Energy Disaggregation: Reducing Model Complexity for Increased Privacy
12/13/2020

1. Introduction

In the United States, the residential and commercial building sectors account for roughly 40% of the total energy consumption, greater than both the industry (32%) and transportation (28%) sectors [1]. This accounts for 40% of US carbon dioxide emissions. To decrease carbon dioxide emissions and reduce the effects of global warming, many research groups are finding ways to decrease energy consumption. Providing building owners with real-time power consumption data down to the appliance level through energy feedback cycles has been estimated to decrease power consumption by up to 12% [2]. Appliance level data can be obtained by installing metering equipment on every appliance of interest, but this method is considered intrusive in that installation invades the privacy and comfort of building owners. Additionally, the total cost to the building owner significantly increases as the number of interested appliances increases and increases even further if the metering equipment requires installation by an electrician.

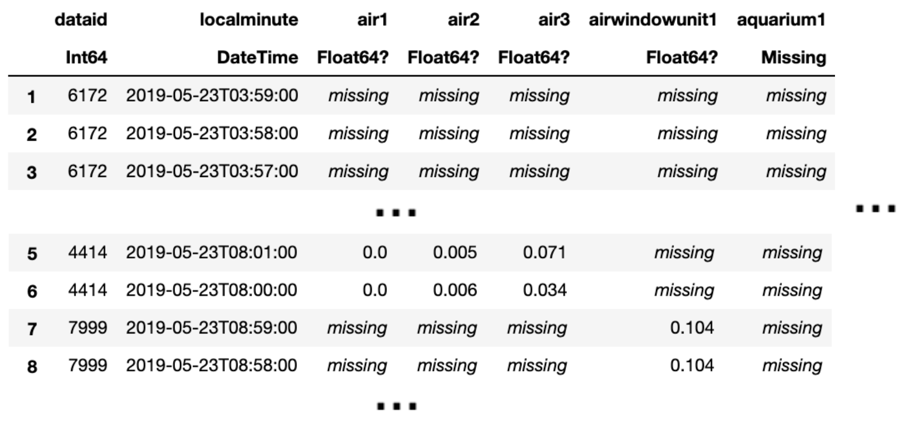
One promising method for obtaining appliance level data in a less intrusive and more cost-effective manner is through energy disaggregation, or Non-Intrusive Load Monitoring (NILM). NILM analyzes aggregate building power consumption data measured through utility-installed smart meters or other commercial metering devices and uses machine learning algorithms to make predictions on appliance level power consumption. Most NILM algorithms discussed in related literature rely on Deep Neural Networks (DNNs) to make their predictions, which are oftentimes complicated and computationally expensive. In most cases, disaggregation cannot be performed locally because of smart meter hardware constraints, requiring data to be analyzed remotely. This exchange of data introduces a risk of privacy invasion, potentially revealing home occupancy and appliance usage patterns to an attacker during the data exchange. In this study, we explore ways in which model complexity can be reduced so that energy disaggregation can be performed on local devices, ultimately improving user privacy.

2. Data Set

I have explored a subset of the Pecan Street Dataset through Dataport, that offers circuit, grid, and appliance-level power consumption data for over 1,000 volunteer homes across the United States. My dataset contained data for 64 single-family homes in Ithaca, New York as well as five surrounding cities. The dataset contained 79 features total, consisting of **real power** (or the power actually consumed due to the resistive load of each device) data in kiloWatts for **74 different home circuits and appliances** as well as a measure of the **grid power** supplied to each home and **two sources of solar power** generated by each home. Every column, or feature, within the dataset was measured at a 1-minute interval from May 1st through May 31st of 2019, with a time-stamp column indicating the time and date at which each power measurement was taken. Another column called "dataid" was included which allowed me to differentiate between each building. All data in both datasets were time series data, with no features (aside from “dataid”) containing categorical, text, or other data types. My goal is to evaluate the performance of separate prediction models for each appliance of interest for each home in my dataset.

3. Data Visualization and Cleaning

Before plotting any data, I decided to visualize the data in its raw form. **Table 1** shows example data from the dataset. Not every home provided time series data for every appliance. In fact, each home had their own unique combination of measured circuits and appliances, making uniformity amongst the homes challenging. Many appliance usage trends are unique to individual homes, making it difficult to impute data from one home based on another. For most cases, homes with appliances containing missing values provided no data at all for that appliance, and although imputing these values using an unsupervised learning method was considered, I ultimately decided not to. Additionally, I found that 24 features were completely missing, with none of the homes in my dataset providing any data for them. They were used to measure appliances existent in homes outside of my subset, so I removed these features, leaving 55 total features.



	dataid	localminute	air1	air2	air3	airwindowunit1	aquarium1	...
	Int64	DateTime	Float64?	Float64?	Float64?	Float64?	Missing	
1	6172	2019-05-23T03:59:00	*missing*	*missing*	*missing*	*missing*	*missing*	
2	6172	2019-05-23T03:58:00	*missing*	*missing*	*missing*	*missing*	*missing*	
3	6172	2019-05-23T03:57:00	*missing*	*missing*	*missing*	*missing*	*missing*	
			...					
5	4414	2019-05-23T08:01:00	0.0	0.005	0.071	*missing*	*missing*	
6	4414	2019-05-23T08:00:00	0.0	0.006	0.034	*missing*	*missing*	
7	7999	2019-05-23T08:59:00	*missing*	*missing*	*missing*	0.104	*missing*	
8	7999	2019-05-23T08:58:00	*missing*	*missing*	*missing*	0.104	*missing*	
			...					

Table 1. Example data from real-power dataset

Additionally, I found that only 52 of the 64 homes provided a full set of 44,640 rows, or examples, for the month. The remaining homes provided significantly less rows (as low as 57% of the full data set), which would cause difficulty in training an accurate model. For this reason, I decided to remove the 12 homes that provided an incomplete set from my dataset. I then divided my dataset into smaller sets based on “dataid” and produced time series plots for grid power provided to each home. **Figure 1** demonstrates sample time series plots for two homes.

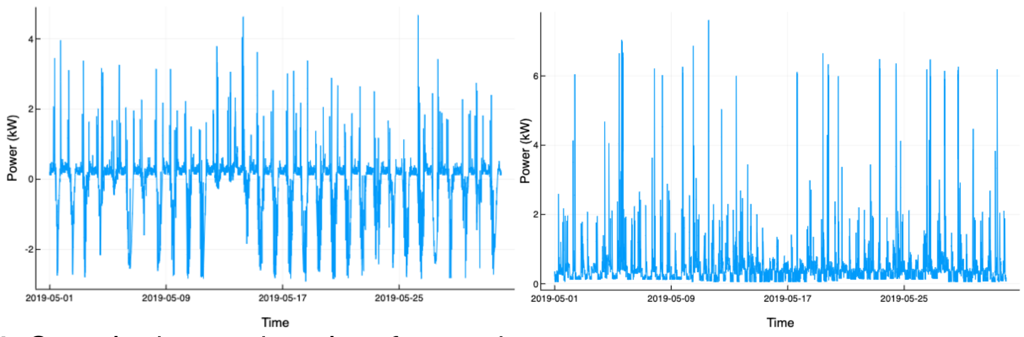


Figure 1. Sample time series plots for two homes

Observing the time series plots, I noticed that some homes followed a trend similar to that of the left plot in **Figure 1** containing negative grid power values while others followed the right, containing all positive values. A negative grid power would mean homes are providing power to the grid instead of consuming power. Realistically, the only instance of homes providing power to the grid would be through solar power generation. I deduced that it was necessary to add grid power with solar power to obtain the true aggregate power demand for each house. Doing so, I produced a new feature, “total_power” for each home that followed a trend similar to that of the right plot in **Figure 1** which no longer produced negative power values, following the expected trend.

For consistency amongst appliances, I wanted to perform the energy disaggregation task on appliances that met the following two criteria: 1) the appliances should be existent and measured in as many homes as possible and 2) the appliances should account for a significant contribution to the aggregate power consumption of homes. Criterion 1 was important to maximize the number of energy disaggregation models generated per appliance. Criterion 2 was important to maximize the benefit of the energy disaggregation models. Appliances with large power demands are typically the ones with the most potential for energy savings. I used **Figure 2** to visualize which appliances met my criteria. I then used element-wise multiplication to multiply the histogram vector by the average power consumption vector to find that “waterheater1”, “air1”, “drye1”, “refrigerator1” and “car1” were the features of greatest power consumption and abundance. I used the power consumption of these appliances as my labels for the energy disaggregation task.

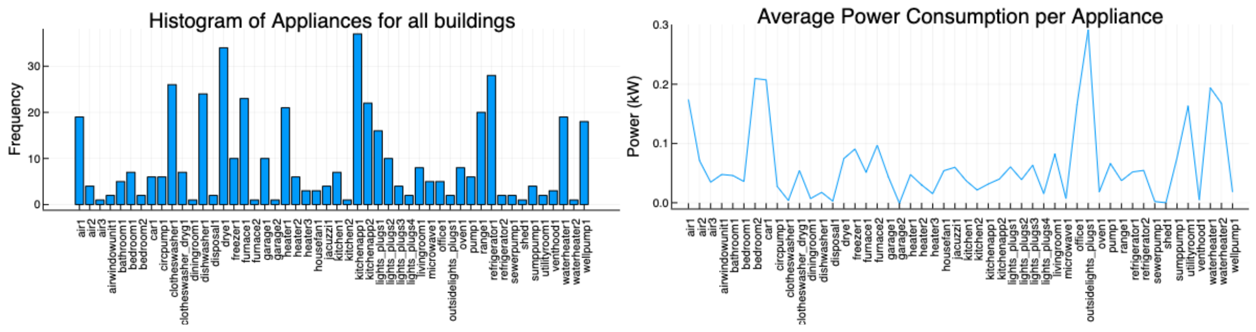


Figure 2. Histogram for all appliances in dataset (left) and average power consumption per appliance (right)

Energy disaggregation models are typically built on features limited to real and reactive aggregate total power, with the label being real power for a selected appliance. In this study, I considered the possibility of other appliances being a part of the feature space, since I thought there may be a correlation between devices such as a washer and dryer, or kitchen lights and the refrigerator. To visualize feature correlation, I plotted correlation matrix heatmaps for every home in my dataset, with example homes highlighted in **Figure 3**.

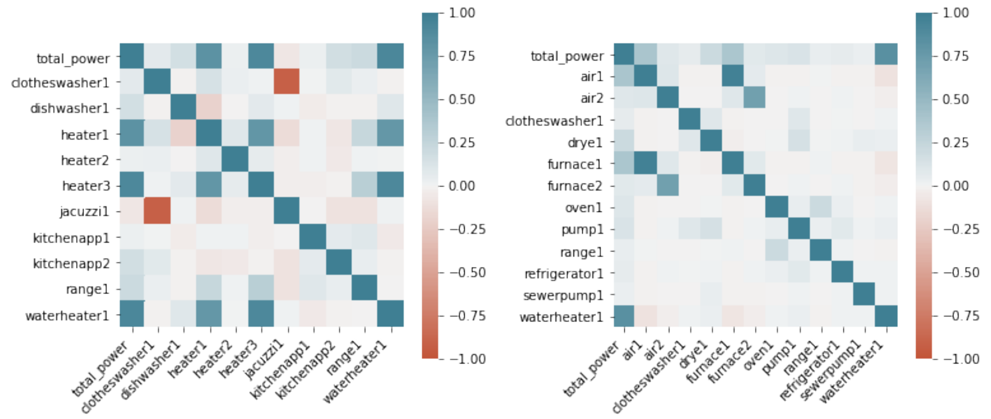


Figure 3. Correlation plots for two sample homes

From the correlation plots, I observed that there is in fact more correlation existing between features than simply between total_power and individual appliances, such as a strong correlation between “heater3” and “waterheater1” in the left plot and between “air1” and “furnace1” in the right plot. From this, I decided to manually choose appliances with strong correlations to my appliances of interest when creating feature spaces for each home.

4. Model Selection and Approach

Knowing my dataset contains continuous time series data, I decided to build a regression model for making predictions. I used linear regression with a quadratic loss function and no regularizer because it was straightforward and would serve as a good baseline. I let $X \in R^{nxd\_h}$ be my feature space for an individual home, consisting of the home’s total aggregate building power as well as its manually selected appliance features, denoted as d_h. I let $Y \in R^{n}$ be my training label, consisting of the interested appliance. I let $w \in R^{d}$ be my linear model coefficients generated using ordinary least squares. 80% of the dataset was used as my training set and 20% was used as my test set. My results are displayed in **Figure 4**.

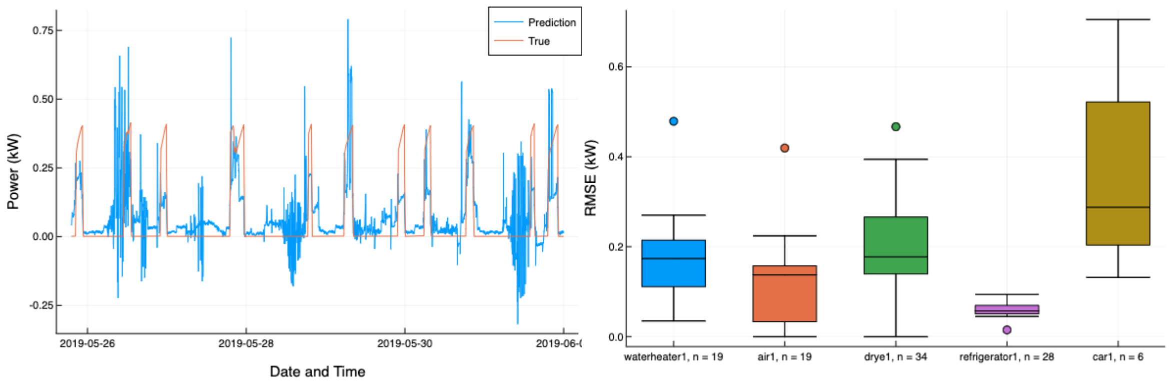


Figure 4. Example predictions for "waterheater1" on a single home (left) and a box plot of the root-mean-square error for all buildings and appliances

The left plot in **Figure 4** allows us to visualize the prediction accuracy of the ordinary least squares model. Because models were trained for 5 appliances for every home within my dataset, plotting all model fits in this way would be cumbersome and would not provide any insight into model performance. Instead, the right plot depicts model performance through root-mean-square error for all buildings and appliances of interest within my dataset. "refrigerator1" showed the least variance in model accuracy, consistently achieving a rather small root-mean-square error. "car1" showed the greatest variance and greatest error overall. These findings are discussed further in the Results and Discussion section.

Next, I tried various feature engineering approaches to try to improve the performance of the ordinary least squares model. Because I was using time series data and knowing that a correlation could exist between power measured at time "t" and time "t-1" or any other time in the past, I decided to use autoregressive features in addition to my ordinary least squares model. I used "t-1" and "t-2" as additional features for all appliances already in my feature space. I obtained slight improvements compared to the accuracy found using ordinary least squares regression, and for this reason decided not to include a box plot of the results.

I then decided to consider other loss functions that might help improve my model given the characteristics of my dataset and other regularizers to minimize overfitting. Before doing so, in an attempt to avoid model overfitting, I included a function that would incorporate cross-validation into the development of each of my models. Standard k-fold cross-validation does not work well with time series data because of the way the method is structured. K-fold cross-validation separates the training set into k segments and uses each of the k segments as the "temporary testing set" for its "temporary training set", consisting of the unused segments. The problem with this method is that the "temporary testing set" could contain data that occurred before the "temporary training set", essentially causing the model to make predictions on the past. Since past values were shown to have a correlation with future values (when I included autoregressive features, model performance was improved), using this method of cross-validation may lead to model overfitting. To avoid this issue, I used Forward-Chaining cross-validation during my model training.

Forward-Chaining cross-validation creates fold restrictions that k-fold cross-validation does not incorporate. Instead of allowing any of the k folds to act as the validation set, its choice is restricted to always occur after the events of the “temporary training set”. The size of the “temporary training set” grows from 1-fold to k-1 folds, while the “temporary testing set” always occupies the fold immediately following the last fold encompassed by the “temporary training set”. **Figure 5** depicts how Forward-Chaining cross-validation works.

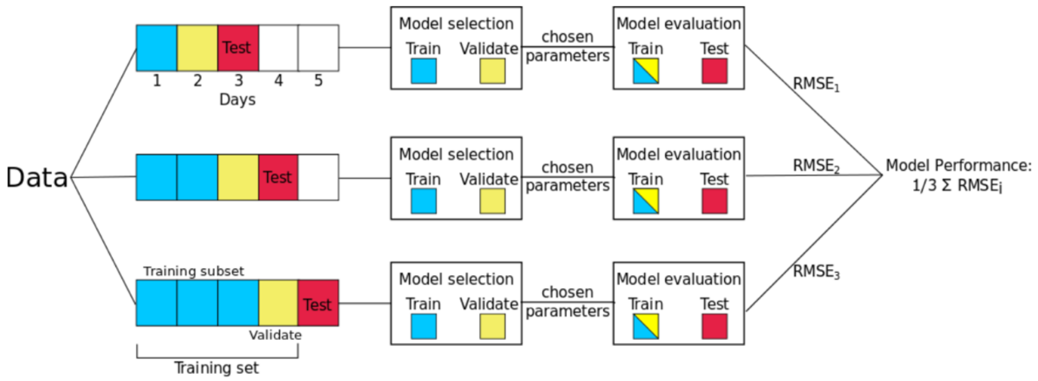


Figure 5. Diagram illustrating Forward-Chaining cross-validation method [3]

After incorporating cross-validation, the first loss function I considered was L1 loss. Looking into the time series plots of a few appliances, I noticed that some devices have two states, meaning their power consumption is one value when the appliance is turned ON and zero when it is OFF. I hypothesized that the previous quadratic model was treating ON states as outliers. Knowing that L1 loss handles outliers better than quadratic loss, I considered L1 loss with each of L1 and quadratic regularizers included to minimize overfitting. My results are depicted in **Figure 6.**

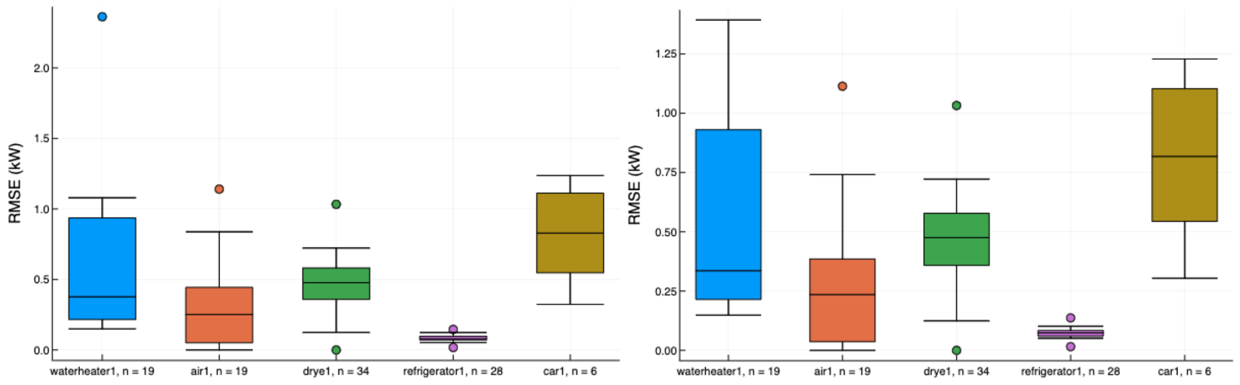


Figure 6. Results from L1 loss with L1 regularizer (left) and quadratic regularizer (right)

Comparing L1 loss with an L1 regularizer to that with a quadratic regularizer, we see that the quadratic regularizer shows improved prediction accuracy compared to the case of the L1 regularizer. However, both cases perform worse than that of ordinary least squares regression. After using L1 loss, I reevaluated its reliability for this problem by considering the fact that spikes in power consumption are actually not outliers. It is difficult to model power consumption while tending to the OFF state, treating ON states as outliers because in reality, making predictions based on this will always predict far less power consumption than what is truly being used and the energy disaggregation model won't be able to be trusted.

I then considered Huber loss with L1 and quadratic regularizers. Huber loss benefits from both L1 and quadratic losses by combining their effects into a piecewise loss function. For large errors, the Huber loss function takes on the characteristics of the L1 loss function, enabling it to handle outliers well. For small errors, it takes on the characteristics of the quadratic loss function and incurs less of a penalty than L1 loss would incur. My results are depicted in **Figure 7**.

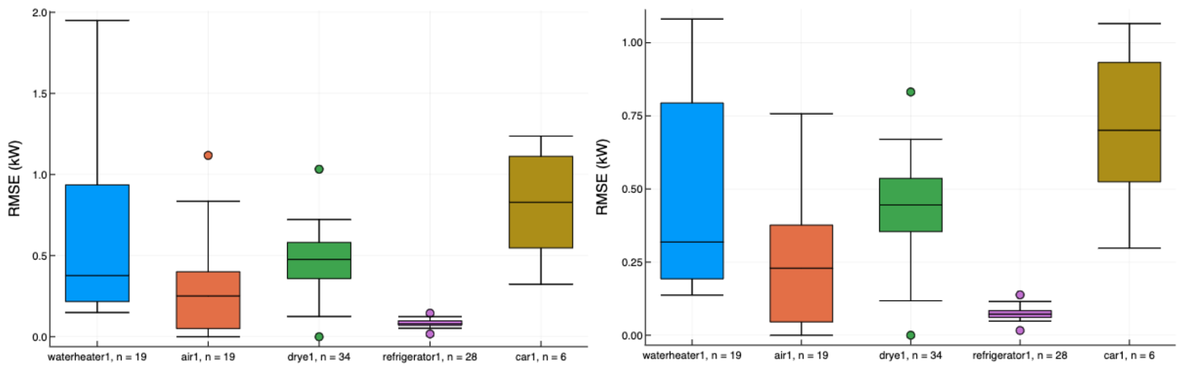


Figure 7. Results from Huber loss with L1 regularizer (left) and quadratic regularizer (right)

Using Huber Loss, we once again see better performance with a quadratic regularizer than with an L1 regularizer, but still less accuracy than that obtained from ordinary least squares. Additional details are provided in the Results and Discussion section.

5. Results and Discussion

Mean RMSE	Quadratic loss w/ Autoregression	L1 Loss, L1 Reg	Huber Loss, L1 Reg	L1 Loss, Quadratic reg	Huber Loss, Quadratic Reg
Waterheater1	0.1753	0.60848	0.5868	0.55129	0.4795
Air1	0.10817	0.2933	0.28046	0.2593	0.24029
Drye1	0.2016	0.44849	0.44849	0.4481	0.42249
Refrigerator1	0.0587	0.0827	0.08299	0.0736	0.07403
Car1	0.3562	0.8130	0.8130	0.80335	0.70675

Table 2. Summary of results from each model tested

Observing results in **Table 2**, we see that the feature engineered quadratic loss with no regularizer case outperforms all other cases. The appliance "refrigerator1" consistently showed the least error, while "car1" showed the greatest. Looking further into the time series plots for each appliance, I found that there were four main types of appliances: those that have two states (ON/OFF), those that are always on, those that have a finite number of states, and those that have an infinite number of states. Of the five appliances used in this study, “refrigerator1” was the only appliance that was considered an “always on” appliance. I hypothesize that this was related to its high accuracy—appliances that are always on without much deviation in power consumption are certainly easier to make predictions on. “waterheater1”, “drye1”, and “car1” each appeared to be either purely two-state or finite state with very few states, which I hypothesize is related to their poor prediction performance.

Overall, I might consider using these models to make predictions on “refrigerator1”, or other continuous use appliances. I would be hesitant using them to make predictions on other appliance types, since high accuracy is a key component to the usefulness of energy disaggregation. Through this study, I was able to determine that with the use of some submeters, disaggregation predictions can be improved. In cases that the appliance of interest cannot be physically submetered easily, aggregate smart meter data in combination with specific appliances that have a strong correlation with the appliance of interest can help prediction accuracy.

6. Future Work and Weapons of Math Destruction

Throughout my study, I created models for each appliance of interest of each building in my dataset. I had hopes of combining my prediction models and hyperparameters for each home into one that could be used for one appliance on all homes within the dataset, but I was unable to. During my study, I optimized for the hyperparameter lambda (multiplied by regularizer term), but ran into trouble with determining a single lambda to use for all homes. Each home seemed to have their own lambda value that minimized error, but I could not think of a way to combine their effects, so I did not include this as a part of my study. Additionally, I had access to a second dataset containing reactive power measurements for each appliance. In energy disaggregation research, real and reactive powers are often used as features to improve accuracy for predictions. I decided not to use it because my machine would not handle 4 million total rows of data. As a part of future work, I would like to include both an optimization for lambda and reactive power features.

My results can indirectly pose the issue of being a weapon of math destruction. I don’t think the results when provided to the intended party pose much of a threat. Homeowners will gladly use the data to identify energy inefficient appliances within their home to replace for electricity bill savings. However, energy disaggregation results can be used to determine when a person is home or not, as well as patterns associated with where they are within their home. If a data attacker were to gain possession of this data, they could potentially harm the user. This is the main motivation behind the study—finding a simple, less computationally expensive approach to the energy disaggregation problem would create less risk to the homeowner in that neither aggregate smart meter data nor energy disaggregation prediction data would need to be shared remotely. If all computation is done locally, the user’s privacy and safety can drastically increase. Considering fairness, I believe that it is not an important criterion for choosing an energy disaggregation model since demographics and other personal information are not considered.

References

[1] “Buildings and Climate Change,” Environmental and Energy Study Institute, accessed September 14, 2019.

[2] Chen, Victor L., et al. “Real-Time, Appliance-Level Electricity Use Feedback System: How to Engage Users?” *Energy and Buildings*, vol. 70, 2014, pp. 455–462.

[3] Cochrane, C., “Time Series Nested Cross Validation,” Towards Data Science, accessed December 10, 2020.