-processing data and SVR, RFR models

4.1 introduction

It has chosen to use Python 3.0 to write the model, to be able to use the sktlearn library, pandas and numpy library, which allows to manage huge chunck of data very quickly and store data into: pickle form for the Running test and csv/xlsx form to transport the result into excel. Libraries are sets of functions and classes that facilitate the writing of complex programs, giving the programmer pre-fabricated packages to use to code programs.

4.2 Data preprocessing

As we so in chapter 2, before to use our dataset, is import to preprare and shape it in the better which let us have the best accuracy on the prediction. The process is divided into 4 main point:

1. *Choice of only residential ID*
2. *Add solar features, as PV production, wind speed, irradiation, temperature*
3. *Isolate singular customer*
4. *Lumped loading generation*

Here there is a flow chart of the process:

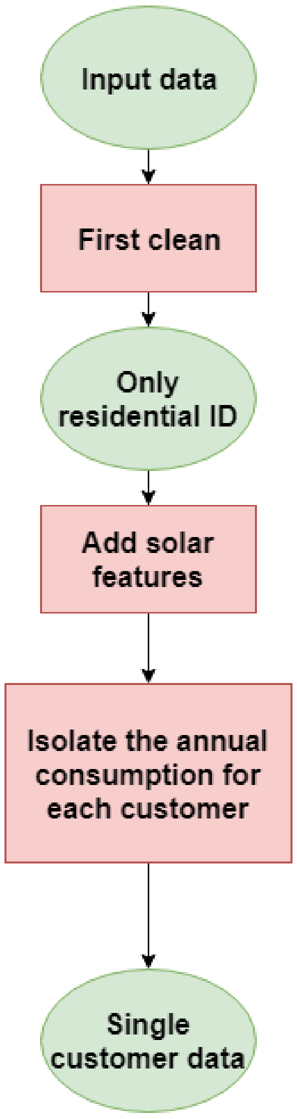


Figure 4.1: Flow chart of the Pre-processing data

4.2.1 Choice of only residential ID

From the total data we need to extrapolate only the residential customers to feed the grid model. I took from the SME and Residential allocations only the residential ID, simply loading the csv file, and holding the ID only if in the Code column i found 1, which means residential customer. After that i save the clean dataframe into pickle file, named residential ID pickle.

The Pickle is an internal Python data saving format that allows you to serialize the saved data and transform it into a python expression, which can be easily managed and manipulated using software that has access to this language. In this way it allows to load and save large amounts of data in less time, allowing intermediate rescue operations to be performed without affecting the efficiency of the program .

Due to the large amount of data, it is preferred let the orginal customers file separeted and i create a while loop system to clean simultaneously all six group of data, just launching in parralel the following script, which is going to read the ID inside the residential ID pickle, that was made before, and compare it to the ID inside the energy datasets. The next figure is a explanetion of the flow chart of the program, which is formed by 3 while cycle that can select only the customers which we had selected before.

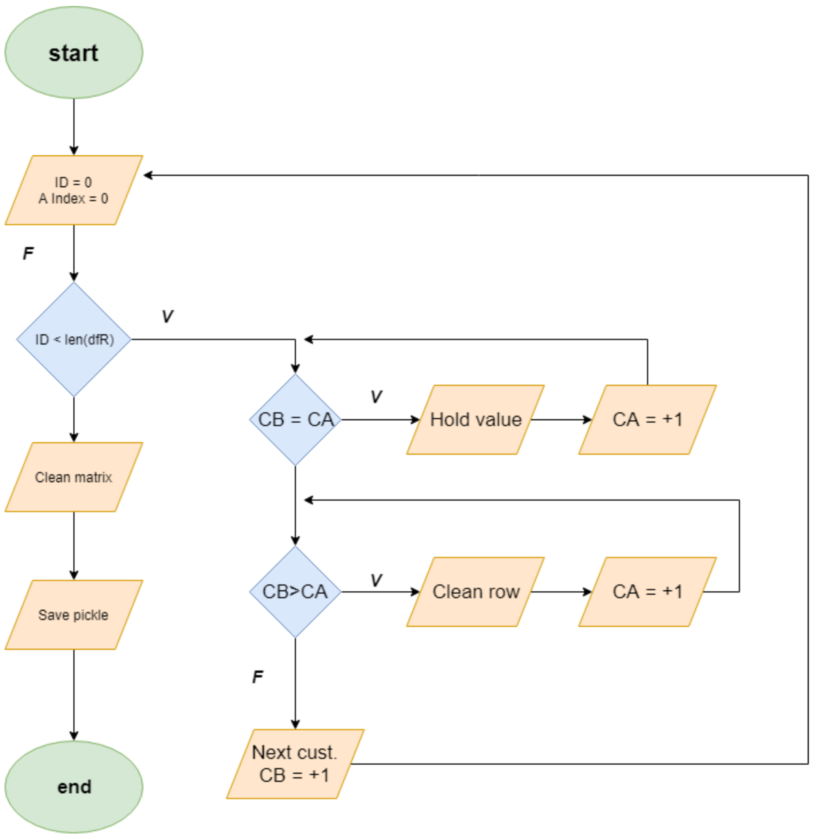


Figure 4.2: Flow chart of the first cleaning

The first block of the program is used to load the library pack.

```
#%%
# Load library
import pandas as pd
import numpy as np
import os
import time
import datetime
# Path to set working directory
PATH = "/Users/d18123388/Desktop/Codici lavoro"
```

The second block load the customer file and the residential ID pickle and sort both matrix for the ID customers, and reset the dataframe index.

```
#%%

# Define the correct A and B customers
```

```
        # B is the ID refferal
        # A are the customers
        dfCA= pd.read_csv("/Users/d18123388/Desktop/
            progetto erasmus/Thesis/samuele dublin
            project/not_clean_data/origin/File6.txt",sep
           =" ")

        # Set directory
        os.chdir(PATH)

        dfCB = pd.read_pickle('residential_ID.pickle')

        #%%
        #Re-oder matrix for customer_ID
        dfCB = dfCB.sort_values('residential_ID')
        dfCA = dfCA.sort_values('customer_id')

        # reset working index
        dfCB = dfCB.reset_index()
        dfCB = dfCB[['residential_ID']]
        dfCA = dfCA.reset_index()
        dfCA = dfCA.drop('index', axis=1)
```

The second block is the function block: first are definited the counters, and the indexs for matrix A (CA) and B (CB). After that, the main while-cycle are inizialited, and another two while-cycle are initialized from it. The first cycle checks if the ID in the big customer matrix are inside the reference customer ID, which we want to hold. If it find it, the value on big datamatrix are held, instead if is not found the function goes on to the next reference index. The process goes on until the end of A matrix or B matrix (reference matrix). At the end the new dataframe with only residential customers are saved into pickle form, with the name that we give it in the input line at the lunch of the program.

```

        #%%
        # Ask the name of the save-file
        nome = input("Nome del file che stai salvando!:
             ")

        #%%
        # Function
        # OP = optional
        def Clean_listA():
```

```

# Set index for the while condition

cb = int(input('Insert ID_B=0 to start, digit 0: '))
ca = 0
print("Start cleaning process!")
counter = 4999
countera = 0
counterb = 0
counttot=0

while cb < len(dfCB) and ca < len(dfCA):
# First condiction of first while cycle
counter += 1
if counter == 5000:
ts = time.time()
st = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
print("Data: " + st)
print("counter: " + str(counter))
counter = 0

while  ca < len(dfCA) and dfCB['residential_ID'][cb] == dfCA['customer_id'][ca]:
# OP: print('Hold Value! ')
ca += 1
countera += 1
counttot += 1
if countera == 100000:
ts = time.time()
st = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')
print("Data: " + st)
print("counter a : " + str(countera) +"\t"+str(counttot))
countera = 0

# second condition of the decisional tree

while ca < len(dfCA) and dfCB['residential_ID'][cb] > dfCA['customer_id'][ca]:
# OP:print("Cleaning!")
dfCA['customer_id'][ca] = np.nan
```

```
        ca += 1
        counterb += 1
        counttot += 1
        if counterb == 100000:
        ts = time.time()
        st = datetime.datetime.fromtimestamp(ts).
            strftime('%Y-%m-%d %H:%M:%S')
        print("Data: " + st)
        print("counter b : " + str(counterb) +"\t"+str(
            counttot))
        counterb = 0
        # Brake while cycle and go on next customer!
        if ca < len(dfCA) and dfCB['residential_ID'][cb
            ] != dfCA['customer_id'][ca]:
        print("prossimo cliente")
        cb += 1


        # Brake while cycle/// Finished to read the
            matrix

        print("End reading, going to drop Nan")
        dfCA.dropna(thresh=3, inplace = True)
        # Save to pickle the new-dataframe!
        dfCA.to_pickle('df'+nome+'_clean.pickle')
        return

        # Calling the fucntion
        Clean_listA()

        #%%
        # Load cleaned matrix, to check it
        dfCA_pulita = pd.read_pickle('df'+ nome +'
            _clean.pickle')
```

The next script is used to join the six different matrix into one Big data-matrix, with the data of all 4000 customers from the midnight of the 13/14 July of the 2012/2013 till the 31 december of 2014.

```
        #%%
        # Import library pack
        import pandas as pd
        import os

        #%%
        # Set working path
        PATH = "/Users/FX504GE - EN069T/Desktop/Codici
            lavoro/partial pickle"

        os.chdir(PATH)

        #%%
        # Load the first starting clean dataframe
        df_tot = pd.read_pickle('dfFile1_clean.pickle')

        # Cyle For to merge the clean pickle
        for numb in range(2,7):
        df_pickle = pd.read_pickle('dfFile'+str(numb)+'
            _clean.pickle')
        df_tot = pd.concat([df_tot, df_pickle],
            ignore_index =True)

        #%%
        # Saving the total matrix
        df_tot.to_pickle('df_tot.pickle')

        #%%
        # Check of the Big-datamatrix
        df = pd.read_pickle('df_tot.pickle')
```

4.2.2 Add solar features

Now to add more features to the data i took PV generation data from (metti da dove lo hai preso.. chiedere a Keith). The PV solar dataset strats from 1 january (2014), so i align the solar data con the right bigdata matrix, which starts from 13 July.

```
        #%%
        # Load library
        import pandas as pd
        import os
        # Set working path
        PATH1 = "/Users/d18123388/Desktop/progetto
            erasmus/Thesis/samuele dublin project/
            not_clean_data/origin"

        #%%
        # Define the function
        def preprocess_Solar_matrix():

        # Load PV solar matrix
        os.chdir(PATH1)
        df1 = pd.read_csv("Solar_Power_Matrix.csv", sep
            =',')

        # Hold the data of one year
        df1 = df1.loc[:17519]

        # Select and Extract the last 8200 values
        # of PV solar matrix
        df8200 = df1.loc[9310:]

        # Merge the data to create
        # one and half year of solar data
        df = pd.concat([df8200, df1], ignore_index =
            True)

        # Change comma with dot, to transform data
        # to float form in next section!
        df = df.stack().str.replace(',','.').unstack()
        df.to_pickle("Solar_matrix.pickle")
        return

        # Launch the function
        preprocess_Solar_matrix()
```

Now is time to merge the two dataframe to create the final matrix. First part of the script loads the two dataframes. The second block changes the value of solar matrix in float type number (spiegazione di cosa sono). The last block joins the two dataframes, using their index, into a bigger one, the final dataframe, with 8 columns : time', 'energy target variable', 'tamb', 'vs', 'GF', 'Tc', 'neta c','PV power'.

```

        #%%
        # Set the new directory
        PATH ="/Users/d18123388/Desktop/Codici lavoro"
        os.chdir(PATH)

        #Load the big datamatrix
        df_energy = pd.read_pickle("partial pickle/
        energy_tot_in_day.pickle")
    df_Solar = pd.read_pickle("Solar_matrix.pickle
        ")

    #%%
    # change in float the value of solar matrix!
    df_Solar.tamb = df_Solar.tamb.astype(float)
    df_Solar.vs = df_Solar.vs.astype(float)
    df_Solar.GF = df_Solar.GF.astype(float)
    df_Solar.Tc = df_Solar.Tc.astype(float)
    df_Solar.neta_c = df_Solar.neta_c.astype(float)
    df_Solar.power = df_Solar.power.astype(float)

    #%%
    # Combine the two datamatrix
    df_final = df_energy.join(df_Solar)
    df_final.to_pickle("Final total matrix/
        Final_Datamatrix.pickle")
    df_final.to_csv("Final total matrix/
        Final_Datamatrix.csv", sep=',')
```

This matrix can be used to generate a single Lumped customer which should represent the total load which the transformer sees at the Low Voltage side.

4.2.3 Isolate singular customer

As now, we have a Big datamatrix with all 4225 customers in one big dataframe which cover one and half year of energy consumption and solar production and ambiental data, as temperature, irrandiance, wind speed. The next program takes 144 random customer, from the residential ID dataframe, and extrapolates them to makes single customer dataframe and datamatrix.
First the program load the working library with the working directory; after that it load the Big datamatrix, with all customers and all datas, and set the customers as index for the dataframe . To extrapolate 144 random customers, first i create a new vector filled with the numbers from 0 to 4225. After a shuffle this vector is set as the new index or the residential matrix and then all the dataframe is sorted by the new index, so i shuffled the residential ID.
Done this process, the program takes the first 144 rows of the dataframe, and save it into a pickle wich is the customers ID pickle.

```
    #%%
    # Load the matrix
```

```
import os
import pandas as pd
import random
import numpy as np
PATH = ("/Users/FX504GE - EN069T/Desktop/Codici
    lavoro/partial pickle")
os.chdir(PATH)

# Load Big datamatrix
df = pd.read_pickle("Big_datamatrix.pickle")

#%%
# Strutturare l'estrazione dei customer!!
# set customers  as index

df.set_index("customer_id", inplace=True)
df.head()

#%%
#------>>> Section used to generate the 144
   customers to do the test!

# generare 144 customers, con ID = 1002 - 7443
   , ma randomico


# Load residential ID
df_customers = pd.read_pickle("Residential_ID.
   pickle")
# reset origial index
df_customers.reset_index(inplace = True)

# Generate a vector from 0 to 4224
arr = np.arange(4225)
# Shuffle the generated vector, to have random
   order
np.random.shuffle(arr)

# transform my vector into a dataframe
df_arr = pd.DataFrame(arr)
# join the new column with customers dataframe,
# to transform later in its index
df_customers = df_customers.join(df_arr)

```

```
        # Drop unneeded column
        df_customers = df_customers [["ID", 0 ]]

        # set the new index and sort the matrix by it
        df_customers.set_index( 0 , inplace = True)
        df_customers.sort_index(inplace = True)

        # Take the first 144 customers
        df_customer =  df_customers.loc[:144]



        #create a pickle of the dataframe of the random
            customers
        os.chdir(PATH)
        df_customer.to_pickle("df_customer_ID.pickle")
```

The second block is the function which: reads the customers in the customer ID pickle, reads the Big datamatrix, extrapolates one by one the customers of the customer ID pickle, and after join the single customer dataframe with the PV solar data, saves them into a specific folder.

```

        #%%
        # Load customers list to extrapulate
        #their annual energy comsuption
        df_customers = pd.read_pickle("df_customer_ID.
            pickle")

        # Define the function
        def Customer_generator():
        # set the for cycle
        for ID in df_customers['ID'] :

        # Extrapolate the single customer Dataframe
        df_customer_ID = df.loc[float(ID)]

        # Reset the index of the new df and prepare to
            join it
        df_customer_ID.set_index("time", inplace=True)
        df_customer_ID.reset_index(inplace = True)
        df_customer_ID.sort_values("time", inplace =
            True)
        df_customer_ID.reset_index(inplace = True)
        df_customer_ID = df_customer_ID[...
        ...["time", "energy_target_variable"]]

        # Load PV solar dataframe
        os.chdir("/Users/FX504GE - EN069T/Desktop/
            Codici lavoro/Final total matrix")
        df_Solar = pd.read_pickle("Solar_matrix.pickle"
            )

        # Join the dataframes
        df_ID = df_customer_ID.join(df_Solar)

        # Save the result
        os.chdir("/Users/FX504GE - EN069T/Desktop/
            Codici lavoro/Customer matrix")
        df_ID.to_pickle("df_customer_" + str(ID) + ".
            pickle")
        return

        #%%
        # Launch generator
        Customer_generator()
```

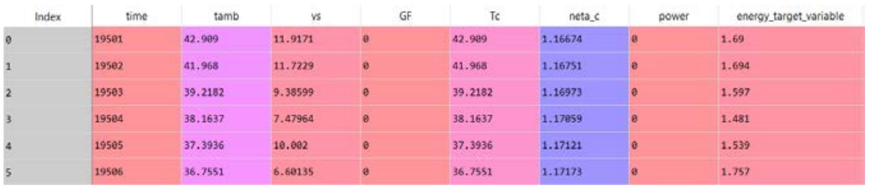
4.2.4 Lumped loading generation

To match the requirements od the simulink model, it is necessary to cluster six by six the 144 customer into 24 lumped loads. This program first loads the library pack and the customer ID pickle with the ID list which we made before; after that it uses a double for cycle, to read the ID, load six customer dataframe, joins them toghether, sets index and sorts for "time" column to summ by time the data. At the end, it resets the original index and save the lumped loads into a selected folder.

```
        #### Group 144 customers in 24 ####

        #%%
        # Load pack library
        import os
        import pandas as pd

        #%%
        # Set working directory
        PATH = ("/Users/FX504GE - EN069T/Desktop/Codici
        lavoro/partial pickle")
    os.chdir(PATH)

    # Load customers ID
    df_customers = pd.read_pickle("df_customer_ID.
        pickle")

    #%%
    # Cycle to produce the Lumped customer matrix
    # grouped by 6

    # Make an empty matrix
    df = pd.DataFrame()

    # Setting counters
    Counter0 = 0
    Counter1 = 6

    # Set a double for cycle
    for NUM in range(1,25,1):

    for x in range(Counter0,Counter1,1):

    # BLock to load matrix
    ID = df_customers["ID"][x]
    os.chdir("/Users/FX504GE - EN069T/Desktop/
        Codici lavoro/Customer matrix")
    df_summ = pd.read_pickle("df_customer_" + str(
        ID) + ".pickle")

    # Join the different dataframe in group of 6
    df = pd.concat([df, df_summ], ignore_index =
        True)

    # Increase the counter
    Counter0 = Counter0 + 6
    Counter1 = Counter1 + 6

    # Sort data per time!
    df.sort_values('time', inplace=True)
    df.reset_index(inplace = True)
    df.drop('index', axis=1 , inplace =True)

    # Sum by time columns
```

```
        df_tot = df.groupby("time").sum()
        df = pd.DataFrame()

        # Reset index, to get "time"
        df_tot.reset_index(inplace = True)

        # Save it
        os.chdir("/Users/FX504GE - EN069T/Desktop/
            Codici lavoro/Customer matrix/Lump Customer"
            )
        df_tot.to_pickle("Lump_" + str(NUM) + ".pickle"
            )
```

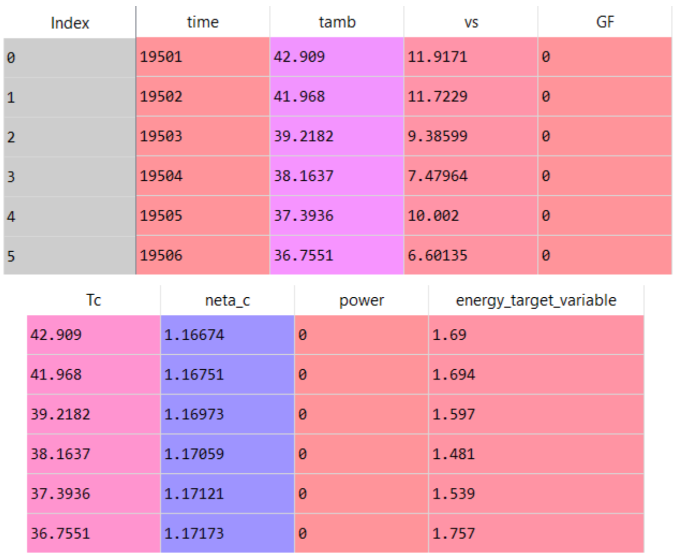
At the end of this process, i have 24 lumped loading which i can use as input data to feed my prediction models.
Here there is an example of the final result:



Index	time	tamb	vs	GF	Tc	neta_c	power	energy_target_variable
0	19501	42.909	11.9171	0	42.909	1.16674	0	1.69
1	19502	41.968	11.7229	0	41.968	1.16751	0	1.694
2	19503	39.2182	9.38599	0	39.2182	1.16973	0	1.597
3	19504	38.1637	7.47964	0	38.1637	1.17059	0	1.481
4	19505	37.3936	10.002	0	37.3936	1.17121	0	1.539
5	19506	36.7551	6.60135	0	36.7551	1.17173	0	1.757

Figure 4.3: Head of dataframe of single customer

In the next figure 4.4 there is a zoom of the 4.3; where you can better see the dataframe structure of individual customers. The first column represents the index, automatic assigned by python, while the subsequent ones are reserved for the actual data of the customer.
In order there are: detection date, environment temperature, wind speed, irradiance, solar panel cell temperature, solar panel efficiency, panel power output and finally the energy consumed at the time of detection.



Index	time	tamb	vs	GF	Tc	neta_c	power	energy_target_variable
0	19501	42.909	11.9171	0	42.909	1.16674	0	1.69
1	19502	41.968	11.7229	0	41.968	1.16751	0	1.694
2	19503	39.2182	9.38599	0	39.2182	1.16973	0	1.597
3	19504	38.1637	7.47964	0	38.1637	1.17059	0	1.481
4	19505	37.3936	10.002	0	37.3936	1.17121	0	1.539
5	19506	36.7551	6.60135	0	36.7551	1.17173	0	1.757

Figure 4.4: Dataframe zoom of a single customer

4.3 SVR model and RFR model

I built a single script in which i have both model,SVR and RFR, and giving to it a input dataframe, let us select the training and test day, giving as result a comparison between the output of the two model, with their score value, R^2, and MSE, Mean Square Error.
First of all the program loads the working library.

```python
#%%
# Library
import warnings
import os
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing, svm, model_selection
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing, svm, model_selection
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error
```

```
        import matplotlib.pyplot as plt
        from matplotlib import style
```

After that, it loads the working directory for the next steps.

```
        #%%
        # FIRST BLOCK: quality of life tools
        warnings.filterwarnings("ignore")

        # Set working directory
        # codici lavoro
        Codici_lavoro= "/Users/FX504GE − EN069T/Desktop
           /Codici lavoro"

        # Final total matrix
        Final_matrix = "/Users/FX504GE − EN069T/Desktop
           /Codici lavoro/Final total matrix"
        # Customer matrix
        Customer_matrix =  "/Users/FX504GE − EN069T/
           Desktop/Codici lavoro/Customer matrix"
        # Lump Customer
        Lump_customer=  "/Users/FX504GE − EN069T/
           Desktop/Codici lavoro/Customer matrix/Lump
           Customer"
```

Here the program loads the selected matrix and is required that the desired output must be set as the last column of the datafrtame. Then i defined the function to make the predictions. When launched, it asks to select the training day and the target prediction day, by giving in input a number, knowing which 0 = 14 july 2013*; is also required to return how many columns do the dataframe have. The input line command transforms the selected number in the correct time-index frame and extrapulates the selected training day and target day.

```
        #%%
        # SECOND BLOCK: Extract the df from the
           original matrix
        os.chdir(Final_matrix)
        df = pd.read_pickle("Final_DatamatrixV2.0.
           pickle")

        #−−−> set the column of the prediction as the
           last column!
        df = df [['time','tamb', 'vs', 'GF', 'Tc', '
           neta_c',
        'power', 'energy_target_variable']]

        #%%
        ##############################################
        # Define function

        def Jarvis_prediction():


        ''' Set the input comand for the choise of the
           day of training and predicted day'''

        td = int(input("Select training day: "))
        gg = td*48
        pd = int(input("Select test day: "))
        ggp = pd*48

        ''' X and y for training with 70% rule, and
           these are the reference to all day train'''
        # set how much column do you have in the matrix
           ?
        cc = int(input("How much column do you have?: "
           ))
        X = df.iloc[gg:(gg + 48),0:(cc - 1)].values
        y = df.iloc[gg:(gg + 48),(cc - 1):cc].values

        # step 4 --> extract Xp,yp for a all day
           training with X,y and a all-day prediction
           with Xp,yp

        Xp = df.iloc[ggp:(ggp + 48),0:(cc - 1)].values
        yp = df.iloc[ggp:(ggp + 48),(cc - 1):cc].values
```

Now the program transforms our dataframe into a vector and only for the SVR is applyed a scaling function to the data.

```
        # step 5 --> transform X and y with
           standarscaling to have X_svr and y_svr,
        # for X_rfr and y_rfr just use a copy of X and
           y, and same for Xp,yp

        ''' For X '''
        X = np.array(X)
        X_svr = preprocessing.scale(X)
        X_rfr = X

        ''' For Xp '''
        Xp = np.array(Xp)
        Xp_svr = preprocessing.scale(Xp)
        Xp_rfr = Xp

        ''' For y '''
        y_svr = preprocessing.scale(y)
        y_rfr = y

        ''' For yp '''
        yp_svr = preprocessing.scale(yp)
        yp_rfr = yp
```

Here,are set the two predictor model, taken from the library. The SVR model is a standard model, meanwhile the random forest is choose with a number of 1000 decisional tree, using a mean square error decisional criteria, "random state" and n_jobs is set to parallelize the training process using all the possible cores of the computer.

```
        # FOURTH BLOCK: SVR and RFR analysis
        # step 1 --> call the SVR e RFR

        svr = svm.SVR()
        rfr =  RandomForestRegressor(n_estimators=1000,
        criterion='mse',
        random_state=1,
        n_jobs=-1)

        svr_day = svm.SVR()
        rfr_day =  RandomForestRegressor(n_estimators
            =1000,
        criterion='mse',
        random_state=1,
        n_jobs=-1)
```

Now the program prepares the training set and the test set, thank to $model_selection$ line, where it's chosen the names of SVR and RFR variables, have to divide the two sets by the test size value, which usually is set to 80% train and 20% test. To try to control the overfitting effects, in this case the SVR is set to 70%-30%, meanwhile the RFR is set to 60%-40%.

```
        # step 2 --> define train-set, test-set, cross-
            validation function
        # Allert: we need to do it only for 70%
            training-day and  30% prediction, if we use
            all-day we don't need

        '''X_svr and y_svr are scalend!'''
        X_train_svr, X_test_svr, y_train_svr,
            y_test_svr = model_selection.
            train_test_split(X_svr, y_svr, test_size
            =0.3)

        '''X_rfr and y_rfr could be used without
            scaling'''
        X_train_rfr, X_test_rfr, y_train_rfr,
            y_test_rfr =  model_selection.
            train_test_split(X_rfr, y_rfr, test_size
            =0.4)
```

The next section launchs the fit function to train the two models. For the first block i train the model with the splitted data, as saw before, instead the second block is a fit using all the day data to train the two models, in oder to have a all day training set and after that going to use another day as test set. Basically is a 50%-50% train and test size, with the double all total data, so instead of using a 48×8 dataframe, is used 96×8 dataframe.

```
        # step 3 --> fit the two models with day data,
            so 48 element each

        ''' For 70%-30% fit'''
        svr.fit(X_train_svr,y_train_svr)
        rfr.fit(X_train_rfr,y_train_rfr)

        ''' For all-day fit, is use their respective
            X_svr, y_svr and X_rfr, y_rfr as train set
            !!! '''
        svr_day.fit(X_svr, y_svr)
        rfr_day.fit(X_rfr, y_rfr)
```

The next section use the train data and test data to do the prediction and save them into new vector, to do later a score test.

```
        # step 4 --> predict the output array

        ''' train section prediction, 70%-30% day (for
            the rfr is 60%-40%, to control overfitting),
             using train set '''
        y_svr_train_pred = svr.predict(X_train_svr)
        y_rfr_train_pred = rfr.predict(X_train_rfr)

        ''' test section prediction,70%-30% day (for
           the rfr is 60%-40%, to control overfitting),
            using test set '''
        y_svr_test_pred = svr.predict(X_test_svr)
        y_rfr_test_pred = rfr.predict(X_test_rfr)

        ''' train section prediction with all-day
           training set '''
        yp_svr_train_pred = svr_day.predict(X_svr)
        yp_rfr_train_pred = rfr_day.predict(X_rfr)

        ''' test section prediction with all-day test
           set '''
        yp_svr_test_pred = svr_day.predict(Xp_svr)
        yp_rfr_test_pred = rfr_day.predict(Xp_rfr)
```

Here the program, use the predictions and the real data, to extract the meean square error of the prediction, which is scaled, and the score of the model.

```
        # step 5 --> Error analysis by MSE and R^2

        ''' SVR accuracy: MSE, mean square error, R^2
           for SVR, train on 70%, prediction on 30% of
           the day'''

        print('SVR-MSE train (70-30): %.3f, test: %.3f'
            % (
        mean_squared_error(y_train_svr,
           y_svr_train_pred),
        mean_squared_error(y_test_svr, y_svr_test_pred)
           ))

        print('SVR-R^2 train (70-30): %.3f, test: %.3f'
            % (
        r2_score(y_train_svr, y_svr_train_pred),
        r2_score(y_test_svr, y_svr_test_pred)))

        ''' RFR accuracy: MSE, mean square error, R^2
           for RFR, train on 70%, prediction on 30% of
           the day'''

        print('RFR-MSE train (70-30): %.3f, test: %.3f' % (
        mean_squared_error(y_train_rfr, y_rfr_train_pred),
        mean_squared_error(y_test_rfr, y_rfr_test_pred)))

        print('RFR-R^2 train (70-30): %.3f, test: %.3f' % (
        r2_score(y_train_rfr, y_rfr_train_pred),
        r2_score(y_test_rfr, y_rfr_test_pred)))

        ''' SVR accuracy: MSE, mean square error, R^2 for SVR, train on all-day, prediction on all-day'''

        print('SVR-MSE train (all-day): %.3f, test: %.3f' % (
        mean_squared_error(y_svr, yp_svr_train_pred),
        mean_squared_error(yp_svr, yp_svr_test_pred)))

        print('SVR-R^2 train (all-day): %.3f, test: %.3f' % (
        r2_score(y_svr, yp_svr_train_pred),
        r2_score(yp_svr, yp_svr_test_pred)))

        ''' RFR accuracy: MSE, mean square error,R^2 for RFR, train on all-day, prediction on all-day'''

        print('RFR-MSE train: %.3f, test: %.3f' % (
        mean_squared_error(y_rfr, yp_rfr_train_pred),
        mean_squared_error(yp_rfr, yp_rfr_test_pred)))

        print('RFR-R^2 train: %.3f, test: %.3f' % (
        r2_score(y_rfr, yp_rfr_train_pred),
        r2_score(yp_rfr, yp_rfr_test_pred)))
```

At the end, the program plots the residual between the real data and the predicted data, to give back an idea of how is distribuited the error.

```

        plt.scatter(y_rfr_train_pred,
        y_rfr_train_pred - y_train_rfr,
            c='black',
            marker='o',
            s=35,
            alpha=0.5,
            label='Training data')
            plt.scatter(y_rfr_test_pred,
            y_rfr_test_pred - y_test_rfr,
            c='lightgreen',
            marker='s',
            s=35,
            alpha=0.7,
            label='Test data')

            plt.xlabel('Predicted values')
            plt.ylabel('Residuals')
            plt.legend(loc='upper left')
            plt.hlines(y=0, xmin=-10, xmax=50, lw=2, color=
                'red')
            plt.xlim([-10, 50])
            # plt.tight_layout()
            #plt.savefig('slr_residuals.png', dpi=300)
            plt.show()

            return

            #Call the function
            Jarvis_prediction()
```

A the end we have a day-ahead prediction which can be used to fill our power matrix and make a power flow analysis with the help of matlab and simulink. Now it's important try to understand how much impact the accuracy of the prediction on the grid's losses.

4.4 Bench test matrix

The main aim of this project is to see how the accuracy of the predction model afflicts the total grid's losses, to understand how much accured must be our prediction to fit the minimum requirement for the grids. It is important to understand if the prediction could be dangerous for the grid health and for the power supply sistem.
To analyse these points i recovered the model built by Ing. Beccari, who built a simulink model to test the impact on battery energy storage sistem

into the network. For convenience it will be re-used the same simulink model to do the power flow analysis.

To try to understand how the R^2 impacts on the total losses error in our grid, i made a bench test matrix, to have a wide range of R^2 of one day.
Down below there are the scripts that make the bench test matrix to use into matlab and simulink model.
First section is a utility section for coding.

```
##### Creation Benchtest for losses analysis ######

#%%
import os
import pandas as pd

from sklearn.metrics import r2_score
import warnings

#%%
# FIRST BLOCK: quality of life tools
warnings.filterwarnings("ignore")

# codici lavoro
Codici_lavoro= "/Users/FX504GE - EN069T/Desktop/Codici lavoro"
# Final total matrix
Final_matrix = "/Users/FX504GE - EN069T/Desktop/Codici lavoro/Final total matrix"
# Customer matrix
Customer_matrix =  "/Users/FX504GE - EN069T/Desktop/Codici lavoro/Customer matrix"
# Lump Customer
Lump_customer=  "/Users/FX504GE - EN069T/Desktop/Codici lavoro/Customer matrix/Lump Customer"
```

The second section loads the lumped load energy values and stores them into one dataframe, 25730x24, where each column is a lumped load energy along the all year and half.

```
#%%
os.chdir(Lump_customer)

df = pd.read_pickle("Lump_1.pickle")
```

```
        df = df[["energy_target_variable"]]

        for NUM in range(2,25):
        print(NUM)
        df_join = pd.read_pickle("lump_"+ str(NUM) + ".
            pickle")
        df_join = df_join[["energy_target_variable"]]
        df_join.rename(columns = {"
            energy_target_variable": "energy_lump_" +
            str(NUM)}, inplace = True)

        df = df.join(df_join)

        #%%
        os.chdir(Final_matrix)

        df.to_pickle("Benchtest_matrix_complete.pickle"
            )
```

The next section isolates a single day of the year; in this case the first day inside our matrix, so the 14 July.

```
        #%%
        df1 = df.loc[:47]
        df1.to_pickle("Benchtest_matrix_true_value.
            pickle")
```

The section below loads the Original Bench test dataframe, which will be used to do the accuracy test later, and sum and subtracts a percentage of their value, starting from 1% up to 10%.

```

        #%%
        df = pd.read_pickle("
            Benchtest_matrix_true_value.pickle")
        df1 = df

        for NUM in range(1,11):

        df = df - df.multiply(NUM/100)
        df1 = df1 + df1.multiply(NUM/100)
        os.chdir("/Users/FX504GE - EN069T//Desktop/
            Codici lavoro/Benchtest")
        df.to_pickle("Benchtest_acc_-%"+ str(NUM) +".
            pickle")
```

Now using the true values of the original dataframe, the programs computes the R^2 values of the different new Bench test matrices and save them into pickle and xlsx.

```
        #%%
        # caricare matrice vera, estrarre una colonna vettore
        # usare vera a sinistra e quella modificata a destra
        # r2_score(vett_vero, vett_benchtest)

        os.chdir(Final_matrix)
        df = pd.read_pickle("Benchtest_matrix_true_value.pickle")
        df = df[["energy_target_variable"]]

        for NUM in range(1,11):
        os.chdir("/Users/FX504GE - EN069T//Desktop/Codici lavoro/Benchtest")
        df1 = pd.read_pickle("Benchtest_acc_+%" +str(NUM) + ".pickle")
        df1 = df1[["energy_target_variable"]]
        print(str(NUM))
        print(r2_score(df,df1))
```

```
        ACC = r2_score(df,df1) * 100
        df1 = pd.read_pickle("Benchtest_acc_-%" +str(NUM) + ".pickle")
        df1.to_pickle("Benchtest_acc="+ str('{0:.2f}'.format(ACC))+ ".pickle")
        df1.to_csv("Benchtest_acc="+ str('{0:.2f}'.format(ACC))+ ".csv", index = False)
        df1.to_excel("Benchtest_acc=" + str('{0:.2f}'.format(ACC)) + ".xlsx")
```

At the end of the process there are 20 test matrix to use into matlab and simulink model to see the impact of the prediction accuracy on the grid losses.

Chapter 5

Simulink model

5.1 introduction

This study regarding the impact of forecasts, created by ML algortims, on variable loads was developed using Matlab and a part of Simulink, Simscape Electrical (formerly SimPowerSystems and SimElectronics).
It provides component libraries for modelling and simulating electronic, mechatronic, and electrical power systems. It includes models of semiconductors, motors, and components for applications such as electromechanical actuation, smart grids, and renewable energy systems. In this case it was used to analyse the transmission and distribution of electrical power at the grid level.

5.2 Simulink model

The simulink grid is show in fig 5.1. The figure represents the network general model and configuration of the figure 3.1.
The model used for the powerflow test is borrowed from [24]: the matlab script "Losses with out battery" and the associated simulink model are used. In the model properties part there is a window to initialize the model functions, using Matlab scripts, containing "mat" format vectors to use as input in the simulation.
For model simulations are used phasor type simulation with a 48 seconds of time simulation. It's been chosen this time setting to match the input data setting, which are one value of power every 30 minutes, so 48 relevation for a day. So the time interval between the samples are fixed as $Ts = 2{\cdot}10^{-3}$.

Starting from the left side of the figure, it is possible to identify the 38 kV

power source, the HV/MV transformer and five derived feeders. Four feeders are represented as Lumped loads, meanwhile the fifth is described in full detail.
Regarding the blocks containing the line values, a Three-Phase Series RLC branch block was used where an RL branch type was selected and it is assumed that only balanced loads are presented.

In the figure 5.2 is shown a single feeder in detail.

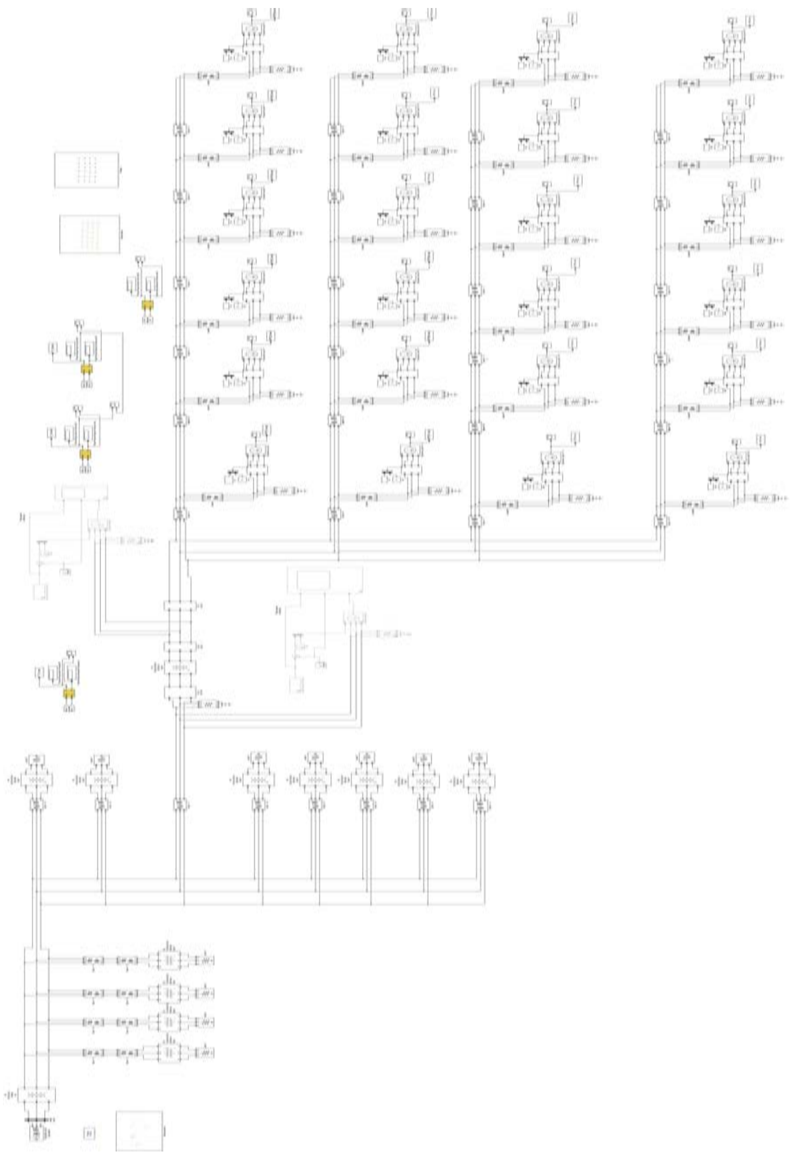


Figure 5.1: Simulink model

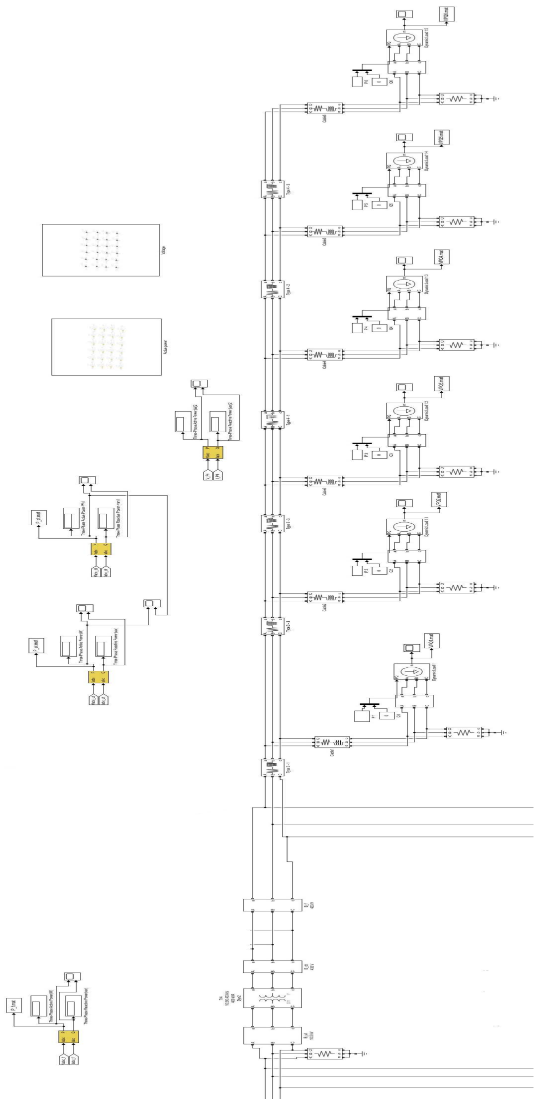


Figure 5.2: Simulink model detail

5.2.1 Three Phase Load Block

In figure 5.2 a feeder is shown in detail: from the left there is a MV/LV transformer between two three phase measurement blocks. after that there are another measurement block, which is usefull if we are goning to set a battery after our transformer.
These “Three-Phase V-I Measurement” block is used to measure instantaneous three-phase voltages and currents in a circuit. When connected in series with three-phase elements, it returns the three phase-to-ground or phase-to-phase peak voltages and currents. Moreover we can chose if we want per unit (pu) values or volts and amperes.
In this case the phase-to-ground voltage measure was chosen.The output values of voltage and current will be reprocessed to obtain the total power of the feeder and let us doing losses analysis.

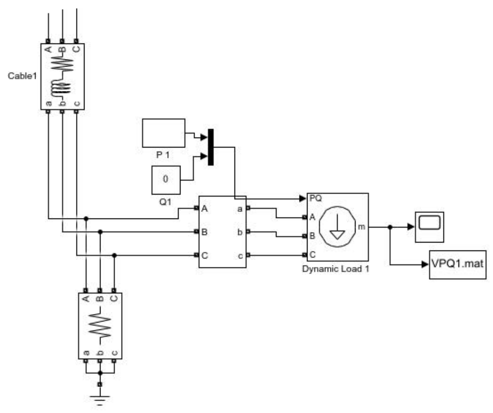
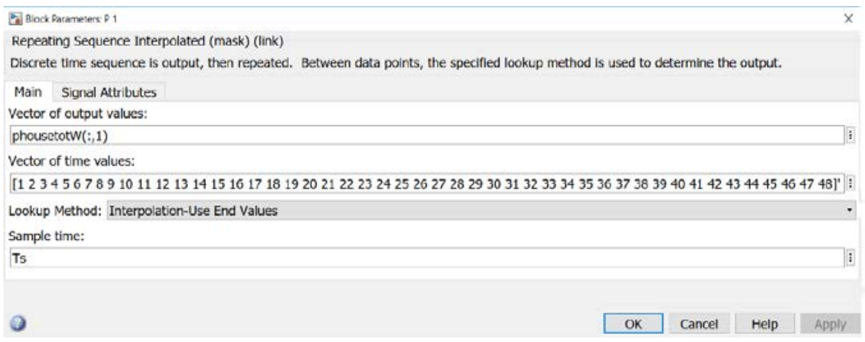


Figure 5.3: Single load model

In Figure 5.3 a single load of the feeder is represented: negative and zero-sequence currents are not simulated, the three load currents are therefore balanced, even under unbalanced load voltage conditions.
This block implements a three-phase, three-wire dynamic load whose active power P and reactive power Q vary as function of positive-sequence voltage. The active and the reactive power of the block are given by external input, using matlab vector defined as $[P, Q]$.
To do it is used the "Repeating Sequence Interpolated" block shown in figure 5.4. Based on the values in “Vector of time values” and “Vector of output values” parameters the block gives back to us a periodic discerete time sequence. In the Vector of output values, the power data from our input datamatrix are called, so if we want to change the input to make more analysis,it’s only necessary to change the Matrix with all power values in matlab script, which we saw before.


Block Parameters: P 1

Repeating Sequence Interpolated (mask) (link)

Discrete time sequence is output, then repeated. Between data points, the specified lookup method is used to determine the output.

Vector of output values: phousetotW(:,1)

Vector of time values: [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48]'

Lookup Method: Interpolation-Use End Values

Sample time: Ts

Figure 5.4: Repeating sequence interpolated block

The Vector of time values rappresent the 48 half hours of the day, which for convenience and ease of calculation they are represented as 48 seconds.
For all the calcuation a $\cos\varphi = 1$ has been assumed for all loads, so the "multiplex vector signal" block takes in input the active power signal and the reactive power signal, which is set to zero.
The output of the "Dynamic Load" block are three signals:

1. positive-sequence voltage in per unit [pu]
2. active power, P, in Watt [W]
3. reactive power, Q, in vars

These values are reported into a matlab variable,which can be an array, timeseries or a structure. The block writes to the output file incrementally and if the output file already exists when the simulation starts, the block overwrites the file.
In our case, we chosen the array format to fill a matrix; in the table 5.1 is shown an example: So in the first row there is the istant time of data

t_1	t_2	...	t_{end}
V_{1-1}	V_{1-2}	...	V_{end}
P_{1-1}	P_{1-2}	...	P_{end}
Q_{1-1}	Q_{1-2}	...	Q_{end}

Table 5.1: Example of output matlab matrix

collection; in the other rows there are the active, reacative power and the voltage values in each load.

Chapter 6

Matlab model

6.1 Matlab code

Down below is shown the matlab code (originaly made by Ing. Beccari, modified for the goal of this thesys [24]) to load the customers csv matrix into matlab enviroment and to compute the total input power of the grid, which will be sent to the simulink model.

Briefly the power flow process can be represented as in fig. 6.1.

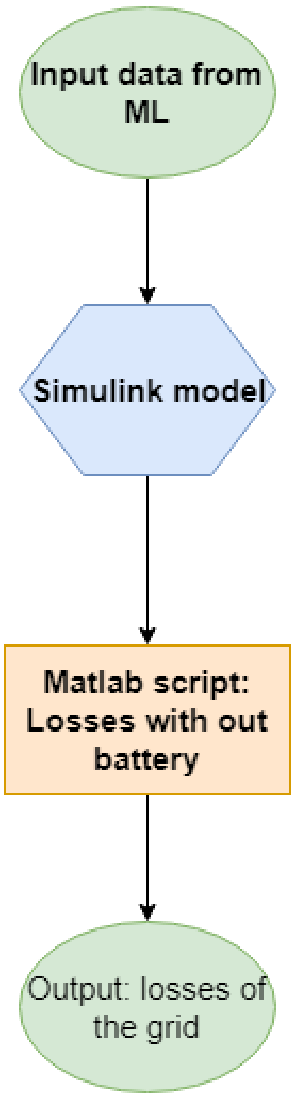

Input data from ML
Simulink model
Matlab script: Losses with out battery
Output: losses of the grid

Figure 6.1: Power flow flowchart

6.1.1 Compute grid parameters

This code section loads the Benchtest matrix storing as "ehousetot matrix". After that the energy is transformed into power and then converted into Watt from KW. At the end there is the summ of the total power absorbed by the feeders.

```
%% Load the Benchtest matrix or the other test
   matrix
ehousetot = xlsread("Benchtest_true_value.xlsx
   ");

phousetot=ehousetot/0.5; %to obtain the power I
    divide by 0.5 because each value
   corrispond to half an hour [kW].

phousetotW=phousetot*(10^3);%THIS IS THE POWER
   IN kW TO USE IN SIMULINK MODEL
% I have to find the power peak in each feeders
   . They are sized on 100kVA

% ADMD
pfeedertot=zeros(48,4);
k=0;

for ii=1:4
pfeedertot(:,ii)=phousetot(:,1+k)+phousetot
   (:,2+k)+phousetot(:,3+k)+phousetot(:,4+k)+
   phousetot(:,5+k)+phousetot(:,6+k);
k=k+6;
end

col_max=max(pfeedertot);
```

Below there is the plotting section where are shown the dayly power trends.

```
%% Plot
h=[1:48]';

figure(3)
hold on
box on
grid on
plot(h,phousetot)
xlabel('Half hours day');
```

```
        ylabel('P [kW]');
        title('Power 24 houses mix with 3 and 4
           bedrooms [kw]');

        figure(4)
        hold on
        box on
        grid on
        plot(h,pfeedertot)
        xlabel('half hours day');
        ylabel('P [kW]');
        title('Power tot in each of four feeders [kw]')
           ;

        figure(5)
        hold on
        box on
        grid on
        for jj=1:24
        plot3(h,jj*ones(length(phousetot(:,jj)),1),
           phousetot(:,jj));
        %rotate
        end
        xlabel('Half hours day');
        ylabel('P [kW]');
        title('Power 24 houses mix with 3 and 4
           bedrooms [kw]');

        %% Find power tot to insert the setpoint
        totP=sum(phousetotW,2); % Power tot LV houses [
           W]
        setpoint=mean(totP); % setpoint

        setpointsinglehouse=mean(phousetotW,1);
        setpointtot=sum(setpointsinglehouse);

        xlswrite('phousetotW',phousetotW);
        xlswrite('pfeedertot',pfeedertot);
        xlswrite('totP',totP);
        save('AA_HouseSelection_Day196','phousetotW','
           totP','setpoint');
```

6.1.2 Matlab Plot and Graphs

There are 3 main graph wich show us the power flow during the day:

1. In the figure 6.2 we can see the power absorbed by the 24 cumulative blocks during a sisngle day.
2. To better understand the 24 power cumulative blocks in the figure 6.4 is shown in 3D representation
3. In the figure 6.3 is represented the LV grid part which consist of four feeders and this graph depicts the sum of the total load in each feeder (in each feeder there are 6 cumulative loads). These values will be useful in the losses considerations

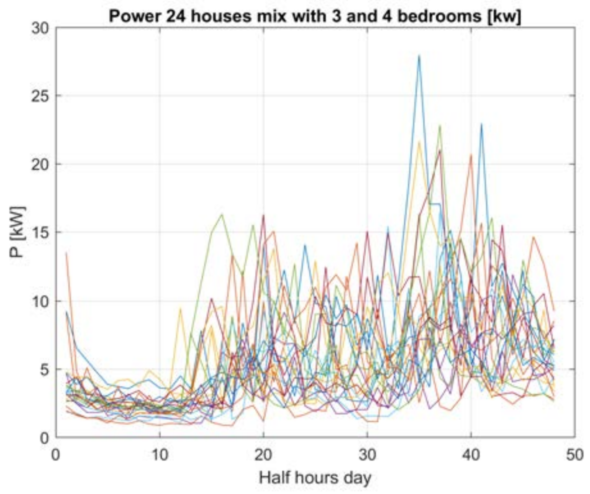


Figure 6.2: Power 24 houses: each colored line is a house

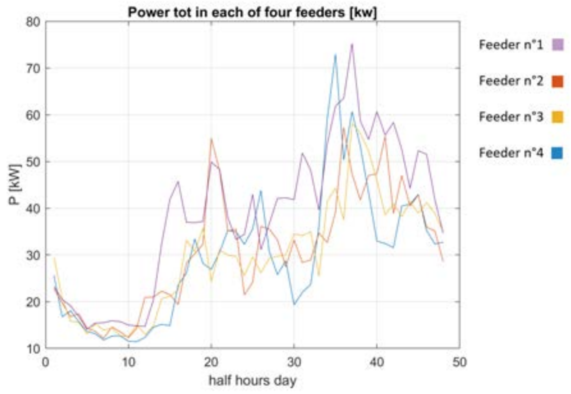


Figure 6.3: Total power in each of four feeders: each colored line represents one feeder

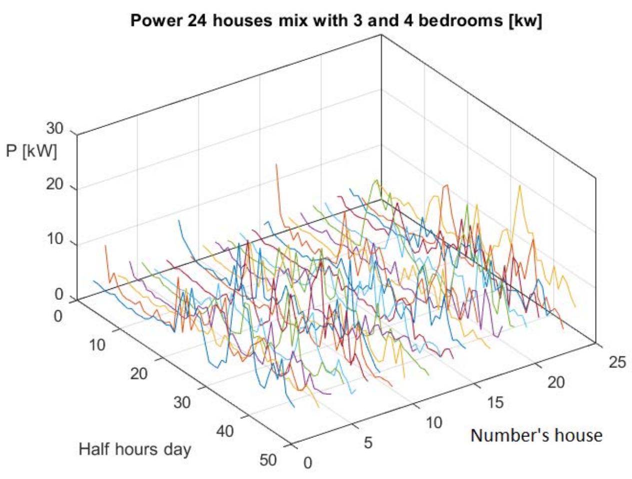


Figure 6.4: Power24 houses in 3D

6.2 Network losses

Is important to make a power flow analysis of the grid, to understand the impact of our prediction on the lossess error.

The network losses are directely proportional to the square of the current and power losses cause production of heat which increase the temperature on the feeders, so lifetime and reliability of the system decrease if the losses increase.
Considering the period of simulation, T, as 48 seconds it is possible to write the network losses as :

$$Loss = \sum_{t=1}^{T} R(t) \cdot I(t)^2$$

In the considered Simulink model, using measurement block (figure 6.5) is possible to get the power by MV and LV side;therefore we can obtain the power losses as subtraction between the two values. Knowing the Voltage and Current values, this block can calculate the active and reactive power (P and Q) and P can be imported into Matlab as file.mat to allowing the next calculations.

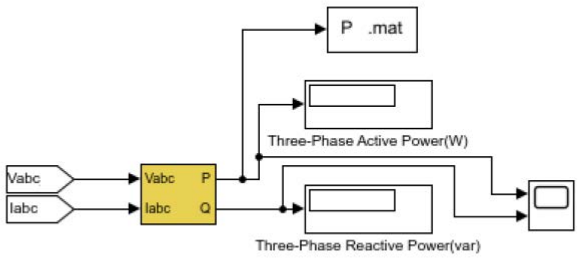


Figure 6.5: Measurement block

So the equation will be:

$$p_{loss}^{transf} = pmeasured^{MVside} - p_{measured}^{lVside}$$

$$p_{loss}^{feeders} = p_{measured}^{lVside} - \sum_{n=1}^{24} P_n$$

where:

1. p_{loss}^{transf} are the losses of the transformer, computed as the differece between MV and LV power measured along the side of the transformer.

2. $p_{loss}^{feeders}$ are the losses in the feeders, which are computed as the difference between the power measured on the LV side of the transformer and the total load power. The total load power is calculated as total unique lumped load.

At each interval the losses are measured and sent to matlab work space in the form of matlab matrix . He matrix has one column for time sampling and the second column stores the power values.
The results are then transformed in Energy losses by the integration of power losses during the time.
With this formula energy are calculeted:

$$E_{loss} = \Delta t \sum_{t=1}^{T} p_{loss}^{t}$$

where Δt is the lenght of the sampling time (half an hour here), T is the total number of intervals (48) and p_{loss}^{t} is the amount of power losses at the istant t.

6.2.1 Losses computation in matlab

Down below there is the script to compute the energy and power losses using matlab.
The first part of the script recalls the active, reactive and time values from Simulink model to create 24 matrix ($n \times 3$). The first colum has the time values from the sampling, the second column has the active power and the thrid column has a list of values about zero, which are the rappresentation of the null reactive power.
To obatain a single summ vector, the ptot vector, the values of the column relative of the active power are summed toghether.
AT the end the first column are extrapolated to use it later for the plotting.

```
%% Losses
clear all
close all

%From Simulink, I import the values of the file
    .mat
for n=25:48
filename=sprintf('VPQ%0.5g.mat',n);
S(n)=load(filename); %it is a struct. I have to
    convert it in a matrix
end
cc=struct2cell(S);
aa=cell2mat(cc);
%size(aa)
bb=permute(aa,[2 1 3]); % to obtain values in
    column and not in rows

% The fist column is the time, second the
    voltage, third the Active Power and
% the fourth reactive power
% I need to extract the third column of each
    matrix and sum toghether to
% have the total power absorbed from the 24
    houses.
p=zeros(length(bb),24);
for m=1:24
p(:,m)=bb(:,2,m); %This values are the same of
    phousetotW [W] present
% HouseSelection_Day196 script
end

```

```
        %i want the power tot on LV side. I sum the 24
            column of each row
        ptot=sum(p,2); %POWER TOT OF THE 24 HOUSES
        time=bb(:,1); %vector of the time. to plot use
            this value
```

Plotting code for Total Power LV.

```

        %% Total power in LV side
        figure(1)
        hold on
        grid on
        box on
        plot(time,ptot)
        xlim([1 48]);
        xlabel('Time [half hours]');
        ylabel('Power [W]');
        title('Total Power LV side Summer day [W]')
```

To compute the transformer losses as difference of the power upstream and downstream, the script uses the values from the measurement blocks set to the right and left side of the MV transformer.

```

        %% Power measured upstream the transformer
        dd=load('P_ut.mat');
        ee=struct2cell(dd);
        ff=cell2mat(ee);
        pt=ff'; %transposed matrix to obtain two
            columns: the first is the time and
        % the second is the power
        p_ut=pt(:,2); %POWER UPSTREAM THE TRANSFORMER:
            extract the second column of the power
        %upstream the transformer

        %p_ut=p_ut(3265:end);

        figure(2)
        hold on
        grid on
        box on
        plot(time,p_ut)
        xlim([1 48]);
        xlabel('Time [half hours]');
        ylabel('Power [W]');
        title(' Power Measured MV side Transformer [W]'
            );

        %% Power measured downstream the transformer
        gg=load('P_dt.mat');
        hh=struct2cell(gg);
        kk=cell2mat(hh);
        pot=kk'; %transposed matrix to obtain two
            columns: the first is the time and
        % the second is the power
        p_dt=pot(:,2); %POWER DOWNSTREAM THE
            TRANSFORMER: extract the second column of
            the power
        %downstream the transformer

        figure(3)
        hold on
        grid on
        box on
        plot(time,p_dt)
        xlim([1 48]);
        xlabel('Time [half hours]');
        ylabel('Power [W]');
        title(' Power Measured LV side Transformer [W]'
            );

        %% Losses calculation
        %% Losses transformer
        p_tl=p_ut-p_dt; %POWER LOSSES OF THE
            TRANSFORMER

        figure(5)
        hold on
        box on
        grid on
        plot(time,p_tl);
        xlim([1 48]);
        xlabel('Time [half hours]')
        ylabel('Power [W]')
        title('Power Losses Transformer [W]')

        figure(6)
        hold on
        box on
        grid on
        area(time,p_tl);
        energylossestrasf=(trapz(time,p_tl))/2;
        xlim([1 48]);
        xlabel('Time [half hours]')
        ylabel('Power [W]')
        title('Energy Losses Transformer [Wh]')
        energylossestrasf %Energy losses in the
           transformer
```

As shown before, for the feeders losses are used the difference LV side transformer power and the total power of the loads. Now, to compute the energy losses, is used the "trapz" matlab command, to calculate the area through the integral of the infinitesimal trapezoids.

```
        %% Losses feeders
        plt=p_dt-ptot; % POWER LOSSES FEEDERS

        figure(7)
        hold on
        grid on
        box on
        plot(time,plt);
        xlim([1 48]);
        xlabel('Time [half hours]');
        ylabel('Power [W]');
        title('Power Losses Feeders [W]');

        figure(8)
        hold on
        box on
        grid on
        area(time,plt);
        energylosses=(trapz(time,plt))/2;
        xlim([1 48]);
        xlabel('Time [half hours]')
        ylabel('Power [W]')
        title('Energy losses Feeders [Wh]')
        energylosses %Energy losses in the feeders

        %% Total losses = feeders + transformers
        totalosses=p_tl+plt; %TOTAL LOSSES FEEDERS+
           TRANS
```

```
energylossestot=energylosses+energylossestrasf
energylossestot
Dayly_losses = sum(totalosses,1);

figure(9)
hold on
grid on
box on
plot(time,totalosses)
xlim([1 48]);
xlabel('Time [half hours]');
ylabel('Power [W]');
title(' Power total losses (Feeders+Transformer
    ) [W]');
```

The next block compute the Voltage values and plots them.

```
%% Voltage values

for n1=1:24
filename1=sprintf('V%0.5g.mat',n1);
S1(n1)=load(filename1); %it is a struct. I have
    to convert it in a matrix
end
cc1=struct2cell(S1);
aa1=cell2mat(cc1);
bb1=permute(aa1,[2 1 3]);

v=zeros(length(bb1),24);
for m=1:24
v(:,m)=bb1(:,2,m); % VOLTAGE OF EACH LOAD
end

%v=v(3265:end);

figure(10)
hold on
box on
grid on
plot(time,v);
xlim([1 48]);
xlabel('Time [half hours]')
ylabel('Voltage [pu]')
title('Voltage of each load [pu]')
```

Efficiency of the grid.

```
        %% Find efficiency
        % epsilon=pout/pin=sum(total load)/sum(losses upstream transformer)
        epsilontrasf=sum(p_dt)/sum(p_ut) % TRANSFORMER EFFICIENCY IN A DAY
        epsilonfeeders=sum(ptot)/sum(p_dt) % FEEDERS EFFICIENCY
        epsilongrid=sum(ptot)/sum(p_ut)  % EFFICIENCY FEEDERS+TRANSF
        percentagefeederslosses=sum(plt)/sum(ptot)
        percentagetransflosses=sum(p_tl)/sum(ptot)

        save('AA_LossesWithoutBattery','time','ptot','p_ut','p_dt','p_tl','v','plt',...
        'energylosses','energylossestrasf','energylossestot','epsilongrid','epsilontrasf',...
        'epsilonfeeders','percentagefeederslosses','percentagetransflosses');
```

Chapter 7

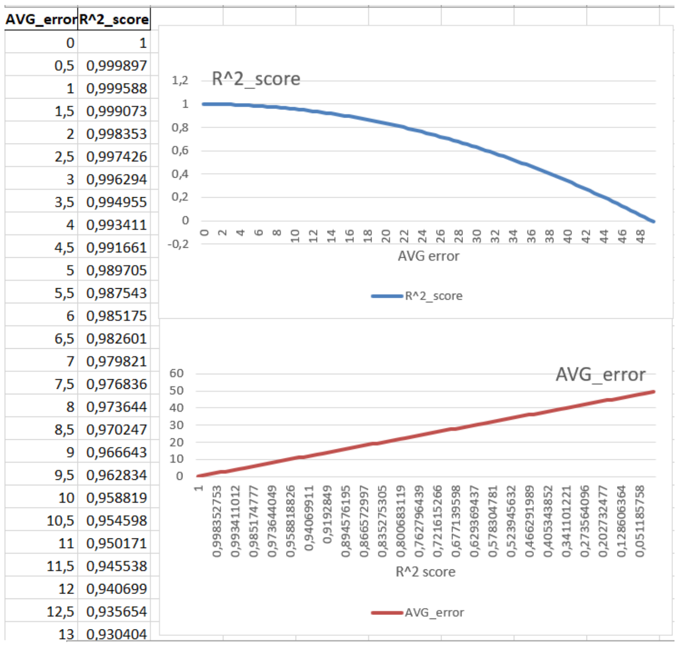
Results

7.1 Bench test result

To understand how the R^2 score can be used to figure out the total losses error of the simulink network, first i used this script (down below) to find a correlation between R^2 and average total dayly error. Here i track only a surplus of an hypotetic estimation of the predicted value, because is the wrost case for the grid, when we have more power demanded and losses, than we expect.

```
        os.chdir(Final_matrix)
        df = pd.read_pickle("
           Benchtest_matrix_true_value.pickle")
        df = df[["energy_target_variable"]]

        for NUM in range(1,11):
                os.chdir("/Users/FX504GE - EN069T//
                   Desktop/Codici lavoro/Benchtest")
                df1 = pd.read_pickle("Benchtest_acc_+%"
                    +str(NUM) + ".pickle")
                df1 = df1[["energy_target_variable"]]

                print(str(NUM))
                print(r2_score(df,df1))
                ACC = r2_score(df,df1) * 100
                print(str(ACC))

                df1 = pd.read_pickle("Benchtest_acc_-%"
                    +str(NUM) + ".pickle")
                df1.to_pickle("Benchtest_acc="+ str('
                    {0:.2f}'.format(ACC))+ ".pickle")
                df1.to_csv("Benchtest_acc="+ str('{0:.2
                    f}'.format(ACC))+ ".csv", index =
                    False)
                df1.to_excel("Benchtest_acc=" + str('
                    {0:.2f}'.format(ACC)) + ".xlsx")
```

This script gives back the excel values of the figure 7.1.
Down below there are the rappresentation of the average error value in relation at the R^2 score. As show in figure 7.1 the error has a linear correlation with the R^2 score: the more the score drops, the more the error raises. It is interesting notice that to have an error in module equal to 10% we must have an $R^2 score = 0.9588$, or 95.88%.



AVG_error	R^2_score
0	1
0,5	0,999897
1	0,999588
1,5	0,999073
2	0,998353
2,5	0,997426
3	0,996294
3,5	0,994955
4	0,993411
4,5	0,991661
5	0,989705
5,5	0,987543
6	0,985175
6,5	0,982601
7	0,979821
7,5	0,976836
8	0,973644
8,5	0,970247
9	0,966643
9,5	0,962834
10	0,958819
10,5	0,954598
11	0,950171
11,5	0,945538
12	0,940699
12,5	0,935654
13	0,930404

Figure 7.1: Theoretical R^2 score vs Avg error

It is important to check these result with the simulink model and the losses estimation matlab script.

7.2 Simulink and matlab result

I run the matlab and simulink model with the different benchtest matrix to analyse the grid losses, and see if they match the expectations of the previous test.

As shown in the figure 7.2 the trend of the graph is almost linear, as within the test in simulink bench test matrices with + or - a certain percentage of error were tested. These matrices have a different R^2 score, but have the same percentage error, which are represented by the flat areas of the graph. Overall considering the machine error, as approximations, the results on the figure 7.2 is quite similar to the figure 7.1.

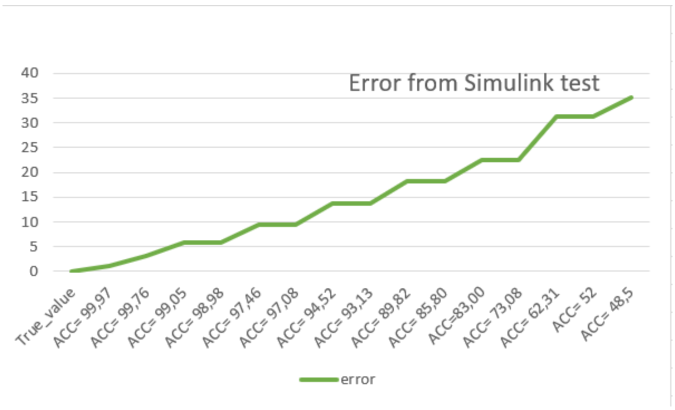


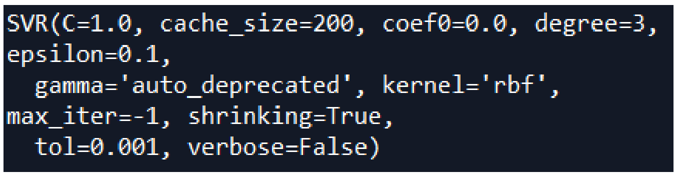
Figure 7.2: Average error from simulink test

So having traced the error trend with respect to our R^2 score, let's analyze the results obtained from the forecast models.

7.3 SVR and RFR model results

Here there are some comparison between SVR and RFR prediction. They use the same input matrix and they are going to predict the same interval of time, first a train-test along the same day, second the train using the day before the wanted one as a train set and as test set the target day. This choice is due to the short prediction interval which we are looking for [27].
The main important point in this article is that, we need good input data, not a large quantity. More data we have, better will be generaly, but in our case, if we want to predict a single day, we must give a single day training set to the models, to allow them to correctly understand the structure of the incoming data.
With the setting shown in figure 7.3, i run the SVR and RFR along 100 times, changing the target day time after time using as input dataset the Lumped custom matrix wich has a structure equal at 4.3.
The structure of the matrix is the same, but the column of power loads

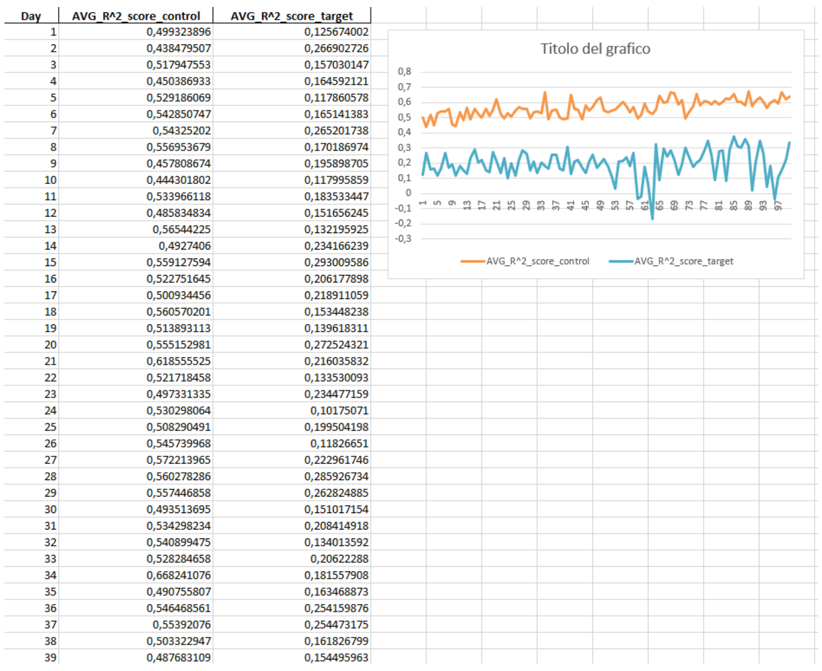
changes, being inclusive of 24 dwellings instead of just one.
The training time is defined as the day before the prediction.



```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3,
epsilon=0.1,
  gamma='auto_deprecated', kernel='rbf',
max_iter=-1, shrinking=True,
  tol=0.001, verbose=False)
```

Figure 7.3: SVR setting

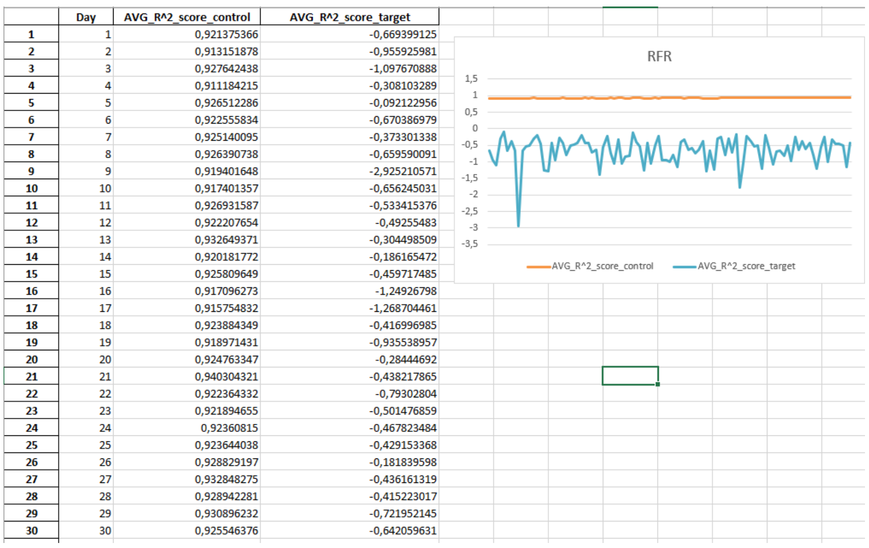
For the SVR the results are shown in figure 7.4. The main problem in this case is the overfitting of the model: the model is quite good to understand the correlation inside the training set, but has a lot of problem when we give as input new data. To overcome this problem is possible modify the C value, trying to raise the test prediction but losing some all overall precision.



Day	AVG_R^2_score_control	AVG_R^2_score_target
1	0,499323896	0,125674002
2	0,438479507	0,266902726
3	0,517947553	0,157030147
4	0,450386933	0,164592121
5	0,529186069	0,117860578
6	0,542850747	0,165141383
7	0,54325202	0,265201738
8	0,556953679	0,170186974
9	0,457808674	0,195898705
10	0,444301802	0,117995859
11	0,533966118	0,183533447
12	0,485834834	0,151656245
13	0,56544225	0,132195925
14	0,4927406	0,234166239
15	0,559127594	0,293009586
16	0,522751645	0,206177898
17	0,500934456	0,218911059
18	0,560570201	0,153448238
19	0,513893113	0,139618311
20	0,555152981	0,272524321
21	0,618555525	0,216035832
22	0,521718458	0,133530093
23	0,497331335	0,234477159
24	0,530298064	0,10175071
25	0,508290491	0,199504198
26	0,545739968	0,11826651
27	0,572213965	0,222961746
28	0,560278286	0,285926734
29	0,557446858	0,262824885
30	0,493513695	0,151017154
31	0,534298234	0,208414918
32	0,540899475	0,134013592
33	0,528284658	0,20622288
34	0,668241076	0,181557908
35	0,490755807	0,163468873
36	0,546468561	0,254159876
37	0,55392076	0,254473175
38	0,503322947	0,161826799
39	0,487683109	0,154495963

Figure 7.4: SVR average score for train and test set

The same test was carried out with the RFR model in figure 7.5



	Day	AVG_R^2_score_control	AVG_R^2_score_target
1	1	0,921375366	-0,669399125
2	2	0,913151878	-0,955925981
3	3	0,927642438	-1,097670888
4	4	0,911184215	-0,308103289
5	5	0,926512286	-0,092122956
6	6	0,922555834	-0,670386979
7	7	0,925140095	-0,373301338
8	8	0,926390738	-0,659590091
9	9	0,919401648	-2,925210571
10	10	0,917401357	-0,656245031
11	11	0,926931587	-0,533415376
12	12	0,922207654	-0,49255483
13	13	0,932649371	-0,304498509
14	14	0,920181772	-0,186165472
15	15	0,925809649	-0,459717485
16	16	0,917096273	-1,24926798
17	17	0,915754832	-1,268704461
18	18	0,923884349	-0,416996985
19	19	0,918971431	-0,935538957
20	20	0,924763347	-0,28444692
21	21	0,940304321	-0,438217865
22	22	0,922364332	-0,79302804
23	23	0,921894655	-0,501476859
24	24	0,92360815	-0,467823484
25	25	0,923644038	-0,429153368
26	26	0,928829197	-0,181839598
27	27	0,932848275	-0,436161319
28	28	0,928942281	-0,415223017
29	29	0,930896232	-0,721952145
30	30	0,925546376	-0,642059631

Figure 7.5: RFR average score for train and test set along 100 days.

A worse problem of overfitting occurs in the case of the RFR model. As you can see in the figure 7.5, the model seems to be able to give excellent answers during the training phase, marked as average score control, but fails in the test phase where it even has negative confidence values. These negative values mean that the model is not able to judge the quality of its forecast. Also in this case, it is necessary to optimize the RFR parameters using other algorithms, such as genetic algorithms.

For the RFR is important to take note about how long is the training time respect the SVR: SVR training time 1 day prediction is 1 seconds, instead for the RFR is 11 seconds, to run the program shown in the script down below, explained by the Flow chart in the figure 7.6.

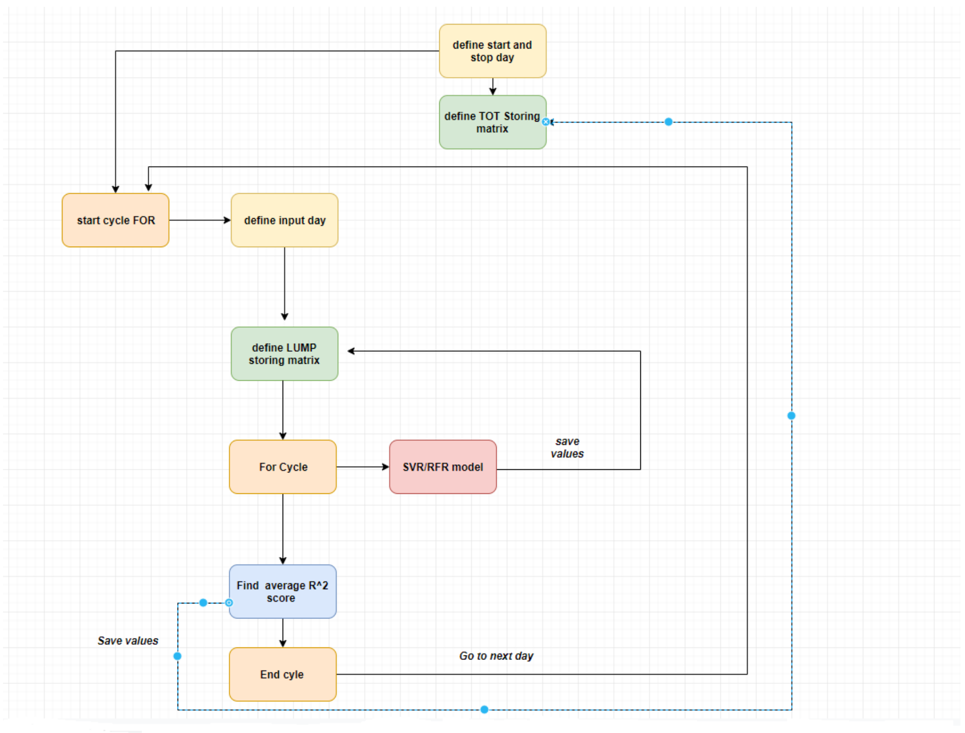


Figure 7.6: Flow chart AVG score generator, for single day and multiple Lump, and multiple day and total Lump score

```
        # inizialize the counter
        start = int(input("starting day (minimum = 1):
            "))
        stop = int(input("stop day (max = 365): "))
        # set the score day matrix for all the day
            predicted
        df_score_tot = pd.DataFrame(index = range(start
            ,stop),columns=["Day", 'AVG_R^2_score_control
            ', 'AVG_R^2_score_target'])

        ts = time.time()
        st = datetime.datetime.fromtimestamp(ts).
            strftime('%Y-%m-%d %H:%M:%S')
        print("Time start: " + st)
        for DD in range(start,stop):
        # define training day as target day - 1
        td = DD - 1
        gg = td*48
        # target day is the first input
        target_day = DD
```

```
    ggp = target_day*48
    #    cc = int(input("How much column do you
        have? (Lump matrix = 8): "))
    # columns on our matrix
    cc = 8

    # storing matrix, 3 columns: lumped customer,
        score of the control, score of the
        prediction
    df_store_data = pd.DataFrame(index = range
        (1,25),columns=['Lumped_customer','R^2
        _score_control','R^2_score_target'])

    for NUM in range(1,25):
    # load the matrx
    df = pd.read_pickle("lump_"+ str(NUM) + ".
        pickle")
    df = df[['time', 'tamb', 'vs', 'GF', 'Tc', '
        neta_c','power', 'energy_target_variable']]

    # select the input // Training value
    # X scaling
    X = df.iloc[gg:(gg + 48),0:(cc - 1)].values
    X = np.array(X)
    X_train = preprocessing.scale(X)

    # y scaling
    y = df.iloc[gg:(gg + 48),(cc - 1):cc].values
    y_train = preprocessing.scale(y)

    # select the input // Target vlaue
    # Xp scaling
    Xp = df.iloc[ggp:(ggp + 48),0:(cc - 1)].values
    Xp = np.array(Xp)
    X_test = preprocessing.scale(Xp)

    # yp scaling
    yp = df.iloc[ggp:(ggp + 48),(cc - 1):cc].values
    y_test = preprocessing.scale(yp)

    # Train e test set
    # X_train_svr, X_test_svr, y_train_svr,
        y_test_svr = model_selection.
        train_test_split(X_svr, y_svr, test_size
        =0.3)

        # SVR calling and fitting
        svr = svm.SVR()
        svr.fit(X_train,y_train)

        # Prediction // control section and target
        y_control= svr.predict(X_train)
        y_target = svr.predict(X_test)


        # control score
        MSE_control = mean_squared_error(y_train,
            y_control)
        r2_score_control = r2_score(y_train, y_control)

        # result score
        MSE_target = mean_squared_error(y_test,
            y_target)
        r2_score_target = r2_score(y_test, y_target)

        # Printing section for debug mode
        #         print("Lump_" +str(NUM))
        #         print("MSE control day prediction = "+
            str(MSE_control))
        #         print("R^2 control day prediction = "+
            str(r2_score_control))
        #         print(

            )
        #         print("MSE target day prediction = "+
            str(MSE_target))
        #         print("R^2 target day prediction = "+
            str(r2_score_target))

        #fill the df_store
        df_store_data['Lumped_customer'][NUM] = "Lump_"
             +str(NUM)
        df_store_data['R^2_score_control'][NUM] =
            r2_score_control
        df_store_data['R^2_score_target'][NUM] =
            r2_score_target

        #    df_store_data.to_pickle("
            Score_SVR_matrix_day="+str(target_day)+".
            pickle")
        #    df_store_data.to_excel("
            Score_SVR_matrix_day="+str(target_day)+".
            xlsx")

        # end firsti cycle, staring the second one
        df_store_data = df_store_data[["R^2
            _score_control","R^2_score_target"]]

        # make the average score values for the day
        df1 = df_store_data.sum()/len(df_store_data)

        df_score_tot["Day"][DD] = DD
        df_score_tot['AVG_R^2_score_control'][DD] = df1
            [0]
        df_score_tot['AVG_R^2_score_target'][DD] = df1
            [1]
        # end cyclo for
        df_score_tot.to_pickle("Score SVR matrix from "
            +str(start)+" to " +str(stop) +" day.pickle"
            )
        df_score_tot.to_excel("Score SVR matrix from "+
            str(start)+" to " +str(stop) +" day.xlsx")
        df_score_tot.sum()/len(df_score_tot)

        #clock sensor
        ts2 = time.time()
        st2 = datetime.datetime.fromtimestamp(ts2).
            strftime('%Y-%m-%d %H:%M:%S')
        print("Time end: " + st2)
```

For our main goal, the short term prediction is better to use the SVR, so we are able to make a more complex model which can update itself more times per day for example, using a different way to predict the data.
If we take one day, and split it into training and test set, our script will be able to use the data of the day before and also of the current day of the prediction to do the forecast. Basically it is possible to use the 70% and 30% rules. Splitting the all data set into, a 70% training set and 30% test set. So we are going to predict a shorter time window, but we can rerun our simulation, more times along the day, being able to predict all the day using the incoming data from the smart metering.

Also the understanding of the data could be improved by having more input data, having a higher sampling frequency, so as to have peaks that are not too different from each other and to make easier the training work for the models.

Chapter 8

Conclusion and futures works

8.1 Conclusion

Combining the information from the Simlunk model and the score test, it is important to notice that only prediction over $R^2 = 0.95$ to have an average error egual at 10% and to remain inside the Voltage limits of the grid for stability.
Moreover the SVR model suits better for short time prediction than the RFR model, which use more computational time to get similar performances.
It's also important to notice that with an higher frequency sampling of energy values, is possible to raise the accuracy of the models.
These results agree with the literature, where we have seen in recent years, that to improve SVR it is appropriate to associate them with optimization systems such as genetic algorithms, PSO. These algorithms in combination can help our model during the training phase and significantly improve the final R^2 score [28] [29].

8.2 Future work

This study can be the base, for an future test on a GA-SVR model, o even more sofisticated machine learning algorithm, as NEAT model. It is possible to try to do the load forecast prediction for all the next year to make the estimation network losses of all the years.
These algorithms, once refined, can be used for both long and short term forecasts, even in fields such as recyclable energies, where the importance of a good prediction of the generated power is essential for a correct management of the powerflow.

Bibliography

[1] D. S. B. Nimat Shamim Dr. Anitha Subburaj, *Model Predictive Control Analysis for the Battery Energy Storage System.* 2017.

[2] V. A. Irena Koprinska Rohen Sood, *Variable Selection for Five-Minute Ahead Electricity Load Forecasting.* 2010.

[3] I. M. Q. M. H. M. Rehan Nawaz Muhammad Awais Shahid, “Machine learning based false data injection in smart grid”, *IEEE*,

[4] R. Sebastian, *Machine Learning with Python (Data Science).* 2015.

[5] A. L. Samuel, “Some studies in machine learning using the game of checkers”, *IBM Journal of reasearch and development*, 1959.

[6] M. H. B. Widroe, Ed., *Adaptive switching circuits*, 1960.

[7] A. Turing, Ed., *Computing machinery and intelligence*, 1950.

[8] P. Langley, *The changing science of machine learning.* 2011.

[9] Lee, Ed., *Automatic speech recognition: the development of the SPHINX*, 1989.

[10] D. Pomerleau, Ed., *Alvinn: An autonomous land vehicle in a neural network*, 1989.

[11] F. U. M. D. S. Weir N., “Automated star/galaxy classification for digitized poss-||”, *Astronomical Journal v. 109*,

[12] C. M. Bishop, “Machine learning and pattern recognition”, *Clarendon press*, 2006.

[13] D. Wolpert, *The Lack of A Priori Distinctions Between Learning Algorithms.* 1996.

[14] D. W. e W.G. Macready, *No Free Lunch Theorems for Optimization.* 1997.

[15] W. P. W. S. McCulloch, “A logical calculus of the ideas immanent”, *The bulletin of mathematical biophysics*, 1943.

[16] F. Rosenblatt, “The preceptron a perceiving and recognizing automaton”, *Cornell Aeronautical Laboratory*, 1957.

[17] W. et al., “Adaptive “adaline” neuron using chemical “memistors””, *Stanford Electron. Labs.*, 1960.

[18] D. A. Freedman, “Statistical models: Theory and practice”, *Cambridge University Press.*, 2009.

[19] C. W. F. Rencher Alvin C., *Methods of multivariate analysis, wiley series in probability and statistics*, 2012.

[20] S. Dey, Ed., *Implementing a soft margin kernelized support vector machine binary classifier with quadratic programming in R and Python*, 2018.

[21] B. S. Alex J. Smola, “A tutorial on support vector regression”, *Statistics and Computing*, 2003.

[22] S. Kullback, *Information Theory and Statistics.* 1959.

[23] A. Tarsitano, *Measuring the asymmetry of the Lorenz Curve.* 1988.

[24] L. Beccari, “A study of the impact of integrating energy storage and pv systems into domestic distribution networks in ireland”, master thesy’s work, Università degli studi di Padova.

[25] *The impact of small scale embedded generation on the operating parameters of distribution networks*, in *Power P. B.* D. of Trade Industry, Ed., 2003.

[26] P. P. B., “Cost and benefits of embedded generation in ireland”, *SEi*, 2004.

[27] T. L. Qian Zhang, “A svm and variable structure neural network method for short-term load forecasting”, *IEEE*, 2010.

[28] P. T. Amit Kumar, “Hybrid ga – svr technique for contingency screening in power system”, *IEEE*, 2012.

[29] W. S.-w. Ju Yi-feng, “Village electrical load prediction by genetic algorithm and svr”, *IEEE*, 2010.