Virtual Sensor Creation to Replace Faulty Sensors Using Automated Machine Learning Techniques

Eushay Bin Ilyas, Marten Fischer, Thorben Iggena, Ralf Tönjes
University of Applied Science Osnabrück
Faculty of *Engineering and Computer Science*
Osnabrück, Germany
{e.bin-ilyas,m.fischer,t.iggena,r.toenjes}@hs-osnabrueck.de

Abstract—**With the introduction of the Internet of Things (IoT) into every area of life, more and more applications are created that rely on information collected from IoT sensors. These applications are developed by third-party developers and depend on a continuous information flow but do not operate the sensor network themselves. However, in case of sensor failures, the flow of information will be interrupted which will pose a problem for the user/application as well as the data provider who might not be able to replace the sensor in time. Depending on the requirements of the application, estimators which are trained through machine learning algorithms, called Virtual Sensors, can ensure the uninterrupted operation of the application. However, for a solution that can be used in practice, the deployment of such a Virtual Sensor needs to be fast enough as well as fully automated, so that no human interaction and domain knowledge is required. This paper describes a framework that is capable of finding correlating IoT sensors in the surrounding environment, selecting a machine learning algorithm, and training a fitting model to run a Virtual Sensor.**

Keywords—IoT, model search, machine learning, virtual sensing, AutoML, faulty sensors

I. Introduction

Over the past years, we have moved closer towards automating everything around us, a process that will be continued for foreseeable future. Caused by the advancements in the development of the Internet of Things (IoT), the world is getting more digitalized day by day. The impact of this progress can be seen in our everyday lives through Smart Cities and Smart Home applications as well as the huge role IoT plays in Industry 4.0. To fully utilize the power of IoT, often many sensors are needed to be deployed in a certain area/environment to give as concrete information about the area under observation as possible. By the estimates of IHS Markit [1], there will be 73 billion IoT devices by 2025. With the increasing number of IoT sensors, there is a need for platforms, which can search these sensors and provide their data to third-party users and applications.

In different IoT domains, especially in the Smart City domain, IoT devices are often deployed in harsh environments and need to operate independently for many years. To extend the lifetime of these devices, a popular approach is to run the devices based on a solar-powered energy source. Due to changing weather conditions, aging batteries, hardware wear tear or other factors, the operation of such IoT devices may be interrupted from time to time. Also, it cannot be assumed, that an IoT system is installed by experts with appropriate redundancies in each domain. Applications that rely on data from these IoT sensors would possibly stop working and the user's experience would suffer.

In this paper, we present a novel approach to create an estimator, named Virtual Sensor, which can be utilized to replace the faulty sensor without the need for human intervention. This Virtual Sensor can be trained through machine learning techniques by using the historical data of the faulty sensor and the neighboring sensors that are highly correlated. The neighboring sensors include all the sensors, which have some relationship to the faulty sensor regardless of the quantity they are measuring. The "relationship" can be defined by the use of ontologies such as the Suggested Upper Merged Ontology (SUMO) [2]. For example, in the case of an outdoor temperature sensor, the potential neighboring sensors can be sensors in the same city or even in a couple of surrounding cities. On the other hand, for a temperature sensor on an industrial floor, the neighboring sensors needed for estimation will be the sensors deployed on the same floor.

One of the important aspects of this research is to create a solution, which can create Virtual Sensors in an automated way, i.e. without the need of a human to analyze the data of the sensors or to create a model describing the environment. Especially if a platform manages a very large number of IoT devices, a manual analysis of the data is infeasible. In such a platform, it will not be possible to visually analyze the data of each sensor to compute similarities between data streams or extract patterns to fine-tune the machine learning algorithms, because of either the unavailability of a human supervisor or the huge number of sensors in an environment. This highlights the need to have an automated approach that selects the data sources, preprocesses the data and selects a suitable machine learning algorithm automatically. While deploying a machine learning model, the constraints of storage space and training time for machine learning algorithms also have to be faced. Less training time is highly desirable to avoid delays in the information flow, which will otherwise reduce the user experience. To deal with limited storage, it is also desirable to deploy the models with small size.

This paper consists of six sections. The related work for creating Virtual Sensors is discussed in section II. In section III, the approach to create Virtual Sensors within the framework is explained. Section IV presents example scenarios in which a Virtual Sensor is created while section IV shows the results, which compare the predicted output of a Virtual Sensor with that of a real sensor. Section V discusses the results presented in section IV. Section VI concludes the presented research.

II. Related Work

A. *Virtual Sensors*

The concept of Virtual Sensors, also known as soft sensors [3], has been researched in different contexts. In one concept, Virtual Sensors are used as an abstraction layer from real hardware. The FP7 project IoT-A for example used "Virtual Entities" to mediate between applications and communication interfaces [4]. In the popular smart home platform OpenHAB, so-called Bindings are used as a virtual representation of a device [5]. The authors of [6] focus on scenarios where applications extract measurements from diverse sensor networks. They introduce the virtual sensors abstraction that enables an application developer to programmatically specify an application's high-level data requirements. The OpenIoT project [7] relies on the W3C Semantic Sensor Networks (SSN) ontology to represent physical and virtual sensors.

In another concept, Virtual Sensors are used to fuse information from different data sources to deduce new knowledge. For this, a model needs to be developed that explains the relation between the data sources. According to [8] three methods are available: a white box model (e.g. a physical relation defined as mathematical equation) if all relations are known; a black box model (e.g. neural networks) if the self-learned model is based purely on the data and not on an analysis; and a grey model if correlations between different data sources are assumed. For example, in [9] the authors describe a semi-supervised JITL (just-in-time learning) framework for non-linear processes as a grey model. However, they use semi-supervised learning where they compare a labeled dataset against an unlabeled dataset to predict future outputs. The Global Sensor Network (GSN) platform [10] uses a combination of both concepts, where Virtual Sensors are used to abstract implementation details as well as to derive new information.

In this paper, we propose to deploy Virtual Sensors to compensate the failure of a physical sensor. In contrast to the abovementioned approaches, correlations between data sources are learned from historical data and models are trained in a fully automated way, independent of their domain or environment.

B. ML Model Selection

Different approaches exist for automated machine learning (AutoML) including hyperparameter optimization [11] of various algorithms [12]. In [13] the authors have presented a survey of the AutoML pipeline. Various AutoML frameworks provide different algorithms and approaches to automating the model creation process. Balaji et al. [14] provides a benchmarking of some of the popular AutoML frameworks while [15] proposes hyper-hyperparameter optimization strategies for such frameworks. In this paper, we have used Auto-sklearn which uses ensemble learning [31]. Ensemble learning combines different models into one single model to overcome the shortcomings of a poor performing model [16].

Another popular technique for model selection or hyperparameter optimization is grid search. Bengio [17] argues that random search is much more efficient than grid search because of the increased dimensions when searching through all the hyperparameters. To deal with this issue, only a selected set of hyperparameters has been used for the grid search which the authors have deemed more useful than others. Additionally, the training time of the model can be influenced by increased or decreased tuning options, like in TuPAQ [18]. To further ensure a better-quality model, a Correlation-based Feature Selection (CFS) [19] approach has been used, where "features" correspond to sensors in this case. Furthermore, a k-fold cross-validation strategy has also been utilized for best model selection [20].

In [21], an automated framework is presented for IoT data-dependent applications, but this research is directed towards data analytic beginners and follows an analyst-in-the-loop approach.

III. Virtual Sensor Creation Framework

The approach presented in this paper provides a method to replace faulty sensors in an automated way. To indicate faulty sensors, a fault/anomaly detection system is assumed to be present in an IoT data platform. The framework will then locate the faulty sensor and search for nearby sensors that are related to it on a higher entity level indicated by their metadata. The raw data is then preprocessed and correlations among the faulty sensor and nearby sensors data are found, see section B. Machine learning techniques, described in section C, are then applied to the created dataset where the historical data from highly correlated sensors will be the predictors and the historical data from the faulty sensor will be used as the response variable. The model created through these techniques will then be deployed as a Virtual Sensor in place of the faulty sensor until it is replaced. Fig. 1 shows an overview of the Virtual Sensor framework.

In the sections below, the necessary elements are explained which enable the framework to create Virtual Sensors.

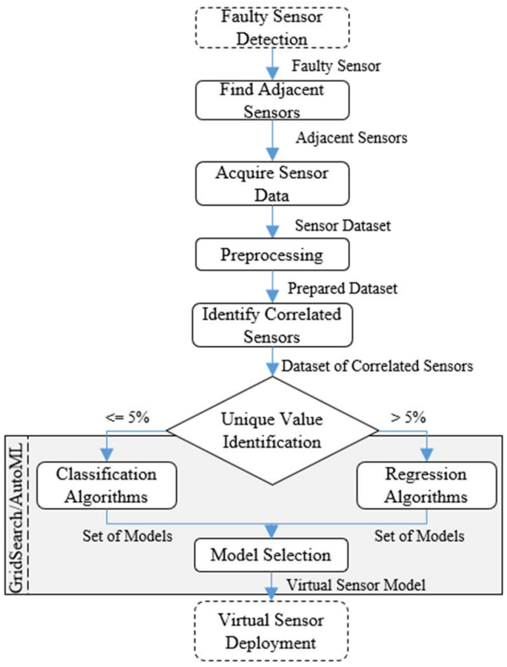

Fig. 1 Virtual Sensor Framework

A. Metadata Enrichment

In order to be able to automatically select suitable data sources, they need to be annotated with metadata that describes the respective sensor. The following attributes have to be defined in the metadata of each sensor.

- Sensor Type (e.g. temperature, electric power, etc.)
- Sensor Location (latitude/longitude/altitude)
- Relationship to a higher entity (e.g. indoor/outdoor, temperature sensor in city x, electrical power output from a refrigerator in a home, etc.)
- Data Type (e.g. numerical, alphanumerical, enumeration, etc.)
- Update Interval (the frequency of data logging)
- Null indication (e.g. NA, None, etc. for missing value)

These metadata attributes are necessary for the automated framework. The sensor type and sensor location allow an IoT platform to search for nearby sensors, which are related to the faulty sensor. The Sensor Location is used to find other sensors that are deployed in the near environment. Sensor Type can also shorten the search in the presence of a large number of IoT devices at the sensor location by searching for sensors of the same type. In the case of poor search results, sensors with different sensor types can be chosen.

The Relationship to a higher entity can further improve the search for correlated sensors. This relationship is a description of the environment in which the sensor is placed. While searching for nearby sensors only sensors with this relationship should be found. For example, the temperature sensors placed in different areas in different cities are related to each other. Similarly, sensors placed inside an office building may be related to each other.

The Data Type of the sensor facilitates the framework in the preprocessing of the historical data, as it indicates the presence of non-numerical data, which should be transformed into numerical values. The Update Interval is the fixed time after which a sensor reports new data into the system. It enables the framework to align the data of different sensors, which are logging data with a different frequency.

Null Indication is the indication of instances where the sensor didn't report any measurement and denotes how this is indicated. This information is used to filter historical data in the preprocessing.

B. Data Preprocessing

Raw historical data of the selected sensors have to undergo certain preprocessing steps before any other operation can be performed on it [22]. The most important step in preprocessing is the time alignment of data from different sensors. It is necessary to align the data from different sensors, which record values at different time instances and maybe even with different update intervals. For sensors recording data at different update intervals, interpolation or extrapolation is used depending on whether the rate of change of values is high or low respectively. Different interpolation techniques exist to insert data [23], performing differently depending on the type of data. Stating a suitable interpolation technique inside the data sources metadata would allow a better alignment of the historical data by the framework. In the absence of this information, simple linear interpolation will be performed. If the faulty sensor has a bigger update interval than the correlating sensors, the values of the correlating sensors, recorded within the update interval of the faulty sensors, can simply be removed.

After synchronizing all data points, null values are removed and categorical features represented as strings are converted to numerical values. Afterward, the framework searches in a central repository for sensor candidates and computes the correlations between those sensors and the faulty sensor. The current implementation uses the Pearson correlation, but other techniques can be used as well. This correlation will indicate which sensors are more closely related to the faulty sensor in terms of the quantity they are measuring. First, the sensors with the highest correlations will be chosen as a set of predictors. The search space can be limited based on different criteria, e.g. the distance. Later, in case sufficient resources are available, also sensor candidates that show only low correlation can be tried as predictors to further improve the accuracy of the Virtual Sensor.

After the set of predictors is determined, the dataset is standardized as it not only removes bias but also enhances the performance of algorithms like ANN [24] and SVM [25].

C. Automated Model Selection

After the data has been preprocessed, the next step is to apply a machine learning algorithm. The first obstacle in choosing a suitable algorithm is to identify whether the system is dealing with a classification or a regression problem. This can be easily decided if the number of unique values in the historical data of the response variable is computed first. If the number of unique values is very small compared to the total number of samples in the dataset, the problem can be defined as a classification problem and a regression problem otherwise. This small percentage can be set prior to the deployment of the framework.

The next obstacle is to decide which algorithm to choose from a plethora of machine learning algorithms. One single algorithm cannot provide a suitable Virtual Sensor for every situation. For example, Linear Regression (LR) can create a model faster than most regression algorithms but it would not be able to create an accurate model of a non-linear data stream. Similarly, for a simple dataset, an Artificial Neural Network (ANN) might require too many resources. Not only will it require more computational resources, but it will also take more time to train and deploy a Virtual Sensor, which will reduce the user experience. Additionally, if both LR and ANN are trained on a dataset and the error scores of LR and ANN are quite close, it will be better to deploy the LR model, following the Occam's Razor principle [26].

One solution is to use a fast algorithm to deploy a model as early as possible while a time-consuming algorithm is trained in the background. This model is deployed as soon as it is ready. The question here is how accurate the Virtual Sensor from a faster algorithm will be and if it is even required to train a complex model in the case where the user can wait for some time to have a more accurate Virtual Sensor. The number of used algorithms and the tuning options can be decreased for a faster deployment of a Virtual Sensor. This can be achieved by simply creating a lookup table containing the time complexity of each algorithm, which enables the framework to estimate the training time (exact time cannot be determined as it is also dependent on the execution environment) of each algorithm for a particular dataset.

For a later evaluation, various algorithms are utilized (e.g. Linear Regression, Support Vector Regression (SVR), etc.) from the Python package scikit-learn [27] and a feed-forward neural network from Keras [28]. A grid search is used to compare all the algorithms with different tuning parameters. For example, SVR is applied with different kernels such as Radial Basis Function (RBF) and Polynomial with different degrees. After all the algorithms have been trained, the best algorithm is selected by comparing the accuracy of each algorithm. The model is then deployed as a Virtual Sensor and the metadata specific to that Virtual Sensor is created following the schema as mentioned in [29]. This metadata can be utilized for the deployment of Virtual Sensors in similar scenarios without training a new model from scratch. For example, if a sensor fails inside a particular building, a search for already deployed Virtual Sensors of the same type in that building can be initiated and if found, the information of the algorithm and its hyperparameters can be used to train a new model instead of applying a grid search. Additionally, if there is only a small difference in the accuracy of the highest-ranked algorithms, the model is selected based on its storage size.

An alternative to the grid search is to use an already developed AutoML framework like Auto-Weka [30], Auto-sklearn [31], etc. To evaluate the feasibility of using such frameworks, auto-sklearn was tested on the same datasets. Auto-sklearn utilizes an ensemble method to create a model. This method has some drawbacks which will be discussed in section V.

IV. Results

To showcase the results of the framework, two scenarios have been created: (a) sensors of the same type are chosen and (b) sensors of different types are used to create a Virtual Sensor. The sensor networks used for the experimentation are deployed by the City Municipality of Aarhus [34] in different solar power plants distributed over the city. They measure the power output of the plants in Watt with an update interval of 5 minutes except in the time between 22:00 to 05:00 every day, as it is the shutdown time of the power plants. To use sensors of different types, a second sensor array in the city is also included in the search space of the framework, which measures different quantities.

A. Virtual Sensor with same Sensor Type

There are 21 different solar power plants all over the city of Aarhus. The description of the sensors and the recorded data can be seen here [34]. Fig. 2 shows the locations of these power plants. The solar power plant installed on top of the public library is arbitrarily chosen as a location with a faulty sensor to create a Virtual Sensor. As explained in section III.B, power plants which are near to the faulty sensor and whose data has high correlation will be selected by the framework as predictors for the Virtual Sensor.

In Fig. 2, the faulty sensor is indicated by a red marker. Sensors with high correlations are shown by green markers while the sensors with mid-level correlation are shown by yellow markers. Although some of the sensors with green markers are not the closest sensors, they are chosen by the framework for creating the Virtual Sensor, because of their high correlation compared to other sensors. TABLE I shows the attributes of the historical data of the faulty sensor.

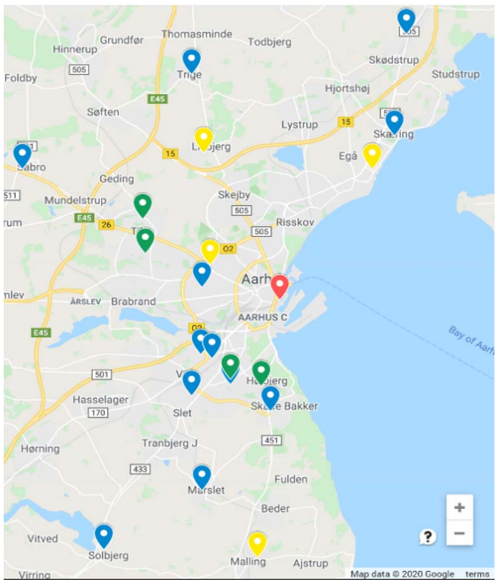

Fig. 2 Solar power plant locations in Aarhus (Denmark)

Percentile values of 25, 50 and 75 are also given to indicate the skewness of data.

TABLE I DATASET DESCRIPTION

Attributes	Value
Samples	24000
Mean	36126
STD	52914
Minimum	0
Maximim	307891
25%	144
50%	10758
75%	51630

This dataset consists of about four months of values. For measuring the accuracy of the model, 33% of the data is separated as hold-out test data. A 5-fold cross validation is chosen for the training of the algorithms via grid search and algorithms are scored based on their Mean Absolute Error (MAE) score. TABLE II shows the top 3 algorithms after training. For the grid search, the training time of the framework was limited to the update interval of the sensor, which was 5 minutes (on an Intel i7 8750h, 16 GB RAM). This ensures, the time required to train and compare all models is within the update interval and would not lead to gaps in the sensor's data stream. The fastest algorithm was LR, which was trained within 50 milliseconds. On the hold-out set, the best algorithm (ANN in this case) scored a MAE of 14898 with a R^2 score of 0.77.

TABLE II BEST RANKED MODELS FOR SAME SENSOR TYPE

Rank	Algorithm	Hyperparameters	Cross Validation MAE
1	ANN	Activation: ReLU Optimizer: Adam	13922
2	LR	N/A	14298
3	Bayesian Ridge Regression	N/A	14299

Autosklearn was also used to train a model with the same dataset. The maximum time limit to train the model was defined according to the update interval, which is 5 minutes in this case. In the absence of this maximum limit, auto-sklearn can continue to train for hours or even more. On the hold-out data, the model scored a MAE of 14296 and R^2 score of 0.78. A comparison of the model predictions from grid search and auto-sklearn can be seen in Fig. 3. Only predictions of the first 50 samples of the test dataset are shown. Test data is indicated with a grey color while predictions from auto-sklearn and grid search models are represented by blue and red lines respectively. It can be seen that the predictions of both models follow the test data quite well.

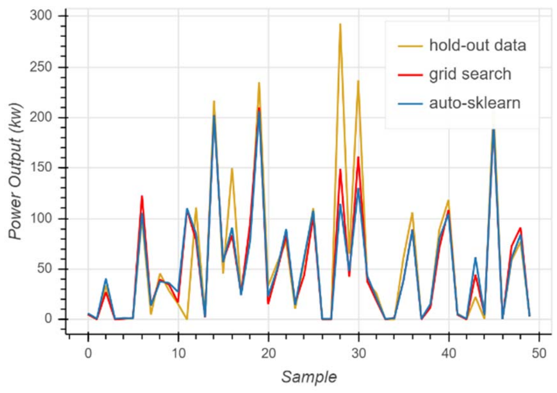

Fig. 3 Model predictions with grid search and auto-sklearn for same sensor types

B. *Virtual Sensor with different Sensor Types*

For experimentation purposes, in this scenario, we ignore the sensors that have been used to create the Virtual Sensor in the previous subsection. Here, to create a Virtual Sensor for the faulty sensor, a sensor array by the City Lab Aarhus has been chosen. This sensor array consists of different sensors, which measure Temperature (°C), Humidity (%), Wind Speed (km/h), Sunlight (PAR), Daylight (Lux), and Battery Level (%). The description of the sensors and their measured quantities can be found on the open data portal of Denmark [35]. Not surprisingly, the historical data from the failed sensor show a high correlation with the Sunlight values only. The sample size for this experiment was around 5800. The reason for the reduced data size is because, contrary to the power sensors, which record data with a 5-minute interval, the sunlight senor records data with an update interval between 10 and 15 minutes. The best 3 algorithms and their scores are shown in TABLE III.

TABLE III BEST RANKED MODELS FOR DIFFERENT SENSOR TYPE

Rank	Algorithm	Hyperparameters	Cross Validation MAE
1	ANN	Activation: ReLU Optimizer: RMSprop	20341
2	ANN	Activation: ReLU Optimizer: Adam	20367
3	LR	N/A	20368

On the hold-out data, the algorithm scored an MAE of 21248 and R^2 score of 0.71. The model created through auto-sklearn has an MAE of 18621 and R^2 score of 0.76 on the hold-out data. The comparison of the model's predictions can be seen in Fig. 4. The results from auto-sklearn are marginally better in this case than the results from the model created through grid searching individual algorithms.

V. Discussion

In section IV, a Virtual Sensor is created from multiple sources. Two different scenarios have been presented, one where the sensors measure the same physical quantity and another where a sensor measuring a different physical quantity is used. This has been done to show the robustness of the sensor selection procedure for a Virtual Sensor. This is important because the sensors selected to create a Virtual Sensor can be different, depending on the situation and the nearby deployed sensors. The proposed framework is capable of selecting these sensors by analyzing the data and the metadata of these sensors.

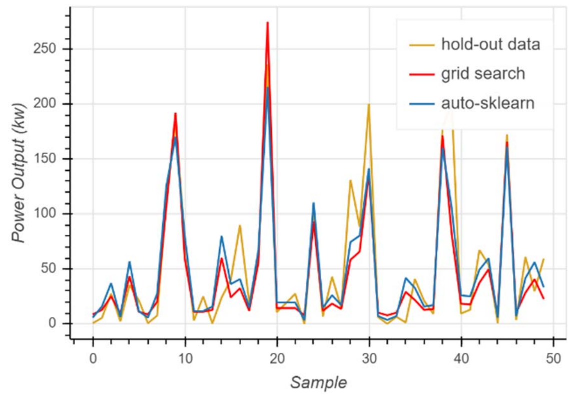

Fig. 4 Comparison of model predictions from grid search and auto-sklearn for different sensor types

A comparison of the models from individual machine learning algorithms and the ensemble model from auto-sklearn has been shown. Both methods show comparable results with different pros and cons. In the grid search implementation, various algorithms with different hyperparameter tuning options are provided. However, these options will have to be reduced when a Virtual Sensor has to be created in a short time, i.e. for sensors with small update intervals. The result would be a very simple model (e.g. only LR model or SVR with different degree polynomials) which may not be able to capture the true relationship between the predictors and the response. However, this is also a benefit for this implementation, as in presence of only a few predictors and a small dataset, a model created through this method will have comparable accuracy to that of any time-consuming algorithm.

Auto-sklearn creates a model through ensemble methods. While it can be argued that a better model can be created through this technique and overcome the No Free Lunch theorem [32][33], it would certainly take more time to train. Another drawback is the size of the model itself. The models created for the dataset presented in section IV have a size of 0.7 kB for LR, 500kB for SVR and 4.9MB for the model created by auto-sklearn. Considering a platform dealing with millions of IoT streams, where thousands of Virtual Sensors can be deployed, the size of the model can become a constraint in which an ensemble model for every Virtual Sensor would become problematic. Although the framework shows promising results in various scenarios, it also comes with limitations of its own. First, as the predictions always have some amount of error in it, Virtual Sensor should not be created in situations where the correctness of data is highly desirable (e.g. on an industrial floor where the functionality of the machines depends on the precision of data they receive from their surrounding). Another disadvantage of Virtual Sensors is that they are not suitable for long term usage in environments where the measured quantity is subjected to data drifts. In situations like these, Virtual Sensors should only be used to provide temporary support until the faulty sensor is replaced rather than deploying the Virtual Sensor indefinitely.

VI. Conclusion

Virtual Sensors provide a useful way to mediate the failure of IoT sensors and allow an uninterrupted execution of applications that depend on the sensor data. Here, automated creation of Virtual Sensors is crucial to any platform, which deals with a large number of IoT sensors and where the creation of such Virtual Sensor cannot be supervised by a human. In this paper, a novel approach has been proposed to create Virtual Sensors to replace faulty sensors in an automated way. The Virtual Sensors use historical data of the faulty sensor as well as correlating data sources nearby. The framework performs all necessary steps required to deploy a Virtual Sensor. It selects the data sources through metadata description as listed in this paper, preprocesses historical data, and trains and ranks machine learning algorithms.

The framework is capable of creating Virtual Sensors by incorporating sensors of any type. This is demonstrated in this paper by creating a Virtual Sensor, which is able to predict the output of a solar power plant by either using a close-by sunlight sensor or other nearby solar power plants. For the ranking of the different algorithms, the framework uses a grid search, but the feasibility of implementing other methods is also shown.

The framework has shown great results without any kind of human intervention. The required knowledge is provided by the data provider in terms of enriched metadata, which contains information about the sensors but without any kind of statistical knowledge.

For the future, we plan to annotate not only the data sources but also the algorithms with metadata. As [36] points out, knowing which algorithm has an advantage over the other depending on the type of data allows to guide the preprocessing step and to reduce the search space in the model selection step. Furthermore, we plan to add different outlier detection techniques to the preprocessing step to further clean the training and test data.

Acknowledgment

This work has been supported by the European Union program Horizon 2020 project IoTCrawler under grant agreement number 779852.

References

[1] 8 in 2018: The top transformative technologies to watch this year URL: https://cdn.ihs.com/www/pdf/IHS-Markit-2018-Top-Transformative-Technology-Trends.pdf

[2] Niles, I., & Pease, A., (2001), Toward a Standard Upper Ontology, in Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds, pp2-9.

[3] Fujiwara, K., Kano, M., Hasebe, S., & Takinami, A. (2009). Soft‐sensor development using correlation‐based just‐in‐time modeling. AIChE Journal, 55(7), 1754-1765

[4] Furdík, Karol & Lukac, Gabriel & Sabol, Tomas & Kostelnik, Peter. (2013). The Network Architecture Designed for an Adaptable IoT-based Smart Office Solution. International Journal of Computer Networks and Communications Security. 1. 216-224.

[5] c't Developer: Heimautomatisierung mit openHAB: neue Bindings und Ausblick in die Zukunft vom 25. November 2014 https://www.heise.de/developer/meldung/Heimautomatisierung-mit-openHAB-neue-Bindings-und-Ausblick-in-die-Zukunft-2462924.html

[6] S. Kabadayi, A. Pridgen and C. Julien, "Virtual sensors: abstracting data from physical sensors," 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks(WoWMoM'06), Buffalo-Niagara Falls, NY, 2006, pp. 6 pp.-592. doi: 10.1109/WOWMOM.2006.115

[7] Soldatos, J., Kefalakis, N., Hauswirth, M., Serrano, M., Calbimonte, J. P., Riahi, M., ... & Skorin-Kapov, L. (2015). Openiot: Open source internet-of-things in the cloud. In Interoperability and open-source solutions for the internet of things (pp. 13-25). Springer, Cham.

[8] T. Hochrein, I. Alig: Prozessmesstechnik in der Kunststoffaufbereitung. Vogel Business Media, Würzburg 2011, ISBN 978-3-8343-3117-5.

[9] Yuan, X., Ge, Z., Huang, B., Song, Z., & Wang, Y. (2016). Semisupervised JITL framework for nonlinear industrial soft sensing based on locally semisupervised weighted PCR. IEEE Transactions on Industrial Informatics, 13(2), 532-541.

[10] K. Aberer, M. Hauswirth and A. Salehi, "Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks," 2007 International Conference on Mobile Data Management, Mannheim, 2007, pp. 198-205., doi: 10.1109/MDM.2007.36, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4417143&isnumber=4417107

[11] Feurer M., Hutter F. (2019) Hyperparameter Optimization. In: Hutter F., Kotthoff L., Vanschoren J. (eds) Automated Machine Learning. The Springer Series on Challenges in Machine Learning. Springer, Cham

[12] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In Advances in neural information processing systems (pp. 2951-2959).

[13] He, X., Zhao, K., & Chu, X. (2019). AutoML: A Survey of the State-of-the-Art. arXiv preprint arXiv:1908.00709.

[14] Balaji, A., & Allen, A. (2018). Benchmarking automatic machine learning frameworks. arXiv preprint arXiv:1808.06492.

[15] Feurer, M., & Hutter, F. (2018). Towards further automation in automl. In ICML AutoML workshop (p. 13).

[16] Dietterich, T. G. (2002). Ensemble learning. The handbook of brain theory and neural networks, 2, 110-125.

[17] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(Feb), 281-305.

[18] Sparks, E. R., Talwalkar, A., Haas, D., Franklin, M. J., Jordan, M. I., & Kraska, T. (2015, August). Automating model search for large scale machine learning. In Proceedings of the Sixth ACM Symposium on Cloud Computing (pp. 368-380).

[19] Hall, M. A. (2000). Correlation-based feature selection of discrete and numeric class machine learning.

[20] Kohavi, R. (1995, August). A study of cross-validation and bootstrap for accuracy estimation and model selection. In Ijcai (Vol. 14, No. 2, pp. 1137-1145).

[21] Chung, C. M., Chen, C. C., Shih, W. P., Lin, T. E., Yeh, R. J., & Wang, I. (2017, June). Automated machine learning for Internet of Things. In 2017 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW) (pp. 295-296). IEEE.

[22] Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2006). Data preprocessing for supervised leaning. International Journal of Computer Science, 1(2), 111-117.

[23] Lepot, M., Aubin, J. B., & Clemens, F. H. (2017). Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. Water, 9(10), 796.

[24] Anysz, H., Zbiciak, A., & Ibadov, N. (2016). The influence of input data standardization method on prediction accuracy of artificial neural networks. Procedia Engineering, 153, 66-70.

[25] Luor, D. C. (2015). A comparative assessment of data standardization on support vector machine for classification problems. Intelligent Data Analysis, 19(3), 529-546.

[26] Buntine, W. (1990). Myths and legends in learning classification rules.

[27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

[28] Chollet, François (2015). Keras. URL: https://github.com/fchollet/keras

[29] Schelter, S., Böse, J. H., Kirschnick, J., Klein, T., & Seufert, S. (2017). Automatically tracking metadata and provenance of machine learning experiments. In Machine Learning Systems Workshop at NIPS.I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[30] Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2017). Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. The Journal of Machine Learning Research, 18(1), 826-830.

[31] Feurer et al., Efficient and Robust Automated Machine Learning, Advances in Neural Information Processing Systems 28 (NIPS 2015)

[32] Wolpert, D.H., & Macready, W.G. (1995). No Free Lunch Theorems for Search.

[33] Rokach, L. (2010). Pattern classification using ensemble methods (Vol. 75). World Scientific.

[34] Solar Power Panel Dataset at Open Data DK. URL: https://www.opendata.dk/city-of-aarhus/solcelleanlaeg

[35] Environmental Dataset at Open Data DK. URL: https://www.opendata.dk/city-of-aarhus/sensordata

[36] Brazdil, P. (1998, April). Data transformation and model selection by experimentation and meta-learning. In Proceedings of the ECML-98 Workshop on Upgrading Learning to Meta-Level: Model Selection and Data Transformation (pp. 11-17).