eptk: Energy Prediction Toolkit

Hardik Prabhu
CloudAEye, Inc.
India
hardik.prabhu@gmail.com

Pandarasamy Arjunan
Berkeley Education Alliance for Research in Singapore (BEARS) Limited
Singapore
samy@bears-berkeley.sg

ABSTRACT

Building energy use prediction plays a crucial role in whole building energy management. In recent years, with the advent of advanced metering infrastructures that generate sub-hourly energy meter readings, data-driven energy prediction models have been implemented by leveraging advanced machine learning algorithms. However, the lack of standardization of model development and evaluation tools hinders the advancement and proliferation of data-driven energy prediction techniques on a large scale. This paper presents *eptk*, an open-source toolkit that enables the seamless development of data-driven energy prediction models. The proposed toolkit helps researchers and practitioners to easily benchmark the existing and new data-driven models on various open-source datasets containing time-series of multiple energy meter data along with relevant metadata. Using the toolkit, we develop and compare the performance of 34 models on two large datasets containing more than 3,000 smart meter readings. *eptk* will be released in open-source for community use.

CCS CONCEPTS

• **General and reference** → **Evaluation**; **Metrics**; • **Computing methodologies** → **Cross-validation**; **Model development and analysis**.

KEYWORDS

Energy prediction, Building energy management, Advanced Metering Infrastructures, and Machine learning

ACM Reference Format:
Hardik Prabhu and Pandarasamy Arjunan. 2022. *eptk*: Energy Prediction Toolkit. In *The 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '22), November 9–10, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3563357.3567410

1 INTRODUCTION

In recent years, data-driven energy prediction models have been extensively studied to improve the operational efficiency of modern buildings, orthogonal to the traditional physics-based or simulation models [4, 5]. These data-driven models are reported to be more efficient because they can map the complex non-linear relationship between a building's energy use and physical and operational characteristics [3, 6]. Despite the increasing amount of research in data-driven energy prediction, it is difficult to compare different models against each other due to the lack of standardized model development tools [1]. It is also widely acknowledged that the availability of various open-source software libraries and tools enabled the development and evaluation of comprehensive models[1].

Motivated by the aforementioned challenges and opportunities, in this paper, we present the design and development of *eptk* – a Python package that enables seamless development of large-scale energy prediction models and benchmarking them on various open source data sets. *eptk* provides reusable and extensible modules to implement various functions such as (i) integration of new and existing energy meter data sets, weather, and other building metadata into a standardized format, (ii) data pre-processing modules for data cleaning and imputation, (iii) a systematic feature engineering module supporting various temporal and aggregation features, and (iv) model implementation and validation strategies using different performance measures. The proposed toolkit offers a new forward-chaining cross-validation technique suitable for energy time series and post-processing ensembling techniques to combine the predictions from two or more models. Using the *eptk* toolkit, we implemented 34 energy prediction models (17 models each on two in-built data sets) and compared their performance. *eptk* is released in open-source[2].

2 ENERGY PREDICTION TOOLKIT

This section presents the architecture and design of the proposed *eptk* toolkit. Figure 1 illustrates the workflow of the *eptk* toolkit and the interconnection of various modules.

2.1 Datasets and Data converters

The first step of energy prediction model development involves loading and parsing available energy time series data and other metadata into a standard format to apply the downstream data processing and analytical functions. *eptk* supports Hierarchical Data Format (HDF5), which is highly flexible for storing and managing arbitrary metadata for each energy meter time series. Suitable data converter modules are provided in *eptk* to convert the energy data in other formats, such as .csv, into a standard format. In the current implementation, the *eptk* toolkit comes with two large datasets: ASHRAE - Great Energy Predictor III [1] and Building Data Genome 2 (BDG2) [2].

The subpackage `eptk.dataset` contains a base class for all the dataset objects. It is an abstract class that standardizes how datasets are downloaded and loaded. All the dataset object inheriting the

BuildSys '22, November 9–10, 2022, Boston, MA, USA

[1] https://energy.acm.org/resources/
[2] https://www.eptk.org/

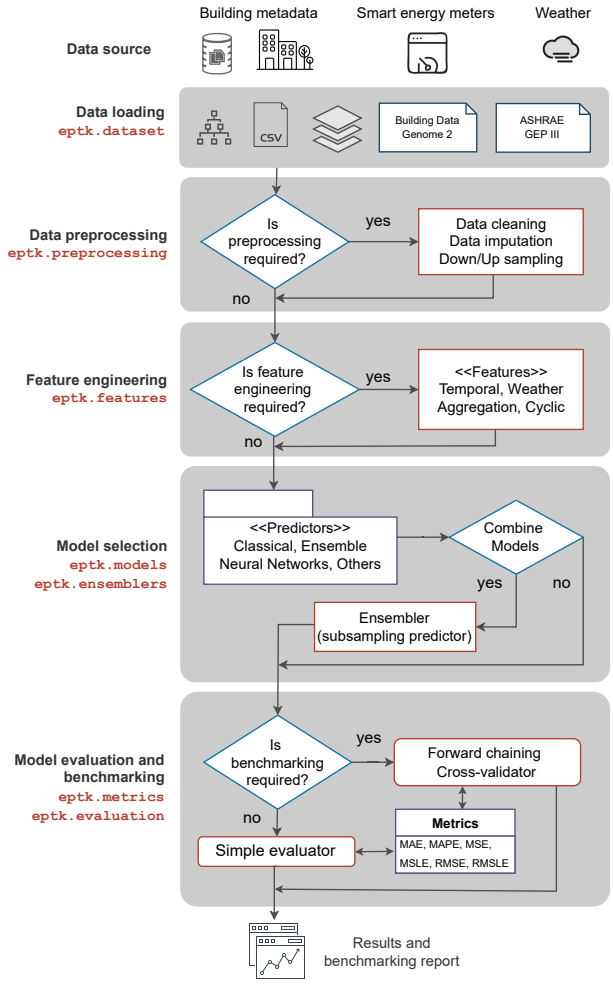

Figure 1: Energy prediction model development using *eptk*.

base class contains a load method that finally returns a data frame(s) for the dataset. While loading, the datasets are first downloaded in the working directory if not found. The data frame returned at the end has standard naming conventions for each attribute. The subpackage *eptk.dataset* aims to allow seamless integration of a large dataset on which the downstream model development workflow could be performed.

2.2 Preprocessing

Since data quality is essential to develop robust energy prediction models, `eptk.preprocessing` provides a collection of methods for data cleaning, imputation, and transformation.

2.2.1 Data cleaning. Data cleaning includes detecting and removing corrupt or inaccurate records from a dataset. The module clean contains methods such as eliminating missing meter readings, removing meter reading values beneath a threshold, removing constant readings, etc.

2.2.2 Data Imputation. For various reasons, many real-world energy meter datasets contain missing values. For instance, in the ASHRAE dataset, the meter readings are recorded periodically, but the weather data has a lot of missing timestamps. To address this, the impute module contains methods such as `add_missing_timestamp` and `impute_weather`.

Moreover, it is straightforward to include additional data cleaning and imputation functions to the *eptk.preprocessing* subpackage.

2.3 Feature engineering

Feature engineering is a process where we leverage domain knowledge to transform the raw data, giving us a better representation of the information well suited for the machine learning task. For example, the temporal features derived from the timestamps alone, such as the hour of the day, will give a better insight into energy consumption. The features supported by *eptk* can be broadly categorized into the following categories. The package provides a vast array of feature engineering methods from `eptk.features` module.

2.3.1 Temporal features. The *eptk* package provides methods to extract some useful temporal features. These are listed below.

(1) Month of the year (Int 1-12)
(2) Day of the week (Int 1-7)
(3) Hour of the day
(4) Year
(5) IsHoliday (Binary 0,1)
(6) Cyclic encoding of the periodic features

The features, such as the *day of the week*, are periodic. In addition to assigning each *day of the week* with an integer value ranging from 1-7, the package also provides an option to encode the periodic feature in a cyclic coordinate system. Whereas the feature *IsHoliday* expects that the geographical location of the building is passed along as well. It could be in the form of the country of origin, longitude-latitude, or site-id, as in the case of the BDG2 dataset.

2.3.2 Weather features. It is well known that weather significantly impacts building energy use. The BDG2 dataset has weather features such as air temperature, dew temperature, and wind speed. In the presence of these features, the package provides methods to extract additional weather features such as relative humidity and temporal and cyclic features.

2.3.3 Aggregated Statistics. The *eptk* package provides methods to group the dataset and aggregate different statistics. The methods are in the form:

```
function_name(df,feature,group_by,kwargs**)
```

In order to create the aggregated values for different statistics, these methods require the data frame (df), feature column as input (feature), and a string or a list of features to split the dataset into different groups and other parameters (kwargs**). The statistics offered are given below.

(1) Min, Max, Percentile
(2) Mean, Median, Standard deviation
(3) Moving statistics over a window
(4) Periodic statistics

```python
# Imports
from eptk.dataset import BDG2
from eptk.preprocessing import *
from eptk.evaluation import CrossValidation
from eptk.metrics import *
from eptk.models.ensemble import *
# 1. Dataset loading
meter, weather, meta = BDG2.load()
# 2. Feature engineering
weather = add_temporal_features(weather, cyclic = True)
weather = include_holidays(weather)
dataset = merge_data(meter, weather, meta)
dataset = to_numeric(dataset, cat = ["primary_use"])
# 3. Prepare data for benchmarking
X, y = prepare_data(dataset, drop = ["building_id"])
# 4. Benchmarking using forward chaining in time
cv = CrossValidation(model= LightGBMPredictor(),
        metrics = root_mean_squared_log_error,
        time_bin_type="month", verbose= True)
results = cv.evaluate(X,y)
print(results)
```

Listing 1: Example Python code using *eptk* toolkit.

2.4 Model development

One of the primary design goals of *eptk* package is to provide an extensive collection of different models and ways to combine them to create powerful predictors and a standardized way to add more models to the existing collection. The module `eptk.models` contains various methods for implementing energy prediction models.

2.4.1 `eptk.models`. All eptk models inherit the base model class `BasePredictor`. It is consisting of the abstract methods `fit` and `predict` and additional useful methods such as `reset` and `load_params`. The `BasePredictor` class is to be inherited while defining new model classes. The models are divided into 4 categories.

(1) Classical - This includes models such as Linear Regression, Ridge Regression, Support Vector Regression, etc.
(2) Ensemble - This includes models such as Random Forest, LightGBM, Catboost, etc.
(3) Neural Networks - This includes models such as Feedforward Neural Network and 1-D CNN.
(4) Others - This includes models linear Generalized additive model and Generalized Linear models.

The model classes are designed in a way that one can extend them to create new model types. In addition, pre-trained models are also supported.

2.4.2 Ensembling. By producing only a single model over the entire dataset, getting accurate predictions may become challenging. If we combine multiple models, the overall accuracy could get boosted. The *eptk* package provides classes for developing combined model predictors. These are child classes of `BasePredictor`. It allows the predictor to be identical in terms of usage with the other *eptk* models with few added parameters, such as passing a list containing model objects as one of the inputs. These classes are listed below.

(1) `Ensembler`: This class enables combining multiple models over the entire dataset and returning an aggregate model.
(2) `SubsamplingPredictor` - This class enables creating different instances of the same model type over different subsets of the entire dataset and then returning an aggregate model.

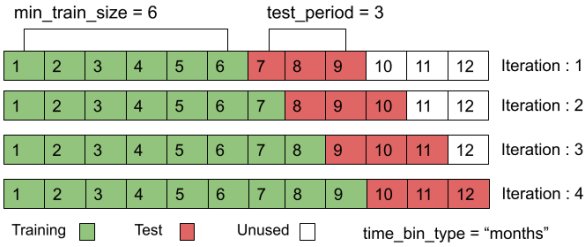

Figure 2: Illustration of forward-chaining cross-validation for a dataset containing 12 months of data.

The aggregation is done by taking the weighted sum of all the models.

$$f_{\text{combined}}(x) = W_1 * f_{\text{model1}}(x) + W_2 * f_{\text{model2}}(x) \ldots + W_n * f_{\text{modeln}}(x)$$

The weight vector $W = [W_1, W_2 \ldots W_n]$ can be set to be uniform, can be taken as an input from the user and also can be optimized automatically while training.

For optimizing the weight vector over the training phase, the training data is split into 2 parts, the models are trained on the first part and then the weights are assigned based on the weighted performance on the 2nd part (validation set). Once the weights are decided, the models are then trained over the complete training dataset and the weighted linear combination of the models is returned as the predictor. The package provides 3 different mechanisms for optimizing the weight vector.

(1) *Meta Regression:* After all the predictors are fitted, the coefficients of the ridge regression with no constant for the regression of the target over the predictions of each individual model over the validation set is taken as the weights.
(2) *Softmax over prediction accuracy:* After all the predictors are fitted, their performance is evaluated over the remaining validation set. The models are assigned weights based on the training performance. A numerically stable *softmax* function is used to assign weights. The negative of mean square error is taken as the input for each model weight for the softmax.
(3) *Bayesian Optimization:* The weights are optimized using Gaussian process regression. First the black box cost function over the weight space is created. All the weights are between 0 and 1. We take negative of mean square error of the weighted predictions with the actual target values as the function to maximize. After optimization we get the vector of weights.

2.5 Evaluation and Benchmarking

The *eptk* package provides various standard metrics for measuring the model performance and a time-based cross-validation to benchmark the model performance over the dataset.

2.5.1 Evaluation metrics. The module `eptk.metrics` offers a variety of standard evaluation metrics commonly used for comparing model performance in the energy prediction domain. The metrics offered are: (a) Mean Squared Error, (b) Root Mean Squared Error, (c) Mean Absolute Error, (d) Mean Absolute Percentage Error, (e) Mean Squared Logarithmic Error, and (f) Root Mean Squared Logarithmic Error.

Table 1: Comparison of performance metrics (lower is better) of 17 energy prediction models on BDG2 dataset (100 meters).

S.No.	Model	RMSLE	MAE
1	**Random Forest**	**0.279**	**2.793**
2	Catboost, LightGBM, and HistBoost [e]	0.319	2.873
3	CatBoost and LightGBM [e]	0.319	2.873
4	CatBoost	0.319	2.873
5	LightGBM	0.318	2.892
6	Histgradboost	0.320	2.907
7	LightGBM (subsampled) [e]	0.333	3.562
8	CatBoost, LightGBM, HistGrad & XGBoost [e]	0.458	3.914
9	Feed Forward Neural Net	0.531	4.199
10	CatBoost (subsampled) [e]	0.333	4.255
11	XGBoost	0.508	4.255
12	KNN	0.409	4.975
13	Linear Regression	0.802	7.078
14	GLM	0.550	9.363
15	Ridge Regression	1.277	12.714
16	Elastic Net Regression	1.526	19.043
17	Lasso Regression	1.765	23.418

[e] Weighted average ensembled models

2.5.2 Time-based Cross Validation. Although several open-source packages, such as `scikit-learn`, offer implementation for performing k-fold cross-validation, these implementations split the dataset randomly into k folds. There is a need for k-fold cross-validation for time series data to split the dataset into temporal bins such as months, weeks, etc. Further, the training process should maintain the chronological order of the train-test split at every iteration. The package offers an approach more appropriate for energy time series, known as forward-chaining (See Figure 2).

2.5.3 Benchmarking. A comprehensive evaluation of model performance is essential to develop robust energy prediction models. To benchmark the performance of one or many models on various data sets, the package provides a `CrossValidation` class within `eptk.evaluation` module. A `CrossValidation` object uses the `EPTKCrossValidator` to split the data into train and test over a series of iterations. The `CrossValidaton` object is to be provided with three parameters that govern the splitting process in each iteration. These are `time_bin_type`, `test_period` and `min_train_size`. A list of evaluation metrics is also passed as input. Over each iteration, the model is trained, predictions are made, and values of evaluation metrics are calculated. In the end, a list containing the aggregated values for all the metrics over all the iterations is returned. The time-based cross-validation, used for benchmarking, can resume evaluation from a stored checkpoint in the event of failure during runtime. Listing 1 shows an example code to benchmark a model on the BDG2 dataset using the *eptk* package.

3 EXPERIMENTS AND RESULTS

We conducted two experiments on the two in-built datasets using our implementation of *eptk* package. Since conducting the experiments on these two large datasets with more than 3,000 smart meter time series is computationally expensive, we randomly selected a subset of meters (100) from each dataset. After applying the required data cleaning and imputation functions, we prepared the final datasets with 40 and 42 features from BDG2 and ASHRAE GEP III. Next, we selected a candidate of 17 models (12 standalone and 5 ensembled) to compare their performance on both datasets. The comparison of 17 models' performance using RMSLE and MAE on the BDG2 dataset is shown in Table 1. The website www.eptk.org contains additional results. The experimental results show the tree-based algorithms' superiority in accurately predicting the meter readings. The neural network model is hard to configure and thus performed sub-optimally on default parameters, trained over 50 epochs. The Random Forest model scored the best in the MAE (2.793) and the RMSLE (0.279) metrics. Models such as GAM, SVR, and CNN exhausted the computational resources after a few iterations and weren't included in the table. Ensembling the tree-based models such as CatBoost, LightGBM, and Histboost performed marginally better in terms of MAE scores when compared with each one individually. The classical linear regression models performed poorly compared to the rest and are shown in the last rows.

4 CONCLUSIONS AND FUTURE WORKS

This paper presents the design and development of *eptk* – a python package for seamless development and benchmarking of data-driven energy prediction models. The proposed *eptk* toolkit offers reusable and extensible modules to extend the toolkit's features. The toolkit facilitates multiple stages of the development of the energy prediction model: preprocessing, feature extraction, model selection, ensembling, and benchmarking. Using a prototype implementation of the toolkit, we developed and compared the performance of 34 models across two in-built datasets. In the future, the toolkit will support additional features such as modules for exploratory data analysis, data visualization, and more complex energy prediction models with hyperparameter tuning and AutoML.

ACKNOWLEDGMENTS

This work is supported by the National Research Foundation of Singapore through a grant (#1645964) for the Singapore-Berkeley Building Efficiency and Sustainability in the Tropics (SinBerBEST) program.

REFERENCES

[1] Clayton Miller, Pandarasamy Arjunan, Anjukan Kathirgamanathan, Chun Fu, Jonathan Roth, June Young Park, Chris Balbach, Krishnan Gowri, Zoltan Nagy, Anthony D Fontanini, et al. 2020. The ASHRAE great energy predictor III competition: Overview and results. *Science and Technology for the Built Environment* 26, 10 (2020), 1427–1447.

[2] Clayton Miller, Anjukan Kathirgamanathan, Bianca Picchetti, Pandarasamy Arjunan, June Young Park, Zoltan Nagy, Paul Raftery, Brodie W Hobson, Zixiao Shi, and Forrest Meggers. 2020. The building data genome project 2, energy meter data from the ASHRAE great energy predictor III competition. *Scientific data* 7, 1 (2020), 1–13.

[3] Yue Pan and Limao Zhang. 2020. Data-driven estimation of building energy consumption with multi-source heterogeneous data. *Applied Energy* 268 (2020), 114965.

[4] Saleh Seyedzadeh, Farzad Pour Rahimian, Ivan Glesk, and Marc Roper. 2018. Machine learning for estimation of building energy consumption and performance: a review. *Visualization in Engineering* 6, 1 (2018), 1–20.

[5] Zeyu Wang and Ravi S Srinivasan. 2017. A review of artificial intelligence based building energy use prediction: Contrasting the capabilities of single and ensemble prediction models. *Renewable and Sustainable Energy Reviews* 75 (2017), 796–808.

[6] Zeyu Wang, Yueren Wang, Ruochen Zeng, Ravi S Srinivasan, and Sherry Ahrentzen. 2018. Random Forest based hourly building energy prediction. *Energy and Buildings* 171 (2018), 11–25.