UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Data Science Curriculum

Artjom Vargunin

Reducing Electricity Cost for PV Prosumers by Load Forecast

Master's Thesis (15 ECTS)

Supervisor(s): Kristjan Eljand, MA
Novin Shahroudi, MSc

Tartu 2023

Reducing electricity cost for PV prosumers by load forecast

Abstract:
The energy industry is experiencing nowadays a paradigm shift due to the switching to renewable energy sources, individual electricity storage and production systems, distributed energy production, electric cars, smart devices etc. The geopolitical situation in the world and climate policy force these changes to develop even more rapidly.

The thesis addresses this topic by considering photovoltaic (PV) prosumers, energy consumers able to produce PV energy, store it into the battery and sell back to the grid. It is assumed that such a household uses exchange-price electricity package from grid electricity provider which means that grid electricity price is not fixed, but changes each hour. The electricity market such as Nord Pool announces hourly electricity prices in advance for the next day. Under these constraints the ultimate goal for PV prosumer can be settled as follows: can household achieve zero electricity bill or even earn money? What would be the strategy for the most cost-effective energy usage?

The thesis is aimed to help solving this problem by providing machine-learning driven forecast for the electricity consumption. This prediction is then used to optimize battery usage and calculate electricity cost for PV prosumer. At that, the following requirements are chased: prediction results in the lowest cost, forecast model is good for all households considered. The outcome of the study is well-reasoned suggestion for the forecasting pipeline with statistically proven supremacy.

Keywords: PV prosumer, load forecast, battery optimisation, electricity cost

CERCS: P176 - Artificial intelligence, T140 - Energy Research

Elektri kulu vähendamine tarbimise ennustusega PV prosumerite jaoks

Lühikokkuvõte: Tänapäeval toimub energeetikas paradigma muutus, mis on tingitud üleminekust taastuvatele energiaallikatele, individuaalsetele elektrienergia salvestamise ja tootmise süsteemidele, hajutatud energiatootmisele, elektriautodele, nutiseadmetele jne. Geopoliitiline olukord maailmas ja kliimapoliitika sunnivad neid muutusi veelgi kiiremini arenema.

Lõputöö käsitleb seda teemat PV prosumerite seisukohalt. Need on sellised energiatarbijad, kes saavad iseseisvalt PV energiat toota, salvestada seda akusse ja müüa tagasi elektrivõrku. Eeldatakse, et selline majapidamine kasutab börsihinnaga elektripaketti võrguelektri pakkujalt, mis tähendab, et võrguelektri hind ei ole fikseeritud, vaid muutub iga tunniga. Regionaalne elektrienergia vabaturg Nord Pool avalikustab elektri börsihinna järgmiseks päevaks. Nende piirangute alusel saab PV prosumeri lõppeesmärgi sõnastada järgmiselt: kas majapidamine suudab langetada elektrikulu nullini või isegi raha teenida?

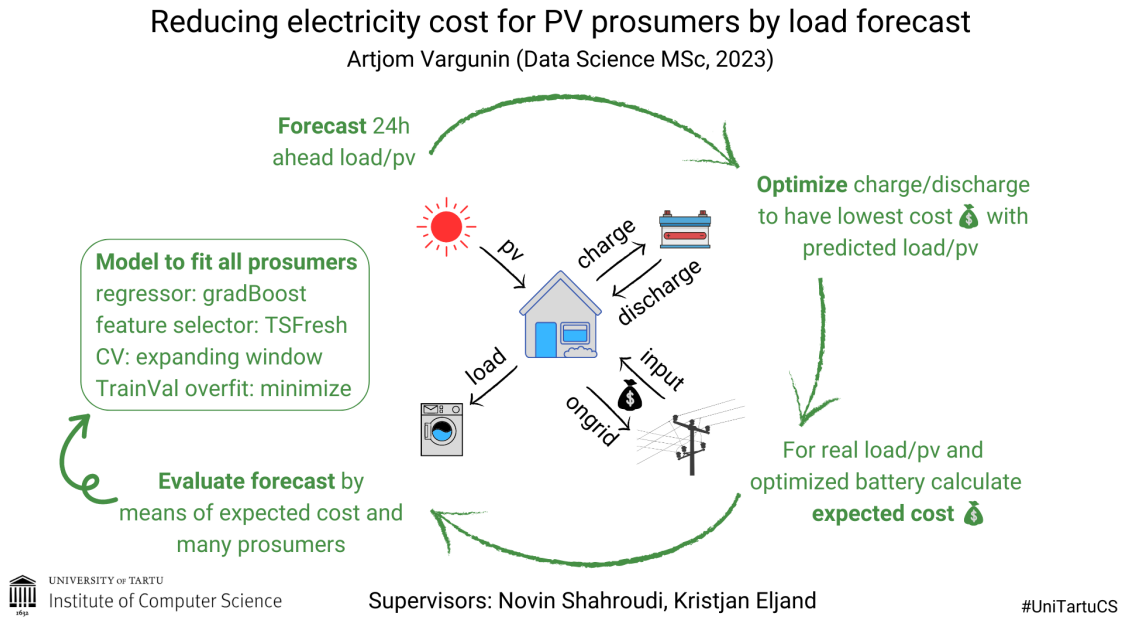
Milline oleks kõige kulutõhusama energiakasutuse strateegia?

Lõputöö eesmärk on aidata seda probleemi lahendada, pakkudes masinõppepõhist prognoosi elektritarbimise kohta. Seda ennustust kasutatakse seejärel aku kasutamise optimeerimiseks ja PV prosumeri elektrikulude arvutamiseks. Seejuures on nõutud, et ennustus tagaks madalaima kulu ja prognoosi mudel on hea kõikide vaadeldud majapidamiste jaoks. Uuringu tulemuseks on põhjendatud soovitus prognoosi mudeli kohta, mille ülimuslikkus on statistiliselt tõestatud.

Võtmesõnad: PV prosumer, tarbimise ennustus, aku optimeerimine, elektrikulu

CERCS: P176 - Tehisintellekt, T140 - Energeetika

Visual abstract:



Contents

1 Introduction to energy prosumerism **6**

2 Background and preliminaries **9**
2.1 Feature engineering 9
2.2 Feature selection 9
2.3 Regression task for time-series 10
2.4 Fighting with hyperparameters 13

3 Methodology **15**
3.1 Data availability 16
3.2 Load forecasting pipeline 17
3.3 Downstream metric 18
3.3.1 Consumption and production prices 19
3.3.2 Battery optimizer 20
3.3.3 Three cost metrics 23
3.4 Evaluating the pipelines 24

4 Experiments **26**
4.1 FusionSolar data analysis and preprocessing 26
4.1.1 Cleaning and preparing the data 26
4.1.2 Exploratory data analysis 28
4.1.3 Balance of the flows 30
4.2 EE data analysis and preprocessing 30
4.2.1 Cleaning and preparing the data 30
4.2.2 Exploratory data analysis 31
4.2.3 Household classification and clustering 32
4.3 Feature engineering and selection 35
4.3.1 Date-time features 35
4.3.2 Lagged features 36
4.3.3 Covariate features 37
4.3.4 Feature selection 38
4.4 On the regressor selection 38
4.5 Finding best hyperparameters 39
4.6 Understanding battery optimization 40
4.6.1 Optimization objective 41
4.6.2 Three cost metrics 44

5 Results and discussions **46**
5.1 Experiments with selected FusionSolar prosumer 46
5.1.1 Test MAE 46
5.1.2 Expected cost 49
5.1.3 Correlation between up- and downstream metrics 50
5.2 Experiments with many prosumers 50
5.2.1 Test MAE 51
5.2.2 Expected cost 54
5.2.3 Correlation between up- and downstream metrics 55
5.2.4 Ensemble pipeline 55

6 Conclusion **57**

References **62**

Appendix **63**
I. Licence 63

1 Introduction to energy prosumerism

According to the McKinsey Energy Insights, the renewables are projected to lead the power generation mix, reaching 80-90% in 2050 [1]. The origins of such a paradigm shift lie mostly on the international agreement by the most industrialized countries for reduction of CO_2 emissions made in past decades. Nowadays we are facing impressive developments of the corresponding technologies drastically affecting industry [1], policy [2], electricity market [3] etc.

From the household perspective, photovoltaic (PV) is one of the most promising, environmentally friendly and affordable renewable energy source. The development of the telecommunication technologies makes the operational side of PV usage very simple and user-friendly in terms of control, monitoring and integration with distribution grids. In this work the focus will be on the PV prosumers, i.e. energy customers who can also produce electricity from the Sun radiation. PV prosumers considered in this study have also possibility to sell electricity to the grid and store it for later usage. Here, the storage device can be electrical vehicle, for instance, but we focus on battery since its availability is not limited as in the case of vehicle.

Nowadays, prosumerism is growing rapidly over the globe and different forms of energy prosumers can be specified [4, 5]. However, there are general signatures common for all prosumers: proactive and sustainable energy consumption. The PV prosumer considered in this study is shown schematically in Fig. 1. Let us discuss the factors which determine its consumption of the grid electricity. Probably the most important contribution stems from the electricity consumption habits as well as best practices obtained from prosumer style of living [6]. Another crucial factor driving the grid electricity usage is related to the generation of the energy by PV prosumer. Two components here, PV and energy storage device, are determined by corresponding technical parameters, like capacity of storage device, its round-trip efficiency and depreciation cost, maximum PV power. There are also external factors involved including available solar radiation, cloudiness, outside temperature and electricity price.

For PV prosumer, the impact and interrelations between all these components is of main concern. If outside temperature is known to be too low (or high) in the next period, then prosumer may want to store more electricity into battery right now to use it later for heating (or air conditioning). If shiny weather is expected next day and battery is expected to be fully charged by solar electricity, then prosumer could sell all battery energy collected so far into the grid for high price. If price is known to be high, prosumer may want to charge battery fully now by buying large amount of grid electricity. The list of such scenarios is long, but it is clear that strong automatisation of decision making process could be very beneficial for PV prosumer.

[1] https://www.mckinsey.com/industries/oil-and-gas/our-insights/global-energy-perspective-2022

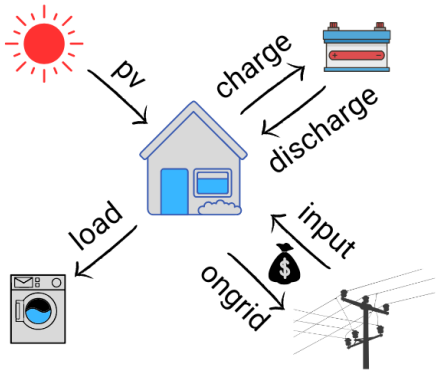


Figure 1. PV prosumer with different energy flows: load and PV (energy consumption and generation), charge and discharge (energy send to and taken from the battery), and input and ongrid (energy taken from grid and uploaded to the grid). Note that load is driven by the prosumer's electrical devices and peculiarities related to their working specifics, for instance, the power load of electrical heating system is strongly affected by the outside temperature, while the amount of light needed to lit up the room is affected by the cloudiness and time of the day. The PV energy is defined mostly by the weather and illumination conditions. Charge and discharge are subject to optimization targeting to decrease the input and increase the ongrid given that load and PV are known. Prosumer's electricity bill is formed based on input and ongrid multiplied by consumption and production prices, respectively.

Today, the energy management and energy sharing by prosuming units is very active field of research [7, 8]. The data science community is also actively challenged by the topic [9, 10, 11]. The building blocks of the problem include load and PV energy forecasts. Different approaches were reported in the literature for these tasks: k-Nearest Neighbour algorithm to forecast energy demand and supply [10], deep learning for monthly consumption [12], different versions of hybrid models [13, 14], neuron networks of different architecture [15, 16, 17, 18], modified time-series models [19, 20], statistical analysis [21].

The PV energy production is strongly driven by atmosphere conditions, for instance, in cloudy conditions solar panels work at 10-25% efficiency only. Therefore, PV forecasting is strongly connected to the physical modelling of the atmosphere and this peculiarity is always addressed in relevant researches. The approaches vary from time series methods for wind and solar forecasting [22] to machine-learning based renewable forecasting [23, 24, 25]. Note that important keyword in this area of studies is the scale.

Namely, different pipelines used for consumption and PV energy predictions are suited for different scale of the problem: prediction of hourly of yearly resolution, household or whole city aggregated consumption etc.

The important issue which affects drastically the performance of the forecasting system is the drift in the data [26, 27, 28, 29]. The drift means the shift in the concept and manifests itself as a change of the feature or target distribution in time. Special measurements are required for predicting under the data drift: drift has to be detected and the model needs to have adaptation mechanism [30, 31, 32, 33, 11].

This thesis is aimed to contribute to the optimization of the PV prosumer lifecycle by trying to reduce its electricity cost by the load forecast. The main innovation of the approach is twofold: the evaluation of best suggested prediction pipeline by the value of prosumer's electricity cost and determination of the best suggested pipeline by how it performs for different prosumers.

This thesis is written in partnership with Eesti Energia (EE), internationally known as Enefit, under the Industrial Master's Program at the University of Tartu.

2 Background and preliminaries

There is usually no way for having well performing model for supervised machine learning out of the box. In order to have one, serious efforts are needed including data preparation, feature engineering, feature selection, finding out suitable regressor or classifier able to understand the patterns hidden inside the data. Usually there are also adjustments present called hyperparameters which are not related to the data, but have to be specified by the practitioner. In this section we briefly review these topics.

2.1 Feature engineering

Feature engineering and selection is a big branch of the research in the data science. Properly engineered features can result in enormous enhancement of the algorithm prediction power. Basically there are two ways to engineer features: manual engineering based on expertise and automated engineering. For the latter case, different approaches have been developed. For instance, the AUTOFEAT package [34] does it iteratively by applying transformations on existing features (various unary mathematical operations) and then applying combinations between features created (simplest binary operations) and so on iteratively. In this approach feature pool increases exponentially with each step.

Completely different approach is used in the package TSFRESH [2] targeting feature engineering for time-series.These features are found by considering dependent-variable time-series formally and applying various mathematical transformations to it. For each target value the prior data window has to be fixed. From these data window the mathematical features are extracted varying from minimal or maximal values and values of various statistics to the Fourier transform or wavelet transform coefficients. The package allows to control the number and complexity of the features extracted which determine feature extraction time.

The success of the manual feature engineering is determined by the expertise in the field under consideration. For PV prosumers, the published surveys on their consuming practices may be very insightful [6].

2.2 Feature selection

Not all features engineered or present in the data are usually of importance. To reduce the curse-of-dimensionality effect and drop redundant features the feature selection approach has to be developed. In the study different methods were used.

To understand how automated selection could be done let us review some approaches. The package AUTOFEAT [34] has module devoted to the selection of the promising

[2] tsfresh.readthedocs.io/en/latest/

features out of the pool of generated ones. During the selection the strongly correlated features are found and only single representatives are left. However, the main idea is to evaluate feature significance not independently, but in combination with other features. This is done by performing many runs as follows. On the selected data subset the list of promising features is selected by training regularized logistic regressor and examining its weights. Then this list is divided by several chunks of promising features. For each chunk the noise filtering is undertaken: some noisy features are added and LassoLARS regressor is trained to filter out promising features less important than noisy ones. By repeating for all chunks, the combined set of selected promising features left after noise filtering is created. After performing final noise filtering on combined set the feature selection is finalised for the run. By repeating procedure many runs with different subsamples of the data, the ultimate set is prepared with most frequently selected promising features. This example illustrates how features are selected by extracting information regarding feature importance from some model.

Another automated feature selection approach is implemented in package TSFRESH. In this case selection is made based on statistical tests between selected feature and the target. For instance, for binary feature tested against the real-valued target the two-sample Kolmogorov-Smirnov test is used. In this test the similarity between two distributions is compared: one is target distribution for the first value of the binary feature and another distribution for another value of binary feature. Under the null hypothesis both distributions are identical meaning that binary feature is not informative.

For real-valued feature tested against real-valued target the Kendall-τ test is applied in TSFRESH. In this case feature and target values are ranked separately and two rank lists are compared by means of rank correlation. Compared to Spearman rank correlation, Kendall-τ is much more robust to errors and discrepancies in the data. In addition, on top of these tests the Benjamini-Hochberg procedure for multiple testing is also applied. This procedure penalizes small p-values happened by chance.

Note that features could be also selected manually, if background knowledge or expertise is sufficient.

2.3 Regression task for time-series

PV prosumer's load data is essentially a time-series of particular granularity: for each timestamp there is particular value of the load. Time-series forecasting is well defined area of the research [35] and many classical approaches exist to tackle corresponding tasks: Holt-Winters, Error-Trend-Seasonal (ETS) and SARIMAX, to name a few. These classical models rely strongly on the data temporal ordering, meaning that the value at particular timestamp is determined by the historical values prior to the timestamp and future values are not accessible. Moreover, the problem is usually low-dimensional, e.g. in simplest case we have only 1+1 dimensions, one dimension for the time and another for the target variable.

The machine learning has shown great results in the field of time-series forecasting [36]. Usually, we assume in the case of machine learning data that samples are i.i.d, or independent and identically distributed random variables. This means that data samples come from the same joint distribution and there is no dependence between samples. In addition it is assumed that data sample represents multidimensional random variable and its dimensions are independent as well. Roughly speaking it means that for machine-learning data given in the form of the table the rows in the table as well as columns are exchangeable.

It is clear that for time-series data put into the table the samples independence and their exchangeability is violated due to importance of the temporal ordering. However, we can map the time-series data into the dataset with the properties expected from machine-learning data. One approach here is to expand 1+1 dimensional time-series problem into N+1 dimensional machine-learning problem by engineering N new features. In this case, it is possible to encode the information on time ordering inside additional dimensions at the same time preserving i.i.d. assumption. The simplest example here is the transformation from 1+1 dimensional time-series data with timestamps into the 3+1 dimensional dataset with new dimensions engineered as the weekday, hour and minute corresponding to the each timestamps. It is clear, that machine-learning data obtained this way is exchangeable and also information regarding temporal ordering is there.

After obtaining dataset with desired features the mapping between the features and the target has to be established. There are many choices available, see, for instance, the list of SCIKIT-LEARN regressors [3]. However we limit ourselves with Decision Tree and Gradient Boost regressors. The former one may be treated as baseline selection, since Decision Tree is known to be weak predictor. At that, it is non-linear regressor with high interpretability. The letter one is potentially strong forecaster. Usually Gradient Boost outperforms well known tree ensemble-based Random Forest algorithm. Moreover, in many forecasting competitions gradient boosted trees show even better results than deep learning-based solutions [37]. On supervised learning tasks, algorithms based on gradient-boosted tree ensembles mostly outperform deep learning models with additional advantage of significantly less training time [38].

Decision Tree. Decision Tree regressor cuts data space of dimensionality determined by the number of the features into small regions each characterized by peculiar prediction. The hyperparameters of the algorithm allow to adjust what features should be used to form the data space and what space cutting rules have to be applied. Fig. 2 demonstrates an example tree of decisions.

Gradient Boost. Gradient Boost [39] combines weak predictors (like Decision Tree) into strong one in the iterative fashion. Let us consider the main steps of the algorithm

[3] `https://scikit-learn.org/stable/`

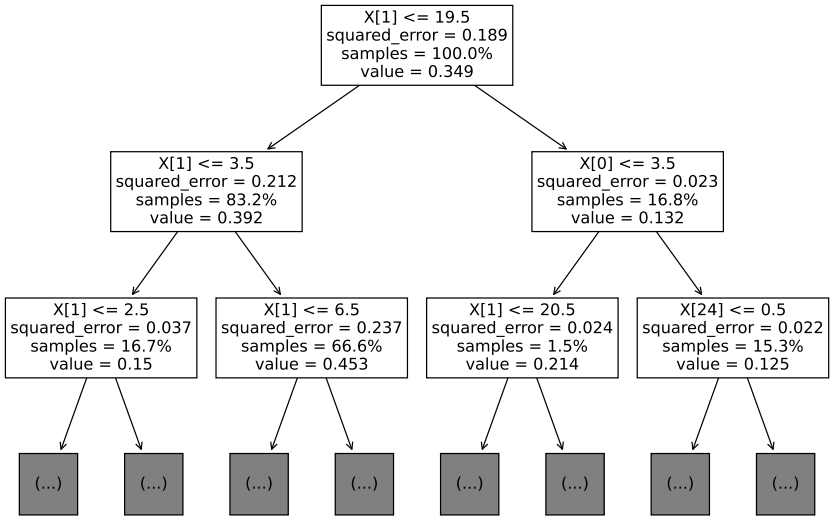


Figure 2. Top layers of decision tree. The decision making is starting from the value of the first feature X[1] from the list X of all features. If sample value is smaller or equal to 19.5 then go the left branch. The condition is satisfied for 83.2% of all samples and predicted target value of 0.392 has error 0.212. The decision making is continued and, if X[1] below 3.5 then for 16.7% of the samples the target value 0.15 has as small error as 0.037. Decision making goes on until all cutting conditions for the data space are satisfied.

for regression task $y \sim x$ with n datapoints:

1. Initialize model with initial constant prediction $F_0 = \mathrm{argmin}_\gamma \sum_{i=1}^n L(y_i, \gamma)$. For quadratic loss function, $L(y_i, \gamma) = (y_i - \gamma)^2/2$, the minimization is straightforward

$$\frac{\partial}{\partial \gamma} \sum_{i=1}^{n} L(y_i, \gamma) = -\sum_{i=1}^{n} (y_i - \gamma) = 0, \quad \text{thus} \quad \gamma = \frac{1}{n} \sum_{i=1}^{n} y_i = \bar{y}. \tag{1}$$

   For quadratic loss function, the initial prediction is just target mean.

2. Fix the number of trees M and iterate for $m = 1 \ldots M$ as follows.

3. Compute local gradient of loss function $r_{im} = -\frac{\partial L(y_i, \gamma)}{\partial \gamma}|_{\gamma = F_{m-1}(x_i)}$. For quadratic loss function

$$r_{im} = y_i - F_{m-1}(x_i), \tag{2}$$

   thus r_{im} is residual. These residuals measure the error of the prediction F_{m-1}.

4. Train Decision Tree regressor $r_m \sim x$ with leaves (bottom nodes of Decision Tree) L_{jm}, $j = 1 \dots J_m$, where J_m is total number of leaves.

5. Find predictions $\gamma_{jm} = \text{argmin}_\gamma \sum_{x_i \in L_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$. For quadratic loss function

$$\frac{\partial}{\partial \gamma} \sum_{x_i \in L_{jm}} L(y_i, F_{m-1} + \gamma) = - \sum_{x_i \in L_{jm}} (r_{im} - \gamma) = 0, \quad \text{thus} \quad \gamma = \frac{1}{n_{jm}} \sum_{x_i \in L_{jm}}^{n} r_{im}, \tag{3}$$

   where n_{jm} is the number of samples in j-th leaf. We see that $\gamma_{jm} = \gamma$ is regression tree prediction for considered leaf.

6. Update combined prediction $F_m(x_i) = F_{m-1}(x_i) + \nu \gamma_{jm} 1(x_i \in R_{jm})$, i.e., if point x_i belongs to L_{jm} leaf, then its prediction is updated by γ_{jm}. Here ν is learning rate ranging from 0 to 1 needed to reduce the overfitting.

By repeating M times the steps 3-6 the final prediction is calculated.

2.4 Fighting with hyperparameters

It is a good practice to reduce the number of the hyperparameters by finding their optimal values. For this, the prosumer's data is divided into test, train and validation sets. The test data starts from the origin of prediction, i.e. the timestamp we want to forecast from. The load prediction is requested by the household for the test data and it is later used to run the battery optimization. Train and validation are used to fit the parameters of the model and validate the utility of the decisions made during model preparation. Important is that the validation and training samples are formed by applying certain splitting strategy.

The validation of the decisions undertaken during model preparation is usually done by cross validation. Cross validation can be used for different purposes. Most important is probably the evaluation different prediction pipelines before making final prediction on test data. The main idea behind cross validation is to fix hyperparameters for prediction pipeline, consider particular train-validation split, train model on train folds with validation on validation folds, repeat for all specified splits and collect error information. By looking on this information we can estimate possible overfitting of the pipeline by comparing test and validation error. If cross validation is performed for two different models or two sets of hyperparameters, then the most promising one can be selected by comparing validation errors. Fig. 3 shows different cross-validation strategies, including usual 5-fold cross validation and rolling and expanding window cross validations often used for time-series data.

The space of hyperparameters is usually multidimensional and the number of all possible hyperparameter combinations need to be tested is very big. The grid search cross

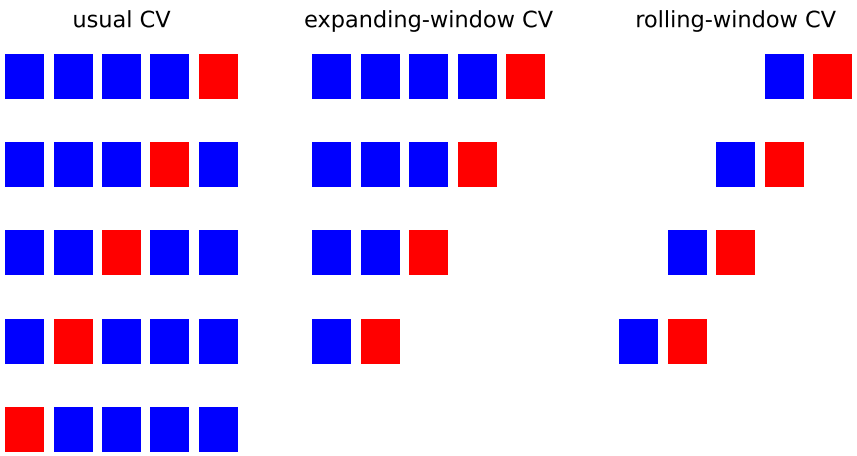


Figure 3. Validation (red) and train (blue) data folds during cross validation (CV). Note that for usual CV the samples are randomly mixed before split, while for expanding and rolling window CV the samples are temporarily ordered.

validation is too time consuming in this case. To increase the speed, the random search cross validation is used. More intelligent searching strategies also exist, for instance, Bayesian optimization [40].

3 Methodology

As compared to usual household able to use grid electricity only to cover its consumption needs, the PV prosumer considered in this study is characterized by ability to consume, store, produce and sell electricity. Schematically, the PV prosumer is shown in Fig. 1 with relevant electricity flows indicated by the arrows. Note that usual household has only input and load flows.

The main goal of this study is to build the pipeline which predicts the load of PV prosumer. Having this prediction and some guidance for the battery charge and discharge, the expected electricity cost could be estimated. Whereas the quality of the load prediction measured by typical measures of quality such as mean absolute error (MAE) on test set and referred to as upstream metric is of great importance, there exists so called downstream metric related to the cost value. The latter could be used as a measure of the prediction utility.

The first innovation of this research is to monitor and make decisions based on the downstream metric - expected electricity cost of the PV prosumer. What decisions are of interest for us? Particular questions which will be answered are the following

- how is accuracy of the load prediction driving expected cost
- how to select between several prediction schemes the best one by monitoring downstream metric

The second innovation of this research is that we are interested in the building of the load prediction pipeline with the following properties

- it performs good according to the downstream metric not only for an individual household, but across many households
- it is flexible enough to adjust to particular PV prosumer on-the-fly, so that resulting pipeline is optimal for average prosumer in the database

Having in mind these two innovations we start with building the load prediction pipeline. The principal scheme of the approach is illustrated in Fig. 4. The main ingredients of the approach used are data preparation, feature engineering and selection, choice of the regressors, and choice of the cross-validation scheme. Different choices results in different load predictions described by the up- and downstream metrics values. By applying statistical hypothesis testing, various decisions regarding the constituents of the prediction pipeline are made and different pipelines are compared to each other.

Various preliminaries of this general approach were introduced in previous section. Below we overview available data and methodological pieces of load prediction, battery optimization, cost calculation and evaluation of prediction pipelines.

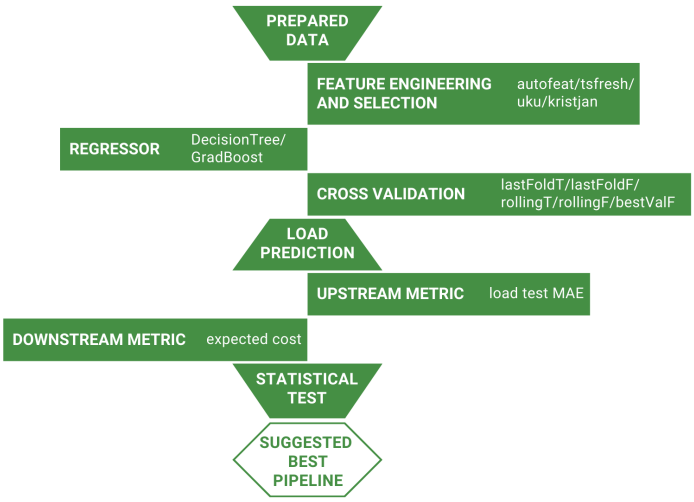


Figure 4. Different steps (from top to bottom) behind the finding out the best load predicting pipeline. Note that there are several selections indicated in the flow chart that have to be done before obtaining load prediction. These selections are evaluated by statistical test applied to up- and downstream metrics.

3.1 Data availability

The data used in this study stems from two sources.

FusionSolar data. FusionSolar is the online interface for the smart PV management system developed by Huawei [4]. The latter system monitors the PV power generation and allows simple access to the PV prosumer's operational data. EE clients classified as PV prosumers can be accessed via FusionSolar. For us, the most interesting information accessible via FusionSolar covers the prosumer communication with the electricity grid (bought input and sold ongrid energy), the prosumer's energy generation (PV energy generated and energy obtained from the battery during its discharge) and the prosumer's energy consumption (due to load and charging the battery). Available energy data unit is kWh. Each data sample has the timestamp with the timezone attached so that transformation to local Estonian times is straightforward. Data resolution is 5 minutes and data comes in cumulative manner. This means, for instance, that meter point measures the PV energy generated during last 5 minutes and reports this value added incrementally to the previous measurement. During the study we had access to three PV prosumer's data covering one year into the past.

[4] `https://eu5.fusionsolar.huawei.com/`

EE data. FusionSolar data source gives access to some EE clients functioning as PV prosumers, i.e. the ones able to generate PV energy, store it and sell to the grid. Another source of the data was also available during the study. It comes from EE data warehouse and covers year 2021 electricity consumption of 100 conventional Estonian households of unknown types, for instance, apartments, private houses, factories etc. Their consumption or load was measured once a hour by the household's meter point having unique identifier (EIC). Together with date-time index of hourly resolution, the load measured in kWh during that hour, postal code and location were also provided. Although these households are not real PV prosumers, we consider them as PV prosumers with PV energy and battery level of fixed values. The latter does not affect our study.

3.2 Load forecasting pipeline

Load forecasting pipeline created in this work consists of several stages and involves several selections need to be fixed as listed below.

Feature selection. In the study we use manual feature engineering relying on our expertise, intuition and established prosumer practices [6]. The details will be given below. The pipeline starts with removing redundant features engineered for considered PV prosumer. Different feature selection options are available: two automatic selections taken from corresponding AUTOFEAT and TSFRESH packages as described in Section 2.2 and two manual feature selections.

Two manual selections suggested by two colleagues and named corresponding to their first names are tested. The Uku's selection contains only two particular features for predicting the load. There was a Kaggle competition organized by EE [5]with the aim to create an energy consumption prediction model for a single household. Among other approaches Uku suggested the model with only two features mentioned and this model achieved top-4 result out of 36 participants. We tested this model after competition deadline and after minor modifications for proposed regressor with almost default parameters was able to reach top-1 position.

The Kristjan's set contains feature selection based on the common sense behind intelligent and spot-price driven electricity consumption. Corresponding features will be described below.

Regressor selection. Two regressors available for the pipeline are Decision Tree and Gradient Boost. The hyperparameters of these regressors are fixed by random search cross validation. The main hyperparameters of interest for Decision Tree are:

- maximal depth: the number of branching layers of the tree

[5]https://www.kaggle.com/competitions/predict-electricity-consumption

- minimal samples on leave: minimal number of samples inside the region of peculiar prediction. This parameter is in strong relation with maximal depth.
- maximal number of features to form the data space.

The main hyperparameters for Gradient Boost are:

- number of estimators: number M of Decision Trees
- learning rate ν
- parameters of the individual Decision Trees

These parameters of great importance because they allow to fit training data very well, but resulting in poor generalisation on unseen test data. This is overfitting problem and it has to be always kept in mind when tuning hyperparameters.

Selection of cross-validation strategy. Validation set is always fixed to be 20% of historical data provided. Selection involves 5-fold, rolling- and expanding-window cross validation. Additional option is to validate on last fold where the most recent data is located.

The default objective of the cross validation is to find the pipeline with lowest validation error. However, this option can suggest the pipeline with overfitting signatures. We implemented the option for the cross validation objective to select the pipeline with reduced overfitting on train folds.

3.3 Downstream metric

The main focus of this study are on PV prosumer, i.e. the client with ability to generate PV energy as well as to store the electricity in the battery. In the previous steps its load prediction was discussed. By having expected load (and PV generation) of the prosumer the battery should be used in most optimal way, meaning that its charging and discharging have to result in lowest electricity cost. The latter is calculated based on the consumption and production prices, i.e. prices used when client buys and sells electricity, respectively. These prices are applied to calculate electricity cost based on the input and ongrid energy, i.e. electricity taken from the grid and loaded to the grid. Fig.1 points also that electricity bill is stemming from input and ongrid.

Let us discuss different topics related to the downstream metric, expected electricity cost, found by optimization of battery usage provived that load prediction is given.

3.3.1 Consumption and production prices

In Estonia, there exist many packages for buying electricity, including spot, universal, and fixed packages[6]. All these packages have different tariffs. In addition, there are several network provider packages. So the number of all possible combinations is large. For prosumers it is important to have packages with volatile prices. Only in this case there is a way to buy cheap electricity and sell expensive one. Therefore, in the analysis below it is assumed that prosumer buys electricity with the use of exchange package[7] which is directly linked to the electricity spot price for Estonia in the Nordpool market[8]. Note also that this assumption is simplification, because in reality some prosumeres have packages for buying electricity with fixed price and only selling energy is governed by electricity spot price.

In the Nordpool market electricity prices are known for some period of time in advance. Usually, each day around 15:00 the next day electricity price is issued. Therefore, for any considered hour the future electricity prices are well known. However, the period of known prices varies depending on the hour of request. If requested at 14:00 then only 9 hours ahead are known. If requested at 15:00 after next day prices are issued, then 33 hours ahead are known.

For selling electricity to the grid the is only single price available - spot price at that hour. Based on this information, we use the following tariffs for buying/selling 1kW of electricity by private customer:

$$\begin{aligned} &\text{consumption price} : spot\ price + 20\% tax + 0.07 Eur, \\ &\text{production price} : spot\ price - 0.01 Eur. \end{aligned} \tag{4}$$

Here 0.07 Eur includes additional fixed taxes and network fees not included into sale taxes of 20%. Approximate value of 0.01 Eur in (4) is the margin substracted as fixed in small-producer agreement. Note that (4) is simplified and rounded (but still close to reality) because the network prices may vary for different clients and periods of time significantly. Therefore for ideal application single value of 0.07 Eur has to be replaced by client-specific one, but in the study we neglect this complexity.

Based on these numbers it is clear that for small spot prices the ratio between consumption and production price at the same hour may reach 2 or more. For instance, if spot price is 0.11 Eur/kW, then consumption price is around 0.2 Eur/kW which is twice higher than production price.

From Eq. (4) we also see that the only way for prosumer to reduce the bills is to buy electricity at cheap hours and sell it on expensive hours. Note also that monetary

[6] https://www.energia.ee/en/abi/hinnad-ja-paketid

[7] https://www.energia.ee/en/era/elekter/elektrileping-ja-paketid?customers=home-customer&packages=spot

[8] https://dashboard.elering.ee/en/nps/price?interval=minute

outcome due to the gambling on the prices becomes larger, if spot price is high. In this respect the last fall was very intriguing, because prices were on average higher then usual due to energetic crisis. As a result, the prosumers were potentially able to earn money or in another sense save a lot on their electricity bills if used the battery optimally to balance the consumption-production.

Also, it would be nice to emphasize that balancing input and ongrid energies on the basis of individual household is also potentially a good thing for the society since selling electricity in hours with peaked electricity price lowers potentially the demand for electricity in these hours. As a result, the electricity prices in peaked hours become more smoothed which is beneficial for all.

3.3.2 Battery optimizer

The ELab team, small R&D unit in EE, had created an optimizer to calculate battery charge and discharge pattern aimed to minimize the electricity cost given that the PV and load time-series are provided. The dashboard with optimizer results is available here [9], however to run the optimizer with one's load prediction the special credentials are needed. The optimizer is written by using Google OR tools [10]. To better understand its capabilities let us consider optimizer input fields. There are several categories for the input: battery description, customer and environment description and frequency market inputs (not used in the study). Let us consider most relevant input parameters and output separately.

Battery inputs. These are the main parameters about the asset, which behavior will be optimized. The parameters are given in Table 1.

These inputs are self-explanatory. Note that battery capacity for FusionSolar data can be estimated from the data: the charge and discharge energies are known together with battery state of charge (in percents). Based on this, one can estimate how many kWh one percent of charge does contain, thus kWh for 100% can be calculated. Regarding round trip efficiency, it is assumed that both charging and discharging efficiencies are equal to the square root of the round-trip efficiency. We also see that optimizer is clever enough to take into account the depreciation of the battery. This means that the optimizer won't charge the battery if the possible cost saving for the household is less than depreciation.

Customer and environment inputs. The local environment is the environment that is connected behind the same main fuse, for example, an household where the battery is installed. Main input parameters and variables are given in Table 2.

[9] `https://eds-optimizer-dashboard.azurewebsites.net/`

[10] `https://developers.google.com/optimization`

variable	description	value
Capacity	The total capacity of the storage device (in kWh)	Fixed by 10
Charge power	The maximum power (in kW) that this asset can consume/produce	Fixed by 5
Initial charge level	The energy level (in kWh) at the beginning of the optimization	Fixed by 3. For FusionSolar data is taken from data
Round trip efficiency	Fraction of energy put into the storage that can be retrieved	Fixed by 1
Min allowed charge level	The minimum charge level (in kWh) that this asset is allowed to have	Fixed by 0.2
Depreciation	The depreciation cost of the asset (in Eur/kWh)	Fixed by 0
Max cycles per period	Number of charging cycles that are allowed	Fixed by 1

Table 1. Battery inputs.

variable	description	value
Consumption prices	List of prices with all taxes included (Eur/kWh) that this customer experiences when consuming electricity from the grid	Calculated based spot price and Eq. (4)
Production prices	: List of final prices (Eur/kWh) that the customer experiences when sending electricity to the grid	Calculated based spot price and Eq. (4)
Main fuse	Size of the main fuse (in Amperes) that limits the total energy production and consumption of the environment	Fixed by 16
Baseload	List of load predictions (in kWh) for the environment	Main input variable
PV production	List of PV production forecast (in kWh)	Fixed by 0. For FusionSolar data is taken from data

Table 2. Customer and environment inputs.

The formula used for prices was introduced in Eq. (4). Here the length of the list of the consumption prices defines the optimization horizon. For example, if 3 hours ahead

values for consumption prices are provided then optimization will be done for next 3 timesteps. At that, the baseload predicition has to be of the same length as consumption prices.

Optimizer output. The output of the optimizer is divided into 4 groups: solver, variables, info and objective. The solver data returns the information about the solver and optimization. Particular fields are listed in Table 3.

variable	description
Optimality	Indicates, whether the algorithm found the mathematically optimal solution or not
Wall-time	Time that it took for solver to find a solution

Table 3. Optimizer solver output.

The variables data shows the key output values regarding the optimal actions, see Table 4. The length of the lists coincides with the one of consumption prices.

variable	description
Charge	How much to charge on each timestep (in kW) to minimize the cost
Discharge	How much to discharge on each timestep (in kW) to minimize the cost

Table 4. Optimizer variables output.

The info data covers the additional information about the asset and the environment, see Table 5. The length of the lists coincides with the one of consumption prices.

variable	description
Battery level	Battery level (in kWh) at the end of each timestep when the battery is controlled by the optimized schedule
Environment from grid	Amount of energy (in kWh) that the environment will consume from the grid, i.e. it is input energy
Environment to grid	Amount of energy (in kWh) that the environment will send to the grid, i.e. ongrid energy

Table 5. Optimizer info output.

The objective data shows the value of the objective function and its components, see Table 6.

variable	description
Value	Total energy cost (in Eur) for the environment for the whole optimization period
Opex	Component of the value field related to the operating cost (energy cost). If opex is negative, the customer is earning money
Depreciation	Compoent of the value field related to the depreciation cost of the battery

Table 6. Optimizer objective output.

Based on this information different usecases can be analysed and conclusions made on the optimal usage of the battery during optimization time depending on baseload, PV, prices etc.

3.3.3 Three cost metrics

In this study we developed three cost metrics to evaluate the efficiency of the battery usage by PV prosumer: cost with user-driven battery, cost without battery and cost with optimized battery.

The first cost metric is the electricity cost with user driven battery. This is how the battery was operated by the user according to the data from FusionSolar dataset without any optimization applied. Thus, for this case the cost value is calculated based on the input and ongrid present in the dataset.

For the second cost metric we used simplest heuristics possible. We calculate the difference between load and PV. If this difference is positive we have to buy corresponding amount of the energy from the grid (create input), otherwise we sell to the grid (create ongrid flow). Based on input and ongrid calculated this way we obtain the second cost metric, where the battery is not being used at all.

The last metric involves intelligent usage of the battery and this is governed by the optimizer discussed above. Let us imagine that there is an oracle who knows the exact load time series for required timestamps ahead. For such oracle prediction, the cost value in this third scenario equals to the cost reported by the optimizer. However, in reality the oracle prediction is not accessible. Instead, we can provide inaccurate load prediction obtained by constructed forecasting pipeline. The cost calculation in the third scenario consists of three steps:

- provide load (and PV) *forecast*
- apply *optimizer* to calculate optimal battery usage strategy provided that load (PV) are governed by forecast

- by using optimized battery and *real* (oracle) load and PV patterns calculate input and ongrid energies and cost value based on production and generation prices.

We call corresponding cost value as expected cost. This is the best approximation for prosumer's electricity bill after using optimized battery. It can be speculated that expected cost has lower bound given by the expected cost calculated for load prediction provided by the oracle. The latter is the same as the cost reported by the optimizer using oracle load forecast.

3.4 Evaluating the pipelines

The aim of the work is to find load prediction pipeline efficient for all prosumers. By pipeline we mean the consequences of selections made to achieve a prediction. These selections correspond to the features, regressor and cross validation related choices explicitly listed in Fig. 4 between the boxes named as "prepared data" and "load prediction".

First, we construct evaluation dataset consisting of the data corrsponding to P prosumers and O origins of predictions common for all prosumers. Thus, evaluation dataset contains essentially $P \times O$ datasets for which load predictions are requested. The promising prediction pipeline has to be applied to evaluation dataset and its prediction results have to be accepted as the best we can suggest. The goal is to find such a pipeline.

In this section different options to build the prediction pipeline for particular prosumer were discussed, including feature selection, regressor selection, several selections for the validation scheme. Different selections will result in different pipelines, let say pipeline A and B. For A and B the upstream metric of interest, test MAE, and downstream metric, expected costs obtained after running optimizer with load predictions, can be calculated. These metrics are calculated for each sample in the evaluation dataset. Having these evaluations for A and B the question arises how to select the best pipeline.

The Friedman statistical test [41] can be used to judge M pipelines. This is non-parametric test aimed to compare performance of multiple algorithms (in our case pipelines) over multiple datasets (in our case samples of evaluation dataset). In Friedman test the pipelines are ranked based on their performance per sample in evaluation dataset and averaged ranks R_i, $i = 1 \dots M$ for individual pipelines are compared. For this Friedman statistic is used

$$\chi_F^2 = \frac{12PO}{M(M+1)} \sum_{i=1}^{M} \left[R_i - \frac{M+1}{2} \right]^2 . \tag{5}$$

For large PO and M, the Friedman statistic is distributed according to χ^2 distribution so that p-value can be easily found and null hypothesis tested. For Friedman test the null hypothesis states that all pipelines emit predictions from the same distribution and thus results in the same accuracy.

The Friedman test does not answer the question which pipeline has the best performance. To approach this question the post-hoc Nemenyi test is used [41]. In the test, the groups of pipelines that differ or differ not are identified. The test makes pair-wise comparison of the pipelines and the difference in performance is considered to be significant if

$$|R_i - R_j| > d_{\alpha,OP,M} = q_{\alpha,M}\sqrt{\frac{M(M+1)}{6OP}}. \tag{6}$$

The value $d_{\alpha,OP,M}$ is called critical distance. To calculate it we use significance level 5%. Values $q_{\alpha,M}$ are taken from the table [11].

The results of post-hoc test will be plotted on critical distance plots, where all pipelines are positioned on the axis of average rank calculated for evaluation dataset. The pipelines whose averaged ranks are closer than critical distance are grouped together meaning that the pipelines in a group have statistically similar performance. The main output of the study will be a report of the pipelines belonging to the leading group on the critical distance plot.

[11] kourentzes.com/forecasting/2014/05/01/critical-values-for-the-nemenyi-test/

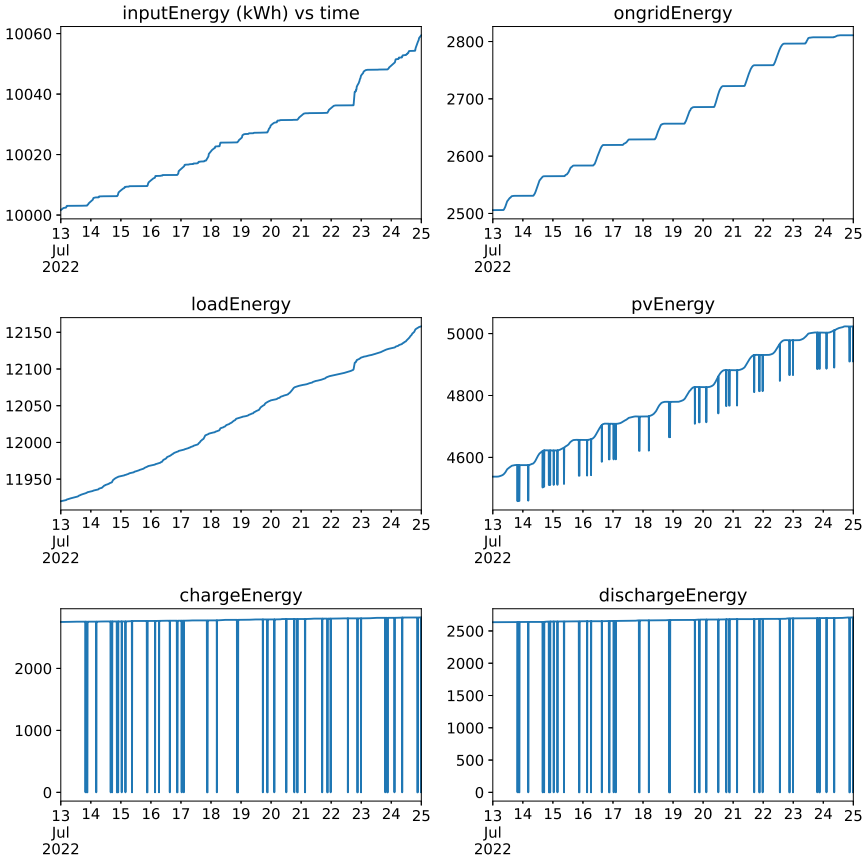


Figure 5. Example raw data (in kWh) from FusionSolar for particular EE client for July 2022. Note the cumulative character of the data and unexpected drops for PV, charge and discharge.

4 Experiments

We start original part of the thesis with exploratory data analysis and steps undertaken to make data ready for further analysis.

4.1 FusionSolar data analysis and preprocessing

4.1.1 Cleaning and preparing the data

In general, FusionSolar data is characterized by the presence of many artifacts. Fig. 5 demonstrates typical FusionSolar data example taken for particular prosumer for the period of two weeks. We clearly see many unexpected drops in the data. Interesting,

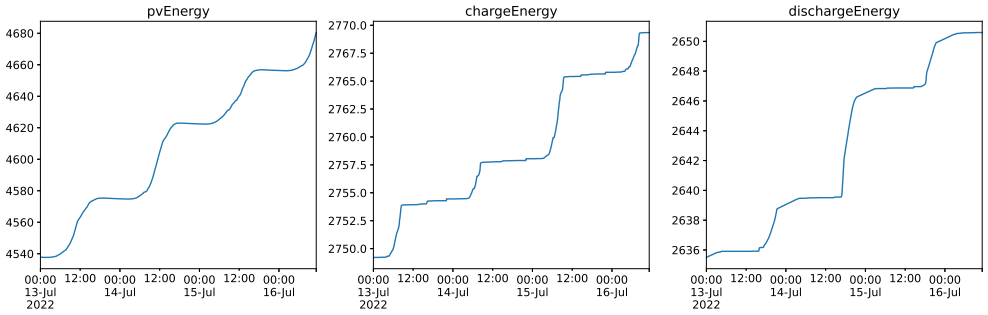


Figure 6. Zoomed in PV, charge and discharge data after removing disconnections. In this example PV and charge energies are well synced, while discharge is lagging. Note negative slope of PV during the nights.

that the drops in PV energy, charge and discharge shown in Fig.5 are found to be always simultaneous. This is also valid for other EE prosumers investigated. Taking this into account and also the facts that data is expected to be cumulative and battery charge cannot decrease to zero within 5 minutes made us conclude that these drops are related to some internal disconnections in the system so that these drops have to be removed. By substituting disconnected values with NA we used PANDAS method INTERPOLATE to fill disconnected values by interpolated ones from neighbour vicinity. After removing disconnections the patterns in the data become more visible, see Fig.6.

One reason why disconnections could be treated as an artifact is related to their distribution. The disconnections are not just a drop in the data within one unit of resolution which is 5 minute long interval. Disconnections may last for several units. In Fig.7 we show that disconnections of particular duration are taking place at exact time. This is not specific for considered EE client, but existed in all client cases investigated. Such a behaviour is likely to be due to system maintainance or other similar reasons so it has to be dropped.

The goal of the preparation phase is to bring the data in the format needed. For further analysis we want to have a data in the hourly resolution with the values indicating the energy usage or production during one hour. This means that the last step in the FusionSolar data preparation is differentiation and aggregation.

By starting with cleared cumulative FusionSolar data given with 5 min resolution we first differentiate it. For instance, this gives us the load energy consumed during 5 min interval. In the aggregation step we sum up differentiated data within each hour. These steps are performed by using PANDAS methods SHIFT and GROUPBY. Described preproseccing leads to the six time-series fluctuating in the vicinity of zero value. The best way to have a look on these time-series is plotting them as heatmaps, see Fig.8. The discussion of these patterns is done in the next chapter.

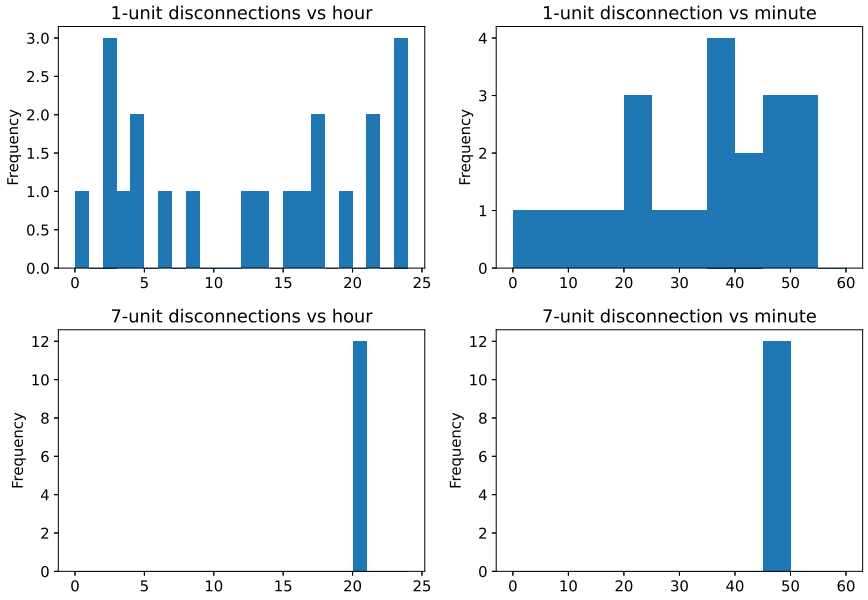


Figure 7. Disconnection statistics. Top: histogram of 1-unit long (5 min) disconnections number vs disconnection hour or disconnection minute. There are in total 21 of such type of disconnections in the data considered. Bottom: histogram of 7-unit long (35 min) disconnections. There are in total 12 of such type of disconnections in the data considered. Note that 1-unit long disconnections seem to be uniformly distributed, but 7-unit long disconnections are always taking place at 20:45.

4.1.2 Exploratory data analysis

For the EE client considered in previous section we can make some conclusions on its energy components, or energy flows. From the Fig.6 we clearly see that PV energy is mostly produced each day around midday. This is expected since we consider summer month July and it is mostly sunny these hours. We also see that solar energy generated is directed mostly directly to the battery to charge it, i.e. PV and charge energy sudden jumps are well correlated. However, the instantaneous correlation with discharge energy is missing, because discharging is happening mostly in evenings so it lags the PV production.

Heatmaps on Fig.8 confirms these conclusion more strongly. By drawing preprocessed time-series by heatmaps allows us to see patterns in the client behaviour. For the considered interval of time the client buys electricity from the grid (input energy) mostly during the night hours. Clear motivation behind this is that usually for flexible electricity packages or electricity spot-price based package the night tariff is the cheapest.

Interesting pattern corresponds to the PV energy generation (PV energy in Fig.8).

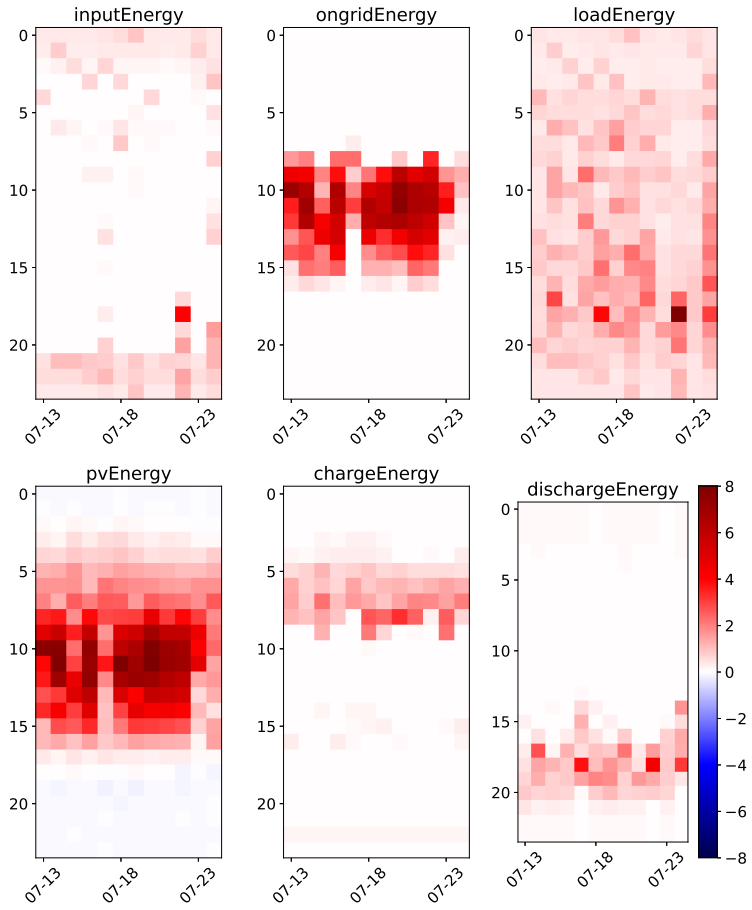


Figure 8. Preprocessed FusionSolar data shown as heatmap with hours and day of the month on vertical and horizontal axes, respectively. Colorbar indicating the value in kWh is common for all energy flows.

The PV energy is produced starting from the early morning and lasts strictly until 16 pm. It may indicate that we are dealing with the client which may have working hours. It is probably not a private household, since the owner wont turn the solar panels off when sun is shining during evening hours. Nevertheless, produced PV energy is mostly sold to the grid (ongrid energy). The energy is sold during working hours obviously because spot prices are the highest these hours reflecting enhanced need for the energy by the market.

Some part of the produced PV energy is used to charge the battery (charge energy) during morning hours. We also clearly see that battery energy is used mostly in evening hours (discharge energy). Discharge energy is directed to cover client's evening loads.

The peak in the load (there is one at 07-22 18:00) could not be covered by battery discharge so that remaining energy is bought from the grid (see peak in input).

Regarding the client's load energy there is no clear pattern in the client's behaviours. We only see increased averaged consumption for hours between 15:00 and 20:00. Note that many different flows contribute to clients load energy: input, PV and discharge energies are involved in some proportion. In general, it is too ambiguous to say that one energy flow transforms directly into another, for instance, input into load, unless all other flows equal to zero.

Notable observation can be also made by looking on the evening hours of PV energy. We clearly see the blueish coloring meaning that PV is negative these hours. This stems from the negative slope of cumulative PV energy seen in Fig.6. How this can be interpreted? The only meaningful explanation is that solar panels may consume energy when turned off or during the nights. This is especially visible for winter months (not presented). Therefore, we assume that solar panels may consume small amount of electricity as well, however, this conclusion requires consultations with FusionSolar specialists.

The sort of analysis presented above is of major importance for establishing the features driving these flows. These features can be categorised into following groups: date-time features (hour, season, day of week etc), covariate features (electricity price in the market, outside temperature etc), lagged features (flow value day ago, average flow value in past days same hour etc). This input will be used below for feature engineering and building machine-learning pipeline.

4.1.3 Balance of the flows

Energy flows existing in the prosumer have to be always checked against balance condition. This means that energy flows which come in to prosumer (input, PV, discharge) have to be balanced by outcoming flows (ongrid, load, charge)

$$input + PV + discharge = ongrid + load + charge, \tag{7}$$

and this condition has to be fulfilled each moment of time for differentiated data. Next we considered cleared differentiated FusionSolar data with 5 min resolution and dropped all differences where interpolation due to disconnections were involved. We confirm that all remaining differences follow nicely balance condition for each 5 min interval separately.

4.2 EE data analysis and preprocessing

4.2.1 Cleaning and preparing the data

In this case data preparation phase was limited to data cleaning only. We have dropped duplicates, checked whether all EICs have the same timestamps and removed duplicated

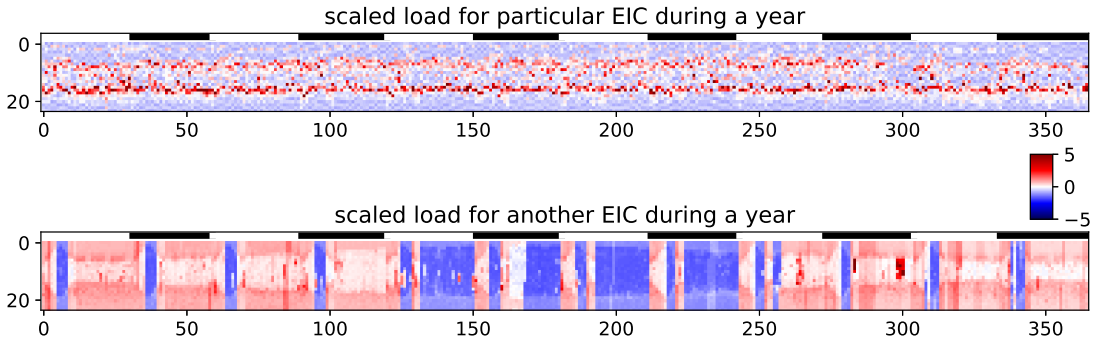


Figure 9. Standard scaled load shown as heatmap with hours and day of the year on vertical and horizontal axes, respectively. Colorbar indicating the load deviation from the yearly average is common for both EICs. White-black dashed vertical line is used to indicate one month duration consecutively from Jan till Dec.

timestamps for individual EICs. We also dropped three EICs whose load time-series were incomplete. As a result, a dataset with 97 EICs was prepared.

4.2.2 Exploratory data analysis

Cleaned EE dataset contains household of various types of load. Two examples are shown in Fig.9

We see that the first EIC is of apartment type. It has increased load during morning and evening hours, while consumption on intermediate hours is below the average. There is no visually significant changes in the pattern as week, month or season changes.

The second EIC has completely different pattern of consumption. On the daily basis this EIC reduces its load in midday. It has visible regular drop in the load in the beginning of each month, probably due to maintenance. Moreover, this EIC lowered significantly electricity consumption in the middle of the year.

The headmaps provide good summaries on the load patterns. However, more detailed investigation can be done. By grouping load data by month, weekdays and hours we create boxplots for two EICs presented in Fig.9. The boxplot in Fig.10 confirm that for the first EIC the monthly and daily load distributions is uniform, however hourly load has two-maxima distribution with increased consumption around 7:00 and 16:00. For the second EIC there are consumption drops in summer months and around midday. Boxplots are also showing that corresponding distributions are skewed because their mean (yellow curve) and median (red marker inside each box) do not coincide.

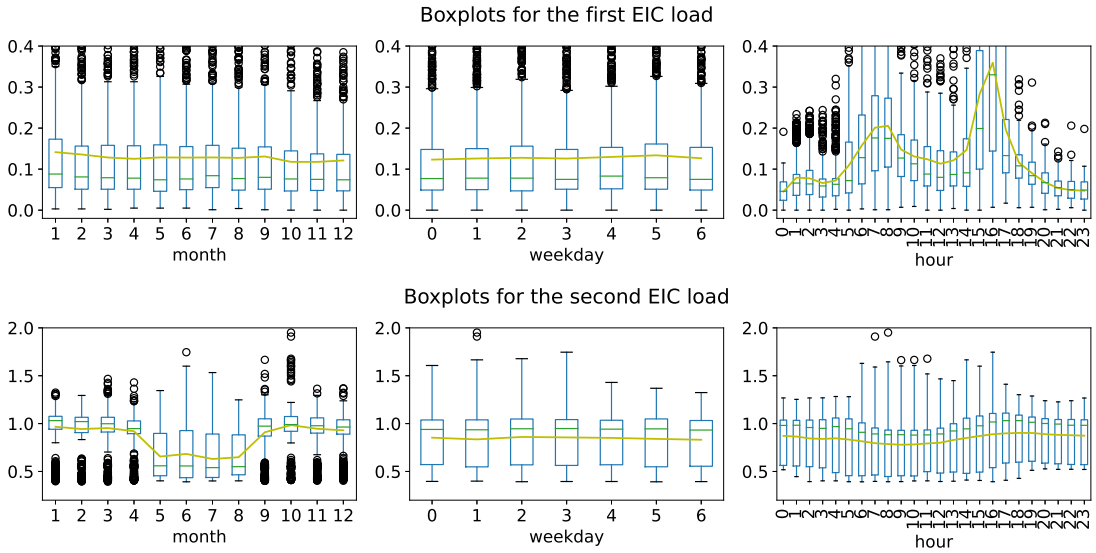


Figure 10. Boxplots to show yearly load data distributions across the months, weekdays and hours. The red marker inside each box indicates the position of the median, while lower and upper edges of the box correspond to first and third quartile, respectively. Black dots indicate outliers and vertical lines connected to box are whiskers. Yellow solid curve corresponds to mean value. Top row of images describe the first EIC, bottom second EIC.

Based on such analysis one can conclude that datetime features (e.g. hour, month) and lagged features (e.g. load last week same hour) may be expressive enough for prediction the load. At the same time, it is clear that there is no universal set of features applicable for all households and some flexibility for choosing them is required.

4.2.3 Household classification and clustering

Although our study is not about classification or clustering available households it may be informative to realize the differences and similarities in consumption patterns by loosely considering this topic. Even naive clustering of the households in terms of their consumption is able to shed the light on various categories of the households sharing the same consumption habits.

Clustering of times-series is challenging problem. The complexity comes from the fact that when time-series is presented as a table then the samples could not be considered as independent ones and temporal ordering between the samples is crucial. There exist specific packages aimed to cluster sequences of ordered data, see, for instance, kernel

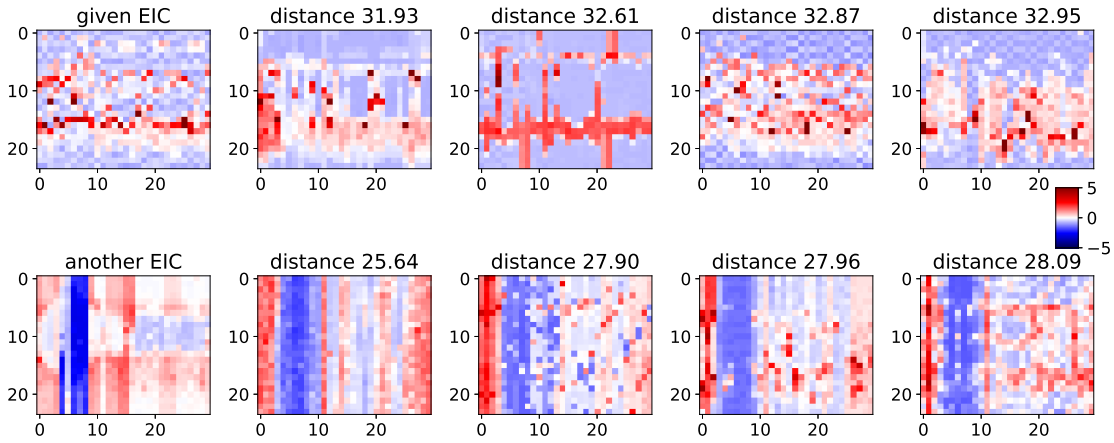


Figure 11. Top: for the first EIC from Fig.9 four nearest neighbours with corresponding distance are shown. Bottom: for the second EIC from Fig.9 four nearest neighbours with corresponding distance are shown. Note that standard scaled consumption for November only is considered. In each heatmap hour and day of the month are used for vertical and horizontal axis, respectively.

k-means or time-series k-means implemented in the package TSLEARN[12]

The presentations of time-series as heatmaps preserve data temporal ordering. The heatmap images can be used for classification or clustering when the comparison pixel by pixel is performed. Below two approaches are applied - KNN and k-means.

For KNN, we calculate Eucledian distance between different heatmaps. For a given household, the selected number of nearest neighbours are then found based on distance between given heatmap and all remaining ones. In Fig.11 four nearest neighbours are shown for both households from Fig.9 by considering November data only. We clearly see that KNN allows to identify the households with similar consumption patterns. Thus, classification task can be solved to some extend. Namely, as shown in Fig.11 the KNN neighbours of the first EIC all have double-peak consumption during a day, while for the second EIC all neighbours are characterised with several days of reduced consumption starting at the end of the first week.

Another straightforward idea is to cluster the households by k-means algorithm. For the algorithm the number k of the clusters needs to be predefined. By throwing k cluster centers onto the data randomly, the expectation (datapoint assignment to closest centroids) and maximisation (calculating new centroids based on datapoints assign to the cluster) process is executed iteratively until centroid positions converge. Note that

[12] https://tslearn.readthedocs.io/en/stable/gen_modules/tslearn.clustering.html

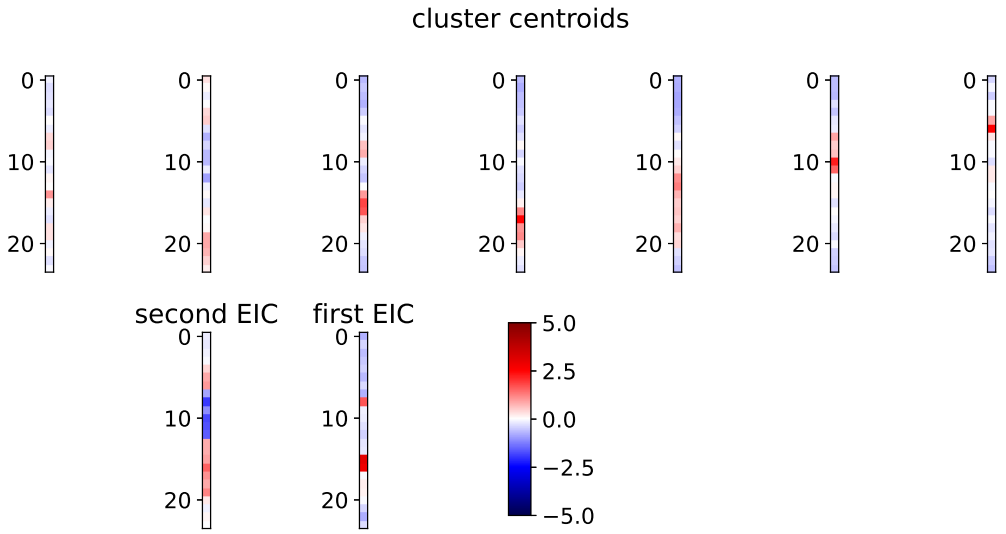


Figure 12. Top row: cluster centers found by k-means for December 30. The value of k is determined by rule of thumb $k = \sqrt{n/2}$, where n is the number of samples. Bottom row: December 30 data of the the first and second EICs from Fig.9 indicating that they belong to the third and second cluster, respectively. Note that December 30. data was standard scaled for all households as indicated by colorbar.

k-means results may be very reliable, for instance, for MNIST dataset the accuracy of k-means may reach 90% [13].

To make analysis of heatmaps more simple we reduce dimensionality and consider only heatmaps for particular day only, let it be December 30. Such a simple approach can serve as a baseline for further analysis.

From. Fig.12 we see that centroids mimic different consumption patterns during December 30. The considered households from Fig.9 stem clearly from cluster two and three. It is also interesting to note that according to k-means analysis the households are distributed between the clusters as {20, 11, 15, 14, 17, 8, 9}, so that distribution is quite balanced.

The main conclusion from this exploratory analysis is that EE dataset contains several categories of the households characterized by very different consumption patterns. This means that in order to describe them effectively, the flexible approach is needed able to adjust to particular household and put more attention to the features peculiar to its consumption.

[13] medium.com/@joel_34096/k-means-clustering-for-image-classification-a648f28bdc47

4.3 Feature engineering and selection

We have seen from exploratory data analysis that efficient load prediction pipeline should use date-time features, covariate features and lagged features. Note that we wont fix manually the feature set peculiar for particular prosumer, but will apply several selection techniques to make feature selection process machine-driven. Let us discuss in details how the general feature pool is build in this study.

4.3.1 Date-time features

Date-time features are stemming from the information given by timestamp. The following 46 features are created:

- month: ordinal categorical feature varying from 1 to 12
- month sin/cos: polar coordinates for month values distributed on a unit circle; month=1 and month=12 both have coordinates (0,1).
- one-hot encoded month interpreted as nominal categorical feature
- day: day of month; varies from 1 to 30(31).
- day sin/cos: polar coordinates for day (same logic as for month sin/cos)
- hour: varies from 0 to 23
- hour sin/cos: polar coordinates for hour
- day of year: varies from 1 to 365
- day of year sin/cos: polar coordinates for day of year
- week of year: varies from 1 to 52
- week of year sin/cos: polar coordinates for week of year
- weekday: varies from 0 to 6
- weekday sin/cos: polar coordinates for weekday
- one-hot encoded weekday interpreted as nominal categorical feature
- quarter: value from 1 to 4
- quarter sin/cos: polar coordinates for quarter
- one-hot encoded quarter interpreted as nominal categorical feature

- is month starts: binary feature with value 1 for first day of the month
- is month ends: binary feature with value 1 for last day of the month

Note Estonian holidays are not considered here, but such a feature may be of relevance as well.

4.3.2 Lagged features

The features of this category are related to the lagged values of the target y (load) and different aggregations of the lagged values. Note that here the prediction horizon has to be taken into account. Since the pipeline will predict next 24 hours ahead starting from the origin of prediction provided, we can build lagged features only with lags larger than 24 hours. This is because the target value has to be treated as unknown for the test interval so that by engineering lagged features for the test data we cannot rely on the target values within test interval. In total there are 51 features engineered as follows:

- from y-24-1 to y-24-24: load value 25..48 hours ago. The shift by 24 hours is determined by the length of the prediction horizon.
- y-24_24hPmean, y-24_24hPstd, y-24_24hPmin, y-24_24hPmax: mean, std, min and max aggregations based on past 24 hours (referred as 24hP) for y-24, i.e. aggregations calculated from the array of the values y-24-1. . . y-24-24.
- from y-1d to y-7d: load value exactly 1..7 days ago.
- y_7dPmean, y_7dPstd, y_7dPmin, y_7dPmax: mean, std, min and max aggregations based on past 7 days (referred as 7dP) for y, i.e. aggregations calculated from the array of the values y-1d. . . y-7d.
- from y-1w to y-4w: load value exactly 1..4 weeks ago.
- y_4wPmean, y_4wPstd, y_4wPmin, y_4wPmax: mean, std, min and max aggregations based on past 4 weeks (referred as 4wP) for y, i.e. aggregations calculated from the array of the values y-1w. . . y-4w.
- y_5wd/2wePmean, y_5wd/2wePstd, y_5wd/2wePmin, y_5wd/2wePmax: mean, std, min and max aggregations based on past 5 working days (referred as 5wd) or past 2 weekend days (referred as 2we) for y depending on what kind of a day y corresponds to (working day of weekend). This is aggregation calculated from the array of the values y-1wd. . . y-5wd, if y corresponds to working day, or from the array of the values y-1we, y-2we, if y corresponds to weekend day.

All engeneering is done by PANDAS method SHIFT by playing with frequencies and periods. For last features the frequency was specified by offsets BUSINESSDAY and CUSTOMBUSINESSDAY.

4.3.3 Covariate features

In this study we consider two covariate time-series: outside temperature and spot price. The reasoning behind this selection is clear. If outside temperature is low (or high), then heating device (or air conditioner) consumes more energy. Similarly, if electricity price is high, the prosumer may intentionally reduce its load during peaked hours. The price was downloaded from Elering [14]as a csv file. For the outside temperature we used rough approximation by neglecting real addresses of the prosumers and using temperature taken from the internet [15]for Tartu. Note that for covariates the past as well as future values are known. For the temperature, the future values are predictions from the weather models. For the price, the horizon for known future values is affected by how the price information is issued, see Sec. 3.3.1. In this study we neglect complications related to the issuing of the covariate future predictions. There are 62 features engeneered from each covariate c:

- from c, c-1 to c-24: covariate value 1..24 hours ago and simultaneous covariate value.
- c_24hPmean, c_24hPstd, c_24hPmin, c_24hPmax: mean, std, min and max aggregations based on past 24 hours (referred as 24hP) for c, i.e. aggregations calculated from the array of the values c-1. . . c-24.
- c_24hPstandard, c_24hPminmax: c value scaled standard and minmax way by c_24hPmean, c_24hPstd, c_24hPmin and c_24hPmax.
- from c+1 to c+12: covariate value 1..12 hours later.
- c_12hFmean, c_12hFstd, c_12hFmin, c_12hFmax: mean, std, min and max aggregations based on future 12 hours (referred as 12hF) for c, i.e. aggregations calculated from the array of the values c+1. . . c+12.
- c_12hFstandard, c_12hFminmax: c value scaled standard and minmax way by c_12hFmean, c_12hFstd, c_12hFmin and c_12hFmax.
- c_12nh/12dhPmean, c_12nh/12dhPstd, c_12nh/12dhPmin, c_12nh/12dhPmax: mean, std, min and max aggregations based on past 12 night hours (referred as 12nh) or past 12 day hours (referred as 12dh) depending on the considered hour for c. We defined day hours as 8:00. . .19:00 and night hours as 20:00. . .7:00.
- c_12nh/12dhPminmax and c_12nh/12dhPstandard: c value scaled standard and minmax way by c_12nh/12dhPmean, c_12nh/12dhPstd, c_12nh/12dhPmin and c_12nh/12dhPmax.

[14]`https://dashboard.elering.ee/en/nps/price?`
[15]`ilmateenistus.ee`

- c_12nh/12dhFmean, c_12nh/12dhFstd, c_12nh/12dhFmin, c_12nh/12dhFmax: mean, std, min and max aggregations based on future 12 night hours (referred as 12nh) or future 12 day hours (referred as 12dh) depending on the considered hour for c.
- c_12nh/12dhFminmax and c_12nh/12dhFstandard: c value scaled standard and minmax way by c_12nh/12dhFmean, c_12nh/12dhFstd, c_12nh/12dhFmin and c_12nh/12dhFmax.
- c_dayMinmax: c value minmax scaled by the min and max values of the covariate during the considered day.

For engineering, the PANDAS methods mentioned previously were used.

4.3.4 Feature selection

After feature engineering step we have collected the pool with more than 200 features. By applying particular selection strategy this number can be significantly reduced. For AUTOFEAT selection the usual output is around 20 features, while TSFRESH selection leaves around 100 features.

Two manual selections from feature pool were also used during the study. The Uku's selection contains only 2 features for predicting the load: outside temperature and hour. The Kristjan's set contains 80 features. Corresponding selection is the following:

- load y lagged and historically aggregated values: from y-24-1 to y-24-24, y-24_24hPmean, y_4wPmean, y_7dPmean, y_5wd/2wePmean
- temperature and price covariates c local values and comparison to daily minimum and maximum: c-12. . . c-1, c ,c+1. . . c+12, and c_dayMinmax

4.4 On the regressor selection

In the beginning of the study we started with Holt-Winters exponential smoothing model and linear regressor implemented in the package DARTS [16].The former one can be used as simple baseline with minimal amount of the parameters to tune. The main idea of the model is to decompose time-series into trend and seasonal components. By minimizing model error, the smoothing factors are found for the components. The parameters which can be tuned are how components contribute to the model: whether additively or multiplicatively.

[16] https://unit8co.github.io/darts/index.html

The linear regression model from DARTS allows to regress on target past values as well as on covariates past and future values. The number of the target and covariate past lags and future shifts are tuning parameters together with the length of historical data.

By implementing grid search cross validation the optimal hyperparameters for these models were found for particular households from EE dataset. However, shortly it became clear that Decision Tree based model outperforms these DARTS regressors so that we did not considered them in later stages of the study.

The main regressors used in this thesis are Decision Tree and Gradient Boost. Other regressors were also tested (linear, regularised linear, Random Forest, SVM, XGB as well as various modifications of SARIMAX model), but not so detailed so we wont stop on them.

4.5 Finding best hyperparameters

By creating prediction pipeline, each selection taken may be considered as fixing some hyperparameter. There exists hyperparameters not related to the regressors directly but to the prediction pipeline itself, for instance, the length of the training data. Since we are interested in the short-term prediction it is correct to assume that recent historical data drives mostly the load prediction and data from previous month/quarter/year/decade is not of big importance. If such historical data is included into training set, this will probably make short-range prediction worse. Thus there is uncertainty related to the length of historical data and this has to be overcome during training phase. Note also that this remark is in close connection to the data shift problem: by considering more recent data for the training the model there is larger probability that data belongs to the same concept.

To find optimal values of the hyperparameters we use validation set as large as 20% of historical data and implement different split options as was discussed in Sec. 2.4. We use the following scenarios:

- last fold is true: validation set corresponds to the most recent historical data located as close as possible to the origin of prediction
- last fold is false: validation fold appears in the historical data arbitrary in time. In this case usual 5-fold cross validation is applied.
- rolling is true (false): rolling(expanding)-window cross validation is used.

In Fig.3 the difference between these options was shown.

In cross validation, the pipeline with lowest mean validation error is usually selected as the best option to proceed. In addition to this default behaviour we implemented the version where all cross validation results are sorted by the discrepancy between mean train error and mean validation error and top-5 list of pipelines is returned. The pipeline

with the lowest validation error from this top-5 list is then returned as the best option to proceed. We hope that this approach allows to select the pipeline with lower overfitting. So two cross validation objectives are used:

- best validation error true: default cross validation strategy.
- best validation error false: strategy aimed to reduce overfitting.

When cross validation strategies are presented we implement random search in the regressor's hyperparameter space. For Decision Tree regressor this space was defiend as follows:

- splitter: best (default) or random
- min samples split: 2(default). . . 10
- min samples leaf : 1(default). . . 5
- max features: auto (default), sqrt, or log2
- max depth: None (default), 3, 5, 7, 10, 15, 20, 30, or 40

For gradient Boost regressor the search parameters are

- n estimators: 50, 100 (default), 150, 200
- max depth: 3 (default) . . . 8
- learning rate: 0.1 (default) or 0.2
- subsample: 0.7, 0.8, 0.9, or 1 (default)

In both cases random state was fixed and loss function was chosen to be absolute error.

4.6 Understanding battery optimization

Previously the battery optimizer was introduced. Let us discuss now how the optimization results look like and obtain necessary intuition in this part of the study.

4.6.1 Optimization objective

To have better understanding on the optimization procedure let us take a look on optimization objective in more formal way. Let us assume for simplicity that depreciation is absent. Let $price_i + \Delta_i$ be a consumption price and $price_i$ production price at time moment indexed by i. Here Δ_i is the difference between consumption and production prices at time moment i. Corresponding input and ongrid energies are $input_i$ and $ongrid_i$ so that electricity cost reads as

$$cost = \sum_{i=1}^{n} [input_i(price_i + \Delta_i) - ongrid_i \cdot price_i], \tag{8}$$

where n defines optimization length. In the experiments considered below $n = 24$. Each moment of time the balance condition (7) is fulfilled. By expressing $ongrid_i$ from balance condition, the cost gets the form

$$cost = \sum_{i=1}^{n} (charge_i - discharge_i) price_i + \sum_{i=1}^{n} (load_i - PV_i) price_i + \sum_{i=1}^{n} input_i \Delta_i \tag{9}$$

The optimizer wants to make the cost as small as possible, even better to make it negative so that the prosumer will earn money. What contributions can be adjusted by optimizer? It is clear that the second sum cannot be changed by optimizer, because it is fixed by the price and the load and PV predictions.

If difference in prices vanishes, $\Delta_i = 0$, then the third sum disappears and optimizer deals with the first term only. Optimizer will try to sync $charge_i - discharge_i$ with $price_i$ in a way to make this term as negative as possible. Of course, optimizer follows here all related constraints, for instance, for maximal charge level and limitations due to the fuse etc. Important is that the battery behaviour does not depend in this case on load and PV, but is determined solely by the electricity spot prices. It is clear that the case $\Delta_i = 0$ will be never realized in reality, but it is still very didactic to have better understanding on this limiting scenario.

In more realistic case, $\Delta_i \neq 0$, the third sum comes into play. Note that this sum is strictly positive. The ultimate goal of the optimizer is to make this term be equal to zero, i.e. reduce input to zero. This is only possible when battery is synced with the load and PV in a way that no need to buy grid electricity appears at all and input electricity flow is zero.

To see how general these conclusions are, let us now consider some example outputs from the optimizer. First, we consider one day long optimization interval and provide to the optimizer different load predictions and see optimized battery patterns. One prediction is so called oracle prediction, i.e. this is perfect prediction for the baseload. Another one is a prediction with the peaked load, see blue curves in Fig.13. These two

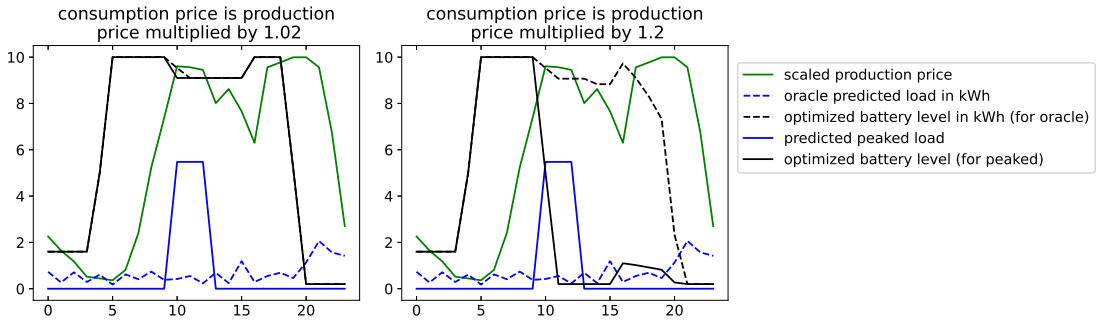


Figure 13. Battery level optimized for 24h ahead for two load predictions: oracle (dashed blue) and peaked (solid blue) prediction. Left: if difference between consumption and production prices is vanishing (corresponding multiplier approaches 1), the optimized battery does not rely on the load, see dashed and solid black curves for oracle and peaked load, respectively. Right: if difference between consumption and production prices is high enough, the battery optimization is driven by the load.

predictions were taken intentionally because they reflect very different load time-series. The idea here is to see how responsible is the optimization to load distinctions.

From Fig.13 we see that dashed and solid black curves pointing to the battery charge level for two different load profiles considered depends strongly on the difference in production and consumption prices. If this difference is vanishing (left subfigure), then optimized battery does not follow the load. Optimizer knows 24h ahead prices and charges battery fully by cheapest electricity available at hour 5. During the optimization period, the price has double-peak behavior with local minimum at hour 16. Optimizer waits until the first price peak comes at hour 10 and then uses energy stored in the battery to cover load or to sell expensive electricity to the market. After that, optimizer waits until the local minimum of the price at hour 16 to restore fully battery level.

In the end of optimization interval, the price is high at hour 19 so that optimizer decides to use all battery energy to cover the load or for ongrid. Note that optimizer does not have any information about the future hours after optimization interval is over. Therefore optimizer prefers fully discharge in the end of optimization to decrease electricity cost as much as it can. As a result, optimizer always discharge fully by the end of optimization so that battery level drops to the minimal value determined by the input for the minimal allowed charge level. The battery pattern described is not sensitive to the load prediction.

When difference in production and consumption prices becomes higher, see Fig.13 right panel, the load distinctions come into play. We see that for peaked load the optimizer decides to discharge fully to cover the peak and avoid buying expensive electricity from

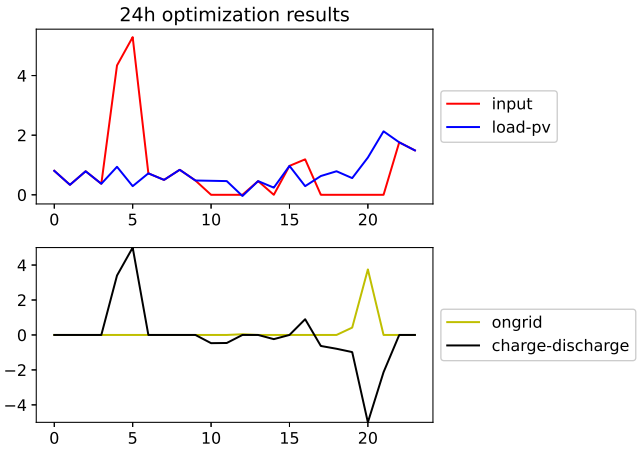


Figure 14. Detailed optimization results for the oracle case from right panel in Fig. 13. By providing load and PV, optimizer calculates battery pattern as well as energies taken from the grid (input) and given to grid (ongrid). Top panel: compares input energy found with the load provided. For considered case PV is practically zero all the time. Bottom panel: compares ongrid energy and charge/discharge found. We plot difference between charge and discharge, which is positive (negative) when charging (discharging).

the grid at hours 10 and 11. After that, optimizer uses local minimum in the price at hour 16 to charge slightly the battery and sell this energy to the grid when price becomes higher at hour 19. For another load predicted by oracle the optimized battery profile is completely different.

Together with battery pattern optimizer calculates the input and ongrid energies based on optimized battery and load and PV provided. In Fig.14 we can clearly see for what purposes the battery energy should be used according to the optimizer. In the beginning of optimization interval input energy is used to cover the load. Also input energy peak at hour 5 indicates that grid energy was bought to charge the battery fully. In the end of optimization interval the battery was fully discharged at hour 20. This battery energy was used to cover the load so that input energy was reduced to zero. The excess battery energy left was sold to the grid as ongrid peak indicates. Note also that Fig.14 demonstrates the intention of the optimizer to reduce input energy as close to zero as possible. In the considered example optimizer succeeded with this task at least for the hours with highest electricity price.

In the next section we discuss another important output parameter which is the electricity cost.

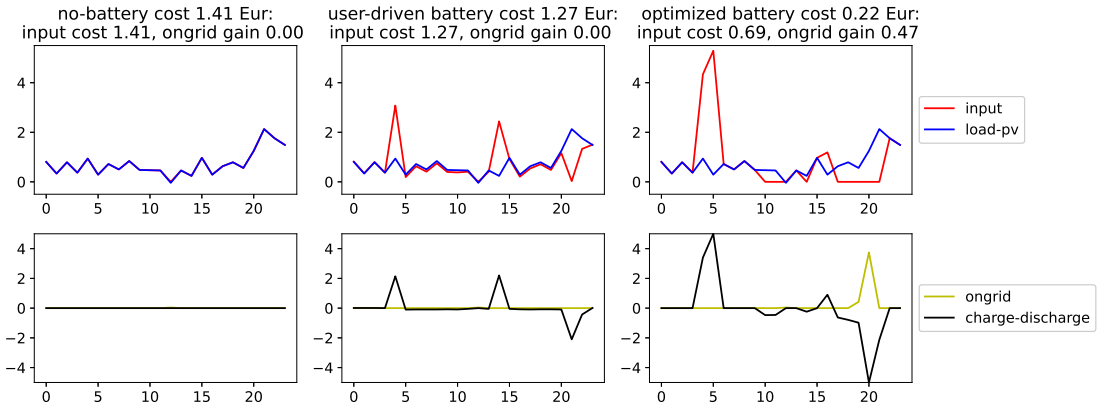


Figure 15. Three cost metrics with corresponding energy flows. Each column of panels corresponds to particular scenario: no-battery, user-driven battery and optimized battery. Cost value with input and ongrid components is indicated on top of each column. The setup is the same as in Fig.14, in particular, consumption price is production price multiplied by 1.2.

4.6.2 Three cost metrics

Three cost metrics were introduced in Sec. 3.3.3. Fig.15 shows the energy flows and cost values calculated for three scenarios: no-battery, user-driven battery and optimized battery. Note that oracle prediction is used in this example. In no-battery case the input is directly governed by load and PV. In the considered case PV was vanishingly small. As a result, there is no ongrid gain. The cost value in the no-battery scenario is the highest.

For the user-driven case, the cost is slightly smaller. We see that user charged and discharged the battery in the hours with suitable prices (prices are shown in Fig.13). However, the amount of stored energy was not big so that possibilities to reduce input were very limited. In terms of the cost, the effect of the user-driven approach to save the money is minor compared to situation with missing battery.

As for optimized battery case discussed previously we see that cost value is indeed significantly reduced. There are two reasons for that. First, optimizer managed to reduce input for expensive hours. Second, optimizer earns money by speculating on the prices with the help of the battery.

Based on these results we conclude that, if the load and PV are known in advance for some period of time, then the behavior of the battery should be calculated by the optimizer. The results of these calculations will be always the best-case solution, meaning that the prosumer electricity cost will be as low as possible. This is of course valid for the oracle prediction discussed in the example. For the inaccurate prediction, however, the expected cost found by means of the optimized battery is still the overall better solution

then no-battery cost or user-driven cost. Therefore, the best we can do by having some load prediction is to drive the battery with the guidance from optimizer. As a result, the only factor influencing the downstream metric, which is expected cost, is the load prediction.

5 Results and discussions

5.1 Experiments with selected FusionSolar prosumer

By knowing the postal code of selected prosumer the weather data was obtained from the internet together with spot prices for the time interval of interest. We consider different prediction pipelines for 24h ahead load forecast. To simulate different experiments each hour in February 2023 is treated as distinct origin of prediction. The production and consumption prices are determined by Eq. (4).

5.1.1 Test MAE

Together with the complex prediction pipelines described above the very basic predictors will be also tested for the reference. These are:

- naive prediction: predict same load hour by hour as was last 24 hours
- constant prediction: constant may be 0 (referred as const0), previous 24h load average (constA), previous 24h load median (constM)
- past 4 weeks average (refered as 4wPmean): for each hour after origin of prediction we consider same weekday same hour past 4 week values and average them

Simulation results are shown for some pipelines in Fig. 16. Here we see test MAE for different origins of predictions located in February. The main outcome from this results is that basic predictors are on average worse than complex pipelines. This means that constructed pipelines seem to be working because they are better than baselines.

Another important message that there is usually no ultimate leader pipeline in terms of test MAE: some pipeline may be the best for particular interval of origins, but totally fail in other interval. This situation is demonstrated in Fig. 16 for two origins marked by vertical dashed line. In first case, the naive basic predictor (yellow) demonstrates superior performance, while in the second case it is badly unsynced with the ground truth (oracle prediction).

Let us consider the pipeline termed as gb 10m uku: this is gradBoost trained on 10 month historical data with Uku's features and no cross validation is used (hyperparameters have default values). This pipeline seems to be so far the best one. In Fig. 17 we plot test MAE and the load ground truth as a heatmaps together for better comparison. We see that load is more or less uniformly distributed with some excess in evenings and several outliers seen as black dots and dark areas in right panel in Fig. 17. Very probably what these outliers have weekly periodicity. For the days where such outliers are missing we have small test MAE: notice two wide light vertical stripes in left panel. However, there are also clear dark vertical stripes in test MAE heatmap. Corresponding predictions clearly cover the 24h intervals with load outliers. From these exploratory analysis we

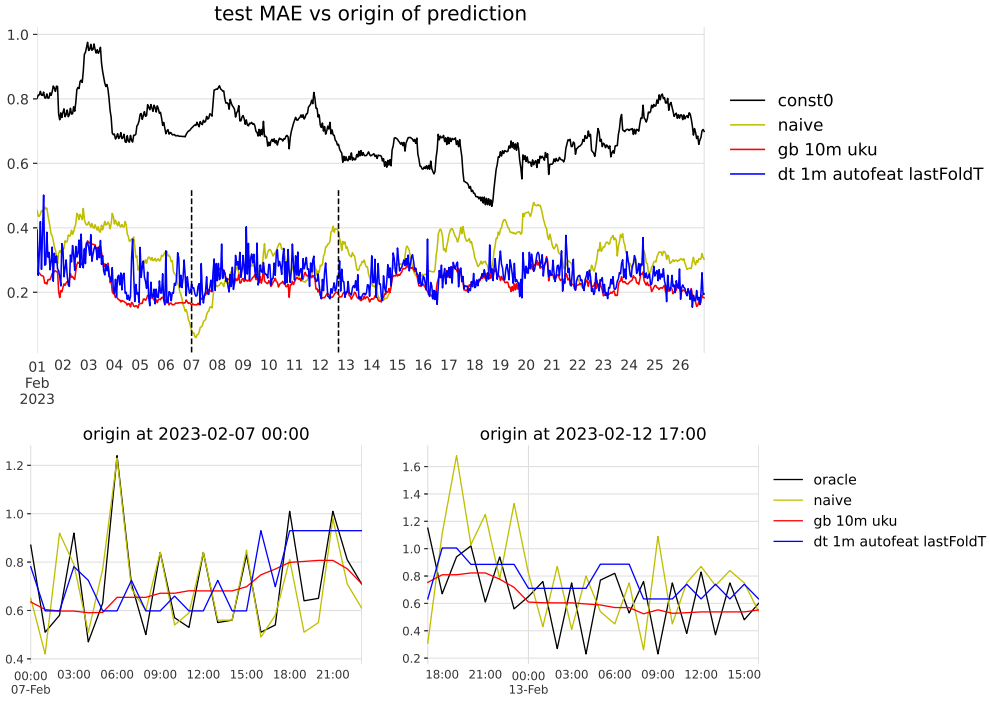


Figure 16. Top: test MAE vs origin of prediction for some 24h ahead predictors: const0 is const=0 prediction, naive is the load as in previous 24 hours; gb 10m uku is gradBoost trained on previous 10 months of historical data with Uku's features; dt 1m autofeat lastFoldT is Decision Tree trained on 1 month of historical data with autofeat-selected features and last fold validation equals true. Vertical dashed lines indicate situations where there is large discrepancy in the test MAE. Bottom: predictions (in kWh) are shown for origins of predictions indicated by vertical lines in top panel. In the left panel the naive prediction almost coincides with the oracle (solid black) one, in the right panel gradBoost prediction has the lowers error.

can suggest that considered pipeline fails to predict weekly seasonality and the feature coresponding to the day of week may have positive impact.

Based on the analysis presented in Fig. 16 one can visually estimate the best pipeline by looking for the one having lowest test MAE. But for the correct estimation the statistical tests have to be performed. Fig. 18 demonstrates critical distance plot for test MAE. We see that leader pipeline called gb 10m uku performs statistically the same as other gradBoost based pipelines. At that, all Decision Tree based pipelines are statistically worse than the leader pipeline. Basic predictors called naive and const0 forecast statistically worse then gradBoost and Decicion Tree based pipelines. This results are expected so that our approach is legit.

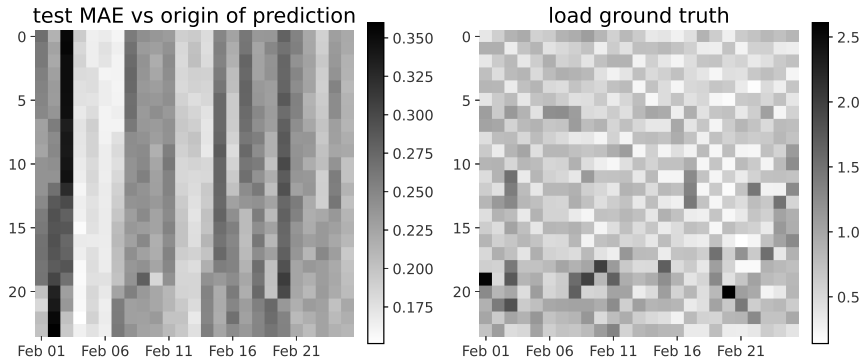


Figure 17. Left: heatmap for test MAE of the pipeline gb 10m uku. Right: heatmap of the target load. In both cases we plot hour on vertical and day in February on the horizontal axis. In left panel the color of the pixel corresponds to the test MAE for the prediction whose origin is determined by the pixel location. In right panel the color of the pixel shows the load value for the hour determined by the pixel location.

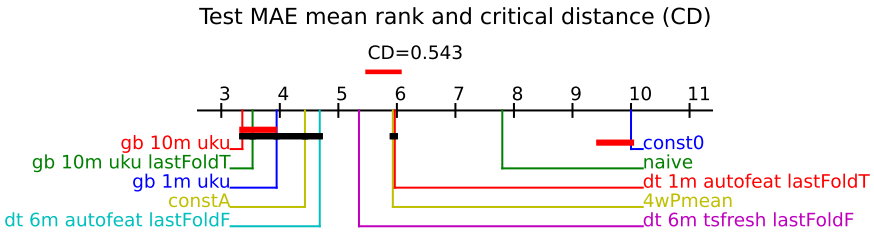


Figure 18. Test MAE critical distance plot. On the horizontal axis the pipelines mean rank is shown. Red section is critical distance. Critical distance is also positioned near the first and the last ranked pipelines for convenience. Black section joins neighbour pipelines separated by the distance smaller than critical one. Here the decoding of the pipeline names is the following: regressor, trainset length, feature selection, cross validation option; for instance, dt 6m tsfresh lastFoldF means Decision Tree trained on 6 month long historical data with TSFresh based feature selection and usual 5-fold cross validation.

Note that in terms of averaged test MAE shown in Table 7 the difference between pipelines may seem minor. However, due to the critical distance plot we are sure that leader pipeline is statistically better then constA, for example.

gb 10m uku	gb 1m uku	constA	dt 6m autofeat lastFoldF	4wPmean	naive
0.225	0.228	0.235	0.234	0.248	0.312

Table 7. Averaged test MAE for selected pipelines.

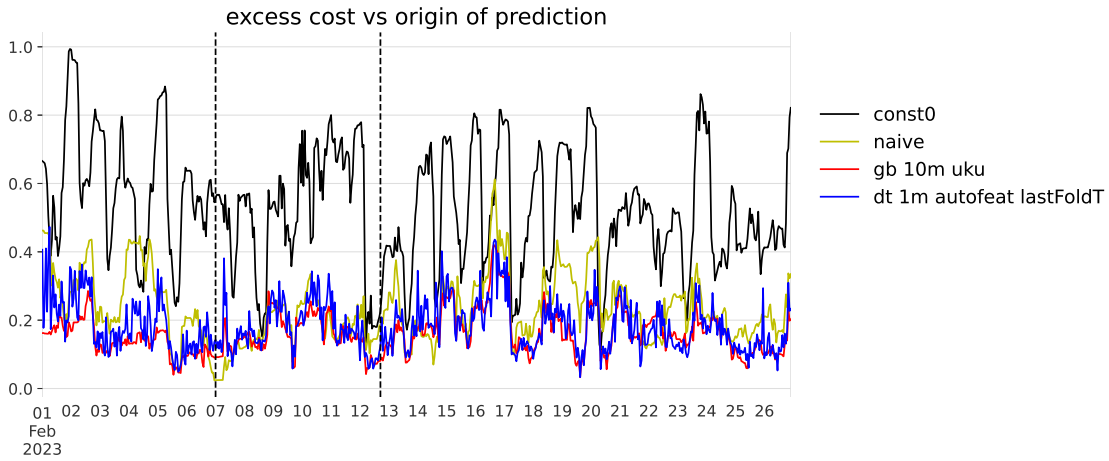


Figure 19. Excess cost (in Eur) vs origin of prediction for some 24h ahead predictors. Predictors are the same as shown in Fig. 16. Vertical dashed lines indicate situations where there is large discrepancy in the test MAE, see Fig. 16.

5.1.2 Expected cost

Let us now discuss how predicted load patterns influences expected cost. For the study, the expected cost metric is not very informative and we will consider excess cost: the difference between expected cost found for the pipeline prediction and expected cost calculated for the oracle prediction. Excess cost measures how much the prosumer pays due to the fact that load prediction is not ideal.

Fig. 19 shows excess cost calculated for the list of origins under consideration. One can notice that test MAEs are usually correlated with expected cost: lower error results in the lower excess cost, compare simulation results for origins of prediction selected by vertical dashed markers. Note also that zero expected cost is accessible only for oracle prediction. It is important checkpoint that pipelines are not producing negative expected cost. Usually it indicates some inconsistency in the appraoch, for instance, wrong optimization results due to improper rounding, unexpected input (depreciation is non-zero) etc.

Next step is to perform statistical comparison of the pipelines in terms of downstream metric, see Fig. 20. The results confirm that the leader pipeline called gb 10 uku outperforms basic pipelines and Decision Tree based ones. The situation is very similar

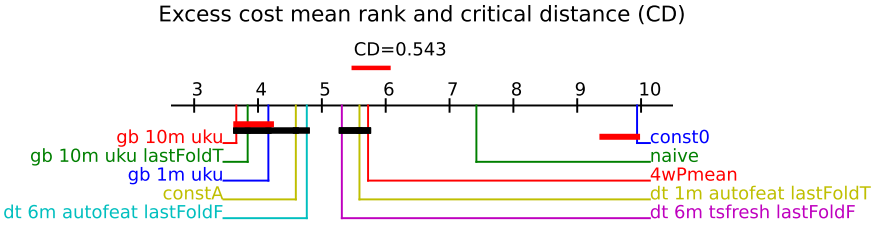


Figure 20. Excess cost critical distance plot. The meaning of the red and black sections is explaind in Fig. 18.

gb 10m uku	gb 1m uku	constA	dt 6m autofeat lastFoldF	4wPmean	naive
0.161	0.163	0.171	0.169	0.177	0.233

Table 8. Averaged excess cost (in Eur) for selected pipelines. Note that for const0 the value is 0.53 Eur.

to what we have discovered previously in terms of upstream metric.

It is also very instructive to look at averaged excess cost for pipelines, see Table 8. One can notice that statistically best pipeline results in only minor cost improvement compared to some basic predictors, for instance, constA. Therefore, it may be not meaningful to proceed with complex pipeline, if simpler pipeline guarantees the relatively close cost value. To fully prove this statement more comprehensive simulations are needed.

In the company the optimizer was used to provide some prosumers the battery patterns to follow. For this initiative the const0 prediction was implemented. For const0 predictor the averaged excess cost is found to be 0.53 Eur. This is more then 3 times larger than averaged excess cost for the leader pipeline.

5.1.3 Correlation between up- and downstream metrics

We already noticed that better prediction leads to lower cost. Fig. 21 illustrates correlation between test MAE and excess cost for some pipelines. We see that correlation is always positive and can be treated as moderate. This gives some support to the idea that better prediction drives cost to the optimal value obtained in the oracle case.

5.2 Experiments with many prosumers

In this part we consider EE dataset and imagine that it corresponds to the PV prosumers. We fix PV energy to be zero and battery initial charge level to be 3 kWh. With these values fixed battery optimization and cost value rely only on load forecast.

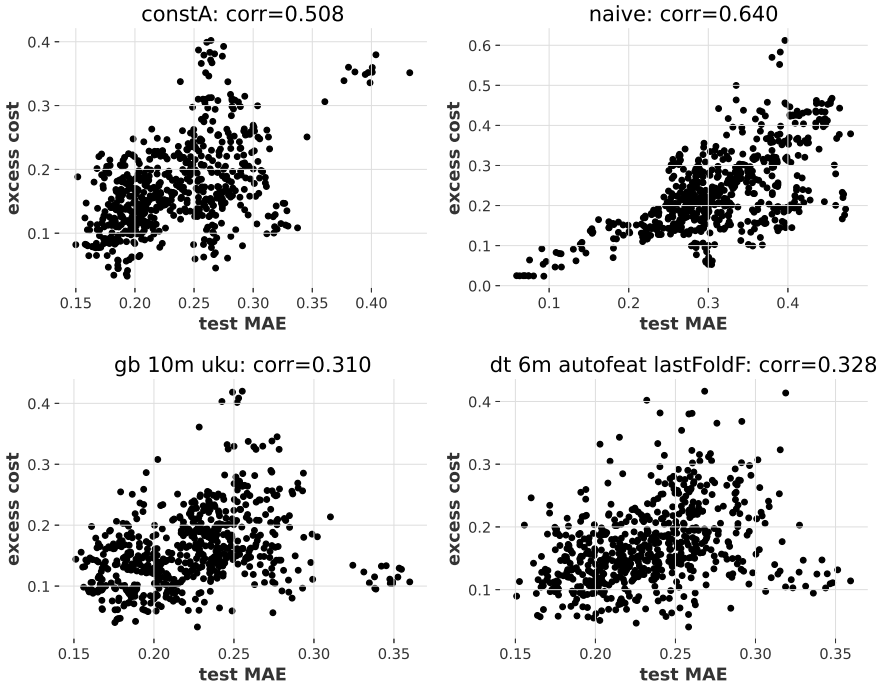


Figure 21. Pearson correlation between excess cost and test MAE for different pipelines.

In case of EE dataset we consider 20 randomly selected households during year 2021 and pick randomly 30 origins of predictions during that year. These origins are common for 20 households selected. The corresponding data is used then in the simulations.

5.2.1 Test MAE

For EE data with 20 households and 30 origins of predictions the test MAE critical distance plot is shown in Fig. 22. We clearly see expected result that gradBoost based pipelines are statistically superior than Decision Tree ones. It is also interesting that basic predictors are among the worst ones except constM which predicts constant value equal to the previous day load median. The latter predictor belongs to the leading group of pipelines, however it is statistically worse than the leader pipeline called gb tsfresh rollingF bestValF.

Clear outcome of these simulations is that feature engineering and automatic selection results in better performance than manually prepared features (uku and kristjan features). Another observation is that cross validation with lesser overfitting (option bestVal false) seems to be overall meaningful.

Based on critical distance plot presented the single pipeline for the EE dataset can be suggested. We have a leader called gb tsfresh rollingF bestValF. Note that it requires

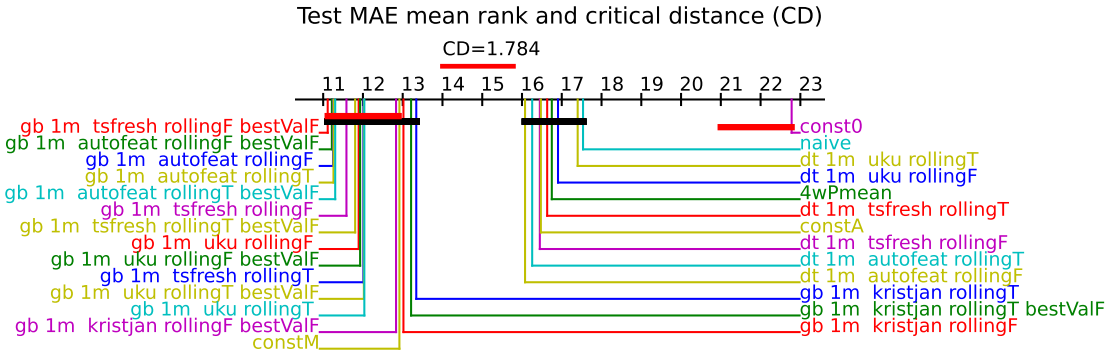


Figure 22. Test MAE critical distance plot for EE data.

gb 1m tsfresh rollingF bestValF	gb 1m uku rollingF	constM
0.182	0.207	0.196
gb 1m kristjan rollingF bestValF	constA	naive
0.203	0.223	0.211

Table 9. Averaged test MAE for selected pipelines.

some time to prepare features since engineering and selectrion is time consuming. If time issue is the main concern, the pipeline with uku features can be proposed as an alternative with faster execution time. Based on the critical distance plot gradBoost with uku features performs statistically the same as a leader pipeline.

For selected pipelines the averaged test MAE is shown in Table 9. Note that averaged test MAE may be ranked not in the order seen in Fig. 22.

It is worth to mention that EE dataset with 20 randomly selected households contains quite challenging data with consumption of the individual households varying in wide limits. Therefore one can ask not for the best pipeline for the averaged household in EE dataset but for the best predictor for the individual household. Fig. 23 addresses this question. We have found the leading pipelines in terms of 20 critical distance plots created for individual households, i.e. these are leading pipelines when test MAEs are grouped by the household. We see that our overall leader called gb 1m tsfresh rollingF bestValF appears to be the leader pipeline only for 3 households.

In Fig. 23 we also indicate in the brackets two most important features found for corresponding households. We see that very often lagged target appears to be important as well as day, hour and covariate past and future values. The discrepancy between leading features for the households indicate that the households have different consumption pattern.

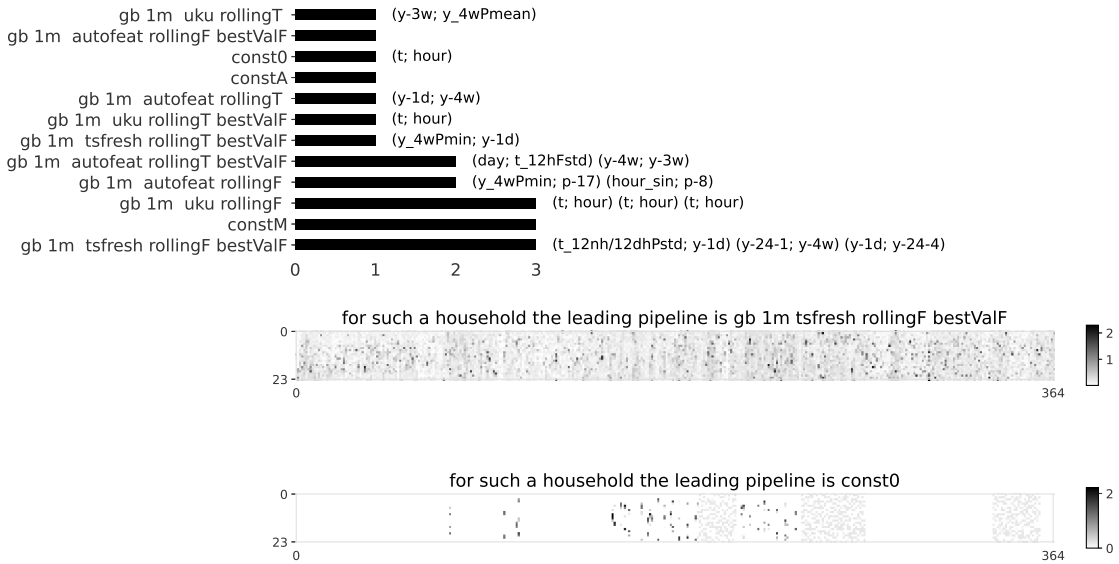


Figure 23. Top: bar plot showing how many times particular pipeline was the leading one among 20 households tested. On the left from each bar we indicate in the brackets two most important features found for corresponding household. Here y, t, p denote target variable, and temperature and price covariates, respectively. Feature engineering was discussed in Sec. 4.3. Bottom: examples of the yearly load data of the households for which leading pipeline is gb 1m tsfresh rollingF bestValF and const0.

gb 1m tsfresh rollingF bestValF	gb 1m tsfresh rollingT	constM	constA
0.128	0.129	0.136	0.155

Table 10. Averaged excess cost (in Eur) for selected pipelines. Note that for const0 the value is 0.256 Eur.

It is interesting that for some households the basic predictor, like constM or const0, becomes the leader one. Fig. 23 shows the yearly data of two households whose leader pipeline is our overall leader and const0, respectively. By looking on the latter case it is clear that there is huge data shift for the yearly load: due to the concept drift the conditional distribution of the target (load) changes in time. It is important to emphasize that our pipelines retrain on-the-fly on sufficiently short historical data. This addresses the shift to some extend. To have a better model able to predict such a data, the mechanism identifying the concept shift and singnaling for model retraining is needed.

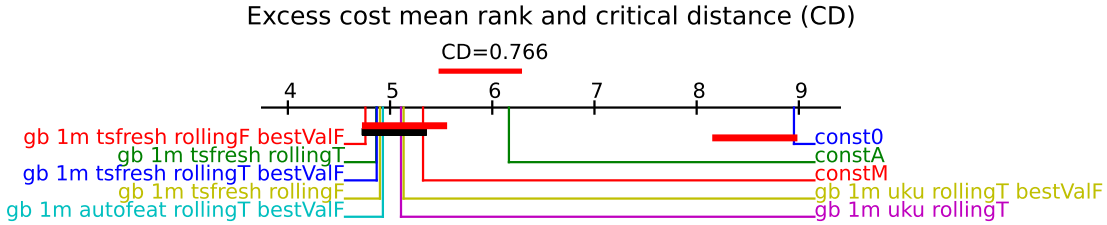


Figure 24. Excess cost critical distance plot for promising pipelines.

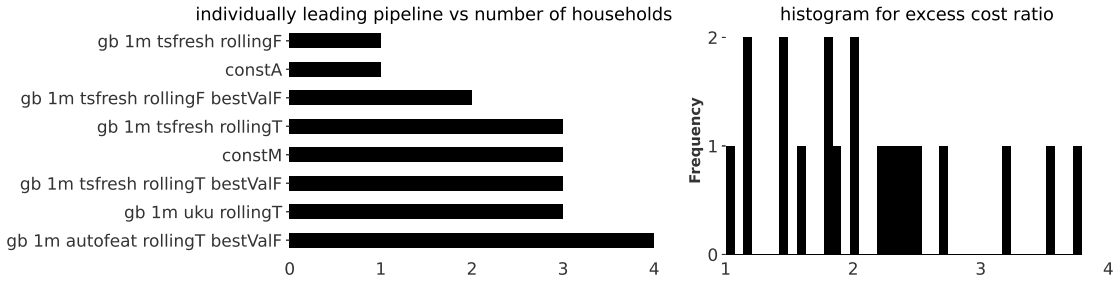


Figure 25. Left: bar plot showing how many times particular pipeline was the leading one in terms of excess cost of individual household. Right: the household-wise ratio between mean excess cost of the overall leading pipeline and mean excess cost of const0.

5.2.2 Expected cost

Based on the results for test MAE we consider the most promising pipelines and calculate excess cost for them for 20×30 load predictions. To analyse the results in terms of excess cost values we create corresponding critical distance plot, see Fig. 24. The leading pipeline found for test MAE, gb 1m tsfresh rollingF bestValF, appears to be on the leading position here as well. Other gradBoost based pipelines show statistically similar performance in terms of excess cost.

The averaged value of excess cost over all origins of prediction is shown in Table 10. We see that leading pipeline results in the excess cost twice smaller than const0 predictor used in the company.

It is important to remember that for EE data we are dealing with many different households. So it is interesting to know how often leading pipeline found in overall consideration appears to be the leading one if we consider only individual households. The bar plot 25 indicates that the overall leading pipeline was chosen as leading one only for two households, while the pipeline called dt 1m autofeat rollingT bestValF was an individual leader for four households.

It is also interesting to see household-wise what was the ratio between averaged

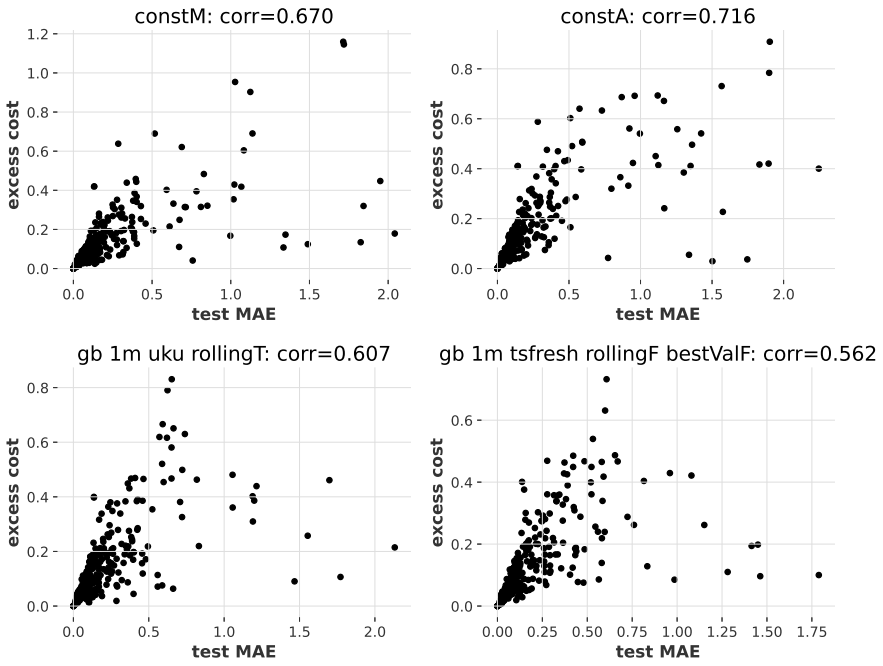


Figure 26. Pearson correlation between excess cost and test MAE for selected pipelines in the experiment with EE data.

excess cost of the leading pipeline to the mean cost of const0. Right panel in Fig. 25 shows that this ratio varies strongly and may reach values close to 4 for some household.

5.2.3 Correlation between up- and downstream metrics

It is expected that excess cost is strongly driven by the accuracy of the load test prediction. For EE data the correlation between excess cost and test MAE is found to be strong, see Fig. 26.

5.2.4 Ensemble pipeline

So far we have generated many promising pipelines. In this situation the natural extension exists: adopt created pipelines for sake of better model created as ensemble on top of selected predictors. The idea is to use predictions from individual pipelines as the features for ensemble model and train separate regressor to map the set of predictions from individual models to the target which is load ground truth.

To develop this idea we have conducted an experiment with four pipelines with autofeat feature selection: gb 1m autofeat rollingF, gb 1m autofeat rollingF bestValF, gb

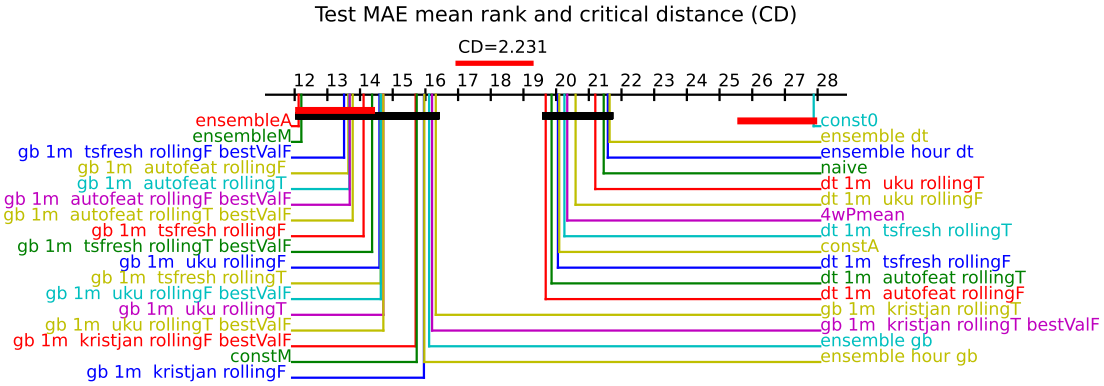


Figure 27. Test MAE critical distance plot with ensemble pipelines included.

1m autofeat rollingT, and gb 1m autofeat rollingF bestValF. These pipelines are used to emit individual predictions for ensemble model.

Ensemble model requires regressor to transform features into target. The simplest mapping is the one whose coefficients are uniform, i.e sample independent. One example here is taking average from individual model predictions. For non-uniform mapping more general regressor is needed. In the experiment we have considered DecisionTree and gradBoost as general regressors. Another component of the ensemble modelling could be related to the adding more features. In the experiment we added one-hot encoded hour as additional feature on top of individual predictions. Thus, we have the following options for ensemble pipeline:

- average and median prediction
- DecisionTree and GradBoost based prediction
- DecisionTree and GradBoost based prediction for features supplemented by hour

The critical distance plot with ensemble pipelines included is shown in Fig. 27. The results are positive: ensemble pipeline is able to beat the former overall leader pipeline, however their performance is statistically the same. This is also evident from mean test MAE values: it equals to 0.177 for both ensembleA and ensembleM which is very close to the value 0.182 reported previously for gb 1m tsfresh rollingF bestValF.

The quality of ensemble pipeline can be estimated. We know individual pipelines it consist of. Therefore the ultimate performance of the ensemble model is to predict sample-wise as good as individual model with minimal test MAE. The average test MAE of such an ultimate ensemble model can be easily found and equals to 0.164. Therefore, ensemble model has room for improvement as large as 7% smaller averaged test MAE.

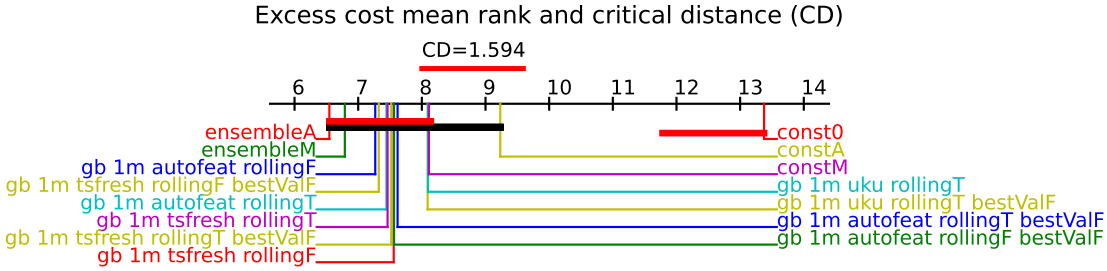


Figure 28. Excess cost critical distance plot with some ensemble models included.

Let us now discuss the excess cost of the ensemble pipeline. In Fig. 28 the corresponding critical distance plot is shown with ensembleA and ensembleM pipelines included. We see again an improvement, however not of statistical significance. For averaged excess cost, the ensembleM and ensembleA pipelines show minor improvement compared to gb tsfresh rollingF bestValF. We did not calculate the excess cost for ultimate ensemble model created with the help of the sample-wise individual pipeline of the best test MAE. However, by taking into account that there is usually strong correlation between test MAE and excess cost, the excess cost for ultimate ensemble can be estimated by finding sample-wise minimal value out of excess costs of individual predictors. Corresponding estimate points also to the possibility to reduce excess cost by 10% by making ensemble model more advanced.

By putting ultimate ensemble on critical distance plots we found also that ultimate ensemble model outperforms statistically all previously created pipelines. This opens one possible direction for future studies: improve ensemble to achieve its ultimate version.

6 Conclusion

In this study we developed an approach for the household-driven prediction of the load energy together with its evaluation in terms of expected electricity cost of the PV prosumer. The approach consists of many decision-making steps and involves statistical testing of the performances of prediction pipelines. Corresponding Python code is partly available online [17].

Particular results for the testing on many households and many origins of predictions are:

- gradBoost regressor outperforms DecisionTree regressor
- automated feature selection outperforms manual one

[17] `github.com/artjom-vargunin/load-forecast-thesis`

- ensemble modelling boosts the performance

The approach could be solidified by applying more intelligent optimization strategy like the one implemented in the package HYPEROPT [42], where optimization is allowed over awkward search spaces, which may include real-valued, discrete, and conditional dimensions.

We made an observation that major factors prohibiting better performance of the created pipelines are complicated mapping from features to target mainly related to the data complexity and data drift seen as abrupt change in the consumption habits. Possible remedies include enhancement of the ensemble modelling through feature engineering or employing more complex regression. Another perspective way is to incorporate adaptiveness mechanism for monitoring and reacting to data shifts.

References

[1] Masters G. M. *Renewable and efficient electric power systems*. John Wiley & Sons Inc, 2013.

[2] Mallon K. *Renewable Energy Policy and Politics: A Handbook for Decision-Making*. Taylor & Francis, 2016.

[3] Morales J. M., Conejo A. J., Madsen H., and Pinson P.and Zugno M. *Integrating Renewables in Electricity Markets*. Springer, 2014.

[4] Ford R., Stephenson J., and Whitaker J. Prosumer collectives: a review, 2016. Dunedin, NZ: University of Otago.

[5] M. Gržanić, T. Capuder, N. Zhang, and W. Huang. Prosumers as active market participants: A systematic review of evolution of opportunities, models and challenges. *Renewable and Sustainable Energy Reviews*, 154:111859, 2022.

[6] Gram-Hanssen K., Hansen A. R., and Mechlenborg M. Danish pv prosumers' time-shifting of energy-consuming everyday practices. *Sustainability*, 12(10), 2020.

[7] Rehman Zafar, Anzar Mahmood, Sohail Razzaq, Wamiq Ali, Usman Naeem, and Khurram Shehzad. Prosumer based energy management and sharing in smart grid. *Renewable and Sustainable Energy Reviews*, 82:1675–1684, 2018.

[8] Parag Y. and Sovacool B. Electricity market design for the prosumer era. *Nature Energy*, 1:16032, 2016.

[9] Jamal Faraji, Abbas Ketabi, Hamed Hashemi-Dezaki, Miadreza Shafie-Khah, and João PS Catalão. Optimal day-ahead self-scheduling and operation of prosumer microgrids using hybrid machine learning-based weather and load forecasting. *IEEE Access*, 8:157284–157305, 2020.

[10] Ana I. Grimaldo and Jasminko Novak. Combining machine learning with visual analytics for explainable forecasting of energy demand in prosumer scenarios. *Procedia Computer Science*, 175:525–532, 2020.

[11] David Toquica, Kodjo Agbossou, Roland Malhamé, Nilson Henao, Sousso Kelouwani, and Alben Cardenas. Adaptive machine learning for automated modeling of residential prosumer agents. *Energies*, 13(9), 2020.

[12] Rodrigo F. Berriel, André Teixeira Lopes, Alexandre Rodrigues, Flávio Miguel Varejão, and Thiago Oliveira-Santos. Monthly energy consumption forecast: A

deep learning approach. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4283–4290, 2017.

[13] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017.

[14] Ayush Sinha, Raghav Tayal, Aamod Vyas, Pankaj Pandey, and O. P. Vyas. Forecasting electricity load with hybrid scalable model based on stacked non linear residual approach. *Frontiers in Energy Research*, 9, 2021.

[15] Prediction of hourly energy consumption in buildings based on a feedback artificial neural network. *Energy and Buildings*, 37(6):595–601, 2005.

[16] Short-term electricity load forecasting of buildings in microgrids. *Energy and Buildings*, 99:50–60, 2015.

[17] Mina Razghandi, Hao Zhou, Melike Erol-Kantarci, and Damla Turgut. Short-term load forecasting for smart home appliances with sequence to sequence learning. In *ICC 2021 - IEEE International Conference on Communications*, pages 1–6, 2021.

[18] Bilal Abu-Salih, Pornpit Wongthongtham, Greg Morrison, Kevin Coutinho, Manaf Al-Okaily, and Ammar Huneiti. Short-term renewable energy consumption and generation forecasting: A case study of western australia. *Heliyon*, 8(3):e09152, 2022.

[19] Zhuang J., Chen Y., Shi X., and Wei D. Building cooling load prediction based on time series method and neural networks. *International Journal of Grid Distribution Computing*, 8:105, 2015.

[20] Norizan Mohamed, Maizah Hura Ahmad, Suhartono, and Zuhaimy Ismail. Improving short term load forecasting using double seasonal arima model. *World Applied Sciences Journal*, 15(2):223, 2011.

[21] Eisa Almeshaiei and Hassan Soltan. A methodology for electric power load forecasting. *Alexandria Engineering Journal*, 50(2):137–144, 2011.

[22] Mahmoud Ghofrani and Musaad Alolayan. Time series and renewable energy forecasting. In Nawaz Mohamudally, editor, *Time Series Analysis and Applications*, chapter 6. IntechOpen, Rijeka, 2017.

[23] Alexandra Khalyasmaa, Stanislav A. Eroshenko, Teja Piepur Chakravarthy, Venu Gopal Gasi, Sandeep Kumar Yadav Bollu, Raphaël Caire, Sai Kumar Reddy Atluri, and Suresh Karrolla. Prediction of solar power generation based on random

forest regressor model. In *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, pages 0780–0785, 2019.

[24] Da Liu and Kun Sun. Random forest solar power forecast based on classification optimization. *Energy*, 187:115940, 2019.

[25] Cyril Voyant, Gilles Notton, Soteris Kalogirou, Marie-Laure Nivet, Christophe Paoli, Fabrice Motte, and Alexis Fouilloy. Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, 105:569–582, 2017.

[26] Alexey Tsymbal. The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, The University of Dublin, Trinity College, Department of Computer Science, Dublin, Ireland, 2004.

[27] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. *ACM Computing Surveys*, 46:1–37, 2014.

[28] Indrė Žliobaitė. Learning under concept drift: an overview. *arxiv.org/abs/1010.4784*, 2010.

[29] Scott Wares, John Isaacs, and Eyad Elyan. Data stream mining: methods and challenges for handling concept drift. *SN Applied Sciences*, 1:1412, 2019.

[30] Mohammad Navid Fekri, Harsh Patel, Katarina Grolinger, and Vinay Sharma. Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network. *Applied Energy*, 282:116177, 2021.

[31] Yuanfan Ji, Guangchao Geng, and Quanyuan Jiang. Enhancing model adaptability using concept drift detection for short-term load forecast. In *2021 IEEE/IAS Industrial and Commercial Power System Asia (ICPS Asia)*, pages 464–469, 2021.

[32] Rashpinder Kaur Jagait, Mohammad Navid Fekri, Katarina Grolinger, and Syed Mir. Load forecasting under concept drift: Online ensemble learning with recurrent neural network and arima. *IEEE Access*, 9:98992–99008, 2021.

[33] Abdulgani Kahraman, Mehmed Kantardzic, Muhammet Mustafa Kahraman, and Muhammed Kotan. A data-driven multi-regime approach for predicting energy consumption. *Energies*, 14(20), 2021.

[34] F. Horn, R. Pack, and M. Rieger. The autofeat python library for automated feature engineering and selection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 111–120. Springer, 2019.

[35] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. Melbourne, Australia, 2018. `https://otexts.com/fpp3/`.

[36] Auffarth B. *Machine Learning for Time-Series with Python: Forecast, Predict, and Detect Anomalies with State-Of-the-art Machine Learning Methods*. Packt Publishing, 2021.

[37] Tim Januschowski, Yuyang Wang, Kari Torkkola, Timo Erkkilä, Hilaf Hasson, and Jan Gasthaus. Forecasting with trees. *International Journal of Forecasting*, 38(4):1473–1481, 2022.

[38] Borisov V., Leemann T., Sessler K., Haug J., Pawelczyk M., and Kasneci G. Deep neural networks and tabular data: a survey. *arXiv: 2110.01889v3*.

[39] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[40] Turner R., Eriksson D., McCourt M., Kiili J., Laaksonen E., Xu Z., and Guyon I. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *arXiv:2104.10201*, 2021.

[41] Demsar J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[42] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, 2013.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Artjom Vargunin**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Reducing electricity cost for PV prosumers by load forecast,
   *(*title of thesis)

   supervised by Kristjan Eljand and Novin Shahroudi.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Artjom Vargunin
09/05/2023