PyARC the Python Algorithm for Residential load profiles reConstruction

SoftwareX 28 (2024) 101878

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

PyARC the Python Algorithm for Residential load profiles reConstruction

Lorenzo Giannuzzo [a,b,*], Daniele Salvatore Schiera [a,b], Francesco Demetrio Minuto [a,b], Andrea Lanzini [a,b]

[a] Energy Center Lab, Polytechnic of Turin, via Paolo Borsellino 38/16, 10152 Turin, Italy
[b] Department of Energy (DENERG), Polytechnic of Turin, Corso Duca degli Abruzzi 24, 10129 Turin, Italy

ARTICLE INFO

Keywords:
Load profiles reconstruction
Python-based software
Non-intrusive methodology
Machine-learning
Data scarcity

ABSTRACT

Load profiling for residential aggregates encounters challenges due to data scarcity and the inadequacy of standard profiles obtained from statistical analyses. In the absence of hourly data, many methods rely on standard profiles, which could lead to significant errors in consumption estimation, especially for evaluating specific aggregates. This article presents PyARC, a Python-based algorithm trainable with customizable consumption data, which addresses the problem related to evaluating the energy consumption of specific aggregates by using typological profiles extracted from similar users, thereby improving accuracy. The algorithm's innovative approach uses Association Rule Mining and Random Forest Classification to reconstruct the load profiles of aggregates, providing a more robust solution for estimating the electrical load with limited data.

Metadata

Nr.	Code metadata description	*Please fill in this column*
C1	Current code version	*v0.9*
C2	Permanent link to code/ repository used for this code version	https://github.com/LorenzoGiannuzzo/PyARC.git
C3	Permanent link to reproducible capsule	
C4	Legal code license	*GNU Lesser General Public License version 3 (LGPL 3.0)*
C5	Code versioning system used	*git*
C6	Software code languages, tools and services used	*Python*
C7	Compilation requirements, operating environments and dependencies	*Python 3.12.1, joblib 1.3.2, matplotlib 3.8.2, numpy 1.26, pandas 2.2.0, scikit_learn 1.4.0, seaborn 0.13.2.*
C8	If available, link to developer documentation/manual	
C9	Support email for questions	lorenzo.giannuzzo@polito.it

1. Motivation and significance

Reconstructing electrical load profiles is pivotal in the energy sector as it provides a detailed understanding of how electricity is consumed over time. This information helps utilities and energy providers optimize grid management, improve demand forecasting, and increase energy efficiency. Moreover, accurate load profiles also typically enable better integration of renewable energy sources, such as solar and wind, by balancing supply and demand. However, One of the major challenges that can arise during a load profiling process is the availability of hourly or sub-hourly consumption data, especially for the residential sector [1]. Many studies have been carried out on load profiling techniques, as shown in [2], but most of them provide software and models that require high-resolution data and additional information such as user habits, appliances, building structure, and occupant behavior to perform load profiling [3–5]. For example, Chuan et al. [6] aimed to establish a model to reconstruct and describe the load profile of different types of users using electrical consumption data enriched with additional information on appliances and building structure. Similarly, in [7] a novel approach was proposed to generate synthetic load profiles with realistic variability and demand peaks using very high-resolution data (15 min resolution). In [8], a study was conducted to investigate the electricity consumption habits of occupants, which is generally a pre-requisite for training load profile reconstruction models, using sub-hourly resolution data. Furthermore, in the article by Piscitelli et al. [9], an implementation of a non-intrusive approach was carried out to reconstruct the electrical load profiles in the building domain using monthly energy bills data, enriched with additional information collected through a telephone survey related to the appliances and habits of the users. In contrast, the article by Lazzeroni et al. [10] proposed a data-driven approach to reconstruct and predict load profiles of single end-users

* Corresponding author at: Via Paolo Braccini 29, 10141 Turin, Italy.
E-mail address: lorenzo.giannuzzo@polito.it (L. Giannuzzo).

https://doi.org/10.1016/j.softx.2024.101878
Received 11 March 2024; Received in revised form 2 September 2024; Accepted 2 September 2024
Available online 13 September 2024

with minimal input data. They created a model that is the combination of clustering and classification processes to predict typical load profiles based on monthly energy bills, obtaining encouraging results compared to other load profile reconstruction techniques, emerging as a solid alternative to load profiling methodologies based on clustering and classification processes.

In summary, load profiling methodologies in the literature have generally used high-resolution data, often enriched with additional information such as information on appliances, habits and occupational activities, with some exceptions such as [10], which however aimed at reconstructing the electrical load profiles only of individual users. This leaves a gap in the literature for methodologies that can reconstruct load profiles at an aggregate level without the need for high-resolution data or extensive user-specific information. In order to fill this gap, this research work presents the PyARC algorithm, which makes use of the methodology described in [1]:

- Propose a practical way to use a data-driven methodology, trainable using datasets from different geographical areas, that can be used to reconstruct the hourly electricity consumption of residential users at an aggregate level;
- Reconstruct the hourly electricity consumption of residential users using non-intrusive machine learning techniques in a simple way that relies only on easily available data, such as monthly consumption data, without any additional information about the users' work activities, habits, or appliances.

2. Software description

PyARC is a Python-developed algorithm designed for the reconstruction of residential aggregate electrical load profiles. It leverages an algorithm for Association Rule Mining to model relationships between Time-of-Use (ToU) data and electrical consumption profiles. The algorithm uses a non-intrusive machine learning methodology that can be used to generate residential electrical consumption profiles at an hourly resolution level using only monthly consumption data (i.e., electrical energy bills). The methodology used to reconstruct the aggregated electrical load profile is mainly composed of three phases: first, identifying the typical load patterns of residential users through k-Means clustering performed on hourly electrical consumption data, supported by evaluation metrics to identify the optimal number of clusters such as Davies-Boudin Index, Elbow Method, and Silhouette Score, then implementing a classification model (Random Forest), to identify, on a monthly basis and for each user, the typical load pattern choosing from the cluster centroids obtained during the clustering process based on features extracted from monthly energy bills and, finally, reconstructing and aggregating the electrical load profile through user-specific rescaling factor based on monthly bills.

2.1. Software architecture

The software provides two main modes of operation: Model Training and Profile Reconstruction, as depicted in Fig. 1.

The Model Training process involves executing the method used to train the classification model and extract the normalized cluster centroids. The input data of this process is the hourly electrical consumption measures and the ToU. The outputs of Model Training are the normalized cluster centroids and the classification model, followed by secondary outputs such as plots and documentation files containing the performance of the classifier during its training and testing phase. Moving on to the Profile Reconstruction, the software provides two different features: the Pre-trained Model and the User-trained Model. For the first option, the Profile Reconstruction method is employed using the pre-trained model, which was previously trained and saved in the software's directory. For the second method, the user-trained model is used instead. To perform the second option, the Model Training process is required to be executed first, to train and store the user-trained model in the software's directory.

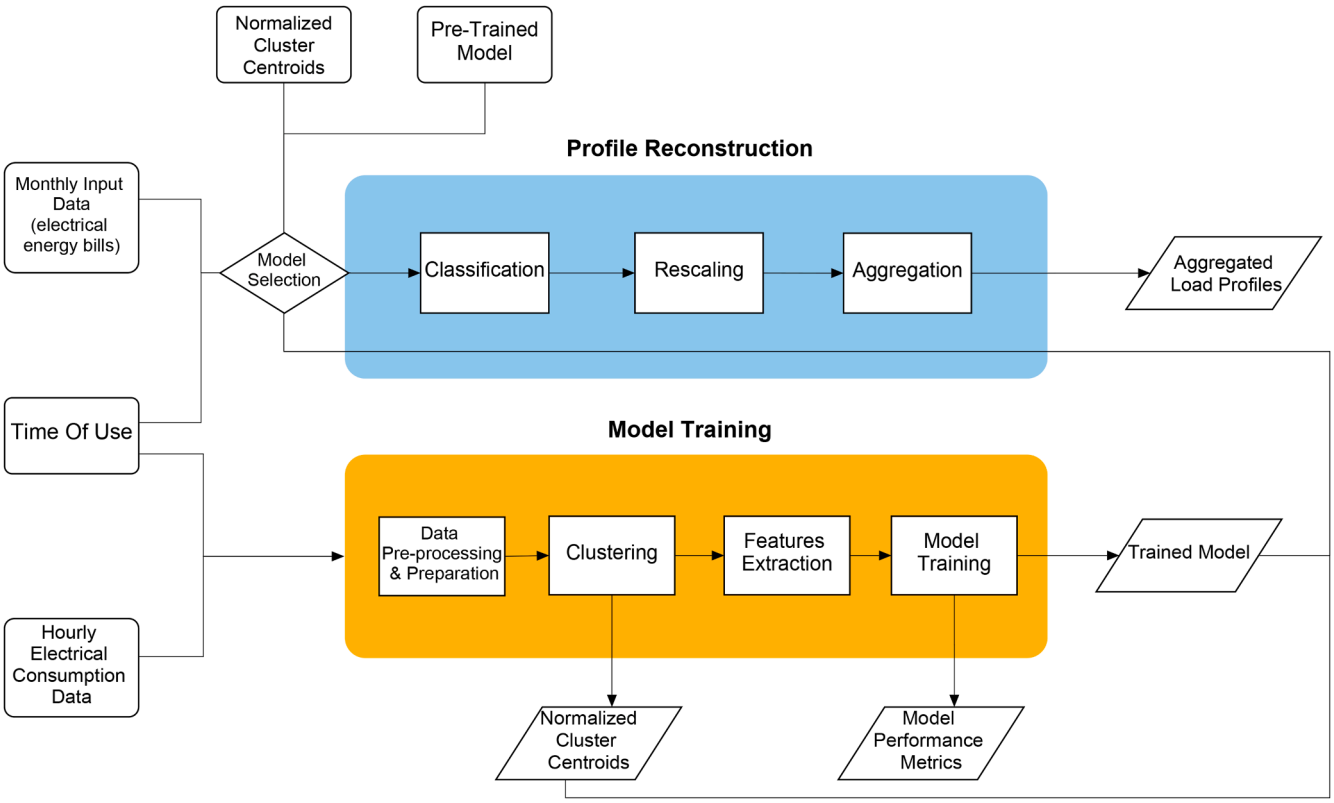

Fig. 1. PyARC features and functionalities.

2.2. Software implementation

The algorithm can be divided into several key components, each playing a specific role in the analysis and reconstruction of consumption profiles, as shown in Fig. 2.

- Data – This folder is populated by the required input data which is necessary to run the software. Data is divided into:
  - Input Training Data – Folder that contains the hourly electrical consumption measures and ToU required to train the user-trained model, in .csv format;
  - Input data – folder containing the monthly electrical energy bills and ToU required to perform the profile reconstruction method, in .csv format;
  - Default Training Data – Folder that contains the data previously used to create the pre-trained model;
  - Centroids – Folder containing the normalized centroids obtained from the clustering process of the Model Training;
  - Output Data – Folder containing the output generated from the Profile Reconstruction process, namely the hourly aggregated electrical consumption in .csv format.
- Docs – This folder contains information related to the performance metrics of the pre-trained and user-trained classification models;
- PyARC – This folder is populated by Python scripts containing software's functionalities.
- Pre-trained Model – Folder containing the pre-trained classification model obtained using the joblib package;
- User-trained Model – Folder containing the user-trained classification model obtained using the joblib package;
- Plots – This folder contains figurative outputs generated by the algorithm.

2.3. Software functionalities

The software provides a set of functionalities for training and using models to reconstruct aggregated electrical load profiles for residential users. The main operations and functionalities are:

- Model Training – This process executes the `train_model()` function to train the classification model using hourly electricity consumption measurements and ToU data. The function starts by delving into required input data files, getting both electrical consumption and ToU information through specialized CSV handlers. The data then undergoes a series of transformations handled by the "DataFrameProcessor", which is responsible for validating the ToU data, normalizing consumption values, dealing with missing or outlier data, and finally calculating monthly consumption averages. The process continues with the clustering process, which consists of a K-means clustering supported by an optimal number of cluster evaluation metrics. The centroids obtained from the clustering process are then saved in the software's directory. Subsequently, the function extracts features from monthly electrical bills and ToU to train and test the classification model, which is then cross-validated and stored in the software's directory. In addition, the process of the creation of the user-trained model is parallelized on the different cores of the machine used by the software user to speed up the software execution.

The model hyperparameters are chosen based on an iterative tuning process handled by the following function:

The function is designed to train a Random Forest Classifier model with hyperparameter tuning using the "GridSearchCV" method. The process starts by initializing a Random Forest model with specific initial hyperparameters, such as using the Gini criterion and allowing trees to expand without a maximum depth limit. A hyperparameter grid is defined, outlining different values for parameters such as the number of estimators, minimum samples to split, minimum samples per leaf, and maximum features. This grid acts as a set of possible configurations for the Random Forest model. The "GridSearchCV" object is then used, incorporating the Random Forest model, the hyperparameter grid, 5-fold cross-validation, and accuracy as the scoring metric. This allows

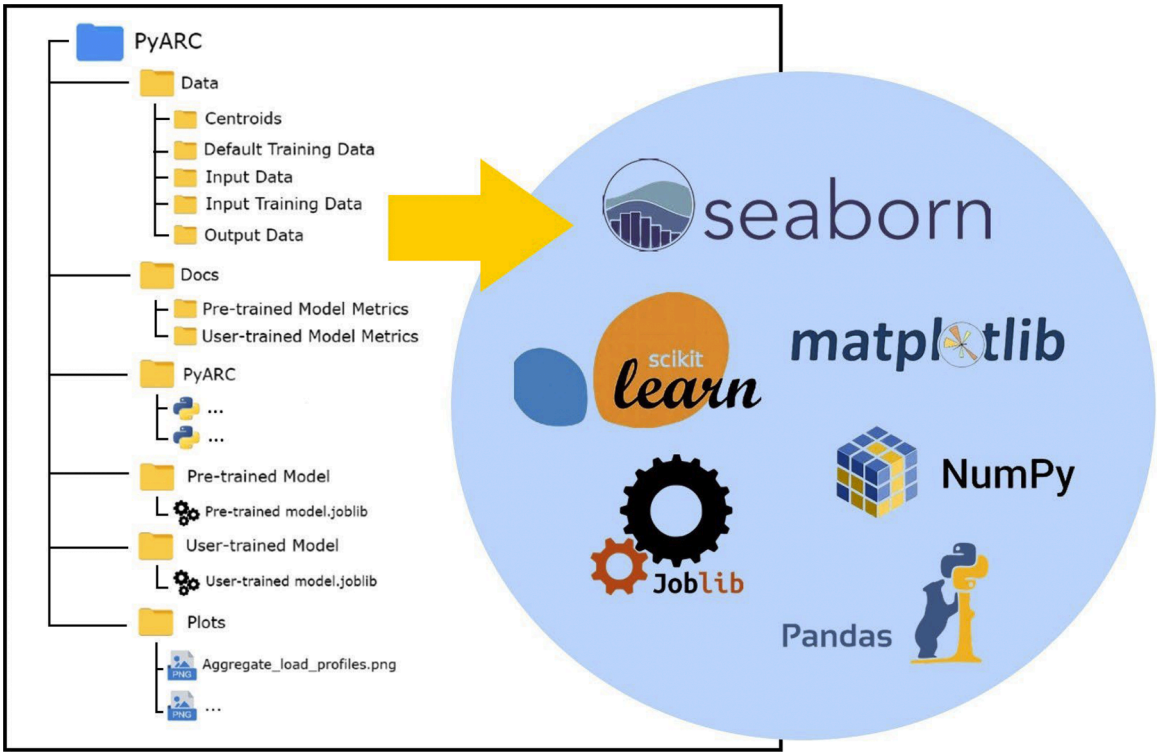

Fig. 2. PyARC repository structure.

```
def _train_model(self):
  rf_model = RandomForestClassifier(criterion = 'gini',
    max_depth = None, random_state = 42)
  param_grid = {
    'n_estimators': [50, 500],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [20, 30, 40, 80],
    'max_features': [1, 2, 3, 4, 5]
  }
  grid_search = GridSearchCV(estimator = rf_model,
    param_grid=param_grid, cv=5, scoring='accuracy')
  grid_search.fit(self.X_train, self.y_train)
self.model = grid_search.best_estimator_
```

exhaustive exploration of hyperparameter combinations, training multiple Random Forest models, and evaluating their performance using accuracy. The fitting method is run on the "GridSearchCV" object, initiating the training process with different hyperparameter combinations on the provided training data. The best-performing model is identified by extracting the "best_estimator_" attribute from the "GridSearchCV" object. This attribute holds the Random Forest model with the hyperparameters that gave the highest accuracy during cross-validation. Finally, the selected best model is stored for future use, accessible via the "self.model" attribute. This contains the entire process of hyperparameter tuning and model training, ensuring the optimized Random Forest model based on the provided dataset. In summary, the outputs are the Random Forest classifier and the normalized cluster centroids, followed by other secondary outputs, such as plots and the classifier's performance metrics.

- Reconstruction using User-trained Model – This process uses the `user_trained_model()` function to reconstruct load profiles using the user-trained model. This function uses the user-trained classification model and the normalized cluster centroids obtained using the `train_model()` function. The input data are the monthly electrical bills and ToU, which are loaded using specific CSV handlers. The function performs feature extraction from the input to perform the classification task using the pre-trained model to identify load consumption patterns for each user on a monthly basis. Subsequently, the function proceeds to rescale the obtained normalized load profiles using user-specific rescaling factors and aggregate them obtaining the aggregated electrical consumption profiles. In detail, the rescaling coefficients are evaluated in two steps:
  - Identify the most recurrent ToU timeframe during daylight hours;
  - Calculate user-specific rescaling coefficients.

In the first step, the most recurrent ToU during the central hours of the day is identified. In the second step, the scaling coefficient is calculated according to Eq. (1).

$$s_{i,m} = \frac{ToU_{el.\ cons.,\ i,m}}{c_{i,m} * E_m} \tag{1}$$

where:

- $ToU_{el.\ cons.,i,m}$ is the electrical consumption in the most recurrent ToU for *i th* user and the *m-th* month;
- $c_{i,m}$ is the sum of the centroids value of the *i th* user in the *m-th* month, during the most recurrent ToU timeframe;
- E_m is the temporal extension, expressed in hours, of the most recurrent ToU in one day.

The rescaling coefficient calculated according to Eq. (1) has been proven to be highly efficient in [1] and [9], obtaining high-performance evaluation metrics, such as the Normalized RMSE (NRMSE) and the Normalized MAE (NMAE). The aggregated load profiles are then exported in a CSV file and saved in the software's directory.

- Reconstruction using Pre-trained Model – This process executes the `reconstruct_profile()` function to reconstruct profiles using the pre-trained model. The process workflow is the same as the `user_trained_model()` function, but the pre-trained model is used instead of the user-trained model. The model is obtained through a training process made with publicly available data, specifically the hourly electricity consumption of users located in London with a time window of one year [7] and their ToU related to a weekday, which is summarized in Table 1 and stored in the Default Training Data directory.

2.4. Software usage

To successfully run the PyARC software, some preliminary actions are needed.

- Installing Required Libraries – open the terminal in the Python terminal and execute the following command:

```
pip install -r requirements.txt
```

This command will automatically install all the dependencies listed in the requirements.txt file.

Table 1
ToU frame used for the pre-trained model.

ToU timeframe name	Hours
Red	11:00–14:00 16:00–19:00
Amber	07:00–11:00 14:00–16:00 19:00–23:00
Green	00:00–07:00 23:00–24:00

■ Verification of installed libraries – After executing the installation command, it is crucial to verify that all the necessary libraries have been successfully installed. This ensures that the PyARC software will have access to the required components for seamless execution.

You can confirm the successful installation by checking the list of installed packages in your Python environment.

■ Input requirements – Input data directories must be populated by CSV files for each of the software functionalities.
- Reconstruct profiles using the pre-trained model – a CSV file named "data.csv" must be placed in the Input Data directory. The data must be structured as follows:

where:

- "User": is a char/string which contains usernames;
- "Year": is a numerical value that represents the year when electrical energy was measured;
- "Month": is a numerical value that represents the months when electrical energy was measured;
- "ToU1, ToU2, ToU3": contains the monthly electrical energy bills expressed in kWh and divided based on the ToU. The column name must be the same as the name of the ToU frames expressed in the "tou.csv" file.

Additionally, a CSV file named "tou.csv" must be placed in the Input Data directory. The data must be structured as follows:

where:

- "Hour": is a numerical value that represents the Hour in which the ToU is divided;
- "ToU": contains the Time of Use subgroup names expressed as char/string.

The software is flexible in handling input data that contains more or less than three ToU frames or that are called with different names.

- Model Training – A CSV file named "train_data.csv" must be placed in the Input Training Data directory. The data must be structured as follows:

where:

- "User": is a char/string which contains usernames;
- "Year": is a numerical value that represents the year when electrical energy was measured;
- "Month": is a numerical value that represents the months when electrical energy was measured;
- "Day": is a numerical value that represents the number of days when electrical energy was measured;
- "Hour": is a numerical value that represents the hours when electrical energy was measured;
- "Consumption": contains electrical energy values expressed in kWh.

Additionally, a CSV file named "train_tou.csv" must be placed in the Input Data directory. The data must be structured as follows the "tou.csv" required for the Reconstruct Profiles process.

- Reconstruct profiles using the user-trained model – a CSV file named "data.csv" must be placed in the Input Data directory, with the same structures described for the Reconstruct Profiles process using the pre-trained model.

Finally, to run the algorithm it's necessary to follow the following steps:

■ Run `main.py`;

The following message will be shown in the command line.

```
Welcome to PyARC! What would you like to do?
  1. Reconstruct Residential Aggregate Electrical Load Profiles using the pre-trained model
  2. Train a new model
  3. Reconstruct Residential Aggregate Electrical Load Profiles using the user-trained model
```

■ Choose an option (1, 2, or 3) using the command line based on the desired action.
1) Reconstruct profiles using the pre-trained model;
2) Train a new PyARC model;
3) Reconstruct profiles using a user-trained model.

After selecting an option, the software initiates, messages tracking the progress of the software processes are progressively displayed in the command line, and output files are generated.

2.5. Software applicability

The underlying assumption of our approach is that there are typical load profiles that describe the hourly consumption patterns of users with certain characteristics. These typical profiles can be identified and classified using clustering methods. Employing a clustering process on normalized load profiles enables the detection and classification of users with diverse characteristics, even in more heterogeneous datasets. As user characteristics become more diverse, the clustering process becomes more adept at distinguishing between these types of users. In our previous work we demostrate that these typical profiles can be identified and classified using clustering methods [1]. However, a challenge when dealing with highly heterogeneous datasets is that the classifier may have difficulty effectively recognizing and categorizing a wide range of

```
User,Year,Month,ToU1,ToU2,ToU3
MAC000016,2012,1,32.27,42.49,9.63
MAC000016,2012,2,7.84,9.50,5.58
MAC000016,2012,4,3.45,4.50,3.31
MAC000016,2012,5,2.83,4.06,3.46
MAC000016,2012,6,3.50,4.82,2.82
…
```

```
Hour,ToU
0,ToU3
1,ToU3
2,ToU3
3,ToU3
4,ToU3
5,ToU3
6,ToU3
7,ToU2
8,ToU2
9,ToU2
10,ToU2
11,ToU1
12,ToU1
13,ToU1
14,ToU2
15,ToU2
16,ToU1
17,ToU1
18,ToU1
19,ToU2
20,ToU2
21,ToU2
22,ToU2
23,ToU3
```

```
User,Year,Month,Day,Hour,Consumption
"MAC000016",2012,1,1,0,0.0275
"MAC000016",2012,1,1,1,0.0735
"MAC000016",2012,1,1,2,0.0395
"MAC000016",2012,1,1,3,0.0305
...
```

user types. Nevertheless, the use of metrics such as the Davies-Bouldin index and the Silhouette score mitigates the risk of selecting an excessive number of clusters, thereby improving the classifier's performance in classification tasks and making it possible to use the software with heterogeneous data sets as well.

3. Illustrative examples

In this section, the process that was undertaken to obtain the pre-trained model is described, and the reconstruction of the load profiles of an aggregate composed by six residential users randomly chosen from data described in [11] is shown, as an example of the software functionalities application. To obtain the pre-trained model, the `train_model()` function was used running the main and choosing option 2 from the command line (Train a new PyARC model), using the data described in [11] and the ToU defined in Table 1 by moving the data from the Default Training Data to the Input Training Data folder, and correctly renaming the files in "train_data.csv" and "train_tou.csv".

```
Welcome to PyARC! What would you like to do?
  1. Reconstruct Residential Aggregate Electrical Load Profiles using the pre-trained model

  3. Reconstruct Residential Aggregate Electrical Load Profiles using the user-trained model
```

As described in Sections 2.1 and 2.2, a K-means clustering was performed to identify the normalized typical load patterns as centroids of the obtained clusters. Then, a Random Forest is trained to build a model capable of detecting the load patterns of residential users based on features extracted from monthly electricity bills. The obtained cluster centroids (Fig. 3) are then saved in the software's folder.

As stated in Section 2, once cluster centroids were obtained, training features were extracted from the input data, and the classification model

```
2. Train a new model
```

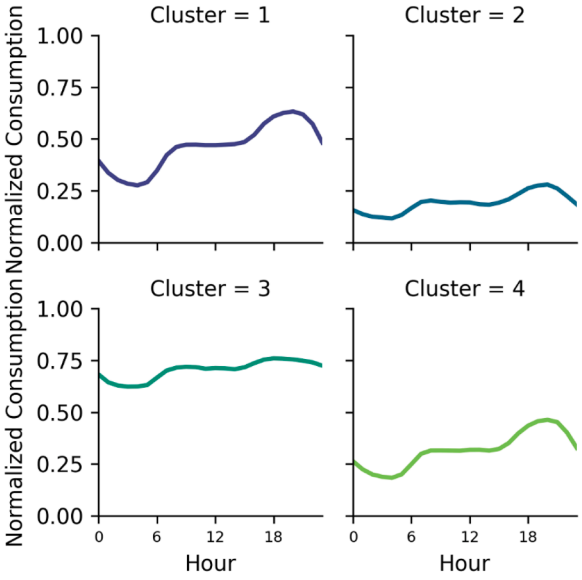

Fig. 3. Cluster Centroids were obtained during the pre-trained model training.

was trained. The obtained classification model performance is summarised in Table 2. In this work, classification performance metrics are evaluated as the classifier's ability to successfully assign a user to its correct cluster, both for training and testing set (70% and 30% of the input data respectively). These metrics are exensively described in our previous work [1]. Feature importance is presented in Fig. 4, where a ranking of the most useful features used for the classification task is shown.

The figure shows that among the selected features, the most relevant for the pre-trained classification model was the total monthly consumption, the consumption in the three different ToU periods, and then the ratio between these values.

As previously stated, the entire process performed to create the pre-trained model can be repeated to create the user-trained model from different input data using the `train_model()` function, resulting in different cluster centroids, optimal number of cluster values, and classification performance metrics. As mentioned above, the `user_trained_model()` function must be used to use the user-trained model.

After obtaining the pre-trained model, an example of the software application is made using monthly data extracted from some randomly selected users from the data described in [11]. The pre-trained model is used via `reconstruct_profiles()` to reconstruct the aggregate load profiles by running the main and choosing from command line option 1 (Reconstruct Residential Aggregate Electrical Load Profiles using the pre-trained model), and by using the input data correctly named and placed in the Input Data folder, following data requirements described in Section 2.

```
Welcome to PyARC! What would you like to do?
    2. Train a new model
    3. Reconstruct Residential Aggregate Electrical Load Profiles using the user-trained model
```

As previously stated, the aggregate load profiles on a monthly basis (Fig. 5) are obtained by combining the classification model and the user-specific rescaling coefficient calculated through Eq. (1).

For this specific example, data corresponding to March are missing for the reasons given in [1] and [11], namely the maintenance of smart meters.

Table 2
Classification performance metrics of the pre-trained model.

Classification performance metrics	Value
Accuracy (train)	90.44%
Accuracy (test)	89.37%

4. Impact

As mentioned in the introduction of this research work, most articles on load profiling rely on high temporal resolution data enriched with additional information on appliances, the number of occupants, their characteristics, and habits. In contrast, this research proposes a software that does not rely on such information-rich datasets, which are often difficult to obtain in real-world contexts. This topic represents an emerging novelty in research, as opposed to the abundance of load profiling methodologies that focus on reconstructing profiles with high temporal resolution for individual users. Having accurate estimates of electrical consumption within an aggregate allows for more robust economic and energy assessments during feasibility studies. For example, the proposed model could be used to support the feasibility studies of Renewable Energy Communities (RECs), helping potential aggregators and stakeholders involved in the promotion and creation of RECs in local areas, even in countries where the deployment of smart meters has reached a high penetration rate, as shown in [1], where a similar code structure was used. Indeed, even in these countries, there is still a lack of publicly available and accessible data that could be used to improve the estimation of energy flows and thus the economic assessment within a REC. As an example, in [12,13] and [14] hourly or sub-hourly data for energy is required for the simulation and optimization of RECs. PyARC allows for analysis and studies through simulation or optimization methods even in contexts in which hourly or sub-hourly data are not easily available. In addition, in real-world scenarios where hourly or sub-hourly data is typically not available, one of the most commonly used methods for estimating electrical load profiles is to use standard profiles, often obtained from large-scale statistical analyses. These profiles, which typically represent the average behavior of a large number of users, often do not accurately represent users in more localized geographic areas. Since aggregates are local aggregations of users, and therefore not geographically extensive, it is plausible to expect that the electrical load of users within a generic aggregate will generally be different (in some cases very different) from the standard load profiles that might be used. Consequently, the use of such profiles, which in most cases consist of one or two profiles per user category (e.g. residential, industrial), would lead to significant errors in the estimation of electricity consumption. On the contrary, the software uses typological profiles, which can be representative of specific users, especially those directly belonging to the aggregate or similar users. Therefore, the typological profiles identified by the model are highly representative of the user behavior within the aggregate and are more numerous than the standard profiles obtained through statistical analysis. As stated in [1], where the PyARC methodology is applied in the context of RECs. In this work, PyARC demonstrated to achieve competitive performances compared to some similar models, such as [10] which is the model with the most similar purpose to those of PyARC among the models in the literature, in terms of Normalized Absolute Mean Error (NMAE) and Normalized Root Mean Squared Error (NRMSE) evaluated between the

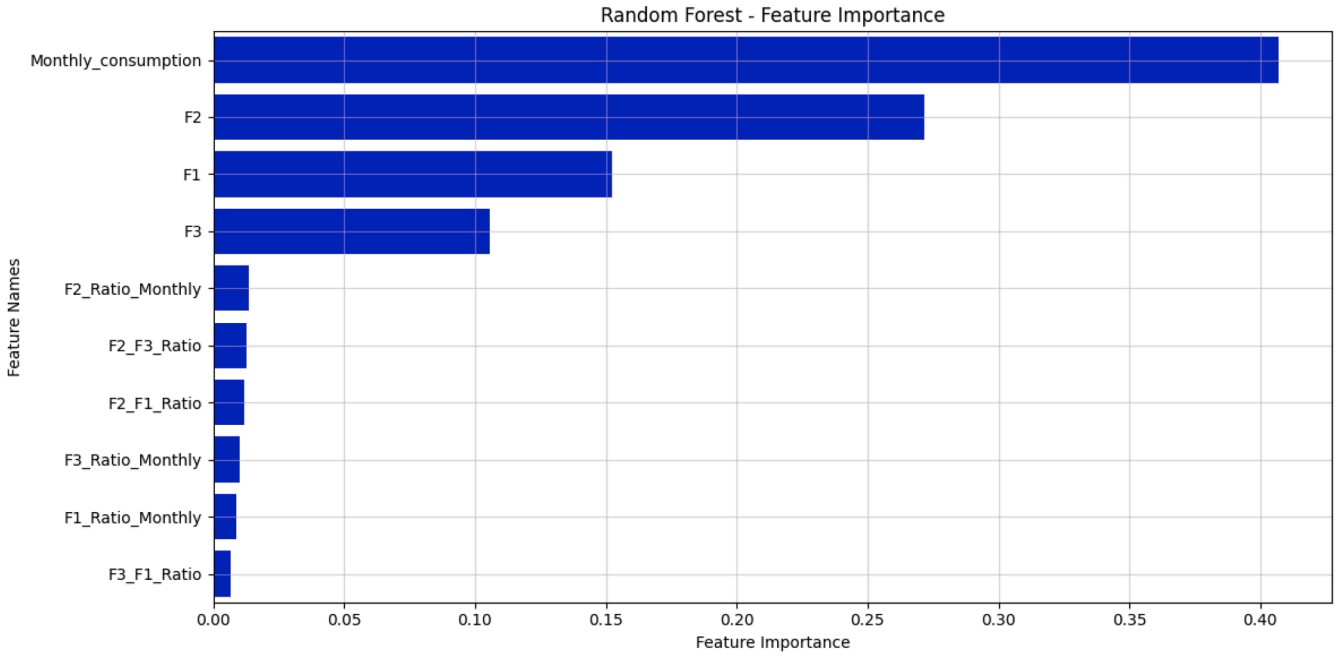

Fig. 4. Feature importance ranking for the pre-trained model.

```
1. Reconstruct Residential Aggregate Electrical Load Profiles using the pre-trained model
```

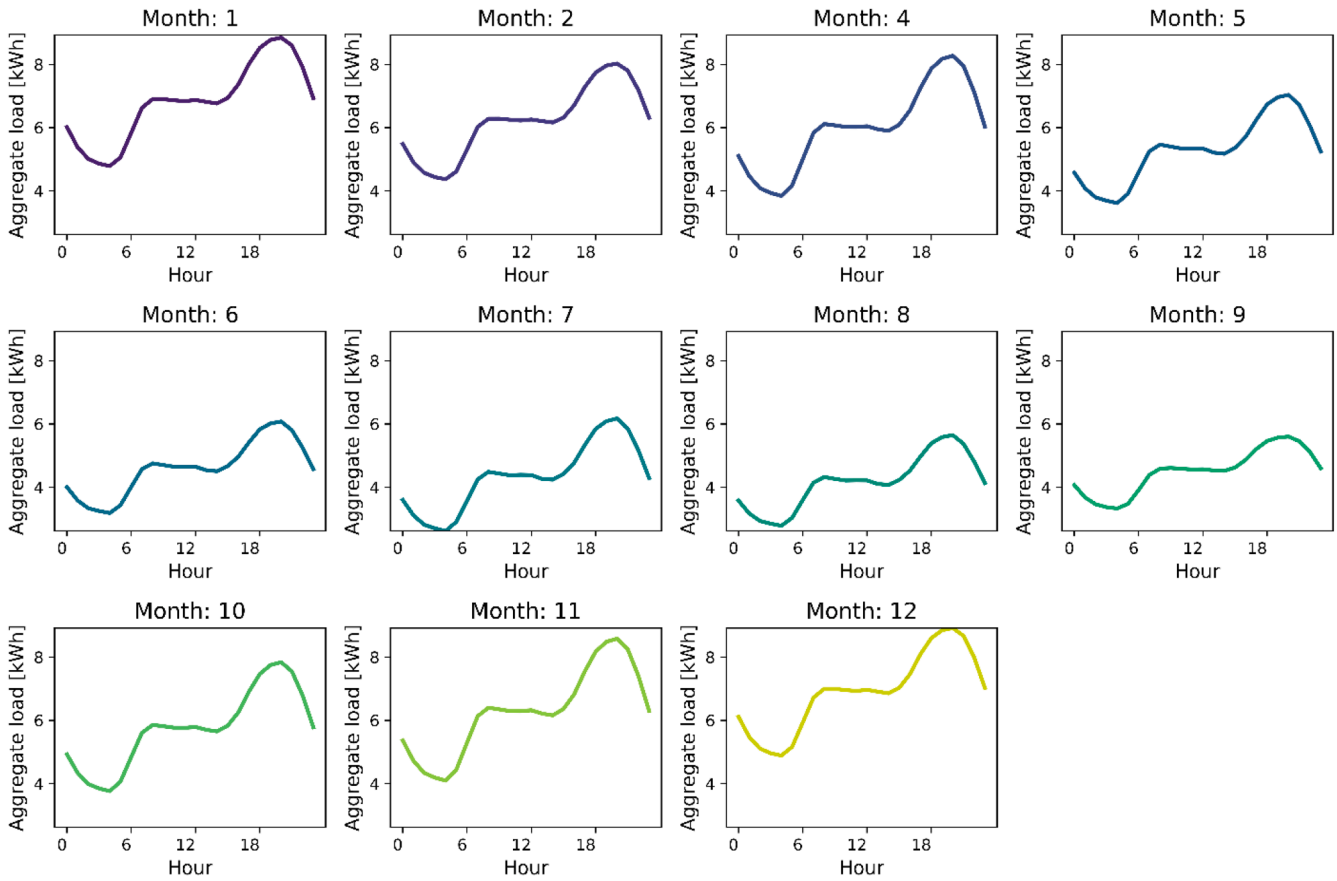

Fig. 5. Aggregated electrical load profiles for each month.

simulated load profiles and the real ones.

5. Conclusions

Load profiling for residential aggregates faces challenges arising from data scarcity and the limitations of standard profiles derived from statistical analysis. To address this issue, our research introduces PyARC, a Python software that can reconstruct the aggregated electrical load profiles for different residential users through easily obtainable consumption data. Unlike traditional methods that rely on rich data sets, PyARC offers a practical solution that applies to real-world scenarios and provides robust estimates of electricity consumption within generic residential aggregates. The software has promising implications for supporting feasibility studies, for example in the context of Renewable Energy Communities (RECs), where accurate energy flow assessments are critical for economic evaluations. Additionally, PyARC could be a valuable tool for stakeholders involved in the promotion and creation of RECs, smart grids, and aggregates in general, offering a way to achieve more accurate energy estimates for economic assessments, even in countries with advanced smart meter deployments. Possible future improvements to the software could include the ability to customize the training and testing data rates at the model-building stage, or the ability to choose from a pool of different alternative models to the proposed one. In addition, another improvement could be the choice of clustering type when identifying typical load profiles, so that the optimal cluster type can be chosen according to the characteristics of the dataset

CRediT authorship contribution statement

Lorenzo Giannuzzo: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Daniele Salvatore Schiera:** Writing – review & editing, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Francesco Demetrio Minuto:** Supervision, Funding acquisition, Conceptualization. **Andrea Lanzini:** Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my code at the Attach File step

Acknowledgments

A. Lanzini, D.S. Schiera, and L. Giannuzzo carried out this study within the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No 1561 of 11.10.2022 of Ministero dell'Università e della Ricerca (MUR); funded by the European Union – NextGenerationE. Award Number: Project code PE0000021, Concession Decree No 1561 of 11.10.2022 adopted by Ministero dell'Università e della Ricerca (MUR), CUP - to be indicated by each Beneficiary, according to attachment E of Decree No 1561/2022, Project title "Network 4 Energy Sustainable Transition – NEST.

F.D. Minuto carried out this study within the Ministerial Decree No 1062/2021 and received funding from the FSE REACT-EU - PON Ricerca e Innovazione 2014–2020.

This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

[1] Giannuzzo L, Minuto FD, Schiera DS, Lanzini A. Reconstructing hourly residential electrical load profiles for renewable energy communities using non-intrusive machine learning techniques. Energy and AI 2024;15:100329. https://doi.org/10.1016/j.egyai.2023.100329.

[2] Wang Yi, Chen Qixin, Kang Chongqing, Zhang Mingming, Wang Ke, Zhao Yun. Load profiling and its application to demand response: a Review. Tsinghua Sci. Technol. 2015;20(2):117–29. https://doi.org/10.1109/tst.2015.7085625.

[3] Tarmanini C, Sarma N, Gezegin C, Ozgonenel O. Short term load forecasting based on Arima and ann approaches. Energy Rep 2023;9:550–7. https://doi.org/10.1016/j.egyr.2023.01.060.

[4] Thorve S, Baek YY, Swarup S, Mortveit H, Marathe A, Vullikanti A, Marathe M. High resolution synthetic residential energy use profiles for the United States. Sci Data 2023;10(1). https://doi.org/10.1038/s41597-022-01914-1.

[5] Yan L, Tian W, Wang H, Hao X, Li Z. Robust event detection for residential load disaggregation. Appl Energy 2023;331:120339. https://doi.org/10.1016/j.apenergy.2022.120339.

[6] Chuan L, Ukil A. Modeling and validation of electrical load profiling in residential buildings in Singapore. In: 2015 IEEE Power & Energy Society General Meeting; 2015. https://doi.org/10.1109/TPWRS.2014.2367509.

[7] Osman M, Ouf M, Azar E, Dong B. Stochastic bottom-up load profile generator for Canadian households' electricity demand. Build Environ 2023;241:110490. https://doi.org/10.1016/j.buildenv.2023.110490.

[8] Alrawi O, Bayram IS, Al-Ghamdi SG, Koç M. High-resolution household load profiling and evaluation of rooftop PV systems in selected houses in Qatar. Energies 2019;12:3876. https://doi.org/10.3390/en12203876.

[9] Piscitelli MS, Brandi S, Capozzoli A. Recognition and classification of typical load profiles in buildings with non-intrusive learning approach. Appl Energy 2019;255:113727.

[10] Lazzeroni P, Lorenti G, Repetto M. A data-driven approach to predict hourly load profiles from time-of-use electricity bills. IEEE Access 2023;11:60501–15. https://doi.org/10.1109/access.2023.3286020.

[11] Schofield J.T., Carmichael R., Tindemans S.H., Bilton M., Woolf M., Strbac G. Low carbon London project: data from the dynamic time-of-use electricity pricing trial, 2013 2016. https://doi.org/10.5255/ukda-sn-7857-2.

[12] Fina B, Monsberger C, Auer H. Simulation or estimation?—two approaches to calculate financial benefits of energy communities. J Clean Prod 2022;330:129733. https://doi.org/10.1016/j.jclepro.2021.129733.

[13] Lazzari F, Mor G, Cipriano J, Solsona F, Chemisana D, Guericke D. Optimizing planning and operation of renewable energy communities with genetic algorithms. Appl Energy 2023;338:120906. https://doi.org/10.1016/j.apenergy.2023.120906.

[14] Weckesser T, Dominković DF, Blomgren EMV, Schledorn A, Madsen H. Renewable energy communities: optimal Sizing and distribution grid impact of photo-voltaics and Battery Storage. Appl Energy 2021;301:117408. https://doi.org/10.1016/j.apenergy.2021.117408.