CHARGING POTENTIAL OF V4G THROUGH V2G STANDARD PROTOCOL FOR CCS

David Wannier[1], Jean-Marie Alder[1], Helena Pereira[1], Jérémie Vianin[1], João Carlos Ferreira Da Silva[1], Jérôme Treboux[2], Dominique Genoud[2], Gregorio Bonadio[3], Toufann Chaudhuri[4], Georges Darbellay[5], Luc Dufour[6],
[1]Institute of Sustainable Energy (IEE), HES-SO Valais Wallis, Switzerland; [2]Institute of Information Systems (IIG), HES-SO Valais Wallis, Switzerland; [3]ElectrInfo Sàrl, Switzerland; [4]GreenMotion SA, Switzerland; [5]OIKEN SA, Switzerland; [6]EDF, France. david.wannier@hevs.ch, jean-marie.alder@hevs.ch, helena.pereira@hevs.ch, jeremie.vianin@hevs.ch, joao.ferreiradasilva@hevs.ch, jerome.treboux@hevs.ch, dominique.genoud@hevs.ch, gregorio@electrinfo.ch, tchaudhuri@greenmotion.ch, georges.darbellay@oiken.ch, luc.dufour@edf.fr

ABSTRACT

The eVIP (Energy Visualisation Integration and Prediction) project aims to predict the load curve of electric vehicles in a semi-private context related to hotels and restaurants. Using the gradient boosted tree algorithm, it is possible to predict the consumption of a hotel with an accuracy of approximately 83.8% with non-intrusive devices. By using this prediction and the data collected when an electric vehicle is being charged at the hotel's charging station, the peak consumption of the hotel can be optimized.
We have also opened the way for V4G (Vehicle for Grid) to allow bi-directional energy flows in this semi-private micro-grid and propose flexibility services to Distribution System Operators (DSO).

INTRODUCTION

Several hotel owners have gathered to create a network of electric vehicles (EV) in the Val-d'Hérens touristic valley (Switzerland). In addition to offering the necessary mobility to tourists, there is a desire to promote the image of a green and ecological region through the Green-Mobility concept (https://bit.ly/2YuYQ87 reportage of the French-speaking Swiss television), which received the 'Best innovation in services in addition to accommodations' award at the Worldwide Hospitality Awards 2016.
In the next few years, we expect a significant increase of the electric vehicle fleet, which will increase the grid electricity demand. During periods of high consumption, charging electric vehicles can overload the local hotel's electric network. Self-consumption of local photovoltaic (PV) leads to a lower grid power demand, ensuring the use of renewable energy. The new standard protocol vehicle to grid (V2G) for Combined Charging Systems (CCS) could help to store PV energy and reuse it during the night. It could also propose the flexibility to the DSO through the vehicle for grid (V4G) concept. V4G is defined as the ability to recover the energy stored in the battery to support the electrical network, thus ensuring a role of balance between production and consumption and V2G is the protocol implemented by the industry to standardize bi-directional energy flows [1].
In this paper, a novel concept, including mobile application for EV charging management is presented. The goal for hotel customers is to interact with the eVIP system and to charge their vehicles while sending data about their desired departure time and battery state of charge. Then, a prediction model using machine learning (ML) and the gradient boosted tree (GBT) algorithm processes these data. The objective is to predict the hotel's electric consumption for the next 15-minute period to prevent overloads of the electrical network. A prediction application programming interface (API) has been developed and deployed to a production server as well as dashboards to display energy flows.

METHODOLOGY

We used two track unified process (2TUP) methodology [2] to define business needs in the functional branch, and in parallel test different ML algorithms in the technical branch.
Figure 1 represents the methodology used in the technical branch for the prediction part of this study. It is based on the CRISP-DM methodology [3] and has been adapted to the project needs and available resources.

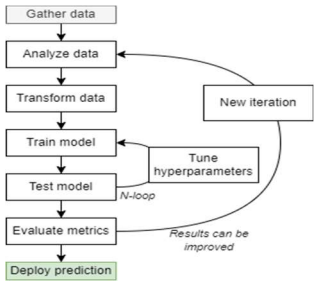

Figure 1: Methodology used to train and improve prediction model inspired by CRISP-DM [3]

We fusioned both functional and technical branches and implemented the architecture described in Figure 2.
All hotels, clients, and vehicles information such as specification of on-board charger type, battery size, IP addresses and access tokens are stored in a relational database, while all data, extracted from the hotel's smart meters measuring electricity consumption and production, is stored in a time series database.
The front and back-end servers are connected to the same VPN as the charging stations and databases, allowing the application to use the necessary data and automatically control the stations.
Using our H2020 GoFlex web application, which is an electricity monitoring dashboard, hotel managers can

control the charging stations manually or automatically and analyze electricity production and consumption graphs.

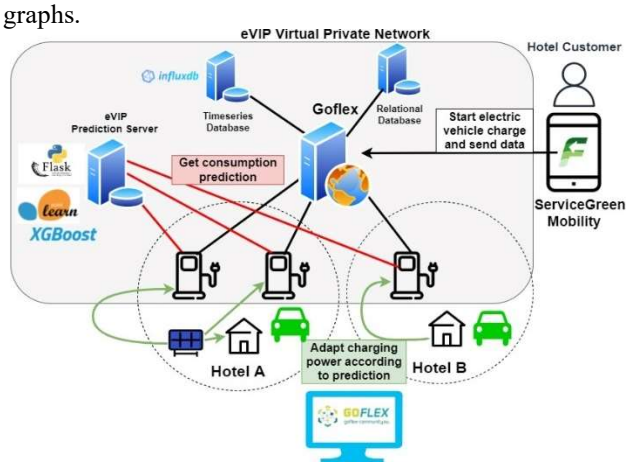

Figure 2: eVIP architecture

The electric consumption prediction model is stored on the time series database. Charging stations' microcomputers are sending consumption data to the eVIP prediction server to retrieve the consumption prediction for the following 15 minutes. With this prediction the microcomputers can adapt the power of the charging station to anticipate the increase in consumption of the hotel.

Then, we developed a mobile application to add end user interaction and get additional information, including battery state of charge at arrival and the estimated time of departure of the EV. Considering all this information and the prediction algorithm, the microcomputer can dynamically control the charge to smooth out the hotel's peak consumption when charging the vehicle (Figure 3) or propose V4G flexibility to the DSO.

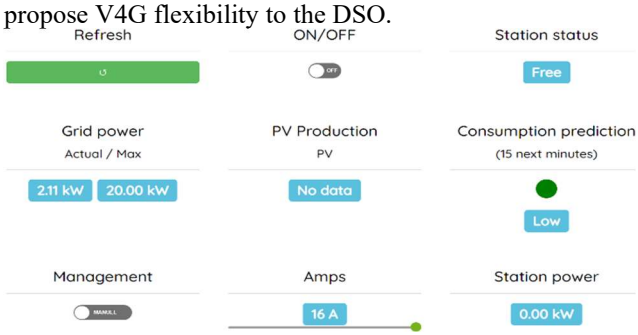

Figure 3: Hotel's dashboard

ServiceGreenMobility smartphone application

The Service Green Mobility smartphone application offers hotel guests a map containing the geo-referenced charging stations and V4G availabilities. Guests activate the charge while specifying the percentage of battery remaining in the vehicle and the desired departure time. The status of the vehicle can be detected (starting, charging, charged). With a V2G enabled vehicle, we can use the vehicle's battery as a buffer to further smooth out the hotel's peak consumption while ensuring that guests have their vehicle ready for departure.

The application is connected to the Goflex system, which allows us to fetch the list of available stations and their status (available, not available, charging, V4G ready).

The application allows us to configure values before charging as shown in Figure 4.

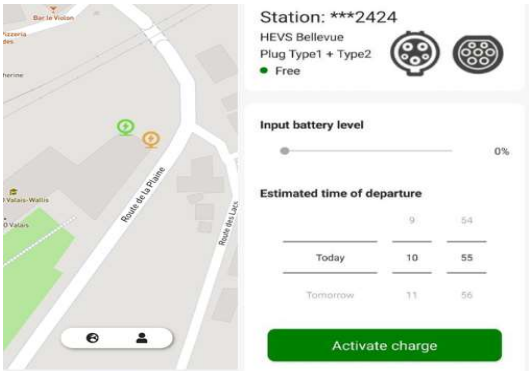

Figure 4: Mobile App ServiceGreenMobility (in orange on the map means V4G ready)

In the next section, we will describe the ML algorithm.

EVIP Prediction server for V4G and consumption peak smoothing

Several machine learning tools and algorithms have been tested in the 2TUP technical branch [4] and we have selected the gradient boosted tree algorithm. Gradient boosting is a machine learning algorithm used for regression and classification problems that computes an ensemble of weak prediction models to create a more robust prediction model [5]. In this case, it consists of several decision tree algorithms, which are optimized by an arbitrary differentiable loss function.

The classification problem in this study consists of predicting the power consumption for the next 15 minutes to automatically set the amp level of the electric vehicle charging stations to prevent overloads. We compared different machine learning algorithms to predict four clusters (each representing a consumption level from low to high). We will now present data pre-processing and prediction training phases as well as implementation.

Input data set and pre-processing

To train the prediction model, the time series dataset has been used, containing a timestamp and the power value of our pilot hotel with a granularity of one minute. Data was gathered using smart meters monitoring the hotel's consumption daily for 1 year and 4 months, which gives a total of 701'184 samples, excluding a few missing values caused by smart meter errors and general system failures, including power outages. Missing values are generally isolated and only last one minute or two. Around 1% of the total values are missing.

Input data must be processed to comply with the format required by the classification algorithm and to compute the mean power usage for a 15-minute period. To perform pre-processing, we used "Knime data analytics platform" [6]. Pre-processing in this experiment includes the following

transformations:

- Extract month number, day of week, week number, hour, and minute from the timestamp in string format.
- For each timestamp, take the last 15 values and calculate the mean power usage. Ignore the first 15 values as they don't have enough previous values to calculate the 15-minute mean.
- Shift the calculated mean by 15 values to create the class column we want to predict 15 minutes in advance.
- Classify the power usage values (mean and live values) according to pre-defined thresholds listed in Table 1 following preliminary expermients.

Table 1: Thresholds definition for consumption classes

Consumption Class	Value Range
Low	[0W; 1900W]
Medium	]1900W;3700W]
High	]3700W;6300W]
Highest	]6300W; ∞W[

Figure 5 shows that the number of samples are not balanced. Medium consumption comes out to be the most frequent class, contrary to the highest class.

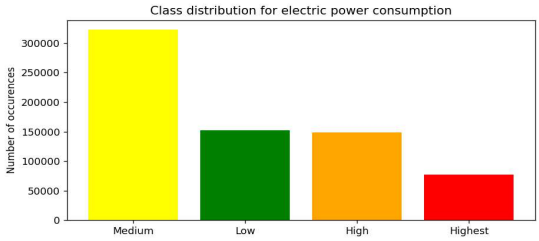

Figure 5: Class distribution bar chart

The initial data set contains missing values that have been processed as follow: if data is not available for more than 15 minutes (15 values), the algorithm skips this time frame and prediction starts again once enough values are available to calculate the mean value. If less than 15 consecutive minutes are missing, the algorithm replace missing value with the previously available value.
Finally, the data set is partitioned to create a training set and a test set. To ensure that the model is trained on a full year, data is sorted chronologically and split at 80% of rows from top, which equals 12.8 months of data. The remaining 20% will be used for testing.

Prediction model training and evaluation

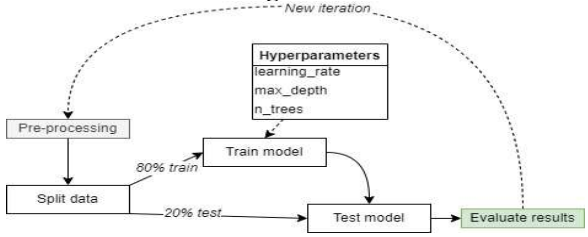

Figure 6: Learner-predictor pattern to train prediction model inspired by [7]

Prediction model has been trained and tested following the method presented in Figure 6.

Parameter tuning
To assess the optimal algorithm parameters, several iterations have been made using different hyperparameters listed in table below.

Table 2: List of tested hyperparameters [10]

Hyperparameter	Description	Tested values	Granularity
Learning rate	weighting factor to limit corrections between predictors to prevent overfitting	[0.01; 0.3]	variable
Max depth	maximum depth of each tree	[3;20]	1
Number of estimators	number of trees to be built	[10; 1200]	50

All other uncovered hyperparameters listed in [9] (Python API section) are set to default for our pilot phase. For each value to predict, the algorithm assigns a prediction score for each class. The class with the highest prediction score is the prediction result. If the prediction score came out to be equal between one or more classes, the highest consumption class is selected to keep a safe approach.

Implementation of the eVIP prediction server
This prediction pattern has been implemented in Python 3.9.6 using "scikit-learn" [8] and "XGboost" [9] ML packages. Python was chosen as the main framework as it is easy to understand and handy when it comes to data processing, validation, and data mining. It is also easy to deploy on a prediction server or directly on the charging station's microcomputer if needed.
Two separated Python programs were created:

- A simple command line interface program to train the prediction model, test the model and gather statistics.
- A lightweight Flask API to fetch the consumption prediction using the prediction model exported from the command line program in json format, which can later be improved and replaced.

We exported the data set with Knime to train the prediction model.

Results analysis of the eVIP prediction algorithm
A python script has been developed to automate multiple train and test iterations with different hyperparameters. The following metrics have been collected during all train and test iterations:

- Accuracy: percentage of correct classifications [8]
- Cohen's Kappa: The level of agreement of the classifier [8]
- F1 score: measure of accuracy that uses precision and recall values [8]
- Receiver operating characteristic (ROC) curve: plot of the algorithm sensitivity according to the prediction score [11].

- Area under curve (AUC) for each class: measure of the accuracy of the prediction distribution model, considering the unbalanced number of samples per class [11].

Each training and testing iterations and all resulting metrics are stored to compare several hyperparameters configurations.

RESULTS

The whole architecture and each process has been tested and validated during the eVIP project.

Microparameters exploration

Several iterations have been made to find the most efficient hyperparameters for the gradient boosted tree algorithm. Only one parameter is being optimized at a time and, once an optimal value has been found, we keep it and use it for the next iteration.

Figure 7 represents the most efficient learning rate (0.18) using other fixed hyperparameters found in previous iterations.

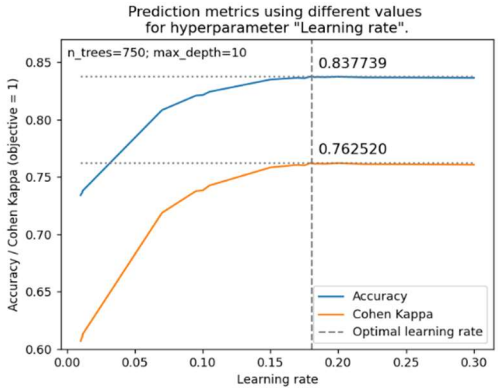

Figure 7: Learning rate hyperparameter tuning results

The maximum depth of a tree has been tuned and an optimal depth is equal to 10 as demonstrated in Figure 8. Some trees might have a smaller depth as the XGboost algorithm optimizes trees to prevent overfitting.

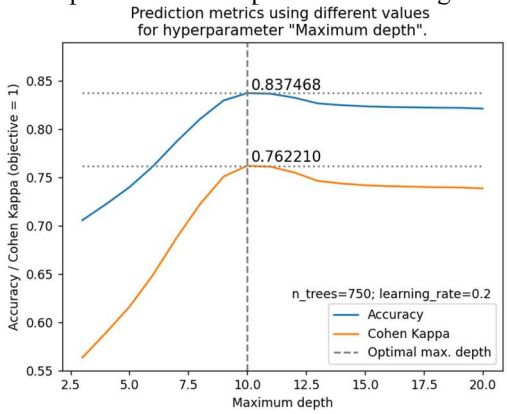

Figure 8: Maximum depth hyperparameter tuning results

The most efficient number of estimators (number of trees) found corresponds to 800 trees as shown in Figure 9.

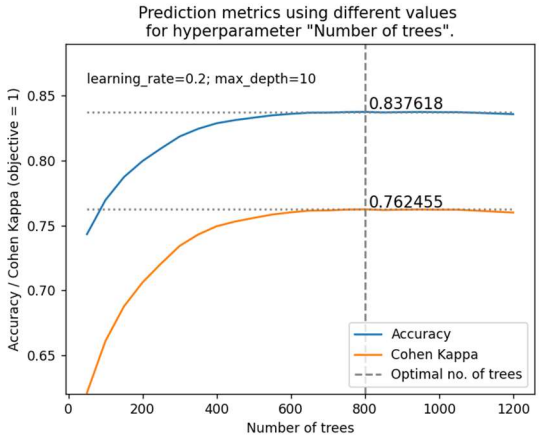

Figure 9: Number of trees hyperparameter tuning results

Prediction results analysis

Table 3 represents the results with the optimal prediction model using the most efficient hyperparameters. At this stage of implementation, the optimization focuses on accuracy to get a simple and comprehensible metric to deploy an efficient pilot API. The objective is to bring accuracy as close to 1 (100%) as possible.

Table 3: Current best prediction results with learning rate = 0.18; maximum depth = 10; no trees = 800

Accuracy	Cohen Kappa	F1 Score
0.838	0.763	0.838

The accuracy obtained shows that 83.8% of predictions are correct without taking in account the unbalanced number of classes.

The confusion matrix in Figure 10 is used to evaluate the performance of the prediction model [12] and can be visualized to spot scenarios where the prediction model was correct or not. The green diagonal represents the correctly classified values. The red cells show where the algorithm failed to classify data.

		Predicted Class			
		Low	Medium	High	Highest
Actual Class	Low	25236	5166	107	6
	Medium	5537	55827	2766	197
	High	122	3496	23939	2231
	Highest	6	339	2785	12477

Figure 10: Confusion matrix for the most accurate prediction model

The number of values classified in the opposite class, e.g., “low” classes predicted as “high” or “highest”, is very low. This means that the prediction algorithm has managed to classify most values in the correct consumption class or in the neighbouring classes, thus limiting the error margin and consequenses of incorrect predictions.

The AUC and the receiver operating characteristic curve (ROC curve) are better metrics to assess the accuracy of an unbalanced multiclass prediction model [13]. The ROC Curve represents the rate of true positive predictions depending on the prediction score (confidence score) for each class [13]. Figure 11 shows ROC curves for each

class against all others, which transforms the single multiclass problem into four binary problems.

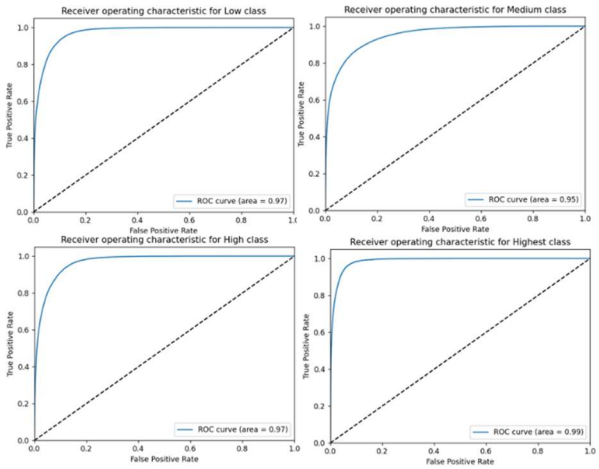

Figure 11: ROC curve for all classes. Low and medium class at the top, high and highest at the bottom respectively.

The AUC for each class is listed in Table 4 and shows that the prediction model has less capabilities in separating the "Medium" class from the others and is prone to more errors than other classes [13]. Appart from that, the algorithm is efficient finding the other classes for most of prediction scores.

Table 4: AUC for each prediction class

Class	AUC
Low	0.97
Medium	0.95
High	0.97
Highest	0.99

EVIP Prediction API deployment

The prediction API developped with Python Flask has been successfully deployed to a cloud virtual machine (VM). To ensure that the API is only accessed by authorized users, an API token must be sent as well as the maximum power in watt. The date is fetched based on the hotel location and generated by the API directly.

CONCLUSION

In this paper, eVIP, a full-scale energy optimization system for hotels with electric vehicle is presented. The mobile application is used by hotel's guests to charge their electric vehicle, which sends useful information to the eVip system about the desired departure time and battery state of charge. The eVIP prediction API is utilized to smooth peak consumptions by optimizing EV charging amps when needed. Finally, V2G enabled EVs can provide electricity to hotels when power consumption is too high or propose V4G for grid stability to their DSO.

The presented prediction model managed to correctly classify 83.8% with promising AUC scores for all classes, the lowest AUC being 0.95 for the "Medium" class, which is the most common class. Some potential improvements have been identified: other hyperparameters and the relationship between them should be assessed. Additional data concerning PV production, weather and consumption data should be added to the model's input to enhance accuracy. Current deployed and running prediction API results should be gathered and analysed.

Acknowledgments

We would like to thank The Ark Foundation, OIKEN SA, GreenMotion SA and ElectrInfo Sàrl, who supported us throughout the project and made it possible.

REFERENCES

[1] RTE: la technologie V2G certifiée pour l'équilibre des réseaux, 2022, https://www.avere-france.org/rte-la-technologie-v2g-certifiee-pour-lequilibre-des-reseaux/

[2] I. Essebaa & S. Chantit, (2018). *Model Driven Architecture and Agile Methodologies: Reflexion and discussion of their combination*. Proceedings of the Federated Conference onComputer Science and Information Systems, 939-948

[3] P. Chapman et al., 2000, *CRISP-DM 1.0: Step-by-step data mining guide*

[4] S. FAFALIOS, P. CHARONYKTAKIS, I. TSAMARDINOS, 2020, "Gradient Boosting Trees", *Gnosis Data Analysis PC*, 1-3

[5] L. Dufour, D. Genoud, B. Ladevie, D. Wannier, *Optimized method to predict energy in a microgrid*, ISEC 2018 conference

[6] M.R. Berthold et al., June 2006, "KNIME: the Konstanz Information Miner", *Workshop on Multi-Agent Systems and Simulation (MAS&S), 4th Annual Industrial Simulation Conference (ISC)*, 05-07 June 2006, Palermo, Italy, pp. 58-61

[7] V. Lakshmanan, S. Robinson & M. Munn, 2020, *Machine Learning Design Patterns,* O'Reilly Media, Inc., Sebastopol, Canada

[8] "scikit-learn", https://scikit-learn.org/stable/

[9] "XGboost", https://xgboost.readthedocs.io/en/stable/python/python_api.html

[10] J. Brownlee, 2016, *Tune Learning Rate for Gradient Boosting with XGBoost in Python*, https://machinelearningmastery.com/tune-learning-rate-for-gradient-boosting-with-xgboost-in-python/

[11] J. M. Lobo, A. Jiménez-Valverde, R. Real, 2007, *AUC: a misleading measure of the performance of predictive distribution models,* Global Ecology and Biogeography

[12] Z. Karimi, 2021, *Confusion matrix*

[13] S. Yang, G. Berdine, 2017, *The receiver operating characteristic (ROC) curve,* The Southwest Respiratory and Critical Care Chronicles, 34-36