Proceedings of the 34th European Modeling & Simulation Symposium (EMSS), 042
18th International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2022 The Authors.
doi: 10.46354/i3m.2022.emss.042

Utilizing Interpretable Machine Learning to detect Dynamics in Energy Communities

Jan Zenisek[1,2*], Florian Bachinger[1], Erik Pitzer[1], Stefan Wagner[1] and Michael Affenzeller[1,2]

[1]Heuristic and Evolutionary Algorithms Laboratory, University of Applied Sciences Upper Austria, Softwarepark 11, Hagenberg, 4232, Austria
[2]Institute for Symbolic Artificial Intelligence, Johannes Kepler University, Altenberger Straße 69, Linz, 4040, Austria

*Corresponding author. Email address: jan.zenisek@fh-hagenberg.at

Abstract

With the growing use of machine learning models in many critical domains, research regarding making these models, as well as their predictions, more explainable has intensified in the last few years. In this paper, we present extensions to the machine learning based data mining technique Variable Interaction Networks (VIN), to integrate existing domain knowledge and thus, enable more meaningful analysis. Several tests on data from a case study concerned with long-term monitored photovoltaic systems, verify the feasibility of our approach to provide valuable, human-interpretable insights. In particular, we show the successful application of root-cause detection in scenarios with changing system conditions.

Keywords: Interpretable Machine Learning, Data Stream Analysis, Root-Cause Analysis, Energy Communities, Photovoltaic Systems

1. Introduction

One method to gain deeper insights regarding a trained machine learning model is to evaluate the impact of the used variables (note: in this work synonymous for "dimensions" and "features") on the final prediction, which is agnostic to the model training algorithm. One goal of such a variable importance (note: in this work synonymous for "impact") calculation may be to use this information to improve and speedup the modeling process by pre-selecting variables for future model trainings. Another one may be to discuss variable impacts with domain experts and verify the model's correctness or even to gain new insights regarding the modeled system. In this work, we focus on so-called Variable Interaction Networks (VIN, cf. Hooker (2004)), representing meta-models which are formed based on variable importance calculations for machine learning models. These VINs display a system's variables and their potential interactions as weighted, directed graph, and hence, provide a visual approach to analyze a system comprehensively, as the examples in the following section 2 show.

In section 3 the fundamentals regarding variable interaction networks are detailed and open issues highlighted. With the aim for more reasonable network structures, we present an extension to the network modeling algorithm in section 4, which enables the integration of domain expert knowledge. In section 5, we present an extension to the network evaluation algorithm, in order to gain deeper insights on streaming data dynamics. In section 6 we present a real-world case study concerned with photovoltaic energy communities, which we used to test the VIN approach with the extensions. For this purpose, first we detail a generated scenario with changing system conditions using real-world data and a simulation model. After

that, we discuss the results collected from experiments with the extended VIN approach. The final section 7 briefly summarizes our key findings and provides additional ideas for further extending VINs.

2. Related work

Variable interaction networks have been successfully applied for different purposes and in a variety of configurations. In the following we provide a brief summery of use cases found in the literature and highlight some differences regarding the network creation algorithms used.

In the work of Kronberger et al. (2011) macro-economic time series are analyzed with VINs to identify potentially interesting dependencies of certain economic indicators, such as inflation, unemployment rates or home sales. The authors use the VIN for visual data exploration as a first step to analyze a new data set and later on inspect the most interesting found dependencies by investigating the underlying detail models individually. A similar approach is chosen by Kommenda et al. (2011) for analyzing a blast furnace dataset and the resulting VIN as well as the underlying base models, which were discussed with domain experts. Both works have in common that they use symbolic regression as machine learning method for the VIN-base models, which allows them to gain even deeper insights, since symbolic regression models are interpretable, *clear-box models* by themselves (Affenzeller et al., 2014).

Winkler et al. (2013) use the VIN approach to calculate and visualize the importance of medical variables for estimating breast cancer diagnoses. Concerning VIN creation details, the authors of this paper describe several methods to calculate the variable impacts, i. e. the VIN edge weights, e. g. using the machine learning model error, the re-training error, or the variable frequency during the training process.

In addition to real-world applications, VINs have also been successfully tested on benchmark data, used to extend and improve the conventional approach. For instance Kronberger et al. (2017) present several measures to quantify the similarity of generated networks, using synthetically sampled data with different noise levels. The work of Zenisek et al. (2020) integrates the similarity measures of Kronberger et al. (2017) and presents a VIN evaluation algorithm for streaming data to detect system changes. These were manually introduced to a synthetic toy problem using two vessels, filled with liquid and connected by a slowly clogging communication path, representing the *concept drift* to-be-detected.

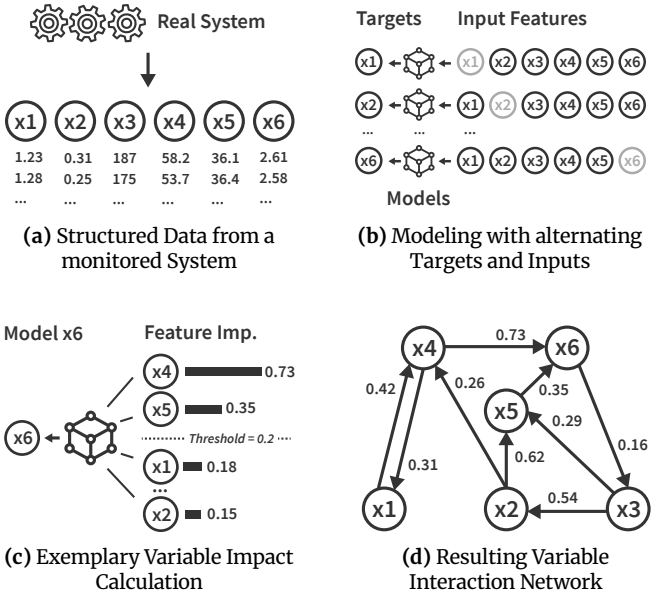
This network evaluation algorithm is discussed in greater detail in subsection 3.2 as it lays the foundation for one of the extensions presented in this work. As mentioned before, another extension concerns the network modeling algorithm, which will be described in subsection 3.1.

3. VIN Foundations and Open Issues

Conventional supervised machine learning methods aim to model a system using a set of variables (reminder: in this work synonymous for "dimensions" and "features") as training input and one specific estimation target, i. e. the model output. Instead of having one specific target variable to be estimated, Variable Interaction Networks (VIN) model a system comprehensively by displaying variable impacts: A VIN is a weighted, directed and potentially cyclic graph with variables as nodes and their impact on each other as directed, weighted edges. The algorithms to create and evaluate VINs are detailed below.

3.1. Network Modeling

As for any other machine learning approaches the starting point for modeling is given by a data set with several variables (e. g. $x1 - x6$) and respective observations (i. e. samples), which represents a real system, monitored over a certain period of time. The format of this data may be a structured data table as in Figure 1a. The first step to create the network is to train a model for each of the available variables, with the respective variable as target and using the others as input (cf. Figure 1b). After this alternating modeling process, the impact of each variable in each of the models is computed, as illustrated for the $x6$-model in Figure 1c. Exemplary methods for the calculation of variable impact are Permutation Feature Importance (PFI) proposed by Breiman (2001) and Fisher et al. (2019), the Shapley Value of Shapley (1953), the derived and more runtime-efficient SHAP method of Lundberg and Lee (2017), or the variable frequency within a genetic programming run as described by Kronberger et al. (2011). A good overview on variable impact calculation as well as alternatives to make machine learning models explainable is given by Molnar (2022). The result of this calculation is a quadratic matrix of real numbers with one column for each target and one row for each input. In a final step the network is created by adding a node for each variable and adding weighted, directed edges for each calculated impact value (cf. Figure 1d). In order to prune the resulting network, different thresholds can be applied as described in Kronberger et al. (2017): For instance, a maximum estimation error may be defined to decide which model should be taken into account when calculating variable impacts. Another useful threshold is a minimum value for the calculated variable impacts (cf. Figure 1c) which controls to create edges only with a certain weight and hence, prevents that the final result is a fully connected graph with many almost irrelevant edges. Because of the described, comprehensive modeling approach, the resulting networks most likely contain cyclic dependencies, which may impair their readability and thus, their feasibility as interpretable machine learning model. Therefore, Zenisek et al. (2020) propose an algorithm to remove cycles by iteratively deleting the weakest link within the shortest cycle until an acyclic network is created.



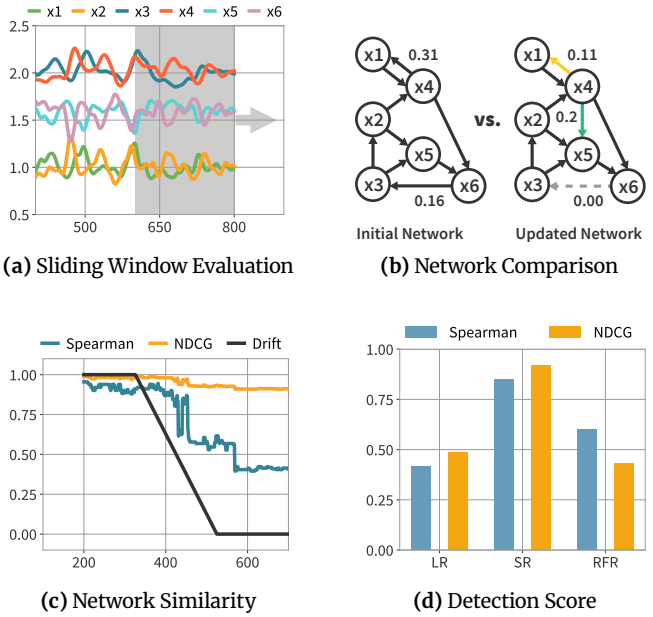
(a) Structured Data from a monitored System

(b) Modeling with alternating Targets and Inputs

(c) Exemplary Variable Impact Calculation

(d) Resulting Variable Interaction Network

Figure 1. Conventional variable interaction network modeling approach: Based on recorded system data, models are trained with arbitrary learning algorithms for each available variable using the others as input. Subsequently, the impact of each variable within each of the trained models is measured and the network is created by adding a node for each variable and weighted, directed edges for their impact others.



(a) Sliding Window Evaluation

(b) Network Comparison

(c) Network Similarity

(d) Detection Score

Figure 2. Symbolic illustration of the VIN based data stream analysis approach on data with a known, because intentionally introduced, drift. Edge design in 2b: black=unchanged, yellow=changed, green=new, dashed=vanished variable impact.

Although these extensions already support the creation algorithm to gain more concise, hence readable networks, which only display the most important connections, the VIN approach still has several flaws: For instance, it suffers from the curse of dimensionality, since the number of potential edges grows quadratically with every added variable. A high number of possible edges not only leads to complex networks, but also increases computation runtime drastically, since some of the variable impact calculation methods are quite resource demanding. Furthermore, most calculation methods are non-deterministic, which may cause numerous valid alternatives for resulting network structures. Therefore, improving the approach regarding interpretability is an open issue. One possibility for improvement is to perform feature reduction, either automatically using a feasible algorithm, or by hand based on the pre-selection of domain experts.

3.2. Network Evaluation

In the work of Winkler et al. (2015) a sliding window machine learning approach is proposed to analyze changes of predictability and variable relationships over time for a set of financial data, in order to gain insights concerning a complex real-world system's dynamics. With the same objective Zenisek et al. (2020) present an algorithm to evaluate variable interaction networks on streaming data in a sliding window fashion, which is illustrated in Figure 2. The VIN evaluation approach for analyzing streaming data is an algorithm which can be performed subsequent to the network modelling approach on which it relies. Therein, data is considered to arrive for analysis continuously, i. e. observation by observation. Either the data is in fact streamed live, e. g. from a sensor equipped system, or it may be replayed to simulate such a real-world situation, as we did in the experiment setup for this work. To diminish negative effects of seldom outliers or short-term trends, the data stream should not be processed point by point, but instead by using a sliding window of a certain size, which moves if new data points arrive (Figure 2a). With the current data partition inside the window, the previously trained machine learning models are re-evaluated and variable impacts re-calculated in order to create a new VIN. In the second step of the evaluation algorithm this updated network is compared to the original one, created on the training data (Figure 2b). Therefore, we make use of similarity measures for graphical structures proposed by Kronberger et al. (2017), in particular the Spearman rank correlation, which compares the ranks of descending ordered variable impacts, and the Normalized Discounted Cumulative Gain (NDCG), which similarly rests upon the impact order but also considers the impact magnitude, i. e. the real-valued impact. The network re-creation and comparison is performed repeatedly with each sliding window move and generates similarity trend lines as in Figure 2c. Assuming that the original network represents a stable system state, one can now easily monitor if and to what degree a system changes over time and could introduce a custom threshold at which the system should be declared "changed" or "drifted". In case the actual drift progression is known, either because it was manually introduced on benchmark data or measured post factum, the correlation

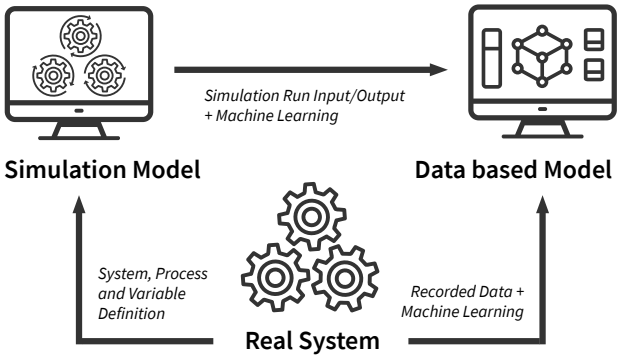


Figure 3. Conventional modeling concept to create digital models — either simulation models (cf. left-hand side) or data based machine learned models (cf. right-hand side) – of real systems. Using the data generated by simulation runs, so-called surrogate models can be created with machine learning for cases where no, or too little data is available for the direct approach.

between network similarity and drift can be calculated. In Figure 2c the known drift is represented by the black colored series *Drift*, which was inverted by $1 - x$ beforehand for the sake of better readability. The correlation value represents a score, how well the VIN evaluation approach detects system behavior – stable and drifting – over time, as shown in Figure 2d.

In the context of this algorithm an efficient impact calculation method is crucial, as VINs have to be re-created with every sliding window move. However, certain variable impact calculation methods are quite time consuming (e.g. Shapley Value) and thus, infeasible for high-dimensional networks as runtime grows quadratically with every added variable. Another open issue is that although drift detection can already be performed on threshold-basis, further analysis potential of VINs towards *drift root-cause detection* has not been investigated yet.

4. Knowledge structured Networks

This section is concerned with an extension to the network modeling process, depicted and described in Figure 1. The process follows the conventional data based modeling approach as shown and described in Figure 3: All information of a real system is used as input to create one global model, be it a computer simulation model or a mathematical model generated by machine learning. In order to deal with the open issues, described in the latter section, we propose another modeling strategy for data based models in general and specifically for variable interaction networks, as illustrated in Figure 4. Following the plain concept of divide-and-conquer, we propose to split a system and the respective data into smaller entities and model each component individually. Due to the decreased complexity of such components, it is easier and less runtime consuming for training algorithms to generate models. Thus, we propose to train multiple models for each component with different machine learning algo-

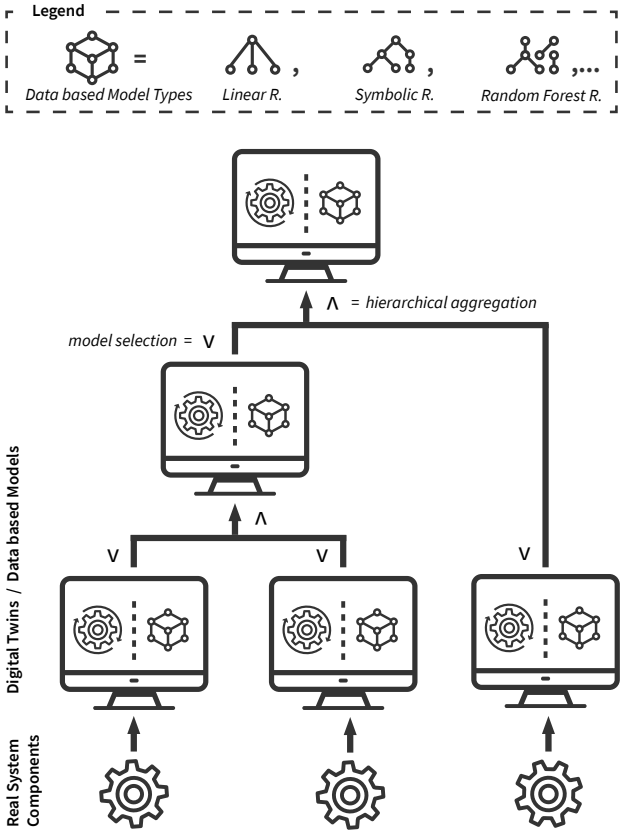
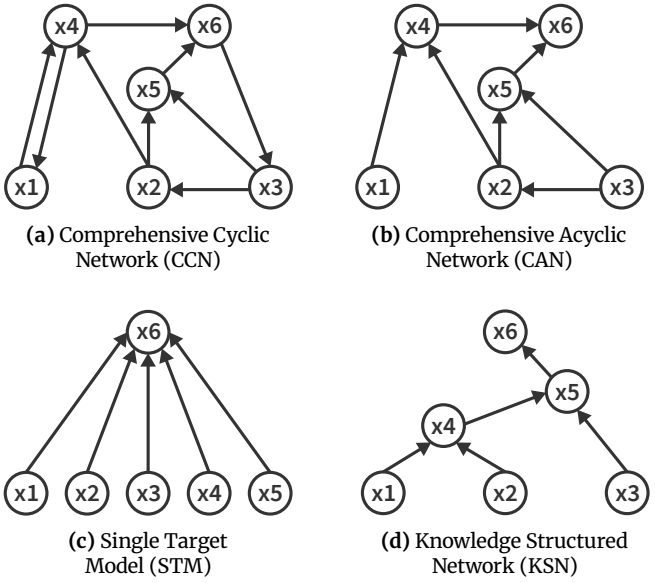


Figure 4. Concept to create (hierarchically) structured models of complex systems utilizing existing domain knowledge. For each system component a simulation model and several different data based models, using arbitrary learning algorithms (cf. figure legend), may be created in parallel. On each hierarchical layer one of them is selected for each system component and aggregated to a larger entity, forming a system component on the next layer – a "lower" component's output is a "higher" component's input, e. g. $OverallSystemRuntime(RT) = RT_{ComponentA} + RT_{ComponentB}$

rithms in parallel. Subsequently the best model for each component can be selected and aggregated by some interface to form the entire system, as shown in the hierarchically organized system in Figure 4. In terms of variable interaction networks the approach can be implemented by pre-selecting variables for each of the base models, depicted in Figure 1. This might be done hierarchically as just described or following any other suiting topology. The part of model selection is performed by creating the network from a collection of heterogeneous model types: instead of having one random forest model for each target variable, an ensemble of different model types is trained in parallel (e. g. random forest, symbolic, linear regression etc.). To form the network, for each target the best model (i. e. lowest test partition error) is selected. With this new modeling approach, we identify four different VIN types, as shown in Figure 5: A CCN represents the conventional modeling approach, although impact and model error thresholds to prune the network may already been applied. By de-cycling the CCN network, using the beforehand mentioned algorithm of Zenisek et al. (2020), the acyclic CAN is created. The STM represents a VIN with only



(a) Comprehensive Cyclic Network (CCN)

(b) Comprehensive Acyclic Network (CAN)

(c) Single Target Model (STM)

(d) Knowledge Structured Network (KSN)

Figure 5. Types of Variable Interaction Networks (VIN), depicted in a symbolic form and without weights for the sake of simplicity.

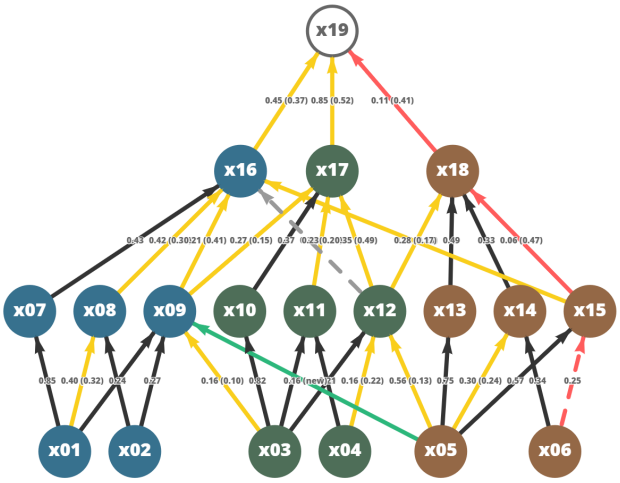


Figure 6. Visualization of a sample drift root-cause detection run on the Resinet real-world dataset, which is concerned with connected photovoltaic systems and further discussed in section 6, with highlighted *hotpath* (cf. the red colored path). The to-be-detected drift is caused by a rapidly degrading battery c-rate in photovoltaic *System C* (cf. brown nodes), which the presented algorithm on the generated hotpath correctly captures.

one variable as impact sink, i. e. with just one underlying model. The herein described new modeling approach is labeled as "Knowledge Structured Network (KSN)", since domain knowledge is necessary to perform the proposed pre-selection of variables. Ultimately, we define an additional network subtype, called "Knowledge Structured Network with Mixed Models (KSN MX)", which is a variant derived from the KSN type. As the name indicates, networks are created using a mix of different model types, as described in the *model selection* part of Figure 4.

5. Drift Root-Cause Detection

This section is concerned with an extension to the network evaluation algorithm, depicted and described in Figure 2. In addition to create more reasonable network structures, we aimed to further extend the VIN-based algorithms for data stream analysis concerning the interpretability of system dynamics so that changes get visually more conceivable to a monitoring domain expert. The following, so-called *root-cause detection* was developed to track down changes over time, and thus, enable the analysis of system drifts in greater detail:

1. Create a VIN which holds the compared edges of the original and the updated network. This *change*-VIN consists of the following edge types: original (unchanged weight), changed (changed weight), new (updated weight > threshold), removed (updated weight < threshold).
2. De-cycle the *change*-graph: identify all cycles and iteratively remove the weakest link within the shortest, still present cycle until all cycles are removed (for the complete algorithm see Zenisek et al. (2020)).
3. Generate all paths from source to target nodes, e. g. using a depth-first-search.
4. Calculate and highlight the *hotpath* – the path with the highest change sum.
5. Track down the root variable by following the reversed graph direction.

Figure 6 depicts a successful sample run of this analysis approach on the real-world *Resinet* dataset, which is further detailed in subsection 6.1. In this example, we used a knowledge structured VIN and were able to track down the specific system component which causes a change, which has been manually introduced beforehand. A more comprehensive analysis is given in section 6, where we present root-cause detection results from analyzing data streams with differently created VINs (cf. section 4) on a larger series of tests.

6. Case Study: Resilient Energy Networks

In the scope of this work we test the effectiveness of the presented methodological extensions on data originating from the project *Resinet* – Resilient Energy Networks. In the following, we detail the project's specific problem instance as well as the experiment setup, and show and discuss the collected results.

6.1. Problem Instance

The *Resinet* project is concerned with analyzing energy networks with regard to their resilience. As part of this, we developed prediction models for

- photovoltaic power production,
- power consumption,
- the State Of Charge (SOC) of batteries,
- grid input/output power

based on data from 188 households from the region of Upper Austria, all equipped with roof-top mounted photovoltaic modules and battery packs. The measurements contain data from 2015 - 2019 and come with several system configuration information, e. g. peak power production, battery capacity. The combined peak power output of the photovoltaic modules per system ranges from 3 kW to 30 kW, the capacity of the installed batteries ranges from 3.5 kWh to 9.6 kWH. As a first step we linked the data with geographic and temporal information (e. g. altitude, twilight times) retrieved from various web services. Most importantly we linked the time series with weather data, such as *global radiation*, *air temperature* and *precipitation sum*, from the Austrian weather forecasting system INCA (Haiden et al., 2009). This enriched data set sums up to roughly a 100 million rows, with 75 partly measured, partly computed variables (cf. feature engineering). Since the system measurements were recorded with a 5-minute frequency, but the INCA weather data is only available on an hourly basis, we re-sampled the time series by taking the hourly mean for each available variable.

After these major and other data pre-processing steps, we were able to develop prediction models with sufficient performance for the listed system components. To investigate network resilience – the ultimate goal of the Resinet project – the analysis of separate models is infeasible, however. The extended VIN approach, on the other side enables comprehensive system analysis. In this particular work, we focus on analyzing system dynamics in terms of a root-cause detection as a detail research question: We aim to identify the reasons which lead to changing or unstable system behavior and thus, the need for resilience. Therefore, we designed following what-if scenario:

> *What if... a small community of 3 systems is sharing its batteries by charging them together for higher network-independence? Can we detect a failing battery pack in such a scenario?* (cf. illustrated in Figure 7).

For this purpose, we used the measured real-world system data, but re-calculated battery states and grid input/output differences, to simulate that the systems are connected and sharing their surplus energy produced (Figure 7). For instance, if the consumption of a system can be covered with the current energy production and the system's battery is already filled, surplus energy is shared equally amongst the other community members and only after that, left over energy is passed to the public grid. This way, a small virtual energy community (Faria et al., 2019) is simulated, which should provide more overall grid independence than unconnected individuals can reach. For each of the experiment runs, we introduced rapid degradation of the charge/discharge function (c-rate) of a random battery at a random point of time, using the data stream generation tool from Zenisek et al. (2018). By this means, we aim to simulate a probable, but not directly observable maintenance problem, which could impede the gained network independence, but potentially remains undetected

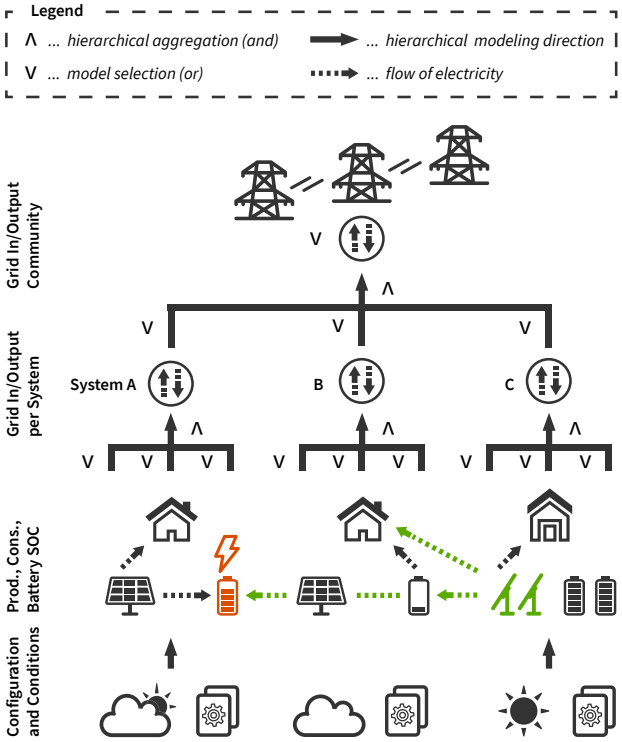


Figure 7. Concept for (hierarchically) structured models (see: Figure 4) applied on the Resinet case study: an energy community of three photovoltaic and battery equipped households (Systems A, B, C), connected to share surplus energy. In this example system A and system C are able to produce energy due to good weather conditions, while system B is not. The photovoltaic power system C covers its own current power consumption, has already filled up its batteries, and hence, shares its surplus energy with its community members A and B. Due to bad weather conditions at system B, no power can be produced, however the shared energy of system C can be used to cover parts of the current consumption and to charge the battery. At system A a battery charge malfunction is indicated, which causes that its own produced, as well as the shared energy cannot be used in full. This however, remains undetected, since we do not assume having hardware to identify battery errors of this kind in our simulation.

due to the compensatory behavior of the interconnected energy community. This concept of using real-world data and enriching it with simulated data is depicted and summarized in Figure 8. Due to novelty of the *energy community*-idea, real-world data on household level is currently not available, which motivated our simulation based approach.

6.2. Experiment Setup

The following items denote the most important parameters of the performed experiments.

Training Algorithms and Models: We developed Linear Regression models (LR, Draper and Smith (1966)) as well as Symbolic Regression models using genetic programming (SR, Koza (1994)) and Random Forest Regression (RFR, Breiman (2001)) models as base for our VIN approach. For the RFR and SR machine learning algorithms we used a configuration similar to Zenisek et al. (2020) to get feasi-

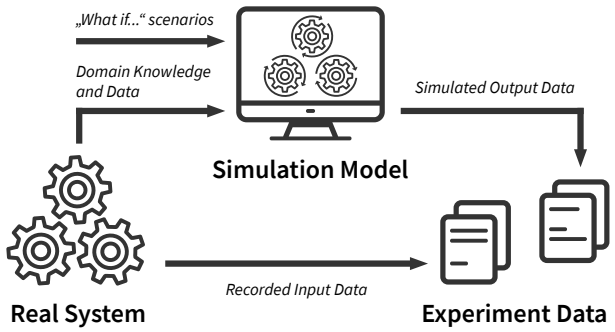


Figure 8. Workflow to generate experimental data: A lean simulation model was created, utilizing the existing domain knowledge regarding energy production and consumption dependencies. Further on, parameters to formulate the described *what-if*-scenario were integrated and the model adapted accordingly. The final experiment data sets consist of original, i. e. real-world input data and the output data, produced by the adapted simulation model, which processes input data and scenario parameters.

ble results, although ML fine tuning is not in the focus of this work: RFR models = trees: 50; M: 0.5; R: 0.3; SR models = 10 ensembled best solution candidates from individual runs; max. tree size: 35, terminals: constants, variables; non-terminals: $+, -, *$, *analyticquotient*, *log*, *exp*, *sin*, *htan*

VIN Creation Thresholds: With reference to the modeling results, we defined no maximum Normalized Mean Squared Error (NMSE) so that any model can take part in the network creation algorithm. One reason therefore was to get a fair comparison between all model types, since VIN structuring is already part of most of the used modeling strategies. An additional threshold could obfuscate the comparison. Moreover, a general maximum NMSE could not have been set, since the results for the *power consumption* models could not be tuned further than a quite high NMSE of 0.791 (cf. Table 1). For the same reasons no minimum variable impact was set. However, the number of variables was already limited to a reasonable set beforehand by domain experts.

Training Data: All models have been trained on real-world and partly simulated time series data, as described in the latter subsection 6.1. For this purpose we use data from 3 random households (denoted as system A, B, C) and the monitoring period 2016-2017. Aggregated results from the test partitions of 10-fold cross validation experiments are given and described in Table 1. More details regarding experiment setup and results of these prediction models will be presented in future work. For the purpose of this work, the herein shown results are already sufficiently good to work as a VIN model base.

Network Types: We trained and evaluated all of the identified VIN types, described in section 4: Comprehensive Cyclic Network (CCN), Comprehensive Acyclic Network (CAN), Single Target Model (STM), Knowledge Structured Network (KSN) and Knowledge Structured Network with Mixed Models (KSN MX). For the latter two types the necessary variable pre-selection was performed based on the hierarchical structure illustrated in Figure 7, which represents existing domain knowledge. Due to the simulated system connections, the pre-selection follows the horizontal layers, however does not separate the systems vertically, since certain variables may influence more than just the system to which they originally belonged. For instance, the *Grid Diff*-model of system C has its own *Battery SOC*, but also the *Battery SOC* of system B in its input vector – see example KSN network illustrated in Figure 6. Thus, all variables of the previous layer are available for any system model in the next one. By this means, energy sharing scenarios can potentially be modelled correctly.

Evaluation Data: The data used for the implemented drift-detection and root-cause detection experiments originates from the same source, as the training data, but from the year of 2018. Based on this, we randomly sampled 24 time series, each with a length of 2160 consecutive events, i. e. hourly aggregated recordings (3 months * 30 days * 24 hours) and simulated 8 battery outages per system on it: We introduced 8 different maximum drift levels per system from 30% to 100% c-rate degradation with a 10% step in between. The drifts start at a random point after a minimum of 250 events and take 250 events (ca. 10 days) to reach their target level. For our tests data has not been shuffled, to be able to monitor the degradation as it would occur in reality.

Evaluation: We used a sliding window size of 100 (ca. 4 days) and a step width of 1 event (1 hour) for all experiment runs, as we gained promising results from prototypical tests with this setting. Within each sliding move, the originally built and the updated networks are compared using the Spearman rank and the Normalized Discounted Cumulative Gain (NDCG).

6.3. Results and Discussion

In Table 1 the Normalized Mean Squared Error (NMSE) for several machine learning models, which serve as a basis for the variable interaction network creation algorithm is given. One can easily observe that except for the *power consumption* target, the models come with quite low estimation errors. We could quickly determine that the reason for the unpredictability of power consumption is due to the lack of data – except for time information (assumption: more power is consumed on weekends) and weather conditions (assumption: e. g. low air temperature causes higher energy consumption of heat pumps), there is simply too less indication in the data to estimate the power consumption of the monitored households for a specific point in time. Overall, for the most system components the Random Forest Regression (RFR) models perform best, closely followed by the Symbolic Regression (SR) models

and with some distance followed by the Linear Regression (LR) models. As explained in the latter section, no maximum error threshold was set for the subsequent VIN creation algorithm. We decided to do so, due to high errors of power consumption models and for the sake of comparability of the already differently structured VIN types. However, the presented NMSE values are used for the Knowledge Structured Network with Mixed Models (KSN MX) to select the best available models per system component automatically. By this means, the KSN MX, which represents the described *Resinet* problem instance (cf. Figure 6), consists of random forest regression models on the lower hierarchical layer (pv production, power consumption, battery SOC) and of symbolic regression models on the upper layers (grid diff, grid diff global).

Table 1. Machine Learning (ML) training results in terms of a Normalized Mean Squared Error **NMSE**-score for several example targets using different ML algorithms. With exception of *Grid Diff Global* the displayed results represent the mean values for the three connected systems.

Modeling Target	LR	SR	RFR
PV Production	0.076	0.060	0.056
Power Consumption	0.931	0.855	0.791
Battery SOC	0.193	0.184	0.114
Grid Diff	0.176	0.104	0.110
Grid Diff Global	0.000	0.000	0.012

The boxplots in Figure 9 and Figure 10 show how well different VIN types with different underlying machine learning models perform on drift detection, which was tested on a set of 24 time series with random, simulated battery malfunctions. For details regarding the experiment setup, the reader is reminded to see subsection 6.2. Both figures provide plots of drift detection scores, given by the correlation R from the interval $[-1, 1]$ between the known, introduced drift on one hand side and the similarity scores from comparing initial with updated VINs on the other hand. In Figure 9 the similarity score is given by the Spearman's rank correlation and in Figure 10 the Normalized Discounted Cumulative Gain (NDCG) is shown. In both cases, we observe that VIN types which utilize algorithms and domain knowledge to tune the network structure, perform better on drift detection than those without modifications. With reference to the VIN modeling extension presented in this work (cf. section 4), this represents our major finding and endorses our approach. The best model by far is the knowledge structured and random forest based network (KSN RFR), with a mean score $R > 0.8$ and quite low variance over all runs. Interestingly, both figures also display the network type KSN MX, which is built with the presumably most sophisticated algorithm, on the second place, with a quite broad spectrum of detection results. By selecting the best estimation model for each system component, the resulting heterogeneous model mix is unintentionally compensating system drifts to some extent. We assume that the inconsistent estimation behavior of the different machine learning models causes the lower detection score and most importantly the large result range. In fact, the KSN MX does not model system behavior representing a stable state, as good as the homogeneous KSN RFR. While overall, more sophisticated models perform better, Figure 10 also shows some promising results for the CCN network type (CCN LR and CCN SR), which does not use any pre-selection of variables. However, they also come with a quite large estimation range. Furthermore, the Spearman based drift detection provides no clear loser – with approximately $-0.3 > R < 0.35$ the range of means for CAN, CCN and STM is far from accurate detection. With the simple STM network type and a $-0.3 > R < 0.0$ range, the NDCG based detection has a clear loser, which in total has a negative, i. e. counter-intuitive, detection record. Regarding machine learning model differences, in Figure 9 with the Spearman similarity measure we observe that random forest and symbolic regression perform on a similar level, while linear regression based networks lag behind. In Figure 10 with the NDCG similarity measure, however, this trend can not be confirmed.

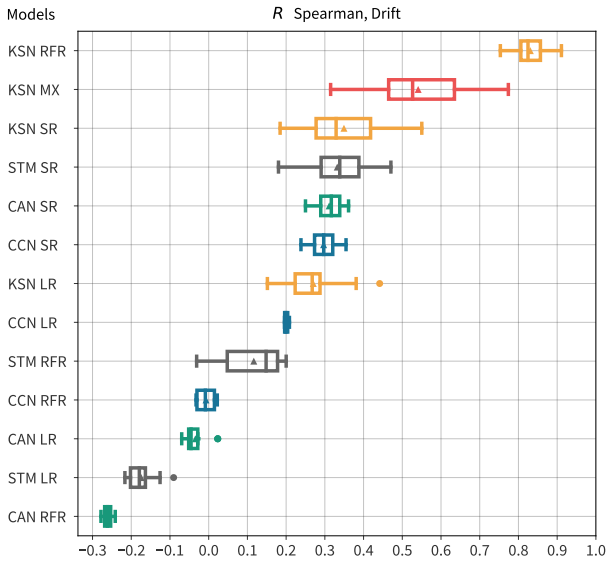


Figure 9. Drift detection results of different VIN models on the described, modified Resinet data sets. The shown *Pearson's R* value is calculated between the **Spearman** based similarity series of initial and re-created VINs and the known drift. Thus, high R-values represent well performing drift detectors. The different VINs tested are ranked by the mean R-value, which is illustrated as triangle within the boxplot-box. Different colors have been applied for each model type, i. e. *STM*, *CCN*, *CAN*, *KSN* and its subtype *KSN MX*. The warmer the color, the more algorithmic and/or domain expert intelligence was applied during modeling, e. g. *KSN MX* uses a knowledge based pre-defined structure and then automatically selects the best available machine learning model, i. e. *LR*, *SR* or *RFR*, per hierarchical layer and component.

In Figure 11 the results for drift root-cause detection are plotted as percentage of correctly identified root systems of each tested VIN type. In the generated 24 test runs,

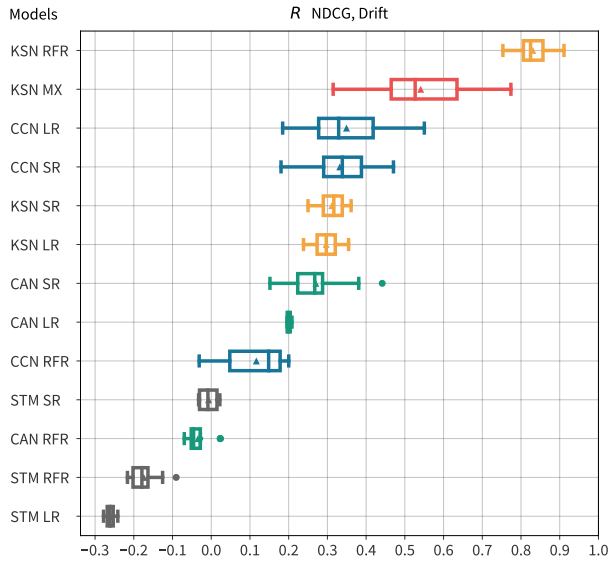


Figure 10. Drift detection results of different VIN models on the described, modified Resinet data sets. The shown *Pearson's R* value is calculated between the **NDCG** based similarity series of initial and re-created VINs and the known drift.

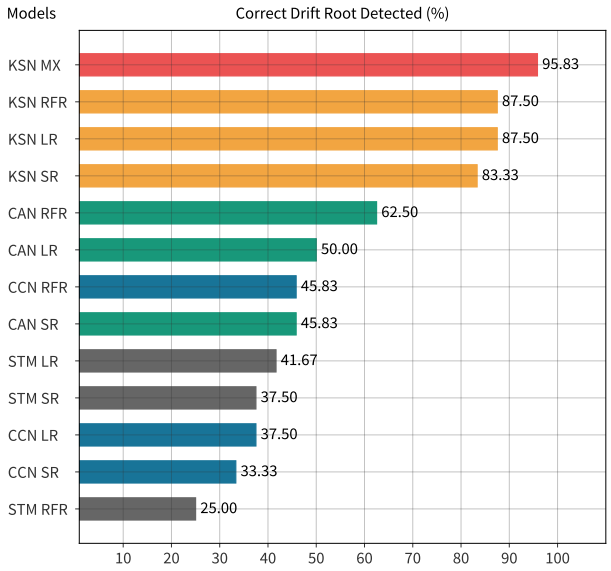


Figure 11. Drift root-cause detection results as percentage of correctly identified roots per tested VIN type.

each of the three systems (A, B, C) is causing a drift equally often (i. e. 8 times), meaning there is 33.33% chance for correct detection by randomly guessing the root. Hence, this score should be topped to prove that the selected VIN type and the drift root-cause detection algorithm, presented in section 5, work as intended. As in the case of drift detection, the scores of more sophisticated VINs are superior to the simpler networks. However, regarding root-cause detection, the outcome is even clearer. While simpler types reach a maximum percentage of 62.5%, all knowledge structured networks detect at least 83.33% roots correctly. In this case the KSN MX achieves the best score with a percentage of 95.83% which represents a single miss in 24 detection runs with differently shaped system drifts. On the bottom of the scores, one can observe that some VINs only perform equal or even worse than random guessing: CCN SR = 33.33%, STM RFR = 25.00%. In terms of the underlying machine learning models, no clear winner can be determined, which indicates that our approach is agnostic regarding the used modeling algorithms.

7. Conclusion and Outlook

In this work, we described two extensions to the Variable Interaction Network (VIN) data mining technique: One concerns the VIN modeling algorithm and proposes to preselect variables for the underlying machine learning models using domain knowledge, develop multiple models in parallel and dynamically select the best model found. The other extension concerns the VIN evaluation algorithm and represents a root-cause detection algorithm to gain deeper insights from VINs, which are evaluated over time on changing system behavior. Results from a case study dealing with dynamics in a small photovoltaic powered energy community showed the potential of the presented VIN extensions: By using the detailed root-cause detection algorithm on domain knowledge structured networks – i. e. by combining both extensions, it was possible to identify hidden system drifts – representing increasing battery malfunctioning inside the energy community – in most (ca. 95%) of the simulated cases.

One obvious extension to this work is the conduction of an even more comprehensive series of experiments – e. g. using different modeling algorithms, evaluation parameters and larger data sets. However, apart from additional tests and experiments, we want to highlight several ideas for further extending the VIN method itself:

Future work may also consider alternative, more runtime-consuming variable impact calculation methods, such as the Shapley Value (Shapley, 1953), SHAP (Lundberg and Lee, 2017), or Feature Interaction using the H-statistic of Friedman and Popescu (2008). With the proposed pre-selection of variables more complex methods may be reasonable to use, as the curse of dimensionality can be kept in check.

Inspired by the presented knowledge structured VINs, another idea to improve the networks' structural correctness – in terms of a theoretical perfect fit to the real-world system – is to use domain knowledge to prune VINs after the creation algorithm. Hence, the network creation could be done as the conventional approach proposes, followed by a removal of unreasonable edges. However, one limiting factor of this approach may be that the underlying models won't be fully reliable anymore, since they are trained with the later de-selected variables connections.

Concerning root-cause analysis, the presented hotpath-based approach might be enhanced by adding some sort of search-stopping criterion. By this means, not only the

correct, drift-causing branch of a VIN could be identified, but also the effectively drift-causing node – which might be different from the root node i. e. some node "earlier" within the directed graph.

Another analysis approach we envisaged for root-cause analysis on VINs, is to transform the problem into a classification task. For this purpose, data for differently shaped drifting system states must be collected and labeled beforehand. By training these states in an offline phase, as we already did with stable system behavior, we could compare a system's current behavior online to the pre-trained VINs and identify the respective state based on the highest similarity measured. However, this approach demands the acquisition of data with labeled system states, which can be a major obstacle from our experience with real-world use cases. In comparison, the root-cause analysis approach we presented in this work, is agnostic to system states – with the exception of a labeled "stable" state – which marks one decisive advantage.

Ultimately, we want to highlight that with the approach to de-compile a complex system into smaller, connected components using existing domain knowledge, a heterogeneous mix of detail models could be developed, which was superior to a single system-wide model. As an extension to this modeling concept, one interesting direction to research might be to investigate the concept's potential for simulation-based optimization scenarios and the field of online learning with constantly evolving detail models.

8. Acknowledgements

The presented work was done within the project "Secure Prescriptive Analytics", which is funded by the state of Upper Austria as part of the program "#upperVISION2030".

References

Affenzeller, M., Winkler, S. M., Kronberger, G., Kommenda, M., Burlacu, B., and Wagner, S. (2014). Gaining deeper insights in symbolic regression. In *Genetic Programming Theory and Practice XI*, pages 175–190. Springer.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Draper, N. and Smith, H. (1966). Applied linear regression.

Faria, P., Barreto, R., and Vale, Z. (2019). Demand response in energy communities considering the share of photovoltaic generation from public buildings. In *2019 International Conference on Smart Energy Systems and Technologies (SEST)*, pages 1–6.

Fisher, A., Rudin, C., and Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *J. Mach. Learn. Res.*, 20(177):1–81.

Friedman, J. H. and Popescu, B. E. (2008). Predictive learning via rule ensembles. *The annals of applied statistics*, pages 916–954.

Haiden, T., Kann, A., Pistotnik, G., Stadlbacher, K., and Wittmann, C. (2009). Integrated nowcasting through comprehensive analysis (inca)—system description. *ZAMG Rep*, 61.

Hooker, G. (2004). Discovering additive structure in black box functions. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 575–580.

Kommenda, M., Kronberger, G., Feilmayr, C., and Affenzeller, M. (2011). Data mining using unguided symbolic regression on a blast furnace dataset. In *European Conference on the Applications of Evolutionary Computation*, pages 274–283. Springer.

Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112.

Kronberger, G., Burlacu, B., Kommenda, M., Winkler, S., and Affenzeller, M. (2017). Measures for the evaluation and comparison of graphical model structures. *Lecture Notes in Computer Science*, 10671:283–290.

Kronberger, G., Fink, S., Kommenda, M., and Affenzeller, M. (2011). Macro-economic time series modeling and interaction networks. In *European Conference on the Applications of Evolutionary Computation*, pages 101–110. Springer.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Molnar, C. (2022). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Accessed: May 19, 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book/.

Shapley, L. S. (1953). A value for n-person games. *In: Kuhn, H. and Tucker, A., Eds., Contributions to the Theory of Games II*, 2.28:307–317.

Winkler, S. M., Kronberger, G., Affenzeller, M., and Stekel, H. (2013). Variable interaction networks in medical data. *International Journal of Privacy and Health Information Management (IJPHIM)*, 1(2):1–16.

Winkler, S. M., Kronberger, G., Kommenda, M., Fink, S., and Affenzeller, M. (2015). Dynamics of predictability and variable influences identified in financial data using sliding window machine learning. In *International Conference on Computer Aided Systems Theory*, pages 326–333. Springer.

Zenisek, J., Kronberger, G., Wolfartsberger, J., Wild, N., and Affenzeller, M. (2020). Concept drift detection with variable interaction networks. *Lecture Notes in Computer Science*, 12013:296–303.

Zenisek, J., Wolfartsberger, J., Sievi, C., and Affenzeller, M. (2018). Streaming synthetic time series for simulated condition monitoring. *IFAC-PapersOnLine*, 51(11):643–648.