CE 295: Data Science for Energy
Final Project Booklet

Spring 2021

Instructor: Daniel Arnold

Don't Shed On Me: Optimal Microgrid Control Using Load Prioritization

Maya Bruguera, Allison Eames, Jake McDermott, Jack Smith, Patrick Soltis

May 7th, 2021

Abstract

Electricity is essential to daily life in the developed world, powering critical systems and services such as hospitals, water supply and wastewater treatment, and other functions. Power outages--such as those driven by increasingly large wildfires and Public Safety Power Shutoffs in California or the recent extreme weather-induced rolling blackouts in Texas--compromise functionality of these critical services. Microgrids that include storage and distributed generation resources can help alleviate some of these stresses, with the ability to isolate or 'island' from the main power grid and distribute power locally. However, microgrids have limited storage and generation available; therefore, the ability to prioritize loads and optimize discharge can help to maximize the benefit that these resources provide and minimize harm. This study creates an optimal storage dispatch schedule based on the priority of serving different loads, as well as storage and distributed generation resources available. Results showed that as expected, mean fraction of load served declines with outage duration, and increases with diesel generator fuel available. Additionally, the model tends to serve a large fraction of load for nodes with relatively low demand despite lower relative ranking, while providing less service to nodes with extremely high demand despite a higher relative ranking.

Introduction

Motivation & Background

The last few years have seen some of the most dangerous and destructive wildfires in California's history. The 2018 Camp Fire created immense financial liability for Pacific Gas & Electric (PG&E), ultimately resulting in their bankruptcy. The 2020 California wildfire season has been billed as the most destructive in history, burning almost 5% of all California acreage.[1] The California Public Utilities Commission (CPUC) has given the investor-owned utilities (IOUs) the ability to conduct Public Safety Power Shutoffs (PSPSs) which allows for the de-energization of the electric grid in places deemed to be at risk of causing additional wildfires.[2] The CPUC does recognize, however, that "a PSPS can leave communities and essential facilities without power, which brings its own risks and hardships, particularly for vulnerable communities and individuals". These risks are especially pronounced for Californians who require access to electricity to power lifesaving medical equipment.[3]

In the wake of these PSPS events, microgrids have emerged as one possible solution to managing the stress and impacts of prolonged power outages. The United States Department of Energy (DOE) defines microgrids as "a group of interconnected loads and distributed energy resources within clearly defined electrical boundaries that acts as a single controllable entity with respect to the grid. A microgrid can connect and disconnect from the grid to enable it to operate in both grid-connected or island-mode".[4] A

variety of challenges exist around the modeling and implementation of microgrids, but their potential benefits over the traditional power grid could outweigh the costs of doing so. In addition to their ability to isolate from broader electrical shutoffs, microgrids' incorporation of various distributed generation units can potentially alleviate concerns regarding the consistent supply of electricity and long-term energy security associated with the existing grid.[5]

Focus of this Study

This study aims to create a model for managing energy generation and storage resources and shedding loads in an optimized manner during an islanding event of a known time horizon. This is accomplished by using load and microgeneration forecasting combined with load prioritization of customer tiers to dispatch distributed resources optimally.

Relevant Literature

A variety of microgrid simulations and studies can be found in literature, examining topics ranging from grid reliability to smart loads.[6,7] Load prioritization schemes are present in many of these sources, since loads must be shed in a controlled and predictable manner as the power supply decreases. Often these are ordered as "tiers" which vary in customer importance. In [8], the first tier embodies all loads that are critical and not to be shed for any reason, including hospitals and 911 dispatch centers.[8] Discretionary loads that can be shed for short periods of time, such as HVAC equipment, are included in the second tier. Finally, the third tier contains loads that are only to be shed to sustain grid stability and prevent a blackout. It includes residential customers and commercial facilities with back-up generation. The literature appears to lack real world examples of how and when to prioritize loads within a microgrid. This study can fill this gap by connecting load prioritization to the real-world impacts of the PSPS events in California. Despite the literature containing some basic examples of load priority tiers, such prioritizations have lacked applicability. This work provides this by linking customer priority to discrete customer types and load profiles found in California.

In addition to managing how power is distributed to the loads connected across the microgrid, internal power generation from distributed generators must be accounted for to ensure adequate power distribution across the microgrid. As noted in [9], the interconnection of these distributed generators in a low-voltage system as proposed in this study may affect overall power system performance.[9] These microsources may be biomass, fuel cells, wind, or others, however, for the purpose of this study only solar and diesel generation will be analyzed. A description of how these generators will be modeled, controlled, and distributed is detailed in the *Mathematical Modeling & Implementation in Python* section below.

Technical Description

Load/Customer Categories & Ranking

When deciding which loads to shed, we have come up with a prioritization of different types of loads. These loads are classified and described in the table below. The guiding principle for assigning the priorities was ensuring that the most critical loads were directly connected to human life and broader social welfare. This led us to prioritizing medical baseline customers that need electricity to power life-

maintaining devices, followed by critical infrastructure like fire stations, low-income residential customers, non-low-income residential customers, and then all other loads:

Table 1. Load Prioritization Categories and Ranks

Customer Category	Rank	Load Source	Description & Justification
Residential (Medical Baseline)	5	CARE Load Profile from CPUC + hourly medical baseline allowance (500 kWh/month or 0.694 kWh per hour)	Residential customers that are on medical baseline tariffs are among the most vulnerable groups with respect to intermittent electricity access. Because customers in this class rely on medical equipment for basic living functions, we prioritize their load needs first.[10]
Non-residential critical facility (e.g., hospital, fire station)	4	PG&E A10[11]	Some non-residential customers provide enormous social benefits to local groups. These include obvious facilities like hospitals and fire stations but could also include less obvious facilities like cell phone towers.[12] Thus, we prioritize this group second.[10]
Residential (CARE/FERA)	3	CARE Load Profile from CPUC	Residential customers that are not on medical baseline tariffs might be on California Alternate Rates for Energy (CARE) or Family Electric Rate Assistance Program (FERA) tariffs.[13] These are meant to assist low-income Californians and these customers might also be at greater risk of wildfire-related hardships (e.g., increased costs from replacing spoiled food are more burdensome on lower income households[14]). Thus, we prioritize this group third.
Residential (non-CARE/non-FERA)	2	PG&E E1[11]	Residential customers that are not on medical baseline or CARE/FERA tariffs are prioritized next. There might be low-income customers that are not enrolled in CARE/FERA. Regardless, during the COVID-19 pandemic a lot of office work has transitioned into working from home. By prioritizing the rest of residential customers 4th, we still allow for meaningful load shedding reduction for average working people. This could be useful for ensuring that heating/cooling remains constant depending on the time of the year, that customers are able to work from home, or that food remains unspoiled in residential refrigerators.
All other customers (e.g., commercial/industrial, agricultural)	1	ENERNOC Open Source Load information (Profile #14)[15]	Our final load priority group is a catch-all group for customers that don't fit into one of the four prior groups. This means all commercial and industrial loads, agricultural customers, streetlights, and even institutional or research loads (i.e., the University of California).

Microgrid Characteristics

The microgrid model created for the purposes of this study was inspired by a hypothetical grid of similar magnitude found in [16] that minimized the set of loads shed while maintaining grid stability.[16] The microgrid model presented below is connected to a medium voltage substation representing the external grid. In the event of this external grid's failure, it can be disconnected from the microgrid using a breaker at the point of common coupling. Local power generation consists of a single diesel generator and two photovoltaic (PV) generators, both of which are attached to a battery capable of storing excess power when power production exceeds demand. A variety of loads that may be encountered in a typical microgrid are included in this model, including residential households corresponding to ranks two, three,

and five in Table 1 above, as well as a non-residential critical facility. Each element of the grid is detailed in Table 1 below, and a schematic can be seen in Figure 1.

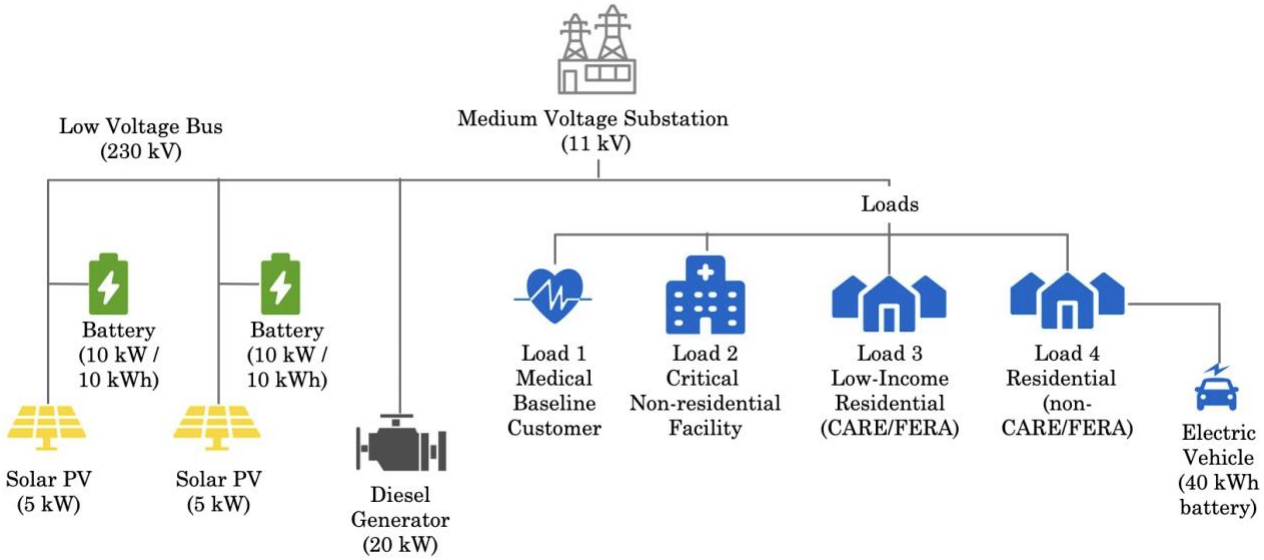


Figure 1. Microgrid diagram, adapted from D'Agostino et. al (2017).[16]

Each node is described in Table 2, including the elements that will be modeled in our optimization.

Table 2. Microgrid Node Descriptions

Characteristic	**Description**	**Modeling**	**Nodes**	**Priority Rank**
Batteries	2 batteries, 10 kWh each	-Battery state of charge -Battery capacity -Energy dispatched or stored at each time step	1,2	--
Solar generation	The PV capacity will be 5 kW capacity (actual generation varies by month and efficiency), and the generation is available to the whole grid.	-Apparent, active, and reactive power dispatched at each time step	1,2	--
Diesel generator	20 kW generator at central location serving the whole system	-Apparent, active, and reactive power dispatched at each time step -Fuel consumed at each time step -Maximum power allowed based on remaining fuel	3	--
Controllable Load 1- Residential Medical Baseline Customer	Load profiles as described in Tables 1 and 2.	Optimal fraction of load served at each time step	4	5

Characteristic	Description	Modeling	Nodes	Priority Rank
Controllable Load 2-critical non-residential facility	Load profiles as described in Tables 1 and 2.	Optimal fraction of load served at each time step	5	4
Controllable Load 3-Low Income Resident (CARE/FERA)	Load profiles as described in Tables 1 and 2.	Optimal fraction of load served at each time step	6	3
Controllable Load 4-Residential (non CARE/FARE)	Load profiles as described in Tables 1 and 2.	Optimal fraction of load served at each time step	7	2
Electric Vehicle Battery	Node 7 has a 100 kWh electric vehicle battery with vehicle to grid capabilities.	-Apparent, active, and reactive power dispatched -Battery state of charge -Battery capacity	7	2

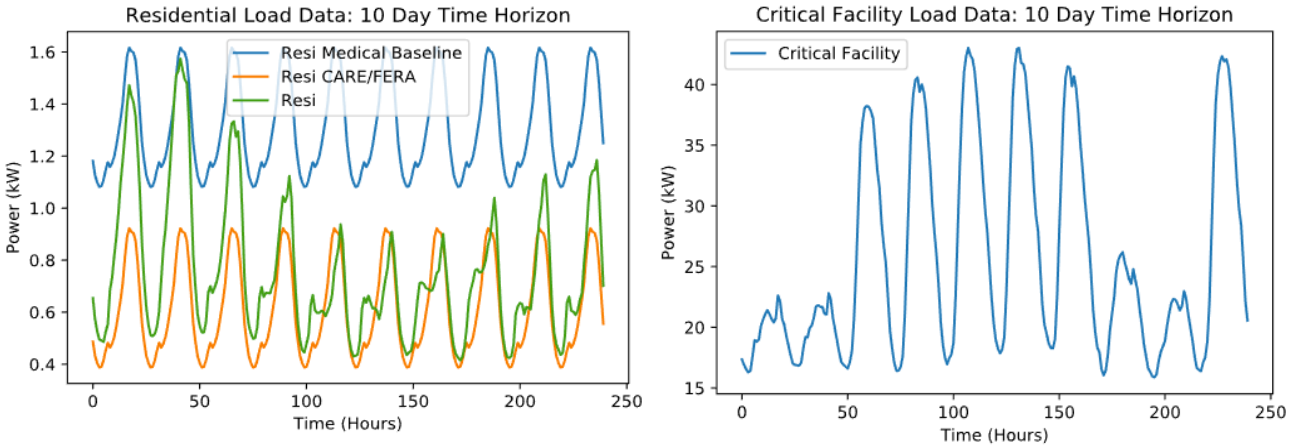


Figure 2. Load profile data for Residential Medical Baseline, CARE/FERA, residential non-CARE/FERA, and critical facility. (Sources: See Table 1)

Mathematical Modeling & Implementation in Python

We model the microgrid as a network with each node representing a single microgrid customer. All nodes (1-7) are connected directly to a central node (0), and no nodes are directly connected to another, as shown in Figure 1. Two nodes have a solar PV module and battery storage (nodes 1 and 2), one has a diesel generator (node 3), three exclusively draw power (nodes 4, 5, and 6), and one draws power and has an EV that can be used as a battery (node 7). We do not disaggregate storage, load, and generation of a single user to multiple nodes, paying attention to only power at the meter. Power flow on lines between nodes is governed by the DistFlow equations outlined in Baran & Wu.[17]

We simulate the microgrid only during its islanded state, over periods of time corresponding to PSPS events. We start with a period of 55 hours, corresponding to the average PG&E PSPS length in 2019.[18] To better understand the system's time flexibility, other outage lengths and diesel fuel availability are also

tested (see Table 3). The simulation operates in one hour intervals, assuming constant power consumed or output by any battery, load, or generator over that hour.

In order to produce the apparent power solar generation parameter, average hourly generation profiles from the Global Solar Atlas for Berkeley were applied.[19] We considered the solar profiles for June (the month with the greatest solar output), December (the month with the lowest output), and September, a month that occurs in the middle of California's wildfire season. Figure 3 below depicts the hourly solar generation profile in Berkeley for each month. Because PSPS events are unlikely in December, and June's profile was nearly identical to September, we chose to only use September's profile in the model.

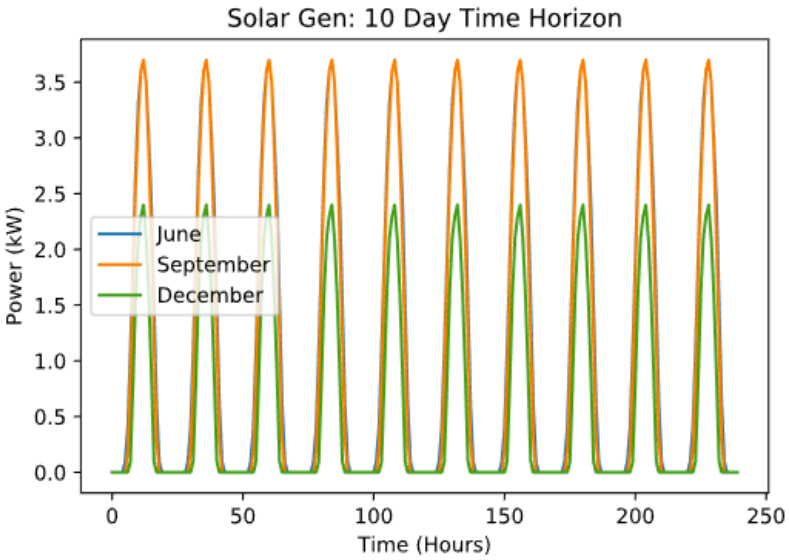


Figure 3. Hourly power output for 5 kW PV system installed in Berkeley, CA in June, September, and December[20]

The table below summarizes the various parameters that will be varied to produce different scenarios.

Table 3. Scenario Matrix

Scenario Number	Outage Duration	Fuel Availability	Time of Year
1	55 hours	6.4 gallons (4-hour supply for 20 kW at full output)	September
2	55 hours	25.6 gallons (16-hour supply for 20 kW at full output)	September
3	55 hours	Unlimited	September
4	5 days	6.4 gallons (4-hour supply for 20 kW at full output)	September
5	5 days	25.6 gallons (16-hour supply for 20 kW at full output)	September
6	5 days	Unlimited	September
7	10 days	6.4 gallons (4-hour supply for 20 kW at full output)	September
8	10 days	25.6 gallons (16-hour supply for 20 kW at full output)	September

Scenario Number	Outage Duration	Fuel Availability	Time of Year
9	10 days	Unlimited	September

Any charging or discharging of the batteries occurs when the model deems it optimal. Power throughput and energy storage limits which keep battery tied to realistic operation are defined by constraints. Appendix A describes the objective function, optimization variables, parameters, and constraints.

Modeling Results

Table 4, Figure 4, and Figure 5 present results from the modeling. As shown in Table 4, mean fraction of load served declines with outage duration, and increases with diesel generator fuel available. The model tends to serve a large fraction of load for nodes with relatively low demand (nodes 1, 4, and 5) even in nodes with lower relative ranking (nodes 6 and 7), while providing less service to nodes with extremely high demand despite a higher relative ranking (node 5). This is likely because the rankings of the nodes are relatively similar in magnitude (ranging from 2 to 5) while the demand of the nodes differ more substantially (demand for Nodes 4, 6, and 7 range from about 0.4 to 1.6 kW, while demand for Node 5 is one to two order of magnitudes higher, ranging from about 15 kW to 45 kW). As a result, in order to maximize the objective function (i.e., the sum of the product of node rank and fraction of load served) the model tends to fully or nearly fully serve Nodes 4, 6, and 7, while providing less service to Node 5, despite the fact that this node was assigned the second-highest rank. This pattern is also illustrated in Figure 4, which shows that the fraction of load served for Nodes 4 (blue), 6 (pink), and 7 (purple) typically remains greater than the fraction of load served for Node 5 (orange).

For scenarios where diesel fuel is constrained, power supplied closely tracks solar generation during the day, while the diesel generator and battery typically provide the bulk of their power when solar is unavailable. In the scenarios where diesel fuel is unlimited, the diesel generator and battery continue to provide more power when solar generation is unavailable, though the fraction of load met is much higher relative to the other scenarios. Notably, even in the unlimited diesel fuel scenarios, the load at Node 5 is not fully met, as power supply continues to be limited by the diesel generator's power output rating (20 kW).

Table 4. Mean fraction of load served across time steps and mean across nodes for each scenario

		Scenario								
	Outage Duration	55 hrs			120 hrs (5 days)			240 (10 days)		
	Diesel Fuel Available	80 kWh	320 kWh	Unlimited	80 kWh	320 kWh	Unlimited	80 kWh	320 kWh	Unlimited
Node	Node 4	1.00	1.00	1.00	1.00	1.00	1.00	0.76	1.00	1.00
	Node 5	0.11	0.37	0.95	0.04	0.14	0.80	0.04	0.05	0.77
	Node 6	1.00	1.00	1.00	1.00	1.00	1.00	0.85	1.00	1.00
	Node 7	1.00	1.00	1.00	0.69	1.00	1.00	0.55	1.00	1.00
	Mean	**0.78**	**0.84**	**0.99**	**0.68**	**0.79**	**0.95**	**0.55**	**0.76**	**0.94**

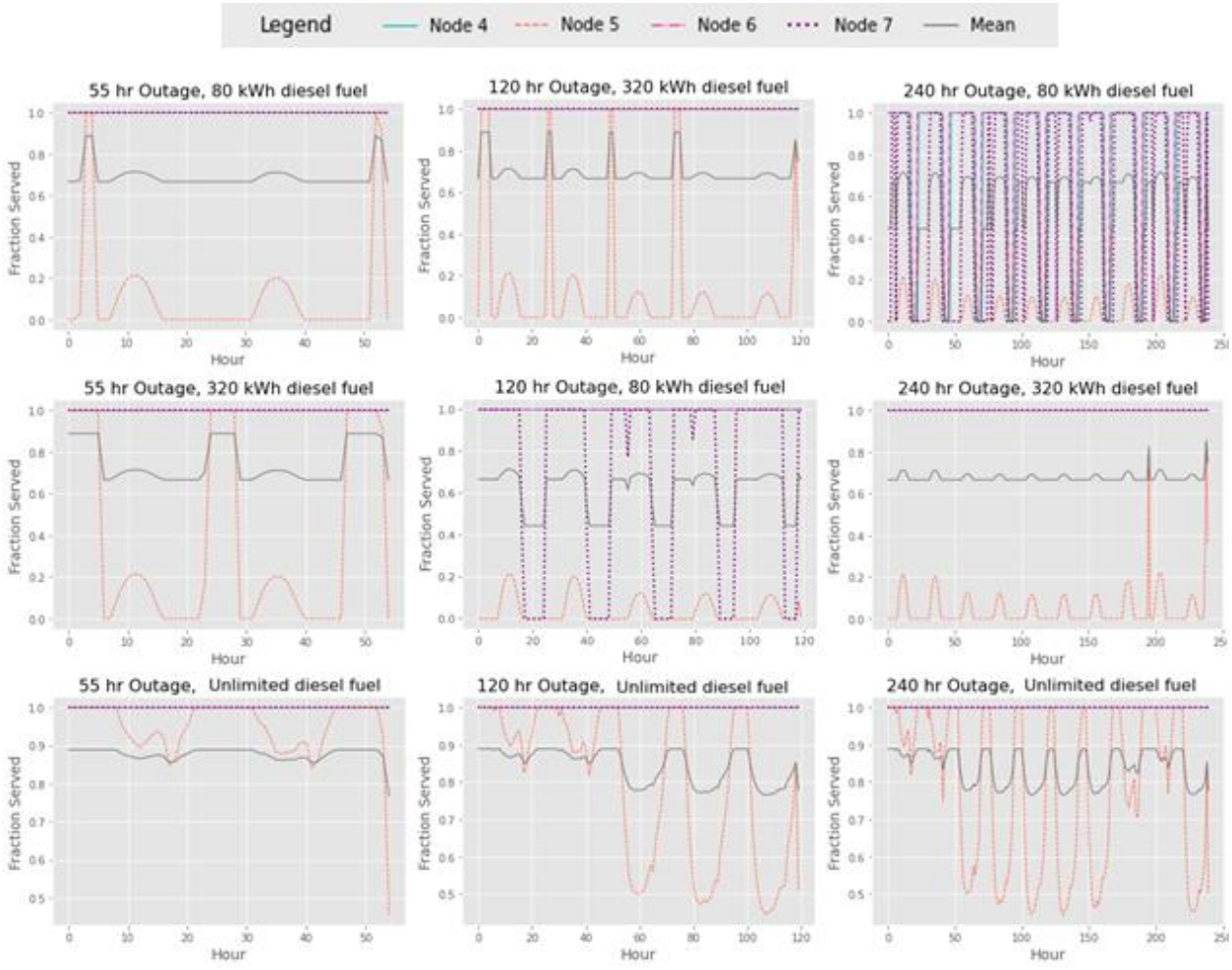


Figure 4. Fraction of load served per node (and mean of nodes) over time under each scenario

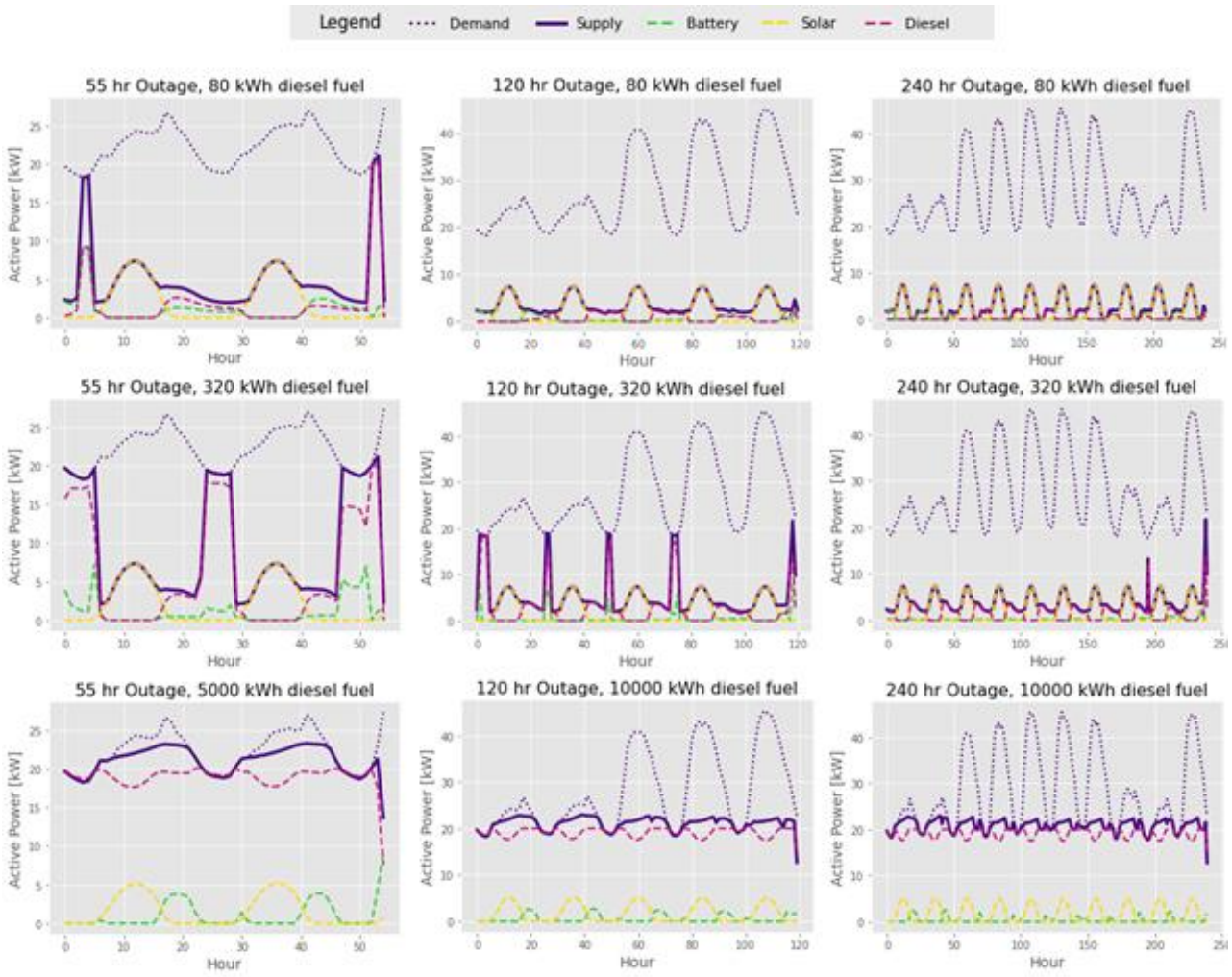


Figure 5. Total active power demand, supply, and generation over time for each scenario

Discussion

The results displayed in Table 4 and Figure 4 above indicate that the microgrid modeled in this analysis can satisfy the power demands of the majority of the loads, regardless of outage duration. One notable exception is Node 5, whose power demand is never fully met even when facing the shortest outage and unlimited diesel fuel is made available to Node 3. This may come across as strange given it is a higher priority than the two nodes that follow it, but the reason behind this becomes clear when examining the relative power demands of each node in detail. Node 5's power demand is significantly higher than other nodes as shown in Figure 2 above, thus the optimization algorithm is incentivized to meet all other load's demands before providing power to Node 5. Adjusting the relative values of the priority rankings could alleviate this issue. In fact, observing how the spacing between priority rank affects the program's choice to serve large, high-priority loads could be a useful method for tuning priority rank values. Specifically, modifying the value of priority rankings such that the difference in magnitude between the rankings exceeds the difference in magnitude between the nodal demands could drive the optimization to fully serve the higher-ranking nodes before fully serving lower ranking nodes. The results from applying the current ranking structure illustrates the tradeoff between serving loads an important but highly energy intensive load, versus serving those that are substantially less energy intensive yet possibly less critical.

Increasing the amount of diesel fuel available to Node 3 has enormous implications for achieving load power demands. The mean demand met is only 55% when the generator produces 80 kWh over a 240 hour outage, but increases by 39% when the generator is provided with unlimited fuel over that same timespan.

The importance of the diesel generator to this particular model is reaffirmed by Figure 5, which shows consistent and relatively low levels of power being provided by both the solar and battery sources. The same cannot be said for power produced by the generator at Node 3, which shows much greater output and variability assuming that it is not limited to 80 kWh. Diesel fuel availability at this node was a key indicator of mean fraction shed; therefore, we recommend that microgrids utilizing a diesel generator have enough diesel fuel to last at least 55 hours, the average PSPS shut-off time. We hope this optimization model can contribute to the wealth of tools for making these microgrids more reliable, efficient, and effective, and even potentially extend their useful operational periods.

In future research, it would be helpful to further stratify the load prioritization. Despite having a 5-tiered system for general load prioritization, as smart homes become more common, there could be an opportunity for further control. Even within the highest ranked medical baseline category, it's reasonable to assume that there are some non-essential portions of that generalized load profile that are less important than the most important loads from the next tiered group. Practically, breaking up large high-priority loads like node 5 would mitigate the optimization's perverse incentive to avoid serving the large load despite its high priority.

Below we define a "microgrid baseline" which is meant to mirror California's medical baseline. The medical baseline program allows enrolled customers to have an extra baseline allowance of 500 kWh per month in recognition of their medical device needs. In a similar fashion, the microgrid baseline is meant to capture some portion of a customer's load that is "essential". From there, we prioritize the essential portions of each tier before prioritizing the non-essential loads. Table 5 below provides additional information on this framework for future consideration:

Table 5. Microgrid Baseline Prioritization

Customer Category	**Rank**	**Description & Justification**
Residential (Medical Baseline): Essential Microgrid Baseline Needs	10	Hourly Essential Microgrid Baseline Need = 0.649 kW + 0.20 + 0.05 kW = 0.944 kW 500 kWh per month (from Medical Baseline Allowance) / 30 days / 24 hours = 0.649 kW .20 kW (Refrigeration needs) .05 kW (other essential load allowance, e.g., heating/cooling/lighting)
Non-residential critical facility (e.g., hospital, fire station): Essential Microgrid Baseline Needs	9	Hourly Essential Microgrid Baseline Need = 5 kW 5 kW will be used as a generic number to represent the range of loads that could be deemed essential (communications technologies, medical equipment etc)
Residential (CARE/FERA): Essential Microgrid Baseline Needs	8	Hourly Essential Microgrid Baseline Need = 0.20 + 0.05 kW = 0.25 kW .20 kW (Refrigeration needs)

Customer Category	Rank	Description & Justification
		.05 kW (other essential load allowance, e.g., heating/cooling/lighting)
Residential (non-CARE/non-FERA): Essential Microgrid Baseline Needs	7	Hourly Essential Microgrid Baseline Need = 0.20 + 0.05 kW = 0.25 kW .20 kW (Refrigeration needs) .05 kW (other essential load allowance, e.g., heating/cooling/lighting)
All other customers (e.g., commercial/industrial, agricultural): Essential Microgrid Baseline Needs	6	Hourly Essential Microgrid Baseline Need = 1 kW Given that this tier is a catch-all for all other customers with varying levels of demand, there is no simple way to characterize the broad range of "essential" demands that could vary based on underlying characteristics (i.e., Agricultural customers would need energy for pumping which is likely not the case for retail or other commercial customers). For this group, we will use a 1 kW per hour baseline.
Residential (Medical Baseline): Non-essential loads	5	All other demand not covered by the Microgrid Baseline
Non-residential critical facility (e.g., hospital, fire station): Non-essential loads	4	All other demand not covered by the Microgrid Baseline
Residential (CARE/FERA): Non-essential loads	3	All other demand not covered by the Microgrid Baseline
Residential (non-CARE/non-FERA): Non-essential loads	2	All other demand not covered by the Microgrid Baseline
All other customers (e.g., commercial/industrial, agricultural): Non-essential loads	1	All other demand not covered by the Microgrid Baseline

Below we provide visuals of what these microgrid baselines would look like. Each graph contains the original load profile, the baseline amount, and the remaining amount of load (total net of the baseline):

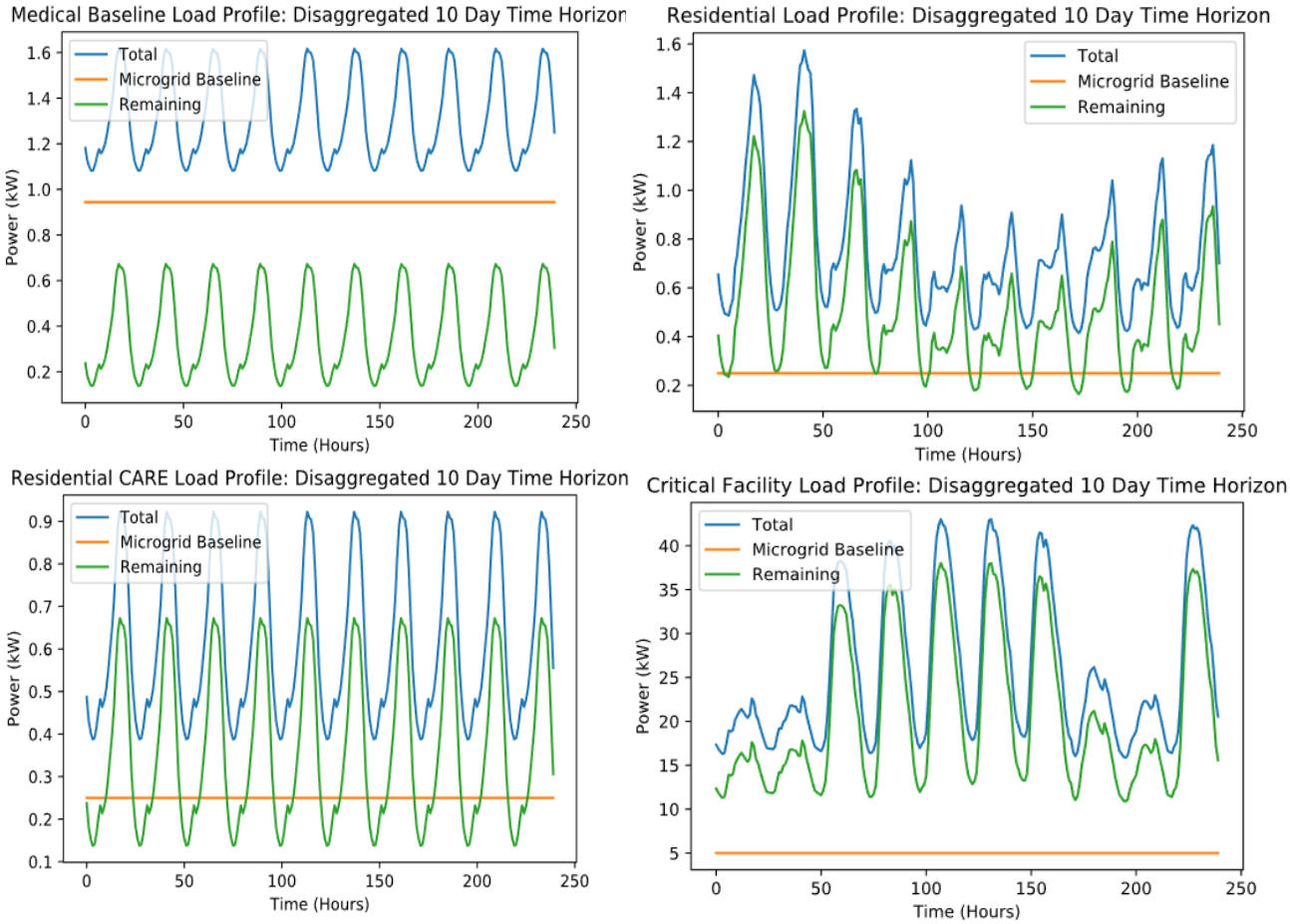


Figure 6. Microgrid Baseline Breakdown

Summary

This study optimized the dispatch schedule of microgrid energy storage resources, given known load priorities, operational characteristics, and system resource constraints. Microgrids may be able to provide continuous power service to critical facilities and homes during PSPS events, and other disturbances to the broader grid that prevent power delivery from outside the microgrid. However, when operating as an island (i.e., not receiving output from the broader grid), a microgrid must provide power using only local generation and storage. The model developed here instructed load scheduling in such an islanded microgrid where local storage resources are unable to meet total demand. While microgrids are not currently permitted to extend over property lines, demonstration projects like the Ecoblock project in Oakland have pushed the CPUC to consider amending the "Own Use" Exemption to accommodate microgrid communities moving forward.[20] As this occurs, it will be essential to create agreement amongst microgrid users to prioritize loads of the most vulnerable during islanding events. As extreme weather events become more frequent and intense due to climate change, and continue to compromise transmission resiliency in the coming years, more and more energy providers may seek to supplement or altogether replace risky transmission with microgrids that can remain self-sufficient for extended periods.[21] In future research, we recommend further stratifying the load prioritization and weighting the loads appropriately to make this model even more relevant to current and future needs.

References

[1] Yan, H., Mossburg, C., Moshtaghian, A., & Vercammen, P. (2020, September 6). *Creek Fire: California sets new record for land torched by wildfires as 224 people escape by air from a "hellish" inferno—CNN*. https://www.cnn.com/2020/09/05/us/california-mammoth-pool-reservoir-camp-fire/index.html

[2] California Public Utilities Commission (CPUC). (n.d.). *Public Safety Power Shutoff (PSPS) / De-Energization*. Retrieved February 25, 2021, from https://www.cpuc.ca.gov/deenergization/

[3] California Public Utilities Commission (CPUC). (n.d.). *What are CARE and medical baseline?* Retrieved February 25, 2021, from https://www.cpuc.ca.gov/General.aspx?id=12196

[4] Ton, D. T., & Smith, M. A. (2012). The U.S. Department of Energy's Microgrid Initiative. *The Electricity Journal*, *25*(8), 84–94. https://doi.org/10.1016/j.tej.2012.09.013

[5] Mahmoud, M., Hussain, S., & Abido, M. (2014). Modeling and Control of Microgrid: An Overview. *Journal of the Franklin Institute*, *351*. https://doi.org/10.1016/j.jfranklin.2014.01.016

[6] Maleki, B., Gandomkar, M., Maleki, T., & Gandoman, F. H. (2014). Method of evaluating reliability of microgrids in Island mode by using load prioritization. *2014 19th Conference on Electrical Power Distribution Networks (EPDC)*, 76–81. https://doi.org/10.1109/EPDC.2014.6867502

[7] Solanki, B. V., Raghurajan, A., Bhattacharya, K., & Cañizares, C. A. (2017). Including Smart Loads for Optimal Demand Response in Integrated Energy Management Systems for Isolated Microgrids. *IEEE Transactions on Smart Grid*, *8*(4), 1739–1748. https://doi.org/10.1109/TSG.2015.2506152

[8] Moran, B. (2016). Microgrid load management and control strategies. *2016 IEEE/PES Transmission and Distribution Conference and Exposition (T D)*, 1–4. https://doi.org/10.1109/TDC.2016.7520025

[9] Basak, P., Saha, A. K., Chowdhury, S., & Chowdhury, S. P. (2009). Microgrid: Control techniques and modeling. *2009 44th International Universities Power Engineering Conference (UPEC)*, 1–5.

[11] California Public Utilities Commission. (n.d.). *SMAP*. Retrieved March 31, 2021, from https://www.cpuc.ca.gov/General.aspx?id=9099

[12] Pacific Gas & Electric. (n.d.). *Pacific Gas & Electric—Tariffs*. Retrieved March 31, 2021, from https://www.pge.com/tariffs/energy_use_prices.shtml

[13] Beam, A. (2020, January 8). *California lawmakers eye back-up power for cellphone towers*. AP NEWS. https://apnews.com/article/0594bd7174cf5ebf09f28b8ae4945289

[14] California Public Utilities Commission. (n.d.). *CARE/FERA Program*. Retrieved March 31, 2021, from https://www.cpuc.ca.gov/lowincomerates/

[15] Initiative for Energy Justice. (2020). California Power Shutoffs: Deficiencies in Data and Reporting.

[16] EnerNOC. (2012). *EnerNOC GreenButton Data*. https://open-enernoc-data.s3.amazonaws.com/anon/README

[17] D'Agostino, F., Massucco, S., Silvestro, F., Fidigatti, A., Monachesi, F., & Ragaini, E. (2017). Low voltage microgrid islanding through adaptive load shedding. *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, 1–6. https://doi.org/10.1109/EEEIC.2017.7977861

[18] Baran, M. E., & Wu, F. F. (1989). Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power Delivery*, *4*(2), 1401–1407. https://doi.org/10.1109/61.25627

[19] PG&E. (2019). PG&E Public Safety Power Shutoff (PSPS) Report to the CPUC: October 26 & 29, 2019 De-Energization Event. https://www.cpuc.ca.gov/uploadedFiles/CPUCWebsite/Content/News_Room/NewsUpdates/2019/Nov.%2018%202019%20PGE%20ESRB-8%20Report%20for%20Oct.%2026%2029%202019.pdf

[20] World Bank Group, ESMAP, SOLARGIS. 2021. Global Solar Atlas. https://globalsolaratlas.info/map

[21] Von Meier, A., & Kammen, D. M. (2021). White Paper: The EcoBlock Project and the "Own Use" Exemption under Public Utilities Code Section 218 – A Way Forward for Privately Operated Microgrids. UC Berkeley: California Institute for Energy and Environment (CIEE). Retrieved from https://escholarship.org/uc/item/1s88n8c6

[22] U.S. Global Change Research Program. (2018). Volume II: Impacts, Risks, and Adaptation in the United States in Fourth National Climate Assessment. https://nca2018.globalchange.gov/

Appendices

Appendix A

This appendix describes the model objective function, optimization variables, parameters, and constraints.

Objective Function

Minimize $-R^T F$

Parameters

	Parameter	In code	Meaning	For every
1	R	R	Priority ranking of different customer categories	Node
2.1	D	D	Apparent power demand	Node
2.2	D_P	D_P	Active power demand	Node
2.3	D_Q	D_Q	Reactive power demand	Node
3	S_S	S_S	Power generated by solar PV (apparent)	Node
4.1	j_{max}	j_max	Maximum energy that battery can store	Node
4.2	j_{start}	j_start	Energy in battery at start of PSPS	Node
5	f_{start}	f_start	Diesel fuel available (in kWh) at start of PSPS	Node
6.1	b_{rating}	b_rating	Maximum power that battery can charge/discharge	Node
6.2	d_{rating}	d_rating	Maximum power generator can produce	Node
7	pf	pf	Power factor (converts active power demand to D)	Node
8.1	V_{min}	V_min	Minimum voltage allowed	Node
8.2	V_{max}	V_max	Maximum voltage allowed	Node
9.1	r	r	Resistance	Line
9.2	x	x	Reactance	Line
10	I_{max}	I_max	Maximum current	Line
11	A	A	Adjacency matrix: A[i, j]=1 if i is the parent of j	--
12	ρ	rho	Parent node index	Node
13	dt	dt	Length of time step (default 1 hour)	--
14.1	η_s	nu_s	Solar inverter efficiency	Node
14.2	η_b	nu_b	Battery inverter efficiency	Node

Optimization Variables

	Variable	In code	Meaning	For every
1	F	F	Fraction of real load served at each node	Node
2.1	l_S	l_S	Apparent power supplied	Node
2.2	l_P	l_P	Real power supplied	Node
2.3	l_Q	l_Q	Reactive power supplied	Node
3.1	b_S	b_S	Battery apparent power dispatched (+) or stored (-)	Node
3.2	b_P	b_P	Battery real power dispatched (+) or stored (-)	Node
3.3	b_Q	b_Q	Battery reactive power dispatched (+) or stored (-)	Node
4.1	d_S	d_S	Diesel power generated (apparent)	Node
4.2	d_P	d_P	Diesel power generated (real)	Node
4.3	d_Q	d_Q	Diesel power generated (reactive)	Node
5.1	S_P	S_P	Solar power generated (real)	Node
5.2	S_Q	S_Q	Solar power generated (reactive)	Node
6.1	s	s	Net apparent power consumed	Node
6.2	p	p	Net real power consumed	Node
6.3	q	q	Net reactive power consumed	Node
7	V	V	Bus voltage	Node
8.1	j	j	Battery state of charge (energy available)	Node
8.2	f	f	Diesel fuel (energy) available	Node
9	P	P	Active power flowing	Line
10	Q	Q	Reactive power flowing	Line
11	L	L	Squared magnitude of complex current	Line

Constraints

Notes on constraints:

- Within constraints, variables and parameters are indexed as [time, node/line].
- Unless otherwise specified, constraints are active and functionally equivalent at all time steps.

	Constraint	Meaning	For nodes/lines
1	$F = \frac{l_P}{D_P}$	Definition: Fraction of load served	All
2.1	$P[:,0] = 0$	No real power flow between node 0 and itself	0
2.2	$Q[:,0] = 0$	No reactive power flow between node 0 and itself	
2.3	$L[:,0] = 0$	No current flowing between node 0 and itself	
2.4	$V[:,0] = 1$	Node 0 voltage is reference voltage, thus 1 p.u.	
3.1	$s = l_S - b_S - d_S - S_S$	Definition of net apparent power consumed	All
3.2	$p = l_P - b_P - d_P - S_P$	Definition of net real power consumed	All
3.3	$q = l_Q - b_Q - d_Q - S_Q$	Definition of net reactive power consumed	All
4.1	$b_S[:,[0,3,4,5,6]] = 0$	No battery (dis)charge at nodes w/out batteries	0,3,4,5,6
4.2	$b_P[:,[0,3,4,5,6]] = 0$		
4.3	$b_Q[:,[0,3,4,5,6]] = 0$		

4.4	$d_S[:,[0:2,4:7]] = 0$	No generator power output at nodes w/out generators	All except 3
4.5	$d_P[:,[0:2,4:7]] = 0$		
4.6	$d_Q[:,[0:2,4:7]] = 0$		
4.7	$S_P[:,[0,3:7]] = 0$	No solar PV power output at nodes w/out solar	0,3,4,5,6,7
4.8	$S_Q[:,[0,3:7]] = 0$		
5	$l_P \leq D_P$	Power delivered cannot exceed demand	All
6.1	$j[0] = j_{start}$	Batteries start with stored energy of j_{start}	All
6.2	$j[t] = j[t-1] - b_S[t-1]dt$	Energy available in battery at any time is the previous hours' energy minus energy dispatched in that hour	All
6.3	$0 \leq j$	Battery cannot have negative energy stored	All
6.4	$j \leq j_{max}$	Battery cannot store more energy than its capacity	All
7.1	$f[0] = f_{start}$	Generators start with fuel supply specified in f_{start}	All
7.2	$f[t] = f[t-1] - d_S[t-1]dt$	Diesel fuel supply is fuel supply at previous hour minus fuel burned in that hour	All
7.3	$0 \leq f$	Generator cannot have negative fuel supply	All
8.1	$b_S[0]dt \leq j_{start}$	Battery does not discharge more energy at any time than available in previous time step	All
8.2	$b_S[t]dt \leq j[t-1]$		All
8.3	$d_S[0]dt \leq f_{start}$	Generator does not use more fuel than available at any time than available in previous time step	All
3.4	$d_S[t]dt \leq f[t-1]$		All
9.1	$-b_{rating} \leq b_S$	Battery cannot charge above power rating	All
9.2	$b_S \leq b_{rating}$	Battery cannot discharge above power rating	All
10	$d_S \leq d_{rating}$	Generator cannot generate above power rating	All
11.1	$P_{ij} = p_j + r_{ij}L_{ij} + \sum_{k=0}^{N} A_{jk}P_{jk}$	DistFlow equation describing real power flow on lines	All
11.2	$Q_{ij} = q_j + x_{ij}L_{ij} + \sum_{k=0}^{N} A_{jk}Q_{jk}$	DistFlow equation describing reactive power flow on lines	All
12	$V_j - V_i = (r_{ij}^2 + x_{ij}^2)L_{ij} - 2(r_{ij}P_{ij} + x_{ij}Q_{ij})$	Voltage drop between two nodes explained by impedance losses along the lines	All
13	$L_{ij} \geq \frac{P_{ij}^2 + Q_{ij}^2}{V_j}$	Definition of squared magnitude of complex current, relaxed to make convex	All
14.1	$\sqrt{l_P^2 + l_Q^2} \leq l_S^2$	Definition of apparent power demand	All
14.2	$\sqrt{b_P^2 + b_Q^2} \leq b_S^2$	Definition of apparent power (dis)charged by battery	All
14.3	$\sqrt{d_P^2 + d_Q^2} \leq d_S^2$	Definition of apparent power output by diesel generator	All

14.4	$\sqrt{S_P^2 + S_Q^2} \leq S_S^2$	Definition of apparent power generated by solar	All
16.1	$\sum_i l_S[t,i] \leq \sum_i b_S[t,i] + d_S[t,i] + S_S[t,i]$	Apparent power supplied cannot exceed apparent power generated at any time	All
16.2	$\sum_i l_P[t,i] \leq \sum_i b_P[t,i] + d_P[t,i] + S_P[t,i]$	Real power supplied cannot exceed real power generated at any time	All
16.3	$\sum_i l_Q[t,i] \leq \sum_i b_Q[t,i] + d_Q[t,i] + S_Q[t,i]$	Reactive power supplied cannot exceed reactive power generated at any time	All
17.1	$v_{min}^2 \leq V_j$	Voltage cannot slip under allowed minimum	All
17.2	$V_j \leq v_{max}^2$	Voltage cannot exceed allowed maximum	All
18	$L_{ij} \leq I_{ij,max}^2$	Current cannot exceed line capacity	All
19.1	$d_S \geq 0$	Generator cannot output negative real power	All
19.2	$d_P \geq 0$	Generator cannot output negative apparent power	All
19.3	$l_S \geq 0$	Apparent power consumed cannot be negative	All
19.4	$l_P \geq 0$	Real power consumed cannot be negative	All
19.5	$S_P \geq 0$	Solar cannot generate negative real power	All

Appendix B

View project GitHub repository for more details.

Predicting Residential Energy Usage Based on Meteorological Data

Siena Moca, Amber Chau, Justin Wong, Juhyun Lee, Kamsey Agu

I. Abstract

Global climate change continues to increase the demand of energy consumption over time as temperatures and weather patterns are more extreme. Increasing demands have led to the need for increasing energy capacity and distribution for customers. In this study, we developed a machine learning model to predict residential energy consumption in the Bay Area geographic zip codes based on meteorological data, such as temperature, humidity, near infrared, and weather. Historic residential energy consumption data from PG&E and NREL National Solar Radiation data served as our validation data. We sought to develop a generalizable model so that such inputs can be applied to other geographic locations outside the Bay Area. The KNN model performed the best based on the evaluation of mean squared error (MSE) values. Such a model serves as key to ensuring efficient operations of residential energy systems as climate change takes its course.

II. Introduction

Motivation & Background

It is commonly accepted that energy consumption will increase in the future as a result of climate change. Reasons driving this trend include economic development, technological developments, and extreme climate (increase and decrease in temperatures) and weather events.

Energy consumption in the United States has been continuously increasing, as seen in Figure 1. According to the U.S EIA (United States Energy Information Administration), the amount of energy use in the U.S. recorded the historical peak. This growth of the amount of energy consumption is causing a vicious circle: the increase of energy consumption causes climate change which again leads to greater demand for energy because the majority of energy sources are still from fossil fuels generating carbon dioxides. Statistics show that 80% of the energy consumed in the U.S is from fossil fuels and only 20% is from non-fossil fuels such as wind, biofuels, and other renewables (Figure 1). It is apparent that energy consumption will continue to grow in the future unless the vicious circle breaks.

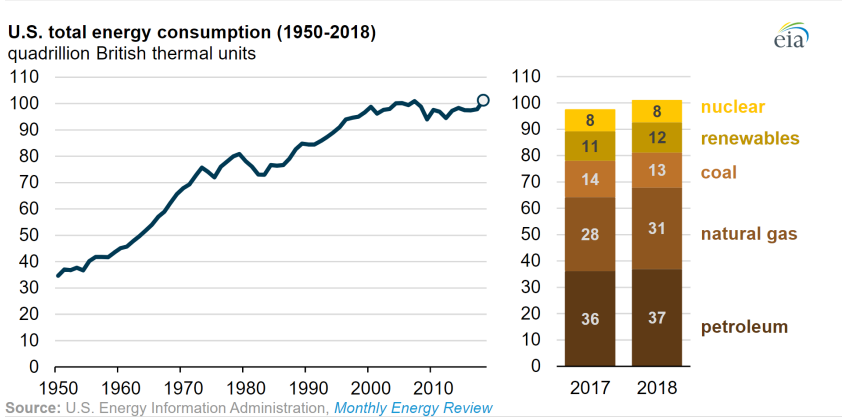


Figure 1. U.S Total energy consumption and energy portfolio (U.S. EIA 2019)

Forecasting energy consumption increase is vital in determining whether or not electricity grids can sustain increased demands and prevent large outages and how it is going to impact climate change. With a better forecast, utilities and stakeholders can plan ahead and invest in improving resiliency and environmental sustainability of their energy system. Recently in Texas, a massive snowstorm has caused power outages not only due to the damage to utility systems but also the very high demand for households to heat their homes as a result of extreme cold weather (Ferman 2021).

Challenges associated with managing this particular energy system includes planning ahead for unexpected climate change events such as wildfires, snowstorms, etc. We also do not know what kind and impact new technologies will help mitigate climate change and may produce a different trend regarding energy consumption.

Relevant Literature

Multiple studies have been conducted in attempting to predict energy consumption through using past energy consumption sets with machine learning methods but have used various input variables such as building characteristics and climate data.

Williams et. al used statistical learning methods including linear regression, regression trees (RT) and multivariate adaptive regression splines (MARS) to predict future monthly energy consumption in residential homes in Bexar County, TX. The authors inputted monthly energy consumption for 36 months and historical temperature and humidity values from the past 30 years. While the MARS model had significantly better performance in predicting future monthly consumption, there were still limitations since there was uncertainty about climate data and regression trees were better for aggregate predictions. Different fuel types were also unknown about space heating.

Zhao and Magoules in addition to regression models, included Artificial Neural Networks (ANNs) and Support Vector Machines or Regression to evaluate the significance of weather conditions determining building energy usage such as temperature, humidity, solar radiation and wind speed. ANNs prediction was useful for solving nonlinear problems and had the highest prediction accuracy when combined with simplified engineering methods.

In New Zealand, Salcedo et. al also used Support Vector Regression (SVR) but added on separately Multi-layer Perceptron (MLP) to predict monthly mean air temperature based on historical monthly mean air temperature in Australia and New Zealand. The researchers found that the SVR algorithm performed the best out of all the other approaches and concluded that Machine Learning methods are appropriate for energy consumption prediction.

Lastly, Wang et. al analyzed models and previous work relating to building energy prediction based on the principle of algorithm integration. They reviewed previous prediction models, such as Random Forest, Gradient Boosting, auto-regressive integrated and moving average (ARIMA) and k-nearest neighbors (KNN). The authors further developed integration prediction models by comparing using "integration learning", which essentially combines the advantages of single models to improve overall performance. The core idea of this “stacking model” was to collect the

differentiation of various base algorithms (each of which can observe data from different perspectives) by constructing an integrated framework. The paper sought robust methods to reduce overfitting and evaluate model performance based on accuracy, generalization, and robustness. Their stacking model was more accurate than any of the individual base models.

Focus of This Study

This study used machine learning and modeling methods to predict the scale of energy consumption increase based on past climate data. For this study, we specifically looked at residential energy usage in kilowatt-hours (kWh). We predicted the scale of energy consumption applicable to the Bay Area, California which can be used to plan increasing energy capacity across utilities, storage, etc.

III. Technical Description

Data Sources

The PG&E dataset contained customer electric (kWh) usage data in the Bay Area. The data is reported by zip code, month, year and four customer types including residential, commercial, agricultural and industrial. The reports from the dataset are publicly available with each report containing 3 months of usage data through the end of the calendar quarter and are provided in CSV file format. For this study, electric usage data was extracted for 2013 to 2020 from the PG&E public database website. Data cleaning was performed by selecting only residential customers and grouping the data by months. The parameters in the data include the zip code, total number of customers, and average and total electric usage in kilowatt-hour.

The NREL National Solar Radiation Database is a collection of half-hourly and hourly values of meteorological data with three common solar radiation measurements, which include global horizontal, direct normal, and diffuse horizontal irradiance (DHI). The dataset majorly covers the United States and certain international locations, and the data was collected at adequate temporal and spatial scales to represent regional solar radiation climates effectively. For this study, the solar radiation dataset was selected for the Bay Area and grouped into monthly data to match the PG&E dataset. The dataset was cleaned by extracting the direct normal irradiance (DNI), diffuse horizontal irradiance (DHI), global horizontal irradiance (GHI), solar zenith angle, temperature and wind speed. The DNI, DHI, and GHI represent the amount of solar radiation from the direction of the sun, the solar radiation on a horizontal surface received from the sky excluding the solar disk, and the solar radiation on a horizontal surface received from the sky, respectively, given in Watt per square meter (W/m^2). The solar zenith angle, temperature, and wind speed are given in degrees, degree Celsius ($\circ\ C$), and meter per second (m/s). The temperature values were converted to degree Fahrenheit ($\circ\ F$). After extracting these variables, the cleaned NREL dataset was merged with the PG&E dataset for analysis and prediction.

The NREL National Solar Radiation Database provides API instructions on how to download their data in Python. According to the instructions, specified API parameters, such as year, longitude and latitude, are passed via a query string for the URL. After defining all variables, a special url is now to be declared. This url is the query string that is used to fetch data from the

NREL API, which returns csv data. Then the command “pd.read_csv(url, skiprows=2)” will load the data which consists of 46 columns including city, state, country, latitude, longitude, time zone, DHI, DNI, GHI, and Solar Zenith Angle.

Zip codes mapped to longitude and latitude coordinates were merged using opendatasoft public dataset. This was used so that the zip codes specified in the PG&E dataset corresponded to a coordinate pair for the NREL API. Since we were interested in solar irradiance data for the PG&E dataset, we first grouped the PG&E data by zip code, coordinate, and year; these served as parameters for the NREL API. At the time of this project, the NREL API was not updated to reflect 2020 solar radiation measurements. Furthermore, there were 6474 unique zip codes-year pairs in the entire PG&E data from 2013-2019. The resulting dataset consisted of 77,689 rows, where each row represented a zip code (with a coordinate pair) at a month and year (between 2013-2019) with additional columns from the PG&E measurement (in kWh) and NREL API (GHI, DHI, DNI, etc. as mentioned previously). We then filtered the PG&E dataset into a bounding box for Bay Area; the South West corner was (36.897966, -123.433313), while the North East corner was (38.593263, -121.381268). Below you can see the bounding box we created in Figure 2.

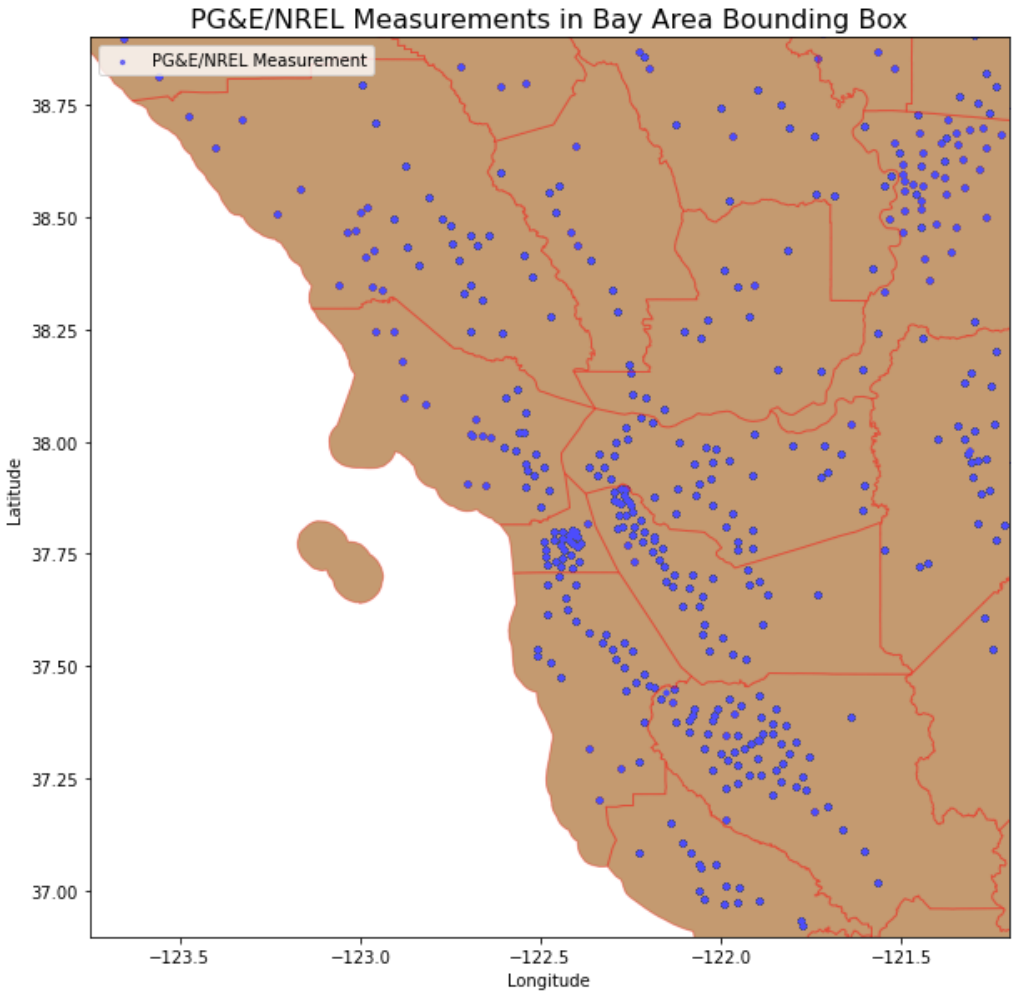


Figure 2. PG&E measurements within our bounding box.

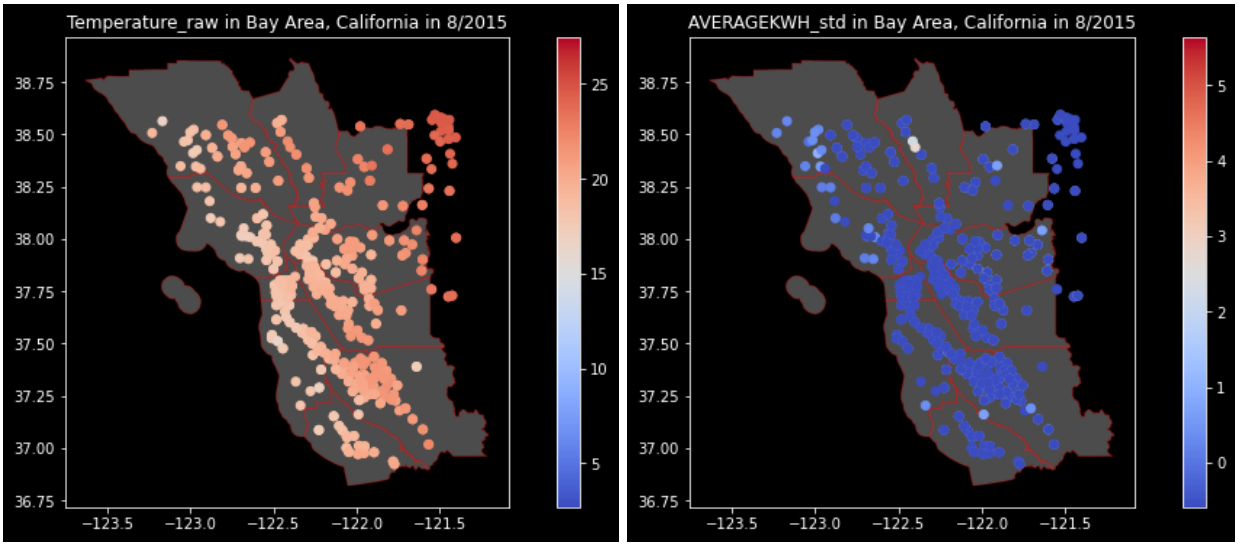


Figure 3. Plot of temperature and average kWh during August in 2015 in the Bay Area

In Figure 3, we plotted the temperature over the region of the Bay Area and the average kWh reported by PG&E for a zip code. We plotted the counties of the Bay Area using red lines on the graph and used a heat map style to plot the relative intensities of each of these measurements. We made similar graphs for each month from 2013 to 2020, in order to visualize the environmental metrics and related average kWh for each region over time.

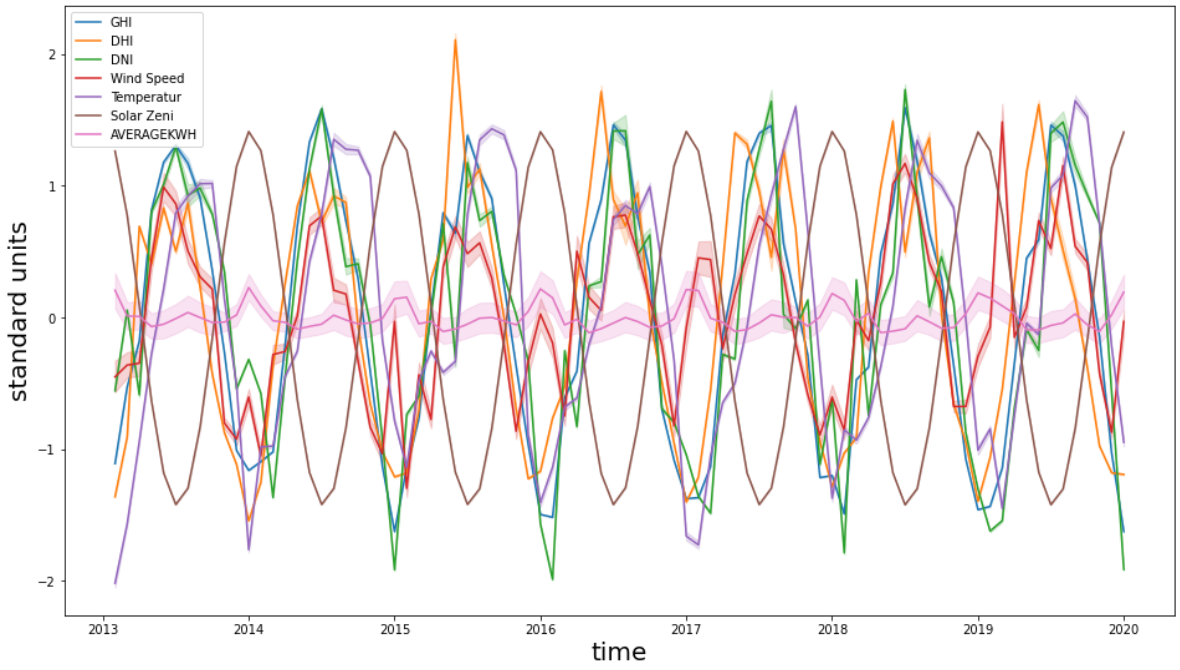


Figure 4. Plot of environmental metrics from NREL and PG&E Data in standard units over time

In Figure 4, we standardized each of our environmental metrics and plotted each of them using the line plot function to see how they changed with respect to time. This step was performed to visualize the relationship between these variables and average kWh and to see if there was a trend and to help guide us in which variables to include in our initial model. After reviewing this,

we decided to include all of the variables from the NREL dataset in our model. Figure 5 shows the PG&E average electricity usage over time from 2013 to 2020 for residential customers.

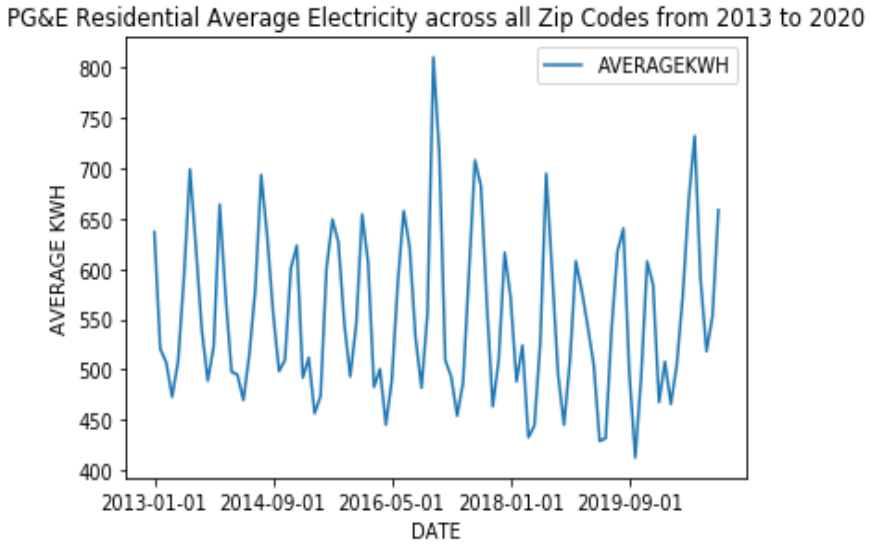


Figure 5. PG&E Average Electricity Usage Over Time

Data Cleaning

We also noticed a few outliers after plotting the standard units for the mean kWh consumed. As a result we decided to only include all values that were within 8.5 standard units away from the mean total kWh consumed from the PG&E data. Additionally, gaps in the data existed especially for certain zips code. Thus, we excluded the zip codes that had less than 84 monthly temperature measurements over the past 7 years or only had measurements for a singular year. The comparison of average electricity consumption of the Bay Area PG&E data before and after data cleaning shown in Figure 6 and 7.

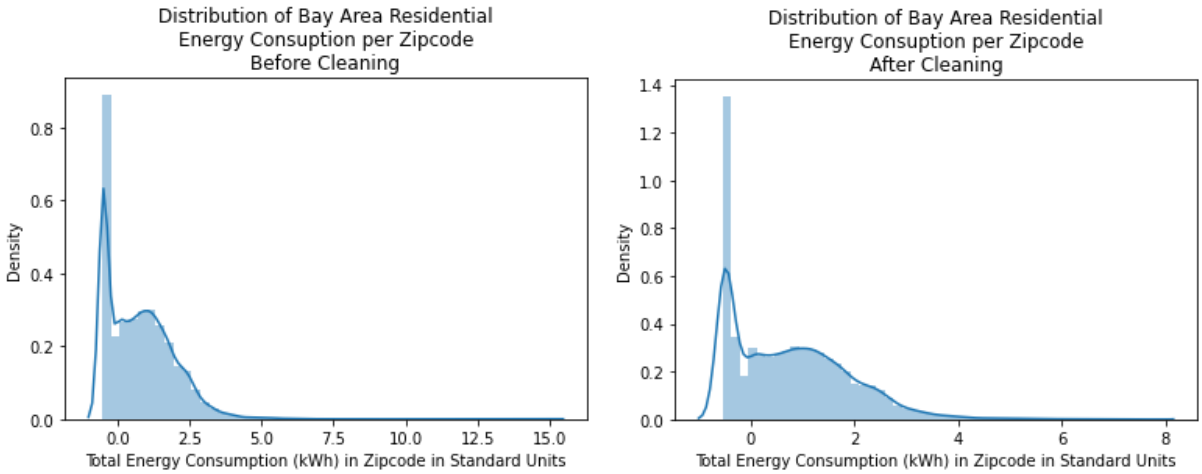


Figure 6. Distribution of mean kWh consumed in standard units before and after data cleaning

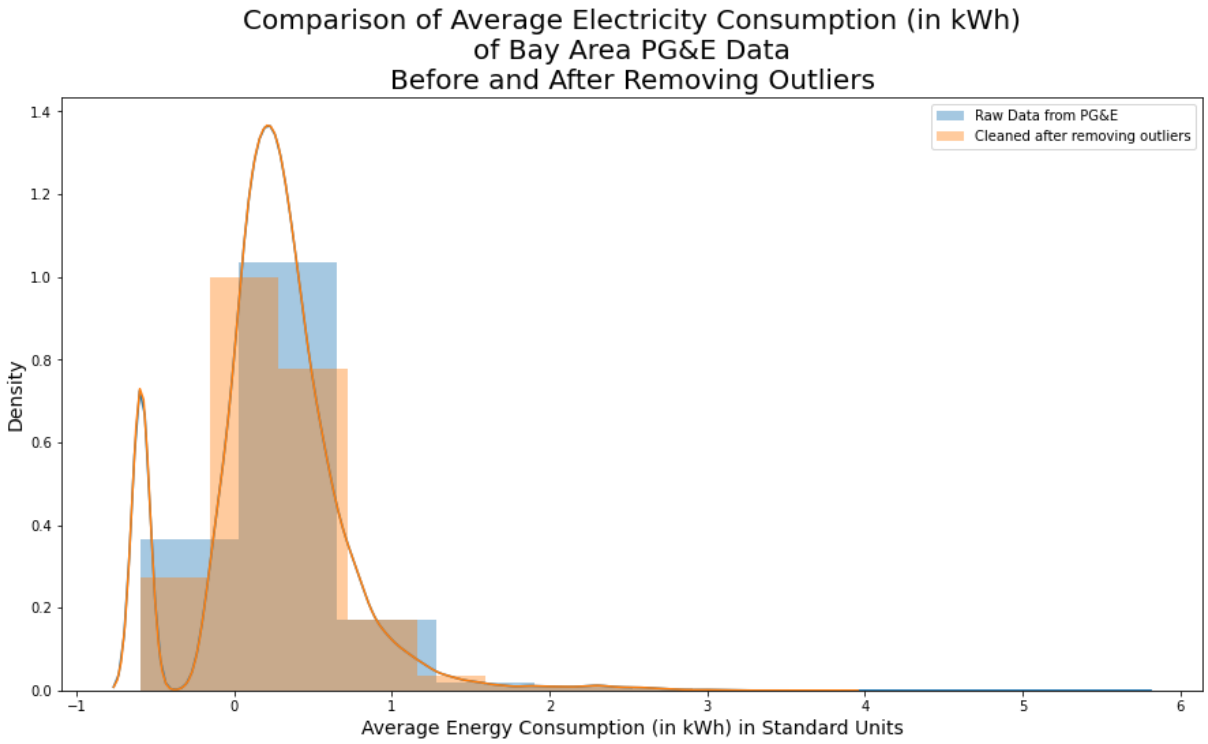


Figure 7. Plot of mean kWh consumed in standard units over time before and after data cleaning

Model Features

Each of the machine learning methodologies used the same X matrix, which consisted of 23 different features. From the PG&E's residential energy consumption dataset, we used month, year, and historical average energy consumption in kilowatt-hours. From NREL's Solar Irradiance Database, we used the following variables: GHI, DHI, DNI, wind speed, temperature, solar zenith angle. Additionally, we lagged certain features by zip code to incorporate the previous three months of data, which made our predictions significantly more robust. The lagged features included the average electricity consumption (in kWh) and all of the NREL solar irradiance variables.

Since our data measurements were as granular as zipcodes, the process of lagging these features was performed using the following:

Start with a blank Pandas DataFrame *df*.
For each unique zipcode *Z*:
Using only measurements corresponding to *Z*, sort the measurements chronologically (using month and year) and call this *S*.
Lag each variable (GHI, DHI, DNI, Wind speed, Temperature, Solar Zenith Angle, and Average KWH) by 1, 2, and 3 months to create an additional 21 columns in *S*.
Concatenate *S* to *df*.
Resulting *df* now consists of 23 columns.

In summary, we lagged seven features (six from NREL and one from PG&E) for three months and included two variables from PG&E, for a total of 23 features to generate our X matrix, which was ultimately a 23136 x 23 matrix.

Machine Learning Methodology Employed

Ordinary Least Squares

The method of linear least squares is to find the best model by finding the model that minimizes the sum of the squares of the error between the observed data and the computed data. By minimizing the squared error, this is expected to reduce the effect of noise on the model.

The data is fit using these parameters, in the form:

$$f(x, \beta) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

The βvalues represent the parameter estimates that are fit to the data, and the x variables are the explanatory variables. Some of the limitations of linear least squares regression is that it is sensitive to outliers, limitations in the shape of the regression over long ranges, collinearity among the features in the X matrix, and poor extrapolation properties. These limitations are solved somewhat by using other machine learning methods, which we will discuss next.

KNN Regression

K-nearest neighbours regression is non-parametric, which means it makes no assumptions about the underlying mathematical model. It is a rudimentary unsupervised learning technique that makes predictions of a new test data point based on the K nearest distance to surrounding data points from the training set. It's also important to note that there's no perfect number of neighbours to use that fits all data sets perfectly. By using hyperparameter tuning techniques, we found that the best hyperparameter, *n_neighbors*, was 10. When training the KNN model, the "y variable" that is essentially matched with the corresponding X vector. This leads to a MSE of 0 for our training data, because the model waits until the testing data in order to do any real work of predicting the classes.

The drawback of using KNN is that it is computationally expensive because in order to predict the "y variable" of a given input, it must compute the distance between the input and all of the training data then take the average y values of the surrounding K points. Additionally, our KNN model belongs to a 23-dimensional vector space. Figure 8 displays some interesting visualizations that show 12 different clusters in the data:

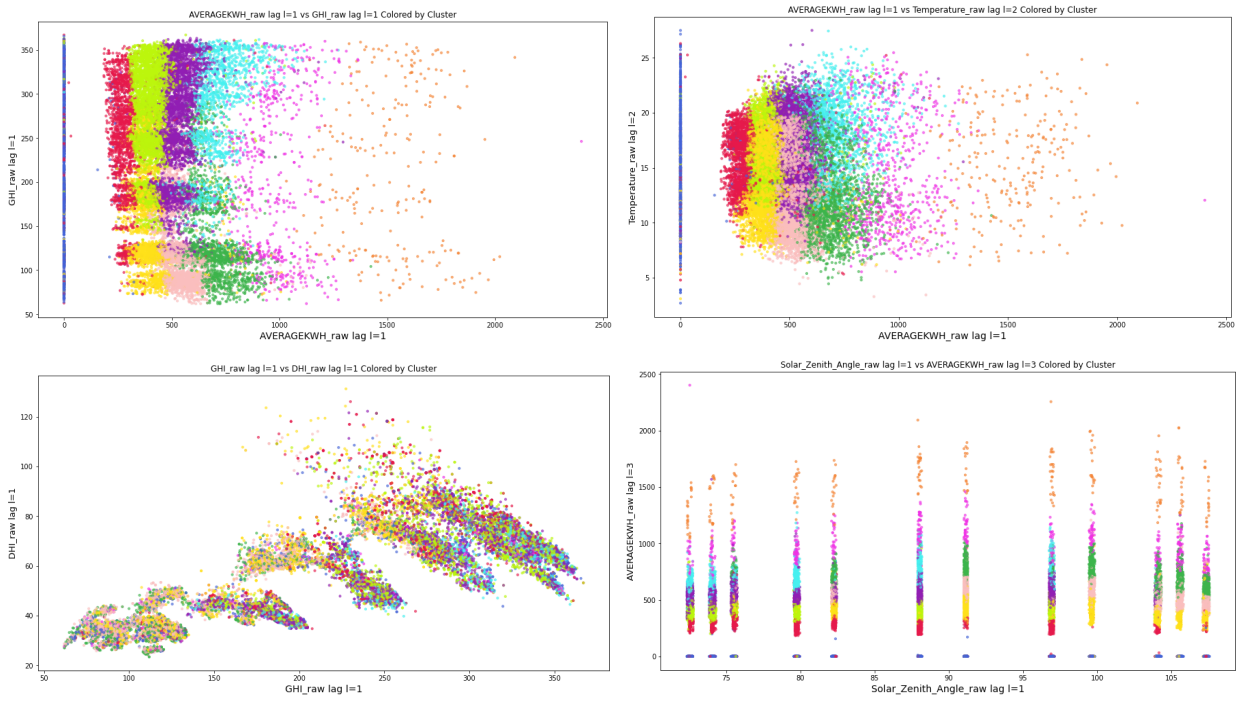


Figure 8: Subplots of 12 Clusters in the Cleaned Dataset

Because of the potentially encountering the curse of dimensionality, we performed principal component analysis (PCA) to reduce the dimensionality. Using the first 17 principal components accounted for at least 99.77% of the original variance in the X matrix. Using the first 9 principal components accounted for at least 95% of the original variance in the X matrix. Further investigation proved that using 23 dimensions had a better test MSE than the dimensionality reduced X matrix (Figure 9).

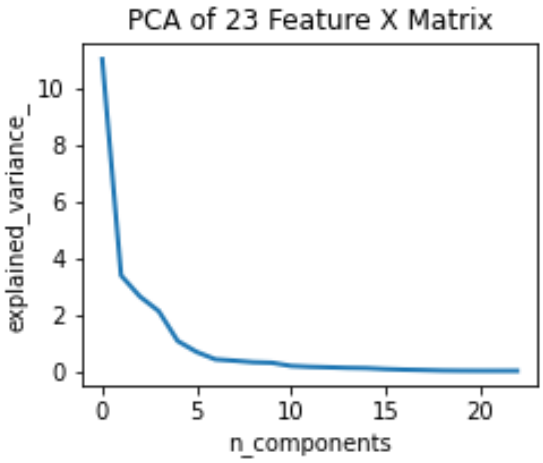


Figure 9: PCA Scree plot of X Matrix

Table 1: KNN MSE values for different X Matrix dimensions

KNN using D dimensions in X matrix	*Training MSE*	*Test MSE*
D = 23 (100% variance)	0.0	5243.805577161207
D = 17 (99.7% variance)	3.140670787779e-22	6217.155193178626
D = 9 (95% variance)	4.005471751402e-22	6242.230004593895

Support Vector Regression (SVR)

The objective function of SVR minimizes the l2-norm of the coefficient vector rather than the squared error. Thus, samples are penalized if the prediction is at least ϵ (epsilon) away from their true target. ϵcan be tuned in order to increase accuracy of the model. We used the following formulation:

$$min\ \frac{1}{2}||w||^2 \quad with\ the\ constraint \quad |y_i - w_i x_i| \leq \epsilon$$

Decision Tree (Randomized Tree)

The Decision Tree model is a non-parametric (like KNN) machine learning model, that has a goal of creating a model that produces the target variable (monthly average electricity consumption) by learning simple decision rules inferred from the data features (which are our environmental metrics).

Decision tree model is one of the supervised learning methods that uses a structure similar to a flowchart which consists of conditions (internal nodes), branches (edges) and decisions (leaves) to predict the value of a target variable. Although this method seems best for classification, it is also widely used for regression. Figure 10 is one of the examples that show the usage of the decision tree model for regression. The target variable predicted in this model is hours played and the predictor variables have roles in making decisions on each branch.

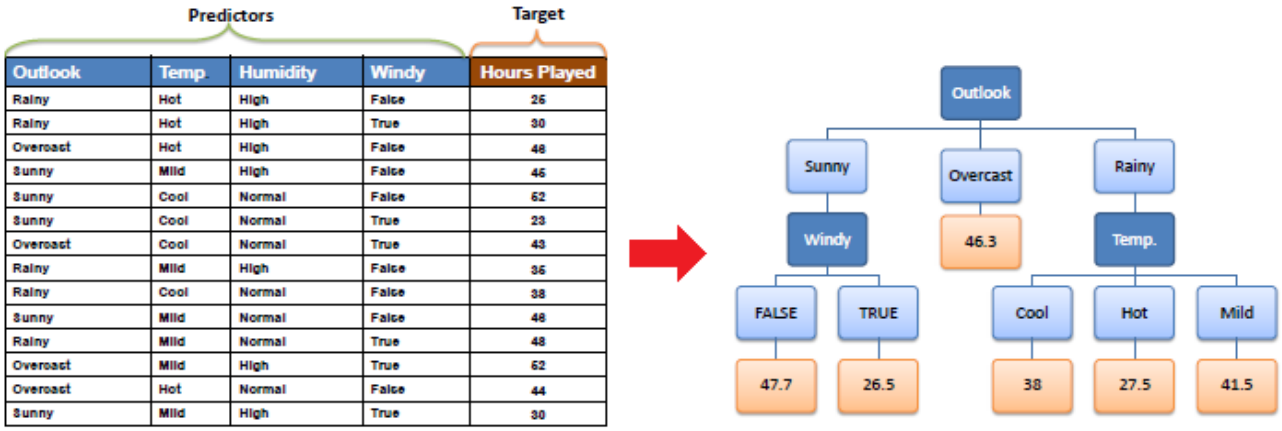


Figure 10. Decision tree model for playing hours (Sayad 2021)

One of the key advantages of this model is that it is simple to understand, interpret and visualize. This model can handle both numerical and categorical data and predict multi-target variables while little effort for data preparation is required. However, this model has a so-called overfitting problem. This problem happens when tree models are too complex that they do not predict well in real cases.

Results

We used functions in the sklearn library to employ each of these machine learning algorithms, and produced a plot showing the training and testing data result after implementing each of these models. Figure 11 and 12 show the estimated results as the thick line and the confidence interval as the band surrounding the lines of the same color.

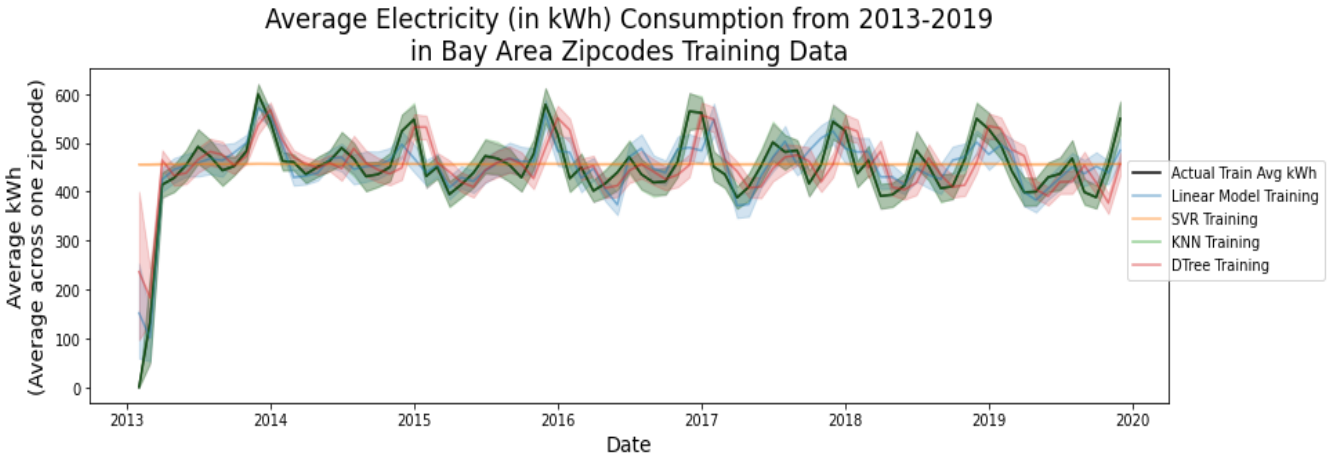


Figure 11. Plotted predicted average electricity consumption for all models using training data

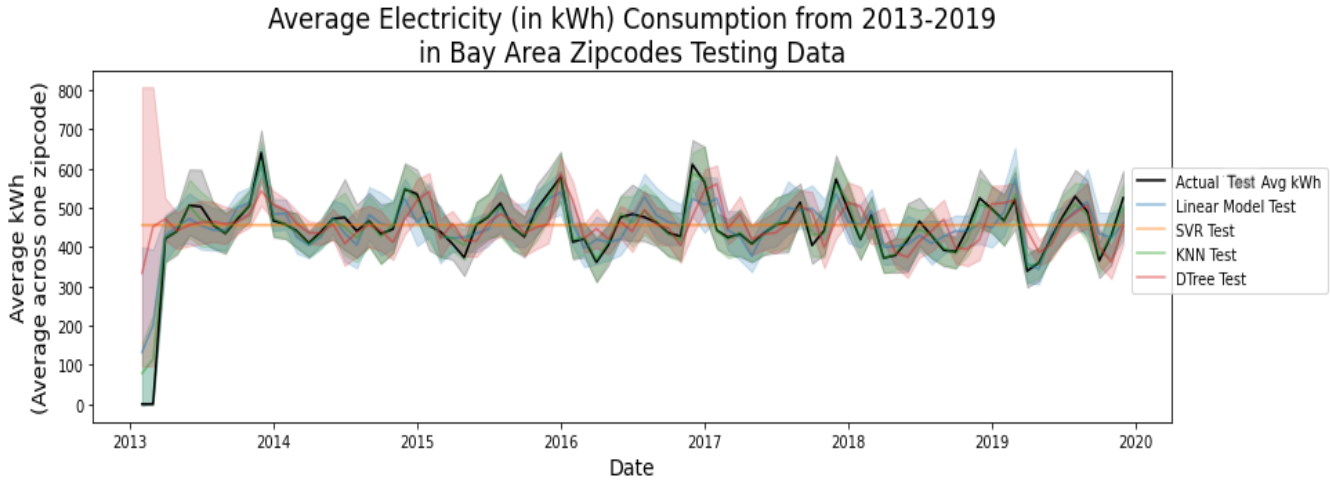


Figure 12. Plotted predicted average electricity consumption four all models using testing data

We also found the training and testing mean squared error for each of the models (Table 2), and found the model with the lowest testing MSE to be the KNN model, and SVR to have the highest MSE. We were thrown off at first when we calculated the MSE of KNN to be 0, however we learned this is because KNN is a lazy learner meaning it does not do the work of classifying the points until the testing stage.

Table 2. Computed MSE after using four different selected models

	Training MSE	*Testing MSE*
Linear Model prediction	11512.800185	12299.512756
SVR prediction	60582.595559	64781.677366
KNN prediction	0.00000	5243.805577
DTree prediction	22361.924220	24123.355030

IV. Discussion

The goal of this project was to create a machine learning model that could predict future residential electricity loads, and it was successfully conducted using NREL and PG&E monthly data from residential consumers after exploring four different machine learning models. A possible application of this model could be to predict future residential energy consumption, based on environmental metrics which will become more extreme as climate change continues. This can help electric utilities to understand when they may experience too much demand and can help them adapt and eventually predict blackouts more accurately and farther in advance to give customers who may be especially vulnerable more warning, such as senior citizens during summer seasons who may experience heat exhaustion if their power and subsequently A/C is turned off.

Some limitations were identified in our project including that household energy consumption loads may be dependent by other variables not related to climate change such as charging an electric vehicle, running an electric appliance, etc. Furthermore, a huge limitation can incur if the NREL and PG&E data is insufficient or inaccurate. Thus, it would be important to explore other energy consumption and climate data sets.

In addition, while the purpose of this study was to predict energy consumption in the Bay Area based on meteorological data such as temperature, humidity, and weather, the key parameter used in the models that increased the accuracy more dramatically than any other variables turned out to be the average monthly electricity usage in kilowatt-hour from the previous three months. The model accuracy suddenly improved after adding those energy consumption data to the X matrix used to train for the machine learning models. The reason for this could be that energy usage tends to increase when the weather is extreme on both sides. Due to this nonlinearity or complexity, it might be difficult for the machine learning to accurately predict future energy consumption without previous months' energy consumption data.

Another limitation of these models is that it could be difficult to apply in real time. Some of the data needed for the X matrix in the models may be difficult or impossible to obtain in a timely manner. For example, to predict energy consumption in June 2021, energy consumption in May 2021 would be needed to run the models. However, the energy consumption data of May 2021

cannot be obtained before June begins, which would present issues. In order to use the model, it is recommended to use data two or three months behind of the target month of prediction. Obtaining this data may be easier if there was a close relationship with PG&E established, meaning potentially preliminary data could be obtained.

Finally, because our model could only learn from data obtained for the years 2013 to 2019, only three months lagging was applied to prevent cutting too much training data. Twelve months lagging could be a better option looking at the seasonal trends of average kilowatt-hour data from Figure 4. However if we were to apply 12-month lagging, the data from the year 2013 would have been removed from the X matrix, which would have cut our available data by about 1/7 which was not ideal. For this reason, we kept the three month lagging which we still felt captured some of the seasonality.

The results were promising; however, there could be some improvements to this model. Firstly, to add more granularity to our models and predictions, it would be preferable to use daily data than monthly data. Moreover, due to the COVID-19 pandemic, it would have been interesting to include 2020 data as well because most residents spent more time in their homes in 2020 compared to previous years. Additionally, a prediction of future electricity usage would have provided more depth to this study. For example, it would have been insightful to predict May or June 2021 electric usage by utilizing weather/environmental forecasts and comparing them with the actual data, but due to data limitations this was not possible as described earlier.

V. Summary

In our project, we predicted monthly energy use in residential buildings and compared estimations based on different machine learning models and methods. Historical energy consumption data and solar radiance data from the Bay Area was used to train and validate our models. Data was taken from PG&E and NREL for the models. The data from PG&E included monthly residential energy consumption while data from NREL included the monthly values of the direct normal irradiance (DNI), diffuse horizontal irradiance (DHI), global horizontal irradiance (GHI), solar zenith angle, temperature and wind speed.

The machine learning models that were used to forecast average monthly energy consumption for residential homes in the Bay Area with climate data included ordinary least squares, KNN, support vector regression, and decision trees. The KNN model performed the best prediction results based on comparing all four MSE values computed for each of the models

Further improvements can be made to ensure more accurate machine learning models, such as utilizing daily data rather than monthly data to add more granularity to our model and predictions. We can also expand our research by incorporating future weather forecasts to predict future electricity, which can be compared later with actual data when available.
However, our energy forecasts can serve as an initial evaluation of the overall impact of climate change on the total demand on the energy grid in the future. This forecast can also be translated to the energy demand on a grid in a region/area. Thus, utilities and organizations can plan ahead to supply and demand imbalance and present power outages.

References:

1.10. Decision Trees—Scikit-learn 0.24.1 documentation. (n.d.). Retrieved March 12, 2021, from https://scikit-learn.org/stable/modules/tree.html

4.1.4.1. Linear Least Squares Regression. (n.d.). Retrieved March 17, 2021, from https://www.itl.nist.gov/div898/handbook/pmd/section1/pmd141.htm

4.1.4.2. Nonlinear Least Squares Regression. (n.d.). Retrieved March 12, 2021, from https://www.itl.nist.gov/div898/handbook/pmd/section1/pmd142.htm

API Instructions—NSRDB. (n.d.). Retrieved March 17, 2021, from https://nsrdb.nrel.gov/data-sets/api-instructions.html

Bay Area (California)—Wikitravel. (n.d.). Retrieved March 30, 2021, from https://wikitravel.org/en/Bay_Area_(California)

Decision Tree Regression. (n.d.). Retrieved March 12, 2021, from http://www.saedsayad.com/decision_tree_reg.htm

Gupta, P. (2017, May 17). *Decision Trees in Machine Learning | by Prashant Gupta | Towards Data Science*. Towards Data Science. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

Kretchmer, H. (2020, July 3). *Chart of the day: How US energy consumption has evolved since independence*. World Economic Forum. https://www.weforum.org/agenda/2020/07/united-states-energy-consumption-since-independence/

Limón, M. F., Sami Sparber and Elvia. (2021, February 15). *2 million Texas households without power as massive winter storm drives demand for electricity*. The Texas Tribune. https://www.texastribune.org/2021/02/15/rolling-blackouts-texas/

Marohl, B. (2020, April 28). *In 2019, U.S. energy production exceeded consumption for the first time in 62 years—Today in Energy—U.S. Energy Information Administration (EIA)*. Today in Energy. https://www.eia.gov/todayinenergy/detail.php?id=43515

McFarland, A. (2019, April 19). *In 2018, the United States consumed more energy than ever before—Today in Energy—U.S. Energy Information Administration (EIA)*. Today in Energy. https://www.eia.gov/todayinenergy/detail.php?id=39092

Moura, S. (2019). *Chapter 4: Machine Learning*.

National Solar Radiation Data Base—OpenEI Datasets. (n.d.). Retrieved March 17, 2021, from https://openei.org/datasets/dataset/national-solar-radiation-data-base

Navlani, A. (2018, August 2). KNN Classification using Scikit-learn. DataCamp Community. https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

PG&E's Energy Data Request Portal. (2021). PG&E Energy Data Request-Public Data Sets. https://pge-energydatarequest.com/public_datasets

Salcedo-Sanz, S., Deo, R. C., Carro-Calvo, L., & Saavedra-Moreno, B. (2016). Monthly prediction of air temperature in Australia and New Zealand with machine learning algorithms. *Theoretical and Applied Climatology*, *125*(1), 13–25. https://doi.org/10.1007/s00704-015-1480-4

Sharp, Tom. *(2020, May 6). An Introduction to Support Vector Regression (SVR). Medium. https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2*

U.S. EIA. (2019, April 16). *In 2018, the United States consumed more energy than ever before—Today in Energy—U.S. Energy Information Administration (EIA)*. https://www.eia.gov/todayinenergy/detail.php?id=39092

US Zip Code Latitude and Longitude. (n.d.). Retrieved March 30, 2021, from https://public.opendatasoft.com/explore/dataset/us-zip-code-latitude-and-longitude/

Wang, R., Lu, S., & Feng, W. (2020). A novel improved model for building energy consumption prediction based on model integration. *Applied Energy*, *262*, 114561. https://doi.org/10.1016/j.apenergy.2020.114561

Williams, K. T., & Gomez, J. D. (2016). Predicting future monthly residential energy consumption using building characteristics and climate data: A statistical learning approach. *Energy and Buildings*, *128*, 1–11. https://doi.org/10.1016/j.enbuild.2016.06.076

Yadav, P. (2018, November 13). *Decision Tree in Machine Learning*. Towards Data Science. https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96

Zhao, H., & Magoulès, F. (2012). A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, *16*(6), 3586–3592.

Talia Arauzo, Dimitris Vlachogiannis, Karl Walter, Weixin Li, Chitra Dangwal
CE 295 Data Science for Energy
May 7, 2021

Final Project Report (Team 3)

I. Title & Team Members

Title: Co-Optimization of In-Home Energy Resources
Members: Talia Arauzo, Dimitris Vlachogiannis, Karl Walter, Weixin Li, Chitra Dangwal

II. Abstract

In this study we modeled and generated optimal operation schedules for smart home energy devices over a 24-hour period (March 15, 2019) located in Berkeley, CA.

First, we modeled three devices – HVAC, a water heater, and a solar photovoltaic (PV) system. We optimized the operation of the HVAC and water heater by minimizing the total daily cost using a PG&E time-of-use (TOU) rate plan.

Second, we performed a co-optimization, where we minimize total costs under different tariff structures – a harsh tariff would pay customers $0/kWh for excess solar generation, while a generous tariff would pay market price.

We found that the HVAC heating schedule varied significantly according to different tariff structures under co-optimization, while, as modeled, the water heater operating schedule remained the same. We demonstrated that there are tangible, although relatively small, benefits of co-optimization (relative to parallel optimization) through reduced consumer costs under harsher tariff regimes, although little-to-no benefit under generous tariff regimes. However, under all tariffs, we found quantifiable grid benefits via a significant reduction of on-peak consumption.

These results may be used to better inform consumer energy device operation and electricity rate design, by quantifying the value and methods of co-optimization.

III. Introduction

A. Motivation & Background

There are several key changes happening on modern grids today: The rise of renewables, the decentralization of power generation (think rooftop solar, micro-grids), and the introduction of dynamic pricing schemes, such as time-of-use rates and demand response events. In parallel to these energy trends there has been an explosion in smart home and connected IoT devices. Taken together these energy and digital trends present an exciting opportunity.

Energy devices in the home may leverage intelligent control strategies to provide value to the occupant and the grid. The occupant may enjoy lower energy bills, improved comfort, and higher ease-of-use. And utilities, with a new grid asset, may call on customers to reduce their load in high-demand times to ease grid strain, and in the long term, defer capacity and transmission upgrades.

There are several smart energy devices on the market already, you can find smart thermostats (e.g., Nest, EcoBee), smart EV chargers, smart heat pump water heaters, and smart plugs.

But these devices operate in isolation and opportunities for coordinated control and co-optimization are underexplored. Now is the best time to explore these opportunities. At the end of 2019 Amazon, Apple, and Google all committed to use a standard communication protocol (Zigbee) [13]. And in February of 2021, Google reversed its prior decision to close off Nest's API to third parties. One rising cleantech startup – Span – is building out a smart electrical panel "so you can intelligently

monitor and control your home energy." Understanding what "intelligent" control should look like in the future will be critical, and is what we aim to explore in this project.

B. Relevant Literature

a) System Setup

There has been extensive research on household devices' energy consumption modeling. It is common practice for appliances to be categorized as interruptible, non-interruptible, and base with power ratings. Some papers only include the basic household devices like the washing machine, water heater, dishwasher, and air conditioner [1,15,17], while others incorporate more secondary devices like hair dryer, vacuum cleaner, coffee maker, etc., reaching over 10 total household appliances [3,6].

Energy generation and storage are also often included in system setups found in literature. [1,3,7,9] included models for renewable energy sources (PV solar panels and wind turbines), batteries, and electric vehicles (EV). [11] included a water tank as energy storage. [12] proposes and compares two system level designs i.e. home energy storage (HES) and community energy storage(CES). In both HES and CES systems heat and electricity storage are used for water and space heating utilities.

b) Optimization Methods

Optimization tools are broadly used across home energy modeling. In [4] and [7], mixed integer linear programming was used to optimize the electricity cost. In [11], a mixed integer nonlinear programming model (MO-MINLP) was used.

Optimization methods using meta-heuristics, that include random exploration and improving by exploitation are utilized continuously in recent years. In [3], the authors deploy the Butterfly optimization algorithm (BOA), a new optimization technique introduced by Aroa and Sing. BOA falls under the umbrella of swarm optimization algorithms, in which each agent shares its experiences with the other butterflies based on distributing the fragrance over the distance. In a similar approach, [1] proposes the dragonfly algorithm to balance the trade-off between exploration and exploitation. That balancing is achieved by emulating the static vs dynamic behavior of a swarm of dragonflies. During the static behavior, the swarm of dragonflies is organized in small groups over a specific area to make prey. In a dynamic swarm, a large number of dragonflies migrate from one place to another over a long distance in order to find the best habitat for their living.

c) Objective Functions

Throughout literature, there are mainly three types of objective function. The first and most common one is electricity cost minimization [1,5,6,7,9,11,15]. In these studies, dynamic pricing schemes were considered. In [9], the paper considers energy transfer both to and from the grid and minimizes the total energy cost of the smart home by buying and selling electricity to the grid under a dynamic pricing scheme. User comfort maximization was another optimization goal through modeling [3,4,5,6,11,15]. [11] integrated thermal comfort and waiting time of the users to optimize the energy usage. Peak load minimization was also considered in [4], which could avoid grid instability and possible grid failure.

C. *Study Focus*

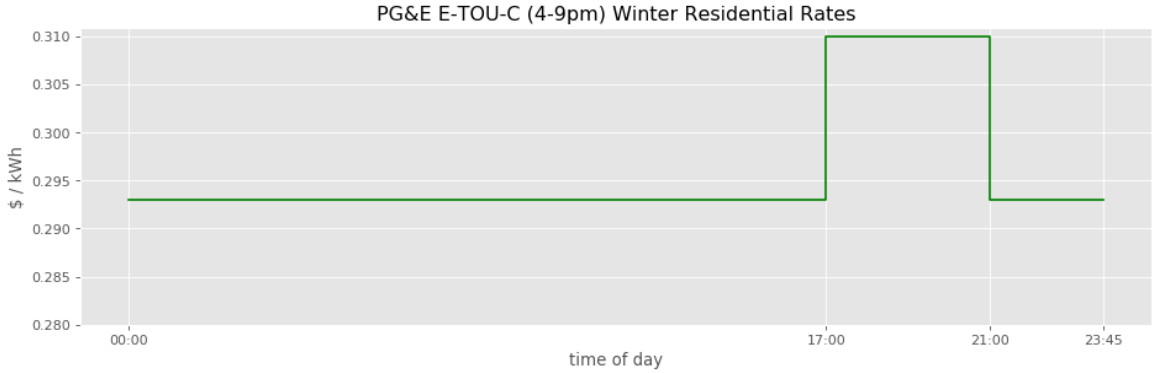
Our project will investigate opportunities for co-optimization between different home energy devices and appliances, aiming to minimize costs subject to device constraints. The goal is to understand and prove how homeowners can optimally reduce energy consumption from multiple sources in unison, reducing customer costs, flattening the electricity "duck" curve, and contributing to energy sustainability efforts.

Challenges to managing this particular energy system optimization project include over-simplifying constraints, accurately accounting for multiple objectives, and selecting the most optimal optimization method.

IV. Technical Description

A. *Time-of-Use Electricity Prices*

The PG&E residential time-of-use rate plan (E-TOU-C) was used which has lower prices before 4pm and after 9 pm ($0.293 / kWh) and higher prices from 4-9 pm ($0.31 / kWh).



B. *Solar Panel Modeling*

The solar photovoltaic (PV) power output was estimated using historic solar irradiation data and several assumptions about the configuration of the PV system. The 15-minute interval irradiation data was gathered from the National Renewable Energy Laboratory's (NREL) National Solar Radiation Database (NSRDB) [16].

The following equations were used to estimate the power generated by the PV system.

$$P_{solar} = (n_{panels} * A_{panel} * S_{module} * \eta)/1000$$

$$S_{module} = GHI * (sin(\alpha + \beta) / sin(\beta))$$

$$\alpha = 90 - \phi + \delta$$

$$\delta = 23.45 * sin(\frac{360}{365}(284 + d))$$

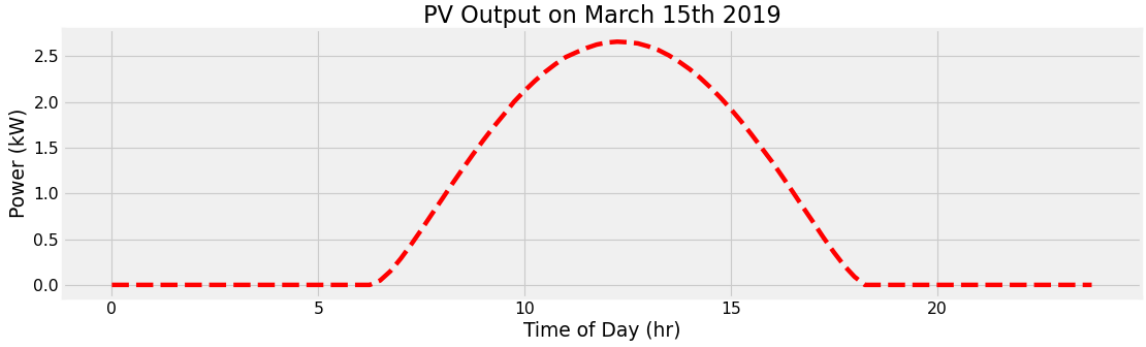
P_{solar} → is the power generated by the PV system $[kW]$

n_{panels} → is the number of PV panels [ ] ( )

A_{panel} → is the total area of each PV panel $[m^2]$

S_{module} → is the solar irradiation incident on each panel's surface $[W/m^2]$
η → is the efficiency of the PV [ ]
GHI → is the global horizontal irradiance $[W/m^2]$
α→ is the elevation angle [ ]
β→ panel tilt angle, measured from the horizontal [ ]
ϕ → is the latitude [ ]
δ→ is the declination angle [ ]
d→ is the day of the year [ ]

System parameter values:
n_{panels}= 8
A_{panel}= 1.63 m^2
η= 0.185
β= 45°



C. *HVAC Optimization*

[8] provided a model of HVAC using the area of windows, ceiling, and walls into account. The model aims to minimize the cost while maximizing the comfort level. [14] provided a simpler model compared to [8], with only the area of the walls considered. Meanwhile, [14] considered the wall temperature separate from the air temperature, and adding an extra layer on the thermal resistance of the outer wall and the inner wall. It also took thermal energy from the sun into consideration.

HVAC is one of the major energy consumption load in and household, hence for improving energy efficiency optimizing and scheduling the thermal load demand from the grid is necessary. We address the problem of optimal HVAC energy use at home while considering the thermal comfort zone of residents.

The thermal model for the house adopted incorporates different sources of heat gain and loss, making some simplified assumptions which lowers the complexity of the in-depth physical model while maintaining overall accuracy.

The following equation details the house thermal model considered:

$$C\frac{dT_{in}}{dt}=\frac{1}{R_t}(T_{out}-T_{in})+\emptyset_{H/A}+A(1-p)\emptyset_{ir}+n_{ac}V_{house}\rho_{air}C_2(T_{out}-T_{in})/3600$$

$C,\ C_2 \rightarrow$ is the capacity of the total indoor air's heat ,
$R_t \rightarrow$ indicates the total resistance against heat flow to the outside of the room or thermal energy from heater/air-conditioner ,
$T_{in} \rightarrow$ is the room temperature ,
$T_{out} \rightarrow$ is the outside temperature ,
$\emptyset_{H/A}$ is the thermal energy flow of the heater/air-conditioner ,
$A \rightarrow$ is the area of the window of the room ,
$p \rightarrow$ is the part of the irradiation of sun which is directly observed by inner layer of indoor walls [],
$\emptyset_{ir} \rightarrow$ is the thermal energy from sun ,
$n_{AC} \rightarrow$ number of air change

The main heat exchange sources considered are solar energy, thermal exchange with the outside environment due to temperature difference between inside and ambient environment and the heat exchange due to HVAC air circulation.

The above continuous state model is changed to discrete step state model by using zero order hold (ZOH) and the resulting difference equations obtained are :

$$T(t_{k+1}) = A_d T(t_k) + B_d u(t_k)$$

where $A_d = e^{A\Delta t} = e^{\left(-\frac{1}{CR}-\frac{n_{ac}V_{house}\rho_{air}C_2}{3600C}\right)\Delta t}$

$$B_d = \int_0^{\Delta t} e^{A\tau} B d\tau$$

where $\Delta t = t_{k+1} - t_k$

$$= \int_0^{\Delta t} e^{\left(-\frac{1}{CR}-\frac{n_{ac}V_{house}\rho_{air}C_2}{3600C}\right)\tau} \frac{1}{C}\left[\frac{1}{R}+\frac{n_{ac}V_{house}\rho_{air}C_2}{3600}1\ A(1-p)\right]d\tau$$

$$= \frac{1}{C}\left[\frac{1}{R}+\frac{n_{ac}V_{house}\rho_{air}C_2}{3600}1\ A(1-p)\right]\int_0^{\Delta t} e^{\left(-\frac{1}{CR}-\frac{n_{ac}V_{house}\rho_{air}C_2}{3600C}\right)\tau} d\tau$$

$$= \frac{1}{C}\left[\frac{1}{R}+\frac{n_{ac}V_{house}\rho_{air}C_2}{3600}1\ A(1-p)\right]\frac{1}{\left(-\frac{1}{CR}-\frac{n_{ac}V_{house}\rho_{air}C_2}{3600C}\right)}\left(e^{\left(-\frac{1}{CR}-\frac{n_{ac}V_{house}\rho_{air}C_2}{3600C}\right)\Delta t}-1\right)$$

$$u(t_k) = \begin{bmatrix} T_{out}(t_k) \\ \emptyset_{H/A}(t_k) \\ \emptyset_{ir}(t_k) \end{bmatrix}$$

It can be observed from the input vector above, we have two exogenous input i.e. $T_{out}(t_k)$ and $\emptyset_{ir}(t_k)$, which depends on the weather data of the day and one controllable input $\emptyset_{H/A}(t_k)$, which is what we are trying to optimize.

The energy consumption of the HVAC thermal energy (cooling/ heating) supplied and can be written as:

$$\emptyset_{H/A} = P_{heat} z_{heat} + P_{cool} z_{cool}$$

Where the heating and cooling power of the HVAC system i.e. (P_{heat} and P_{cool}) is fixed and the variable that determines the energy consumption is z_{heat} and z_{cool} that works as and operation status switch for HVAC (z_{heat} = 1 , when heating is on , otherwise 0 , similarly and z_{cool}=1 , when cooling is on and 0 otherwise).
The day chosen for the analysis done in this study March 15 2019, which is a relatively cold day as can be seen from the weather data, hence only heating load was considered in the HVAC system.

$$\emptyset_{H/A} = P_{heat} z_{heat}$$

The optimization objective takes the energy consumption of the HVAC and minimizes the cost associated with operating cost of the HVAC.

$$min\, z(t) \sum_{t=1} p(t).P_{heat} z(t)$$

As reported in [14], the human thermal comfort level is between 18◦ C to 24◦ C and optimum temperature is 21◦ C. To insure that indoor temperature is maintained within this comfort zone, additional constraint, i.e., comfort constraint is also considered.

$$\left|T_{in} - T_{set}\right| \geq \Delta T_{thers}$$

T_{set} is taken as 21◦ C ΔT_{thers} is taken as 3◦ C. Hence in the HVAC optimization the following constraints are applied:

$$T\left(t_k\right) \leq T_{max}, T\left(t_k\right) \geq T_{min}$$

The physical values of the parameters, taken in the HVAC models are summarized in the table below and are referred from [8],[14].

Physical Parameters	
Capacity of the total indoor air's heat	C=8.12 [kWh/°C] $C_2=1214.4/\rho_{air}$ [J/(kg*°C)]
Heating power of the HVAC [kW]	$P_{heat}=60*A_{floor}/1000$
Total resistance against heat flow [°C/kW]	$R_t=8.02$
Area of the floor/window [m²]	$A_{window}=4$ $A_{floor}=100$
Volume of the house [m³]	$V_{house}=300$
Part of the irradiation of sun directly observed by window	P=0.995
Air density [kg/m³]	$\rho_{air}=1.2$
Number of air change of the house [1/hr]	$n_{air}=0.35$
Minimum temperature	$T_{min}=20$
Maximum temperature	$T_{max}=24$
Initial temperature	$T_o=21$

D. Water Heater Optimization

In modeling the electric water heater (EWH) device, we based our work off [17] and utilized a 15 minute time step. We only modeled the cold water that enters the tank and mixes with present hot water (red outline in Fig. 1); thus, we neglect water mixing post-heating and assume the output water temperature is satisfactory for the user.

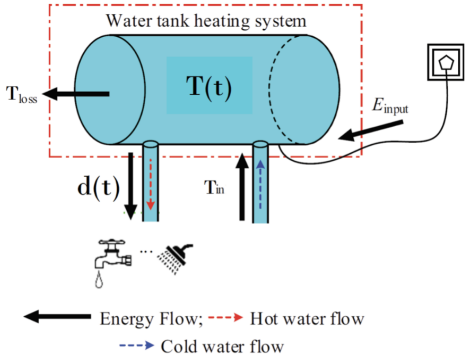


Fig. 1: Energy Flow inside Electric Water Heater

We represent the usage of a 50 gallon water tank for two, discrete tasks: a shower and washing machine cycle. Each water demand has a constant, predetermined flow rate and usage time; the shower (1.6 gpm) runs for 15 minutes from 9-9:15 AM and the washing machine (2.0 gpm) runs for 45 minutes from 6-6:45 PM.

The water inside the EWH is treated as a single body with uniform temperature and instantaneous heat transfer. Applying energy conservation and thermodynamic principles, the temperature inside the water heater can be determined and controlled via multiple constraints and a penalty function.

The following three constraints were employed in the optimization:

1. The initial temperature inside the water heater is a predetermined value T_o.

$$T(0) = T_0$$

2. The temperature inside the water heater must never exceed a predetermined maximum temperature T_{max}.

$$T(t+1) \leq T_{max}$$

3. The temperature inside the water heater can be expressed in state space form and accounts for hot water demands d(t) as well as water heater input temperature T_{in} effects. The state and input matrices model the thermodynamics of the system, with beta representing the input matrix importance factor, which accounts for feasibility.

$$T(t+1) = \underbrace{e^{\frac{-\Delta t}{RC}}}_{A}\left[\frac{M-d(t)}{M}T(t) + \frac{d(t)}{M}T_{in}\right] + \underbrace{\beta QR\left(1-e^{\frac{-\Delta t}{RC}}\right)}_{B}u(t)$$

The objective function employed in the optimization can be decomposed into cost and penalty components. The cost component models the pricing of utilizing the water heater based on TOU pricing p(t). The penalty component penalizes the system when the temperature inside the water heater falls below a predetermined minimum temperature T_{min}; a penalty function was employed rather than a hard constraint, as a hard constraint drove the problem to become infeasible due to thermodynamic constraints. The optimization variables are the temperature inside the water heater T(t) and the state (on/off) of the water heater u(t).

$$\min_{u(t),T(t)} \sum_{t=1}^{n} \underbrace{(p(t)\cdot Q\cdot u(t)\cdot dt)}_{\text{Cost Function}} + \underbrace{\frac{1}{2}\alpha\left[\max_{T(t)}(0,-T(t+1)+T_{min})\right]^2}_{\text{Penalty Function: } T(t+1) \geq T_{min}}$$

The physical parameters, variables, and factors we incorporated into our optimization (referenced above) are as follows:

Physical Parameters	
Thermal Resistance	R = 1.52 °C/kW
Thermal Capacitance	C = 863.4 kWh/°C
Heat Capacity	Q = 4 kW
Heater Mass	M = 50 gal
Initial Temperature	T_o = 60 °C
Minimum Temperature	T_{min} = 55 °C
Maximum Temperature	T_{max} = 65 °C
Input Temperature	T_{in} = 25 °C

Factors	
Penalty Function	α = 1
Input Matrix	β = 10,000

Variables	
Temperature [°C]	T (t)
State [0,1]	u(t)
TOU Pricing [$/kWh]	p(t)
Demand [gal]	d(t)

E. Co-Optimization

In the sections above we have generated an optimal schedule for each device separately. We will now incorporate all three devices (PV, HVAC, water heater) into our optimization.

To solve this problem, we defined a new vector variable, P_{grid} representing the net power (kW) coming from the grid to the home in 15-minute increments:

$$P_{grid} = P_{wh} + P_{hvac} - P_{solar}$$

Where

$$P_{wh}(t) = Q \cdot u(t)$$

$$P_{hvac} = P_{heat} \cdot z(t)$$

We then defined a new total cost vector, C_{grid} where the price per kWh, $p(t)$to consume electricity remains the same, but the price paid for electricity exported to the grid is scaled by β.

$$C_{grid}(t) = \begin{cases} P_{grid}(t) \cdot p(t) \cdot \Delta t & if\ P_{grid}(t) \geq 0 \\ \beta \cdot P_{grid}(t) \cdot p(t) \cdot \Delta t & if\ P_{grid}(t) \leq 0 \end{cases}$$

$$\beta \in [0, 1]$$

We then defined our objective function to be

$$min \sum_{t=1}^{n} C_{grid}(t)$$

We were able to transform this into a solvable system by introducing a slack variable, g and introducing the piecewise function into the constraints. By plugging in different values for βwe were able to evaluate how different tariff schemes impacted our optimization (see Results section).

$$min \sum_{t=1}^{n} g(t)$$

Where

$$g(t) \geq P_{grid}(t) \cdot p(t) \cdot \Delta t \ \ \forall\, t$$

$$g(t) \geq \beta \cdot P_{grid}(t) \cdot p(t) \cdot \Delta t \ \ \forall\, t$$

$$\beta \in [0, 1]$$

V. Results

A. Water Heater

After performing the optimization, as shown in Fig. 2 (left), we observe that the water heater was turned on three times (9.00 am - 9.15 am and 6.00 pm - 6.30 pm) and the activity was related to the water demand from the shower happening from 9 am to 9.15 am and the washing machine between

6.00 pm and 6.45 pm. The water temperature inside the electric heater, as displayed in Fig. 2 (right), barely dropped below the minimum temperature of 55 °C, meaning the implementation of the penalty function to handle the desired minimum temperature constraint performed extremely well. That is in spite of the large volumes of water proportionally extracted from the tank for the activities and instantaneously replaced with water of equal volume but of much lower temperature (25 °C). At the same time our maximum temperature constraint of 65 °C is satisfied throughout the day.

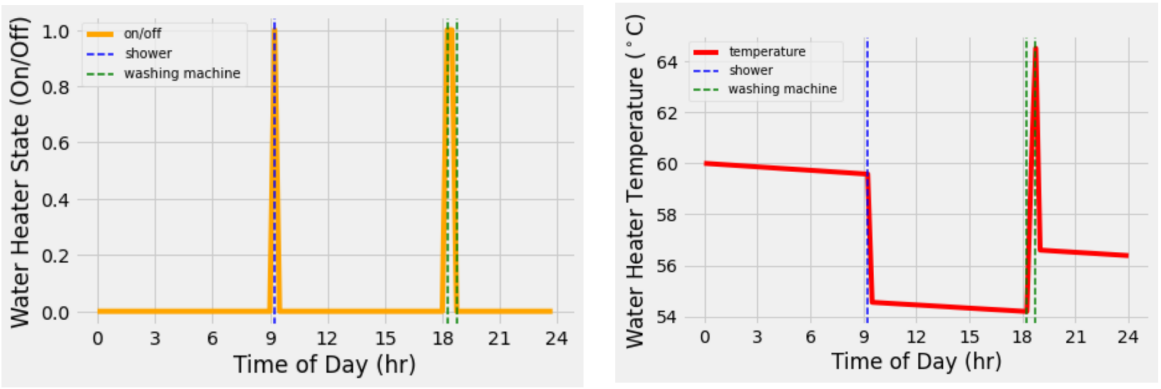


Fig. 2. Left:Water Heater State per Time of Day, Right: Water Heater Temperature per Time of Day

B. HVAC

The optimization result of the HVAC system is shown in Fig. 3. In the HVAC model, the optimal temperature was set between 21 °C and 24 °C. The initial temperature of the house was set to be 21 °C. After performing the optimization, we found that HVAC was not turned on till 3:15 am because the initial temperature was high compared to the minimum set temperature. During daytime from 9.00 am to 6.00 pm, the HVAC was turned on less frequently compared to the rest of the day because of external heat from the sun. There is a larger jump in temperature and longer heating at 4:30 pm. This is because HVAC was trying to heat more to overcome the temperature loss and increase in unit price between 5pm and 9pm. The utility price of the HVAC for the day was $7.99.

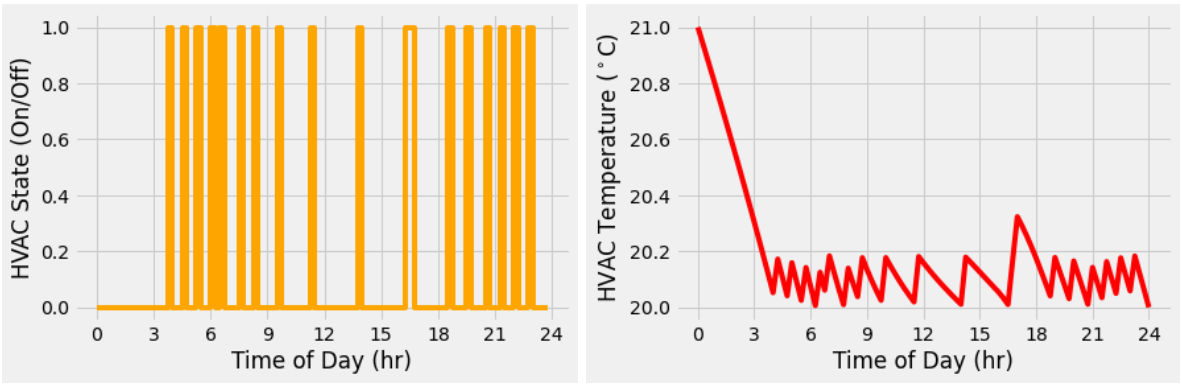
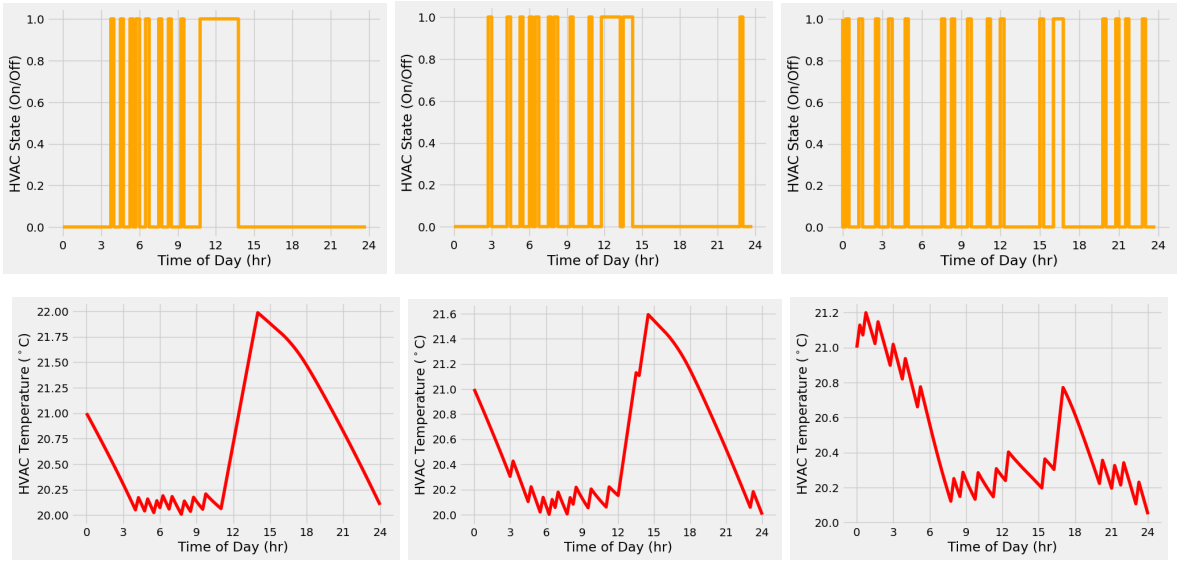


Fig. 3. Left: HVAC State per Time of Day, Right: HVAC Temperature per Time of Day

C. Co-Optimization

In Sections A and B, each device made decisions in isolation. Here we show that the solar output and tariff structures have real impacts on the optimal operating times for HVAC.

The opportunities for co-optimization are defined by the tariff structures they operate within. If excess solar exported to the grid has no value, then the optimization will strongly favor directing that solar-generated electricity to in-home devices (see 0% in Fig. 4). And, on the opposite side, if exported electricity is paid full market value (i.e. the price to export is equal to the cost to consume), then there is little-to-no-incentive to alter home device consumption to align with the times of PV output (see 100% in Fig. 4).



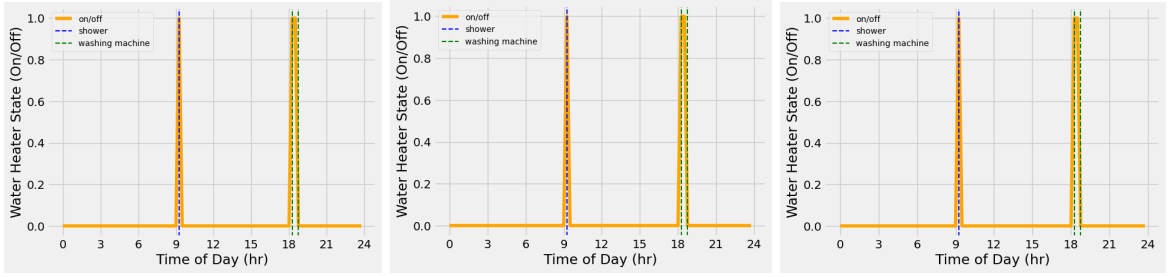
β =**0%** β =**50%** β =**100%**

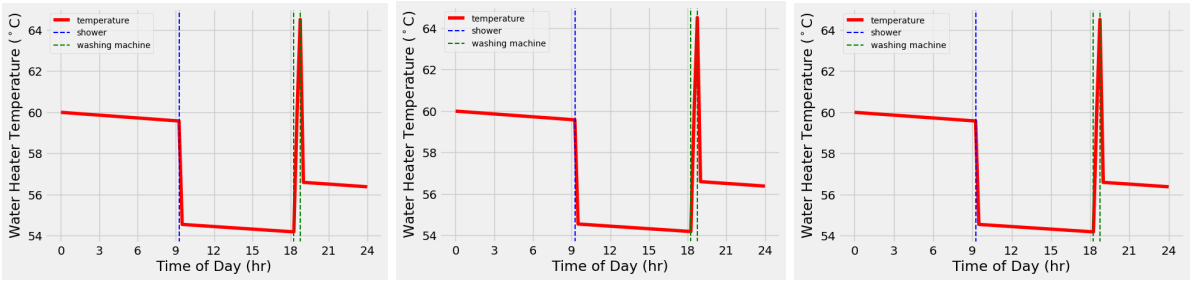
(Excess solar has no value) ----------> (Excess solar is paid full grid price)

Fig. 4. Top: HVAC State at Different % Paid for Exported Solar

Bottom: HVAC Temperature at Different % Paid for Exported Solar

While co-optimization significantly alters the control schedule for HVAC as βchanges, we see no impact on the water heater's state. This is likely because the water heater is quite powerful, and so there is little room to preheat water without exceeding the maximum temperature (65° C). This is especially true given our duty cycle period (15 minutes) which is high for a water heater.





β =**0%** β =**50%** β =**100%**

(Excess solar has no value) ----------> (Excess solar is paid full grid price)

Fig. 5. Top: Water Heater State at different % Paid for Exported Solar

Bottom: Water Heater Temperature at different % Paid for Exported Solar

Above we explore how co-optimization impacts devices' schedule. We can also explore the broader impacts on customer cost and on the grid.

In Fig. 6, we see that customer cost decreases linearly with β - the higher the price paid for excess solar, the lower the cost for the customer.

But, we see that as βincreases, so does on-peak electricity consumption. However, the difference is relatively small, there is less than a 0.3% jump in on-peak consumption between $\beta = 0\%$and $\beta = 100\%$. Also note that the relationship is non-linear - the on-peak percentage is near constant until $\beta \approx 60\%$, at which point the on-peak percentage jumps by ~0.28% .

Fig. 6 shows the tradeoff between consumer and grid interests for different β*s*.

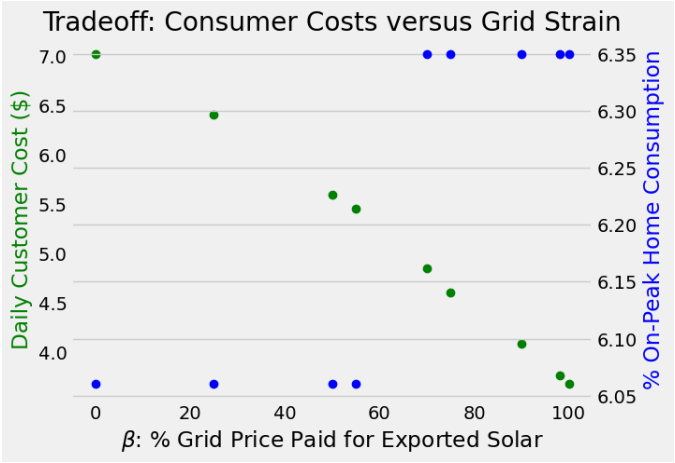


Fig. 6. Customer Costs and On-Peak Consumption vs β

Here we explore the net value of co-optimization relative to parallel optimization. We found that there was a slight net consumer benefit of co-optimization, with the most significant value at lower β values (see Fig. 7). Surprisingly, we found that co-optimization underperformed parallel optimization at higher βvalues, however this was likely due to the slight differences in optimization solvers used for co-optimization (MOSEK) and parallel optimization (XPRESS).

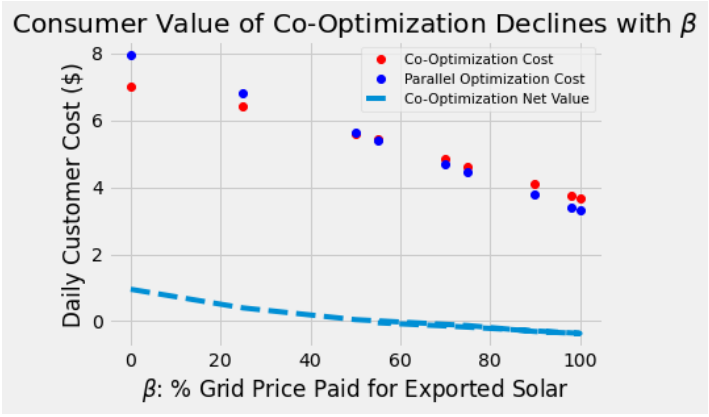


Fig. 7. Daily Customer Costs under Co-Optimization and Parallel Optimization vs β

Although the consumer cost benefit of co-optimization was only slight (and even non-existent at higher βvalues), we see substantial grid benefits - the reduction of on-peak device consumption is significant and consistent across all values of β(Fig. 8).

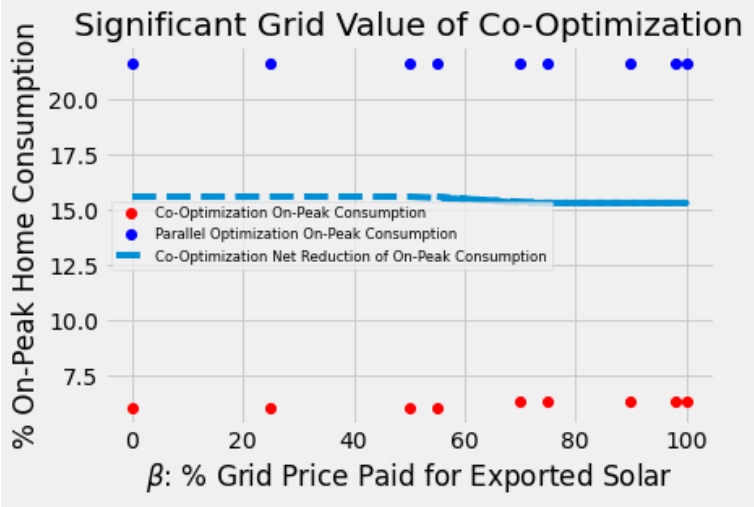


Fig. 8. On-Peak Consumption under Co-Optimization and Parallel Optimization vs β

VI. Discussion

We demonstrated how to model and design an optimal schedule for HVAC and water heater systems.

We also demonstrated that incorporating PV solar output and tariffs into a co-optimization model can significantly affect when devices use electricity, the total cost paid by the consumer, and the on-peak electricity consumption. These results are informative in two key ways:

(1) Automated Decision-Making for Consumers: A centralized decision-making algorithm, with information from the PV system and all home energy devices, has the potential to make more intelligent decisions on behalf of the consumer.

(2) Tariff Design: There is ongoing discussion in California about the future of solar tariffs (i.e. NEM 3.0). Our methodology and results provide useful context to inform tariff design and show

that decreasing the amount paid for excess solar will increase consumer costs while only slightly reducing customers' on-peak consumption.

Improving both automated decision making and tariff design will help to enable the future of low-carbon, flexible, and decentralized electricity grid.

Future Steps

- Adjust time steps to better reflect the duty cycle of the water heater.
- Investigate and adjust HVAC inputs. The system modeled may be undersized.
- Explore implementation of real-time control and optimization.
- Model and incorporate other smart-home energy devices, such as dishwasher and air conditioner.
- Expand model to include additional times and locations.
- Introduce uncertainty into demand of devices to create a more robust, realistic model and framework.

VII. Summary

This project aimed to co-optimize home energy appliances and devices, and resultantly produce a daily cost-optimal load profile for various energy devices in one's home. This study explored the methods and benefits of co-optimization relative to parallel optimization.

In the first part of the study we modeled three devices – HVAC, a water heater, and a solar photovoltaic (PV) system. For each of these devices we used assumed certain equipment characteristics and user consumption patterns.

Using mixed integer programming (MIP), we optimized the operation of the HVAC and water heater by minimizing the total daily cost using a PG&E time-of-use (TOU) residential rate plan and generated the appropriate daily load profile for each device.

In the second part of the study, we performed a co-optimization, where we minimize total costs under different tariff structures – a harsh tariff would pay customers $0/kWh for excess solar generation, while a generous tariff would pay the market price for exported solar electricity.

We found that the HVAC heating schedule varied significantly according to different tariff structures under co-optimization, while, as modeled, the water heater operating schedule remained the same. We demonstrated that there are tangible, although relatively small, benefits of co-optimization (relative to parallel optimization) through reduced consumer costs under harsher tariff regimes, although little-to-no benefit under generous tariff regimes. However, under all tariffs, we found a significant reduction in on-peak electricity consumption, suggesting that co-optimization may provide substantial grid benefits over a parallel optimization.

These results may be used to better inform consumer energy device operation, by quantifying the value and methods of co-optimization. Similarly, the results can inform tariff and rate structure design by quantifying the tradeoffs of consumer cost and grid value under different cost regimes.

References

[1] Hussain et al., Optimizing energy consumption in the home energy management system via a bio-inspired dragonfly algorithm and the genetic algorithm

[2] Zupancic et al., Genetic-programming-based multi-objective optimization of strategies for home energy-management systems

[3] Wang et al., A multi-objective home energy management system based on internet of things and optimization algorithms

[4] Yahia et al., Multi-objective optimization of household appliance scheduling problem considering consumer preference and peak load reduction

[5] Benhmed et al., Novel Home Energy Optimization Technique based on Multi-Zone and Multi-Objective Approach

[6] Mouassa et al., Scheduling of smart home appliances for optimal energy management in smart grid using Harris-hawks optimization algorithm

[7] Dutra et al., A realistic energy optimization model for smart-home appliances

[8] Makhadmeh et al., Optimization methods for power scheduling problems in smart home: Survey

[9] Chandra et al., Energy Management of Smart Homes with Energy Storage, Rooftop PV and Electric Vehicle

[10] Perry et al., Grid-Interactive Efficient Building Utility Programs: State of the Market

[11] Anvari-Moghaddam et al., Optimal Smart Home Energy Management Considering Energy Saving and a Comfortable Lifestyle

[12] Terlouw et al., Optimal energy management in all-electric residential energy systems with heat and electricity storage

[13] Apple et al., Amazon, Apple, Google, Zigbee Alliance and board members form working group to develop open standard for smart home devices

[14] Shakeri et al., An intelligent system architecture in home energy management systems (HEMS) for efficient demand response in smart grid

[15] Wu et al., Multi-objective optimization for electric water heater using mixed integer linear programming

[16] National Renewable Energy Laboratory, NSDRB Data Viewer

[17] Du and Lu, Appliance Commitment for Household Load Scheduling

Clustering Household Energy Consumption

Team 4 (Blue)
Jackson Le, Alvin Zhou, Hamza Chaquiq Elbadre, Michael Wehrmeyer, Ursan Tch[outeng Njike

Abstract

Households account for a considerable amount of the global energy consumption in developed countries. Through the individualized clustering of households and their power consumption can assist in optimizing power consumption to be both cheaper, and more carbon efficient. Global energy demand is increasing every year, and there is an rising need to optimize the distribution of energy to reduce both hazard and cost for the consumer and producers. This can be achieved by providing an estimate of how much energy a household may consume with generalized load profiles. These generalized load profiles can provide straightforward insights on the power demand distributions in an energy system. In turn, gaining a better understanding of the load distribution within an area will make it feasible to create better energy distribution methods and to solve optimization problems that can tackle issues such as energy price and carbon production.

Introduction

(a) Motivation & Background:

40% of the primary energy produced is consumed by buildings in european countries and the USA [6]. Due to the exponential population growth, energy outages are a growing concern as household demands in energy are in constant increase. In addition, lighting and HVAC operating systems account for approximately 90% of building emissions[6]. Households however may have different consumption patterns and a traditional grid fails to capture that. The first step towards a smart grid would be to identify these patterns accurately to better understand the consumer demands and different time of the day.

Determining characteristic load profiles that describe behaviors and lifestyles could help energy companies create different programs for different typical demands. These could include building a more robust grid design system, offering more effective energy reduction recommendations, and improving pricing models. These profiles are also an important first step in solving optimization problems formulated using large amounts of time-series input data [4].

A significant challenge in pursuing this type of analysis is the scarcity of household energy data that is open-sourced. Most available datasets are either limited in the number of households, the length of time where data is gathered, or both. This limits the widespread application of cluster analysis as there is not enough data from one source to create a clustering system that is easily extensible outside of the scope of this project.

(b) Relevant Literature

There exists substantial literature on the application of clustering methods for summarizing time usage profiles from energy usage data. For example, Lavin et al. utilized partitioning-based clustering methods such as k-means to group customer energy use data into distinct profiles [8]. Their results indicated that comparing individual profiles with the representative profile they were clustered with can yield insights on how individual customers can improve their energy use behavior. The authors also noted that energy use profiles vary from month-to-month, meaning that clustered profiles should only be compared with others that are in the same season. Zhang et al. [1] and An et al. [3] also utilized these distance-based clustering methods to analyze an energy use dataset in Austin, Texas and Zhengzhou, China, respectively.

Ruiz et al. demonstrate the use of hierarchical clustering to group hours of the day into clusters based on energy usage [5]. Their results indicated three major clusters in each day: one for early morning hours, another for midday, and one for the evening. In addition, their exploration of their energy use dataset found significant correlation between energy use on weekdays, where most people work during the day, and another link between energy usage on the weekends. These results point to important recurring relationships such as the weekday-weekend dichotomy and/or seasonal trends that must be considered when interpreting the results.

Motlagh et al. applies PCA and two clustering techniques to characterize univariate electricity consumption behaviors in houses with solar panels [2]. These techniques are used to understand different aspects of behavioral patterns: PCA for qualitative “feature extraction from households’ diurnal load curve”, a Hebbian neural network to “identify clusters with dominant behaviors, and SOM to examine “general and specific behaviors” by “decomposing time and frequency components”. PCA revealed many levels of general behaviors, and found that consumption of solar-generated energy was a dominant behavior. The Hebbian technique fails to find a similar trend, although the influence of consumption of self-generated energy can be seen in the patterns. In this section on the Hebbian method, the authors determined that clustering on a specific time segment is not promising. In comparison to the Hebbian method, the SOM method successfully clusters diurnal curves clearly by decomposing different time components into “tasks”.

The work by Teichgraeber et. al indicates a potential use for clustering energy use profiles beyond visualization and interpretation of usage trends. They proposed that representative usage profiles derived from clustering methods can simplify energy systems optimization problems [4]. Instead of optimizing for every customer in the system, the optimization can be performed on the representative in each cluster to reduce the problem scale and complexity. The solution for the representative profiles can then be applied to the individuals in each cluster. Because of the variance in their optimization results when applying their clustering framework, they indicate that additional work is needed to identify the ideal clustering methodology for a given dataset and optimization problem.

(c) Focus of this study

In this report a variety of clustering algorithms are applied to identify representative household usage patterns based on clustered energy consumption profiles. Each method is evaluated based on cluster stability and predictive accuracy through the silhouette scores and cluster distortions, respectively. The cluster results will also provide insights on general patterns on household energy use in the dataset.

Technical Description

(a) Dataset

The data used in this analysis is sourced from the Pecan Street dataset, which contains the data of 25 homes from Texas, California, and New York and their grid energy consumption. Included with the energy use profiles are dummy variables with survey question answers about their air conditioning, dishwashers, solar panels, energy-intensive appliances and means of power generation. Although the dataset includes data on energy usage from three US states, the clustering methods were only applied to the New York dataset because of the lack of time consistency in the other two datasets. The California data is not of the same timescale, and includes measurements sporadically from different years. The data from Texan households contained outliers that made it difficult to identify the proper data to include in the cluster analysis. To reduce the dimensionality of the time series profiles, energy usage data was taken in 15 minute time intervals instead of the 1 second or 1 minute dataset that Pecan Street also provided.

The data provided is given to us with a unique ID for each household along with a timestamp containing the amount of energy that was pulled from the grid and given to the household with a 99% completeness for the Texas dataset and a 100% completeness for the New York dataset. The missing data points were interpolated using a 5th order polynomial. While this may generate data that may not agree completely with the true, unknown energy consumption, this was required to ensure that every household included had a recorded measurement for the entire timespan of the dataset.

After all the missing values were interpolated, the data was reorganized to streamline the process of exploratory data analysis (EDA). The time series energy data was separated by household and the day of measurement. This was implemented using a Pandas multi-index, with a unique key for each combination of household id and date. Every key then corresponded to 96 energy usage measurements across the 24 hours in each day. This format allowed for the extraction of daily usage profiles for every household in the dataset. These profiles are referred to as time series profiles or energy usage profiles in the rest of the report.

(b) Exploratory Data Analysis

The preliminary visualizations of the data are shown below. Figure 1 shows the histogram of the frequency of maximum power draw from the energy grid. The figure shows that the surveyed New York homeowners on average draw the most power from the grid between 4 PM to around 11 PM, with a peak at around 8 PM. Figure 2 describes the consumption of households over May 2nd, 2018. The energy consumption plot for a single day indicates that there are two distinct sets of households: those that only consume energy from the grid, and another set that is able to generate power for the grid using installed solar panels. The survey data was then split among these two sets to identify any general profiles inside them using cluster analysis.

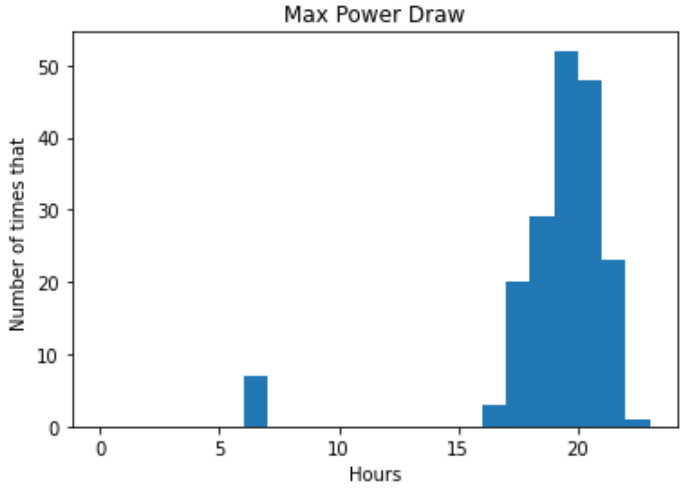


Fig. 1 Mode of peak energy consumption at specific time of the day (NY 2019 Data).

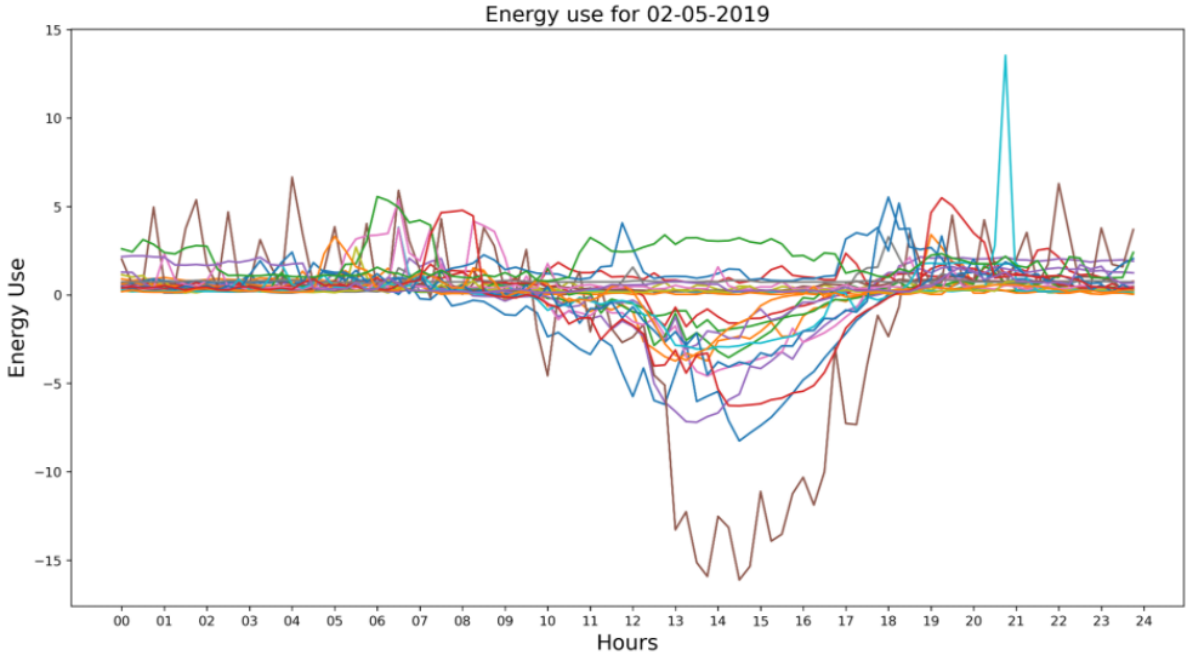


Fig. 2 Energy consumption throughout the day (NY 2019 Data).

(d) Models

The cluster analysis framework can be summarized in three steps: first, the dataset is normalized and then clustering models are applied to identify potential clusters. Finally, the clustering results are visualized and interpreted.

Prior to applying any clustering models, the raw data needed to be pre-processed through normalization to bring the magnitudes of different time-series to similar levels. The need for normalization is illustrated in Fig 2. There are several households with similar usage patterns but have different magnitudes of energy usage. Applying cluster analysis to these households may yield results that place them in separate clusters, which would be undesirable. To remedy this issue, the time series profiles for each household and day were grouped by the household and the day of the week. Then, each profile was scaled such that the maximum and minimum energy consumption in the group would be 1 and -1, respectively. This meant that a different scaler was applied to each household for every day of the week.

Afterwards, the clustering methods are applied and provide clusters with a representative time-series for each cluster. These steps are summarized in the following schema from Teichgraeber et al. [4].

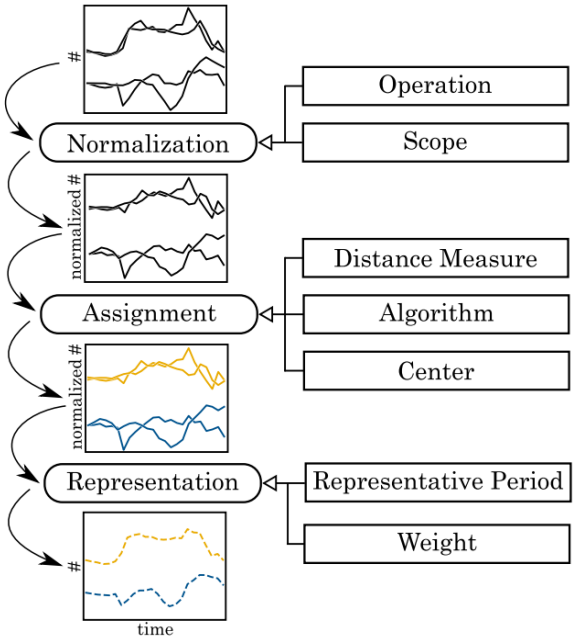


Fig 3. Schematic representation of the model approach [4].

The next step is to define a distance measure to quantify the similarity between sequences as shown in equation 2. A well known family of distance measures is the Minkowski metric that is used in [5]:

$$dist(x,\ y)\ =\ \left(\sum_{i=1}^{n}\left|x_i - y_i\right|^p\right)^{1/p} \quad (1)$$

Selecting $p = 2$ yields the commonly used Euclidean distance metric, while $p = 1$ generates the Manhattan distance metric. Setting $p = \infty$ calculates the maximum distance between corresponding elements in x_i and y_i.

The models that will be used can be divided into two groups: Partitional clustering and hierarchical clustering algorithms [4]. The first group of methods differ in the choice of the distances and cluster centers, but they follow the same algorithm:

1. Cluster centers z are initialized randomly.
2. Each sequence x is assigned to the closest center.
3. Update the centers.

The steps 2 and 3 are repeated until convergence or attaining the maximum number of iterations. The centers are defined as follows:

$$c_k = argmin_{z \in Z} \sum_{i \in C_k} dist(x_i, z)^2 \quad (2)$$

In the case of k-medoids: $Z \in \{x_1, ..., x_n\}$. In the case of k-means: $Z \in \Re^T$ where T is the total number of timesteps.

The hierarchical clustering algorithm starts with $k = N$ clusters, where N is the number of observations in the dataset. The closest clusters are merged together until the desired number of clusters is obtained. The clusters to be merged are decided by minimizing the total within-cluster variance (Ward linkage), however many other linkage criteria exist such as single linkage (merge clusters with two closest points) or complete linkage (merge clusters with minimum distance of the two farthest points). The couple of centers (c_i, c_j) that minimizes the euclidean distance is merged.

For DBScan, the algorithm takes in two inputs, a maximum distance and a minimum cluster size. Then using the Haversine distance between the items within the cluster, DBScan separates the items according to the distance between the centroid of the cluster and the resulting items clustered with it. If there is not a minimum number at least equal to or greater than the minimum cluster size, the items within the new found cluster will be regarded as noise and will not be labeled. The ideal use case for this method is when there are spatial features that are not well understood by other methods such as k-means, k-medoids, or hierarchical clustering.

(e) Metrics

Silhouette score

The silhouette score is calculated for each data point used in the cluster analysis. For each data point x_i in the set of observations I, the equations used to calculate the silhouette score are:

$$a(x_i) = \frac{1}{|C_i| - 1} \sum_{x_j \in C_i, i \neq j} d(x_i, x_j) \tag{3}$$

$$b(x_i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{x_j \in C_k} d(x_i, x_j) \tag{4}$$

$$s(x_i) = \frac{b(x_i) - a(x_i)}{max\{a(x_i), b(x_i)\}} \tag{5}$$

Equation (3) represents the mean distance from the data point to all other points in the same cluster. Equation (4) represents the mean distance from the data point to all other points in the next nearest cluster. Equation (5) combines the first two equations to calculate a score between -1 and 1. The total silhouette score is then calculated by taking the average of all the silhouette scores for the data points. Scores closer to 1 indicate a good fit for the assigned cluster labels, while scores closer to -1 may indicate mismatched clusters

Distortions

The distortion score is determined by summing the squared distances between the data points and the center of the cluster it is assigned to. The metric used can be any valid distance metric, but for the case of the Euclidean distance, the distortion for a cluster classification is given in Equation (d). Here, c_i represents the center of the cluster that the data point x_i is assigned to.

$$D = \sum_{x_i \in I} ||x_i - c_i||_2^2 \tag{6}$$

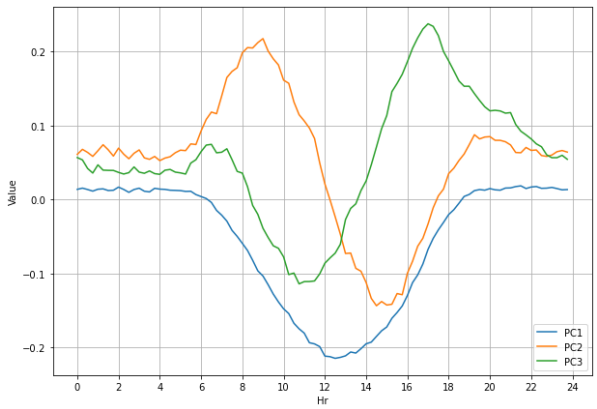
Discussion

(a) Principal Component Analysis (PCA)

Principal component analysis was used to collapse dimensionality of data, reducing noise in data for clustering, and to capture higher-level energy consumption trends. PCA uses linear algebra to identify hyperplanes through the 96-dimensional dataset that describes the most variance in the data.

The higher level trends of the non-normalized data, shown in the figure below, are particularly interesting. For the houses with solar panels, the first PC describes energy generated when the sun is out, and the second and third PCs describe aggregate energy consumed during one part of the day and generated during the other part. These first three principal components describe approximately 70% of the variance in the data. For the houses without solar panels, a regular amount of power consumed throughout the day describes the most variance, with the 2nd and 3rd PCs describing increases in consumption later in the day. These first three principal components describe approximately ~55% of variance in the data.

PC of Houses with Solar



PC of Houses without Solar

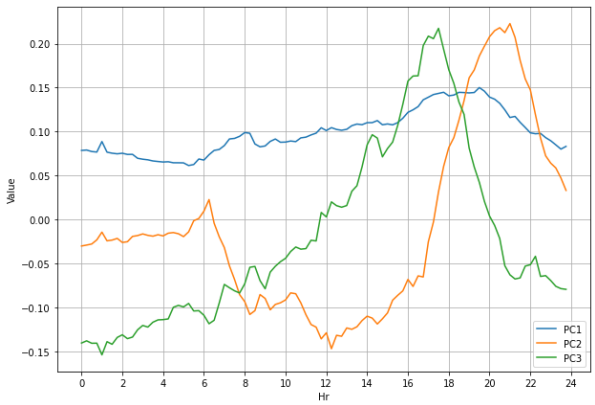
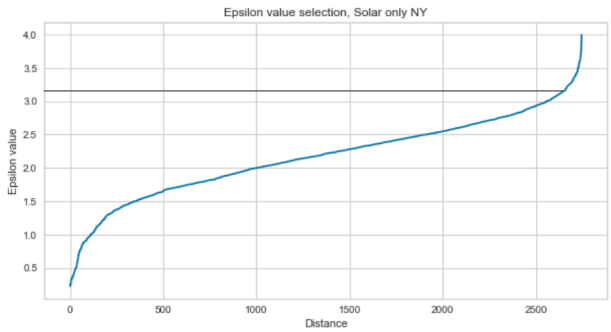


Figure 4: Principal components of houses with and without solar panels.

(b) DBScan

DBScan did not do particularly well for clustering the data both with and without separation of the homes with and without solar panels. In the case with the solar panels, DBSCAN found only one cluster each. This meant that DBScan failed to find any type of distinction between any of the homes in the dataset. This was also observed in the dataset where the homes were not separated, and varying the epsilon value created more noise points than cluster the values into meaningful clusters due to the nature of DBScan using the haversine distance between the clusters. The results did not change with applying PCA to the dataset to reduce the amount of variables that the analysis had to look at.



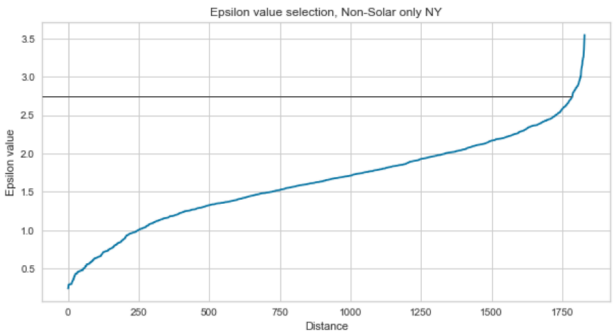


Figure 5. Selecting the optimal epsilon value for DBSCAN.

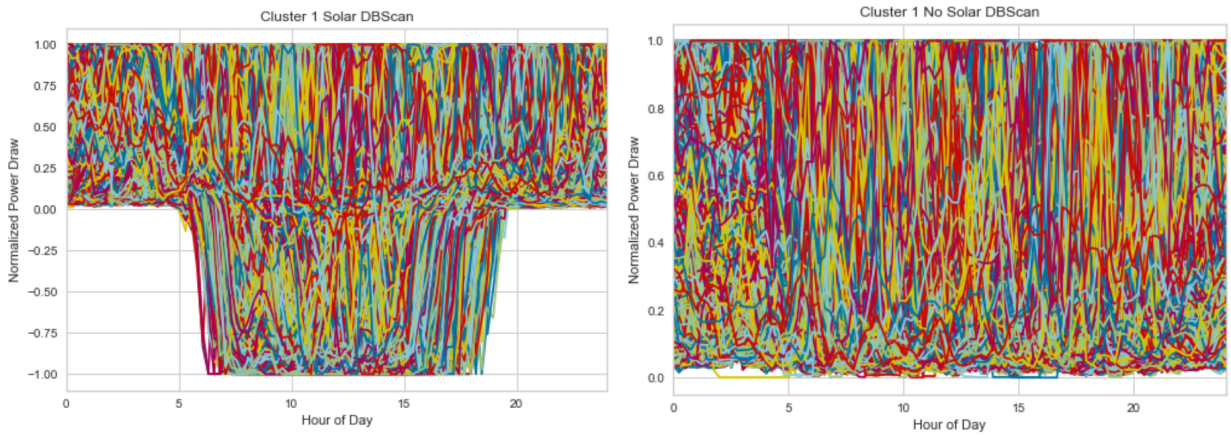


Figure 6. Resulting clusters from the selected epsilon values.

(c) K-means clustering

Unlike DBSCAN, the number of clusters produced in the analysis is predetermined. In this case, the ideal number of clusters was determined by applying the elbow method to plots of the distortions and silhouette scores versus the number of clusters. However, this method does not calculate the optimal number of clusters: it is a heuristic designed to provide a reasonable estimate of the optimal number. Figure 7 shows the distortion elbow plots used to determine the number of clusters for the dataset with solar panel-equipped households. The remaining silhouette score plot and the other pair of elbow plots for non-solar installed households is included in the appendix. PCA was used before k-means in a separate analysis. Similar clustering results and higher distortion scores lead us to present the k-means clustering without PCA.

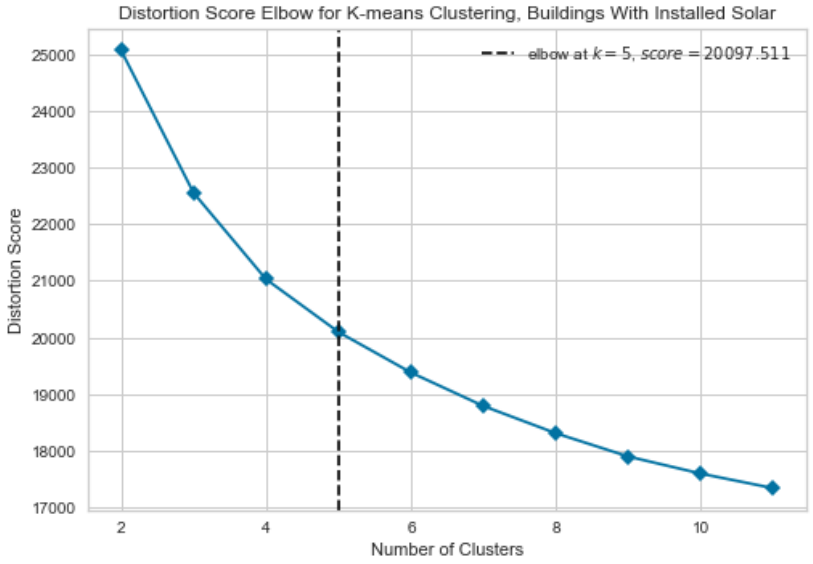


Figure 7. Elbow plot with distortion scores to determine the ideal cluster number for K-means.

The elbow plots indicated that using 5 clusters for applying k-means would be the most ideal. The resulting cluster centers are plotted below in Figures 8 and 9.

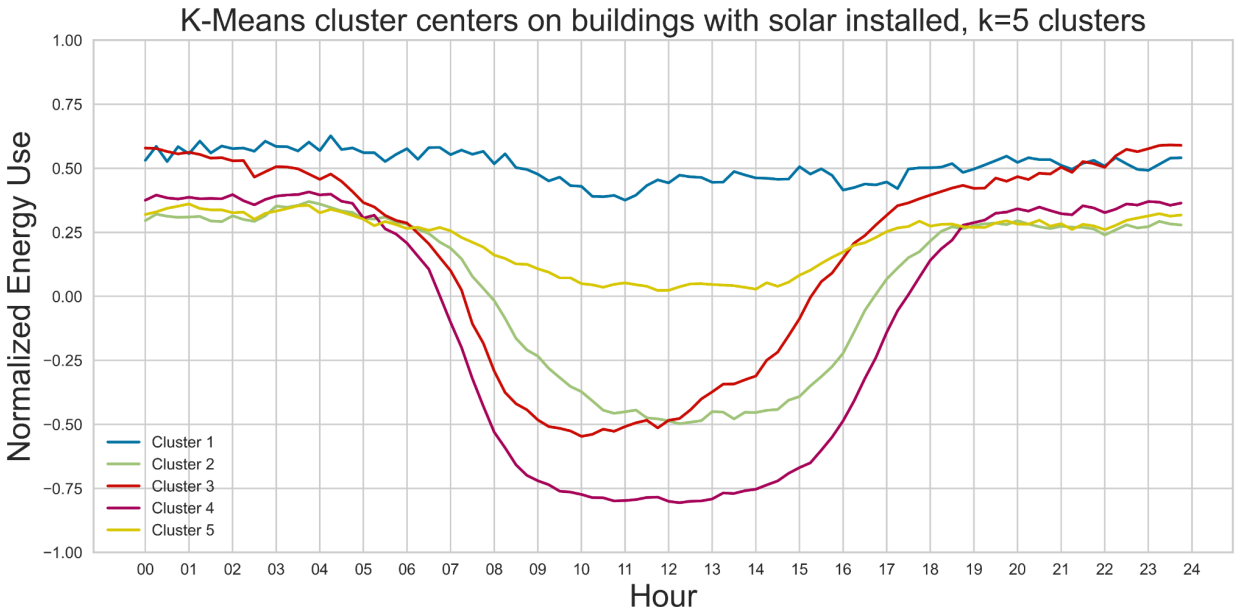


Figure 8. Plot of cluster center profiles for households with installed solar panels.

The clustering results for households with installed solar panels indicate that there are two distinct usage patterns. Cluster 1 (blue) is characterized by the lack of solar power generation, and the other cluster centers can be seen as varying levels of solar power generation during daytime. Cluster 5 (yellow) depicts a use case where solar power partially fulfills household energy needs. Cluster 3 (red) is unique among the representative profiles for solar power generation, as its generated power decreases rapidly in the afternoon. This can be seen in contrast with the other power generating profiles, where the power generation is roughly symmetric across daylight hours.

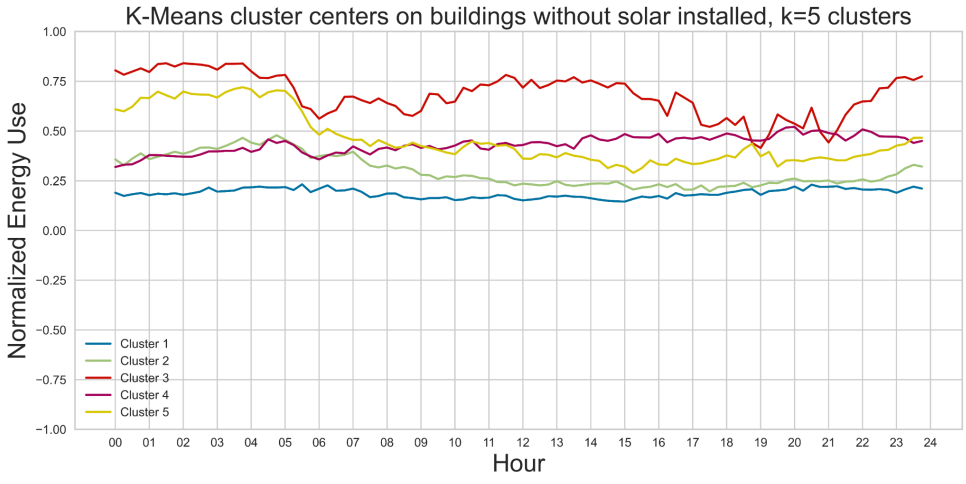


Figure 9. Plot of cluster center profiles for households without installed solar panels.

The cluster centers for the households without installed solar panels indicate several levels of power consumption. Cluster 1 (blue) represents a steady amount of energy consumption across all hours of the day. Clusters 2 (green) and 4 (Purple) have similar usage levels in the early morning hours, but diverge mid-morning, where the energy consumption in Cluster 2 decreases for the rest of the day while the usage in Cluster 4 remains constant. Cluster 3 illustrates a usage pattern that has consistently high energy consumption throughout the entire day. The most interesting representative profile is Cluster 5, which is characterized by high energy usage during the night and lower usage levels in the day. This could be because of the use of electric heating to maintain temperatures throughout the night, which is then switched off when the household members leave the house in the day. Households that frequently have this type of usage pattern could provide a viable target for energy optimization efforts.

(d) Hierarchical clustering

For Hierarchical clustering, the optimal number of clusters (n=3) was determined using the elbow method shown in the figure below:

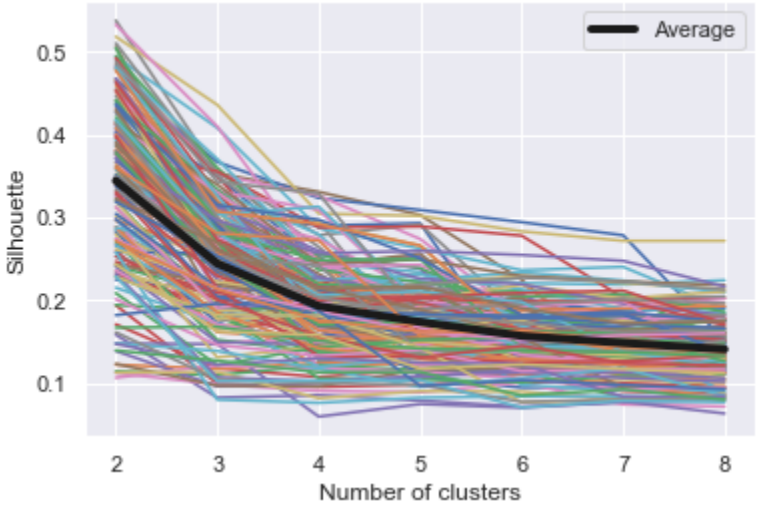


Figure 10. Elbow plot with silhouette scores for hierarchical clustering analysis.

Each line corresponds to a different time series, which was tested for a number of clusters ranging from 2 to 8. The bold line represents the average score of all the analyzed time series.

The final clusters obtained are shown below, where the average energy consumption of each house and the centers of the clusters are plotted.

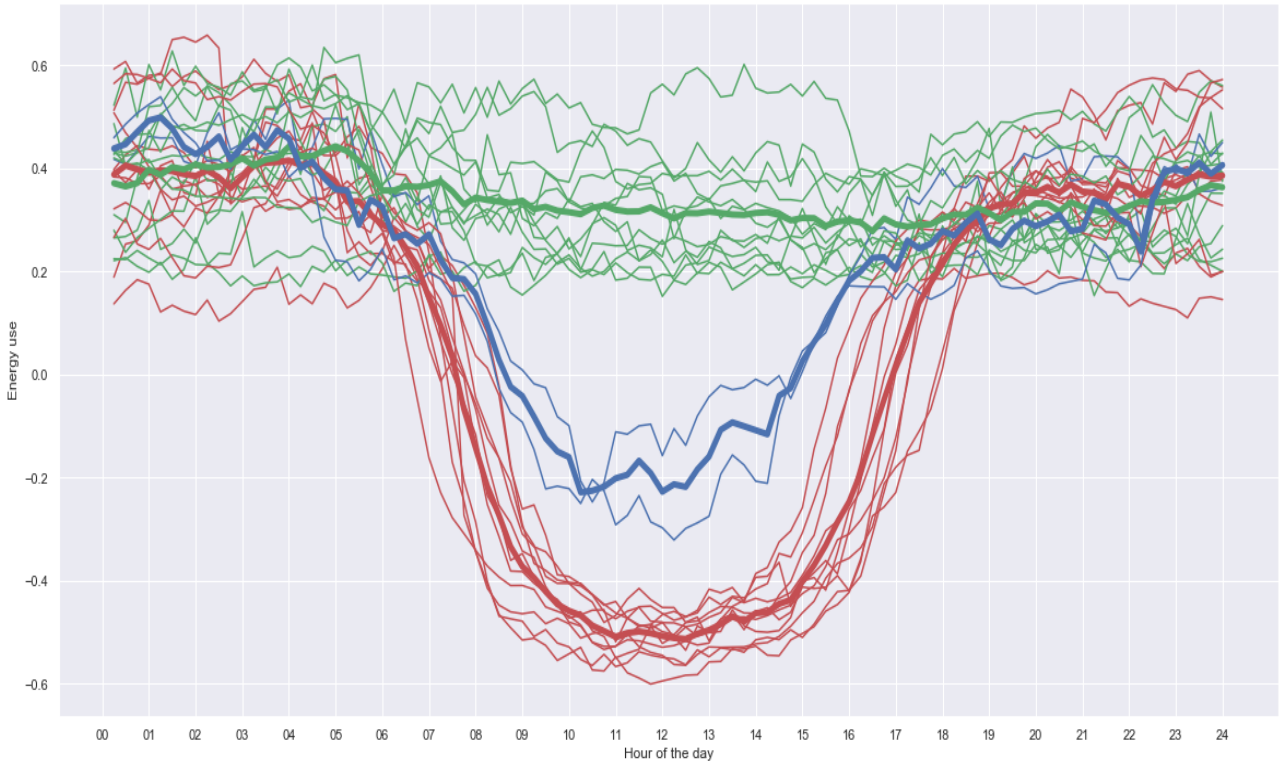


Figure 11. Hierarchical clustering results, time series profile colored by label. The cluster centers are denoted by bolded lines.

We notice that 3 main profiles emerge from this method: Households with normal consumption (green cluster), households using a reasonable amount of solar energy and balancing consumption/production (blue cluster), and households that rely heavily on the solar energy (red cluster).

(e) K-Medoid Clustering

This clustering method is very similar to K-Means. The main difference is that the cluster center is an actual data point instead of the mean of the included data points in each cluster. Similar to K-means, this method also requires the number of clusters, K, to be defined before clustering. In the K-medoid clustering analysis, the dataset was normalized by dividing by the max energy consumption value in each household. The time series profiles (24 hours) for each household across the entire time span of the data set (~5 months) was averaged into a single usage profile. The clustering algorithm was applied and the cluster centers were identified for K=2,3, and 4. The plots of the clusters are shown in Figure 12.

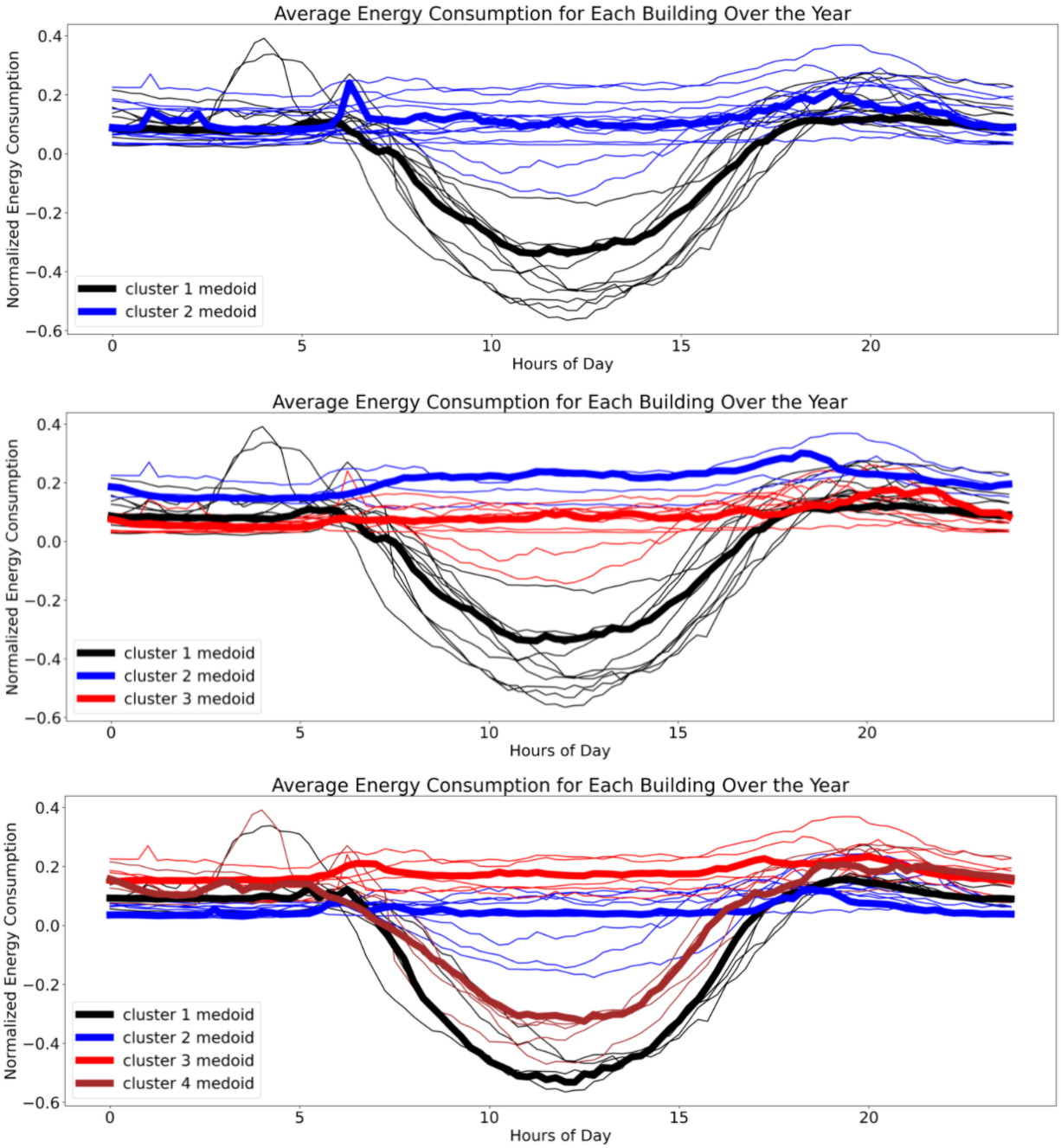


Figure 12. K-medoid clustering results.

From the plots above, selecting K=3 produced clusters with an acceptable amount of overlapping between clusters. In this specific case, the distortion is 720.7 and the silhouette coefficient is 0.458.

Table 1. Metric scores for clustering algorithms

Clustering Type	Distortion	Silhouette Score
DBScan, Solar	36019.98	0.139
DBScan, No Solar	10403.15	0.305
K-means, Solar	20097.51	0.124
K-means, No Solar	7473.97	0.0782
K-medoids	720.7	0.458
Hierarchical Clustering	24496.49	0.244

Summary

Applying both partitioning and hierarchical clustering algorithms to a dataset of household energy usage, measured in 15-minute intervals, yielded positive and interpretable results. After dividing the households into those with installed solar panels and those without, cluster analysis found several distinct usage patterns in each household group. Each type of clustering algorithm with the exception of DBSCAN yielded slightly different results but with the same underlying patterns.

Although the clustering methods used in this report produced viable results, more sophisticated distance measures were introduced in [4],such as the Dynamic time Warping (DTW) and the Shape based distance (SBD), are likely to provide better results. These distance metrics are formulated to account for the intricacies in analyzing time series data and would be much better suited for comparing two different time series vectors than the Euclidean distance utilized in this report. Future work could involve applying the more sophisticated time series analysis methods described previously. The results from using those methods can be compared to the results in this report to quantify the improvements provided by these metrics.

References

[1] Zhang, Z., and Zimet, T. K-means Based Clustering Analysis of Household Energy Consumption, Stanford University, https://uploads-ssl.webflow.com/5d59afed4a2e86d2cbb138a2/5d59db9d8602db04b598766d_project-cs229a.pdf

[2] Motlagh, O., Paevere, P., Hong, T. S., & Grozev, G. (2015). Analysis of household electricity consumption behaviours: Impact of domestic electricity generation. Applied Mathematics and Computation, 270, 165–178. doi:10.1016/j.amc.2015.08.029

[3] An, J., Yan, D., & Hong, T. (2018). Clustering and statistical analyses of air-conditioning intensity and use patterns in residential buildings. Energy and Buildings, 174, 214–227. doi:10.1016/j.enbuild.2018.06.035

[4] Teichgraeber, H., & Brandt, A. R. (2019). Clustering methods to find representative periods for the optimization of energy systems: An initial framework and comparison. Applied Energy, 239, 1283–1293. doi:10.1016/j.apenergy.2019.02.012

[5] Satre-Meloy, A., Diakonova, M., & Grünewald, P. (2020). Cluster analysis and prediction of residential peak demand profiles using occupant activity data. Applied Energy, 260, 114246. doi:10.1016/j.apenergy.2019.114246

[6] L.G.B. Ruiz, M.C. Pegalajar, R. Arcucci, M. Molina-Solana, A time-series clustering methodology for knowledge extraction in energy consumption data, Expert Systems with Applications, Volume 160, 2020, 113731, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2020.113731.

[7] Hsu, D. Comparison of integrated clustering methods for accurate and stable prediction of building energy consumption data, Applied Energy, Volume 160, 2015, Pages 153-163, https://doi.org/10.1016/j.apenergy.2015.08.126.

[8] Lavin, A., & Klabjan, D. (2014). Clustering time-series energy data from smart meters. Energy Efficiency, 8(4), 681–689. https://doi.org/10.1007/s12053-014-9316-0

Appendix

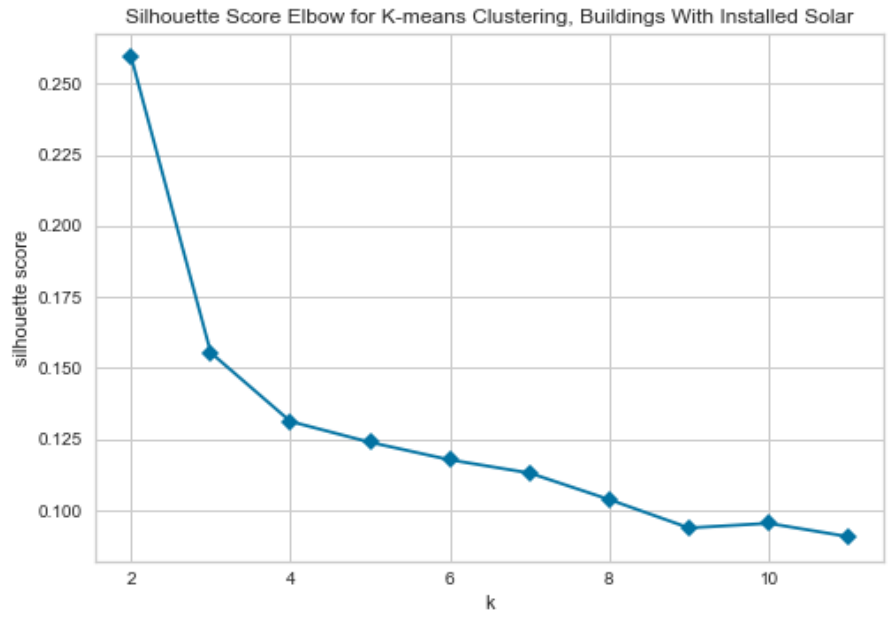


Figure 13. Elbow plot with silhouette scores to determine the ideal cluster number.

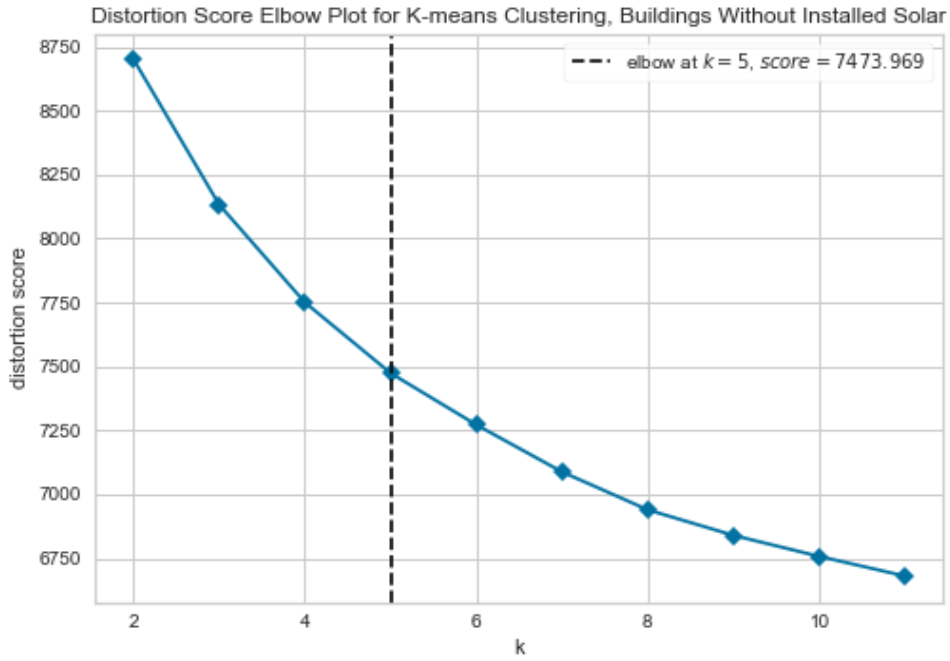


Figure 14. Elbow plot with distortion scores to determine the ideal cluster number.

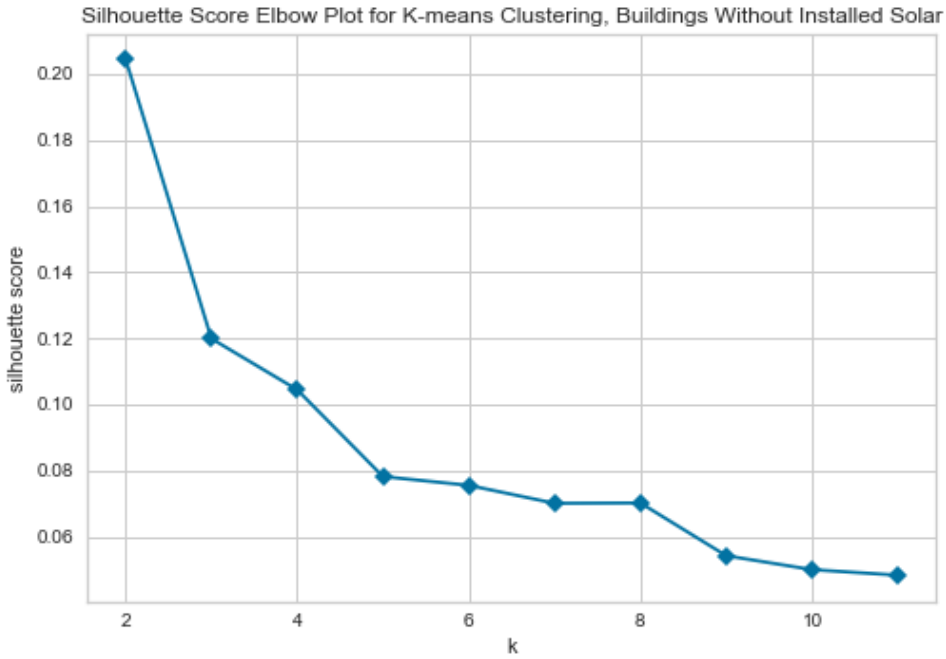


Figure 15. Elbow plot with silhouette scores to determine the ideal cluster number.

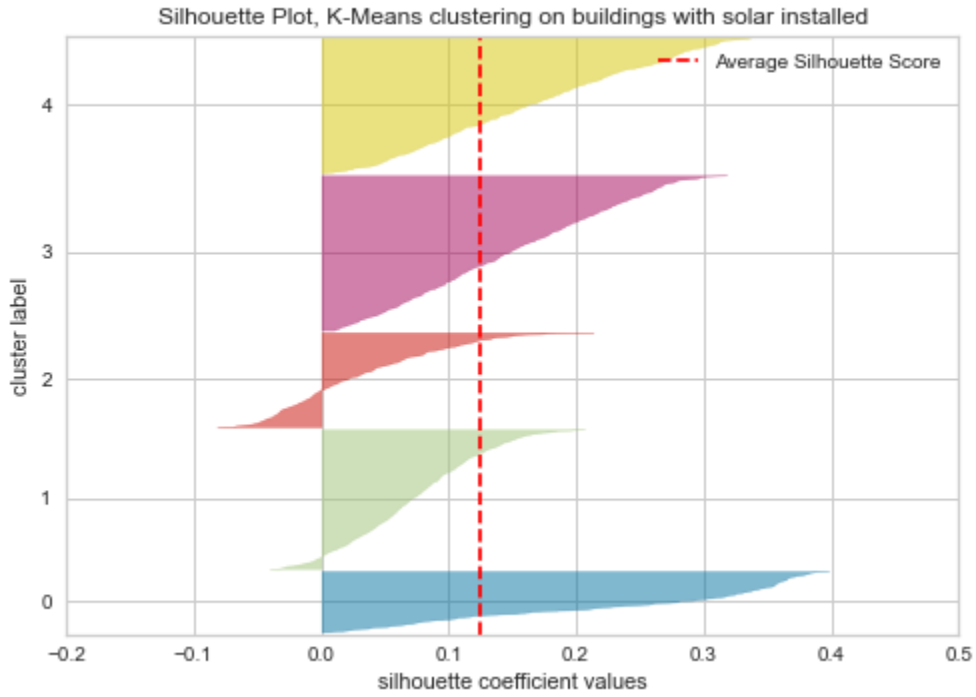


Figure 16. Silhouette plot for clusters found in households with installed solar panels.

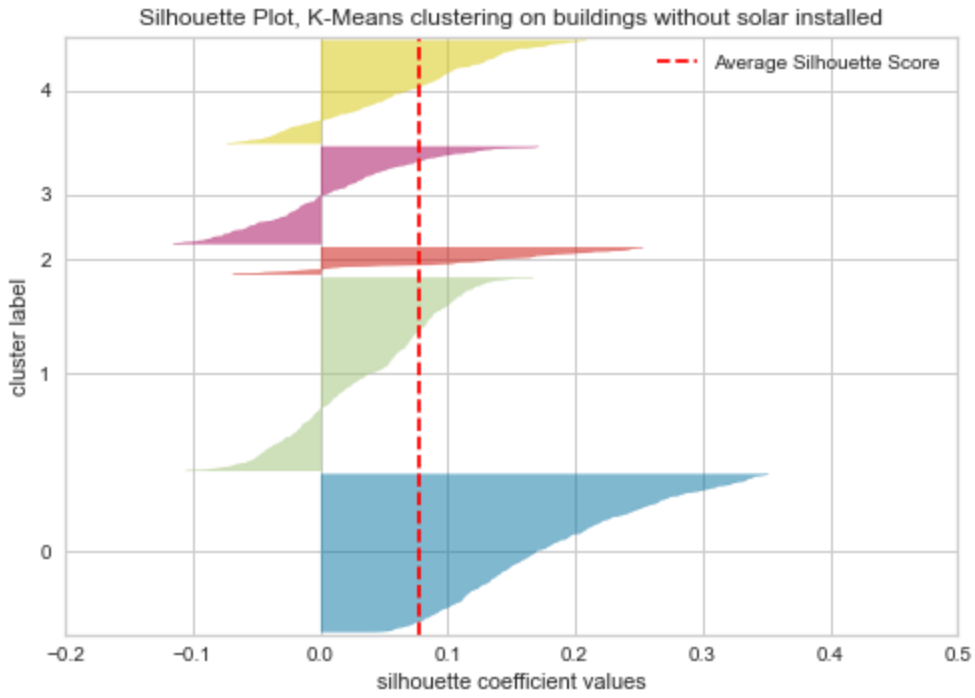


Figure 17. Silhouette plot for clusters found in households without installed solar panels.

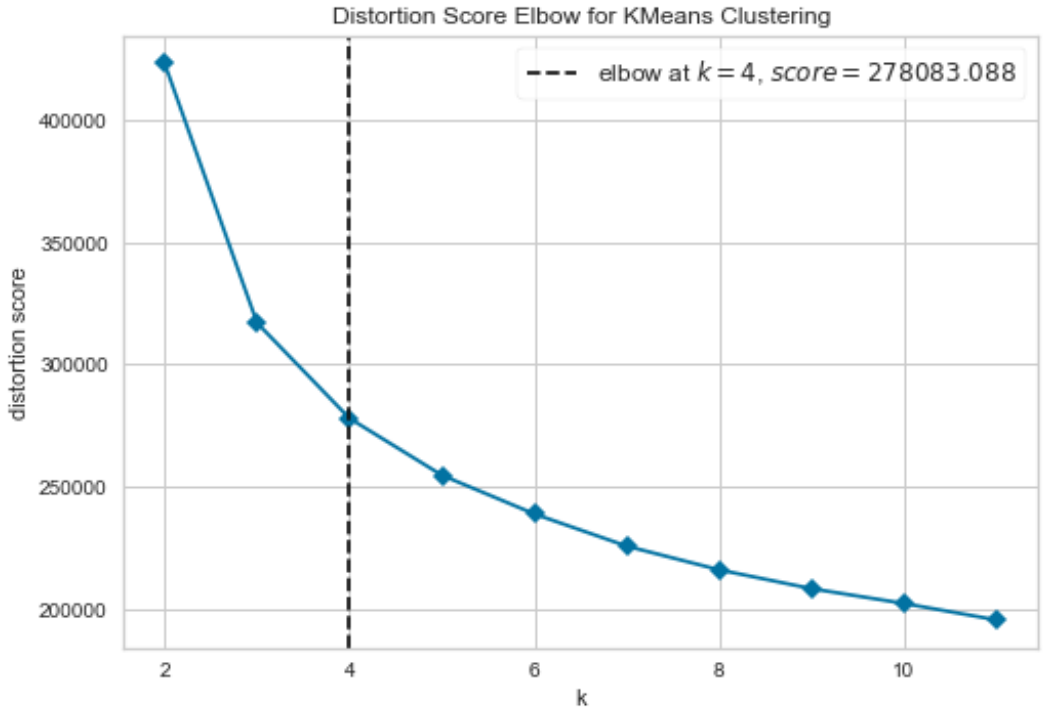


Figure 18. Elbow plot of distortion scores to determine cluster number for PCA projections for houses with solar panels.

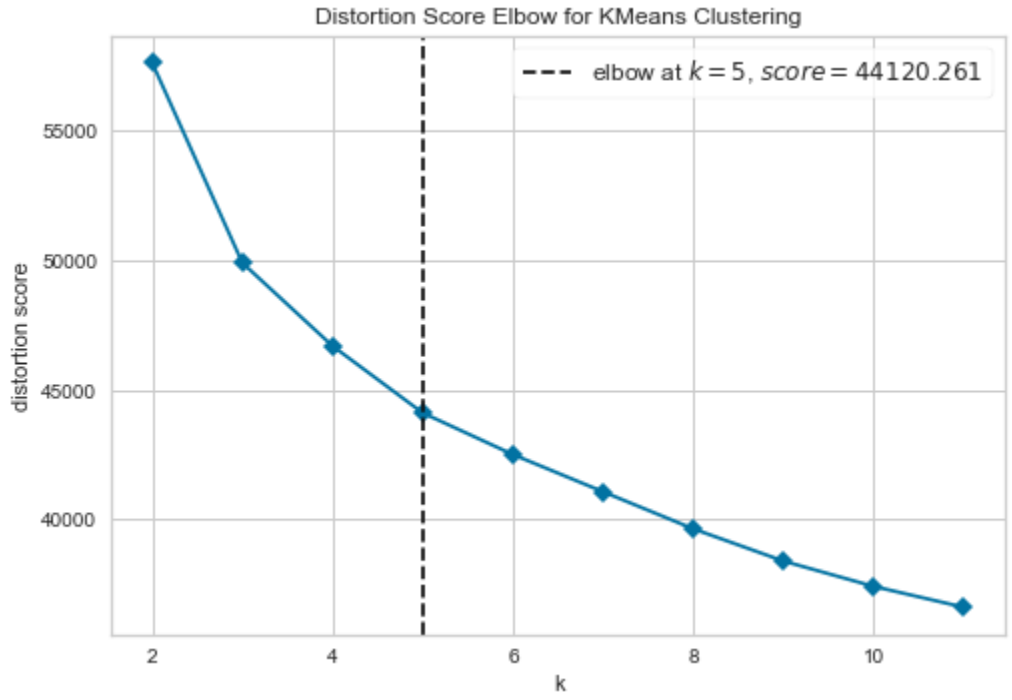


Figure 19. Elbow plot of distortion scores to determine cluster number for PCA projections for houses without solar panels.

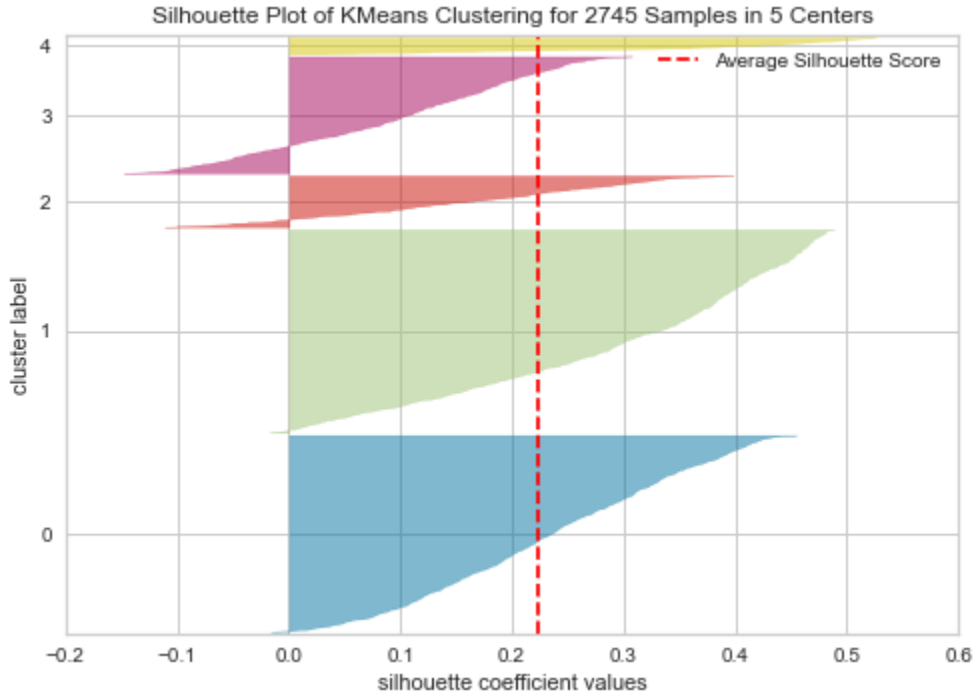


Figure 16. Silhouette plot for clusters found in households with installed solar panels for PCA projections.

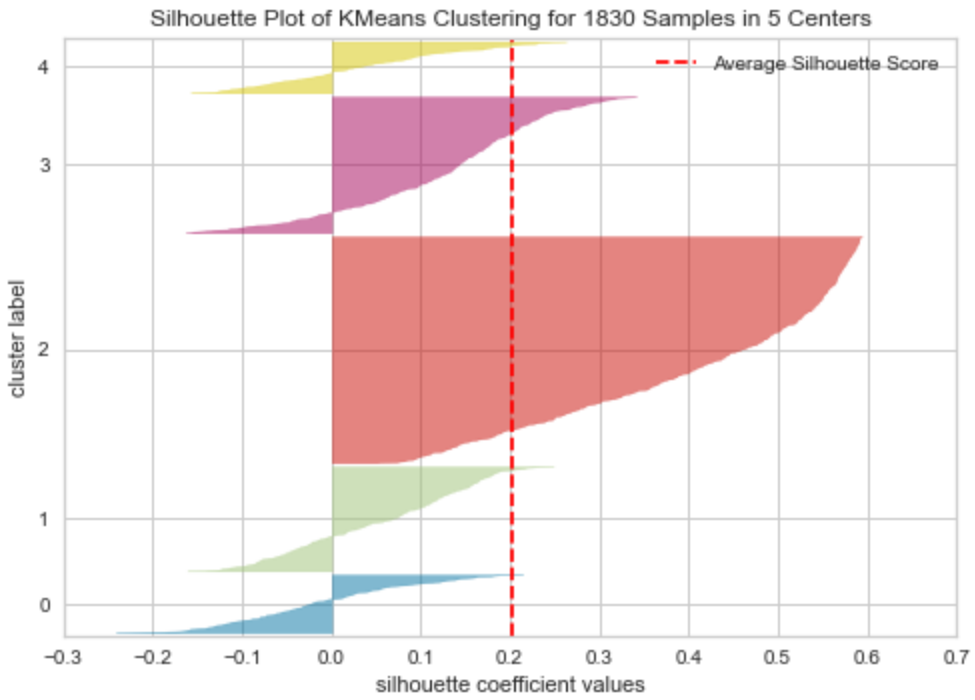


Figure 17. Silhouette plot for clusters found in households without installed solar panels for PCA projections.

Load prediction and operation schedule optimization for distributed generation and storage systems in buildings

Team 5 – Purple

Advisor:	*Daniel*	*Arnold*
Team members:	*Hanzhi*	*Wang*
	Ruiting	*Wang*
	Xingshuo	*Yan*
	Xinyu	*Zhang*
	Zhiyong	*Jiang*

Content

Abstract

The application of more environmentally friendly and energy-saving technologies in the building sector is vital to promoting a greener and more sustainable future, given the large percentage of energy consumed in commercial buildings. We aim to optimize the operation of building distributed energy generation and storage systems based on load prediction, in the hope that we can minimize the operation cost and the amount of carbon emission, while maximizing the lifecycle of the battery storage system.

Linear regression and neural network were used to predict building electricity consumption. We improved the regression score to approximately 0.9. Mixed integer programming optimization was implemented to optimize the operation of building energy storage system. We took multiple objectives into consideration, including operation cost, the carbon emission, and battery aging by environmental emission monetization and battery degradation cost. Optimized hourly PV output power and battery charging/discharging pattern were specified by the result. Under the circumstance that energy demand was sufficiently satisfied, a cost efficient and environmentally friendly energy system was constructed through optimization.

1. Introduction

1.1 Motivation & Background

Energy consumption in residential and commercial buildings accounts for more than 20% of global energy consumption, according to data from the Energy Information Administration. Statistics from the Department of Energy show that, in 2017, building sector energy consumption in the US accounts for about 39% of the total primary energy use. Building energy is one of the most extensive parts of building operation cost, while with utilization of renewable energies and energy storage system, we can drastically cut off the expense as well as the carbon emission of building. However, in application, operation of the renewable energy generation integrated with energy storage systems have always been the pain points given the stochastic nature of renewable energy generation and building loads. We aim to present a comprehensive study that can help to resolve these concerns. Our project was developed following the objectives of accurate predictions of future electricity loads and optimized operation schedule under minimized economic and environmental cost.

1.2 Relevant Literature

1.2.1 Prediction of electricity loads

1.2.1.1 Linear Regression

While predicting building thermal load, artificial intelligence techniques, including support vector machines (SVM) and neural networks have demonstrated better prediction accuracy over classical linear regression methodologies [1]. However, because of their relatively high computational requirements and complexity, AI methods have not been widely applied in industry. Kyungtae and Rogelio [2] introduced a computationally efficient and uncomplicated implementable alternative model to AI-based models, which is the 4-3-5 ARX model. It is a linear autoregressive model with exogenous, i.e., external inputs, indexed with five ambient temperatures and three time zones. The accuracy of a properly indexed ARX model has better performance than that of non-indexed models. For winter and summer seasons' thermal load prediction, the proposed indexed ARX model shows both better generality and accuracy than the hourly indexed or non-indexed models. In our model, we did the similar indexed ARX model.

1.2.1.2 Neural Network

In the scope of predicting building energy, Artificial Neural Networks (ANN) is a generally used data-driven method. It was adopted by most of the literature in the past. Kangji and Hongye [3] presented a hybrid genetic algorithm-adaptive network-based fuzzy inference system (GA-ANFIS) model, in which GA optimizes fuzzy if-then rule based on finding the best parameters of the subtractive clusters, and ANFIS adjusts the premise and consequent parameters to match the training data. The GA-ANFIS model's calculated results showed better performance compared to ANN in terms of the accuracy of prediction. In the application, the proposed model also gets comparable modeling time with ANN.

The hourly energy prediction of a library is applied in the report. It has rough information and fewer environmental variables, which is a common situation in prediction tasks. With different parameter configuration, all the results confirm the time consuming and accuracy performance of the proposed GA-ANFIS model. Byeongho and Dongsu [4] confirmed a different model,

named NARX-ANN, shows better performance in most of the comparisons than the 4-3-5 ARX model [2], indicating that the NARX-ANN model is a more suitable option for the nonlinear load predictions. For optimizing building control, one of the most important things is the control algorithm, and ensuring the algorithm can make the best decision base by predicted building demands. Thus, forecasting accuracy can be considered as one of the most critical factors. For that reason, many researches have been carried out to improve the overall accuracy of forecasting as well. Resulting from abrupt changes in outdoor and indoor conditions, building operating schedules, and storage effect of building materials, the building thermal behaviors tend to show nonlinear patterns during certain time periods. The non-linearity in building thermal behavior significantly increases in the late afternoon and early morning. Therefore, the incorrect forecasting occurs mostly at these moments. Therefore, Byeongho's study [4] focuses on forecasting accuracy, especially at the moments of high non-linearity. The improvement of prediction accuracy at those moments would not only contribute to constructing control algorithms to work more wisely and precisely, but also result in increasement of the overall prediction accuracy.

1.2.2 Optimization of operation

1.2.2.1 Optimization framework

To optimize the energy supply system is a common issue in recent years, and there are many mature researches trying to optimize the system from a wide range of standpoints. The common standpoints in this type of optimization problems are roughly divided into three groups: economic cost, environmental effects and energy sustainability, and both. Michal et al [5] set the pure economic objective as their optimization objectives. They are trying to build a relatively complex energy-supply system in which heat and electricity are provided by various resources, and then minimize the overall capital and operation cost of this designed system. Kentaro et al [6] process the optimization problem from a completely different perspective. They cared about environmental effects, especially greenhouse effects of the buildings. Therefore, their goal is to minimize the expected carbon dioxide emission under different weather scenarios. Divas et al [7] cared about the sustainability of a city, thus their optimization objective is to minimize the energy consumption of the building energy system. When a standpoint is chosen, the other factors would become constraints somehow. When the researchers start the optimization problem with a minimum environmental effect, what they have to pay attention to is the control of installation and procurement cost. We include the pure economic cost in the objective function with greenhouse effect as externality in this system. Greenhouse effect is monetized by shadow price in order to get the minimum negative externality or even the maximum positive externality while ensuring the minimum economic cost. Besides, a relatively longer life time of batteries was considered as another target. By including battery degradation cost in the objective function, the depth of discharging was well controlled resulting in more cycle times.

1.2.2.2 Shadow price

A great number of methods are applied to measure the level of environmental emissions and energy consumption, and one of the most intuitive methods is to give a 'price' for emissions and consumption. The prices for energy consumption are easy to defined considering the market price for energy resources. However, the prices for environmental emissions are not able to be obtained as directly as those for energy consumption. Regulations on environmental emissions

take effect commonly by setting a threshold of environmental emissions for businesses. Recently, those limitations are gradually replaced by tradable carbon credits, which give more elasticity to the market. Companies with large volume environmental emissions could buy carbon credits from the companies with fewer emissions and extra carbon credits. This type of transaction price generated by the market and local regulations is named by shadow price, which quantify the willing of people paying for the externalities created by their activities. To scrape the transaction data and process the market price into shadow price, a lot of work needs to be done regarding of the complexity of market nominal price and related policies. Davis et al [8] gave a comprehensive economic framework to present the carbon dioxide emission externality. Besides, some authoritative institutions have started to document the transaction information including price, size, date and even transaction groups. Related data are easy to find from the websites of those authoritative institutions.

1.2.2.3 Battery lifetime

In operational optimization, it is important to take the battery aging process into consideration, so that the 'optimal' achieved is representative of the real world. Battery degradation can significantly impact the estimation of daily or annual cost. Battery degradation depends on the battery class and the innate chemistry. Lead-acid and lithium-ion have been the most common technologies in building energy storage systems (EES). Sometimes, nickel-metal hydride (NiMH) technology is also used. A variety of factors, including depth of discharge (DoD), discharge rate, ambient temperature, charging regime, battery maintenance procedures, can impact battery lifetime [2].

For lead-acid battery, Dufo-López et al compared three lifetime prediction models [3], including equivalent full cycles to failure [4], 'Rainflow' cycles counting [9] and the Schiffer weighted Ah-throughput model [1]. Smith et al. studied the cycle-life against total cycle of discharge and DoD for lithium–iron phosphate (LFP) battery and lithium–nickel–cobalt–aluminum (NCA) battery [6]. Zhou et al. further reported that for most lithium-ion cell chemistries, an exponential relationship was found between battery life-cycle and DoD, while the battery life-cycle decays quadratically with rise of ambient temperature [2].

Based on the literature, in this project, we utilized LFP battery for building ESS and considers a battery degradation model with DoD as the major factor.

$$Life_{Bat}\ (years) = \frac{1}{\sum_{i=1}^{m} \frac{Z_i}{CF_i}}$$

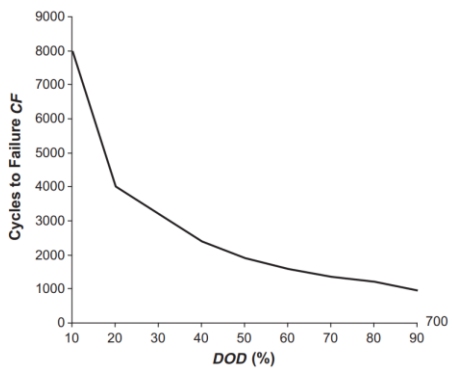


Figure 1. Battery life time based on the range of depth of discharge (DOD)

1.3 Focus of this Study

This study focuses on building energy system operation, with a particular interest in determining the day-ahead operation schedule of the battery storage system with multiple objectives based on prediction of electricity loads. The inputs and outputs of the system are shown in figure 2. Several scenarios are explored, which will be further elaborated in section 2.

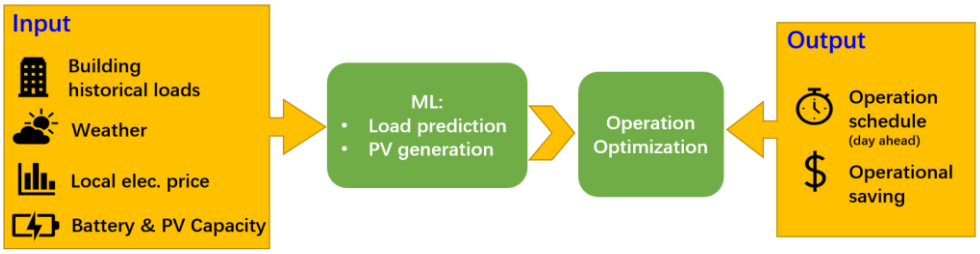


Figure 2. Research framework

2. Technical Description

2.1 Prediction

2.1.1 Data source

load DataFrame (train.csv)

- building_id - Foreign key for the building metadata.
- meter - The meter id code. Read as {0: electricity, 1: chilled water, 2: steam, 3: hotwater}. Not every building has all meter types.
- timestamp - When the measurement was taken
- meter_reading - The target variable. Energy consumption in kWh (or equivalent). Note that this is real data with measurement error, which we expect will impose a baseline level of modeling error. UPDATE: as discussed here, the site 0 electric meter readings are in kBTU.

bui DataFrame (building_meta.csv)

- site_id - Foreign key for the weather files.
- building_id - Foreign key for training.csv
- primary_use - Indicator of the primary category of activities for the building based on EnergyStar property type definitions
- square_feet - Gross floor area of the building
- year_built - Year building was opened
- floor_count - Number of floors of the building

wea DataFrame (weather_train.csv)

Weather data from a meteorological station as close as possible to the site.

- site_id
- air_temperature - Degrees Celsius
- cloud_coverage - Portion of the sky covered in clouds, in oktas
- dew_temperature - Degrees Celsius
- precip_depth_1_hr – Millimeters
- sea_level_pressure - Millibar/hectopascals
- wind_direction - Compass direction (0-360)
- wind_speed - Meters per second

2.1.2 EDA & Plotting data

The original dataset contained more than one thousand and four hundred buildings' records. In order to obtain satisfying results for prediction and optimization, we selected office building 305 in site 3, which had distinct temperature and weather conditions. For load DataFrame, we only kept rows with meter IDs equal to 0 (electricity) since this project only cares about electricity consumption prediction. The dataset had some missing values in the meter reading column. We filled nan (not a number) values with column mean values.

Figure 3 was plotted by the timestamp and meter reading. In the graph, the peaks around 5000th hour in a year can be assumed as summer and winter, since the air conditioning would be the largest contributor to electricity consumption. Also, this can be drawn from figure 4, which demonstrates the relationship between air temperature and meter reading.

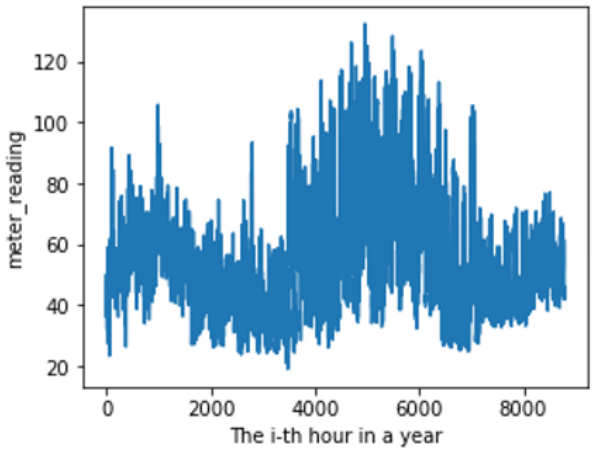


Figure 3. Plot of raw data

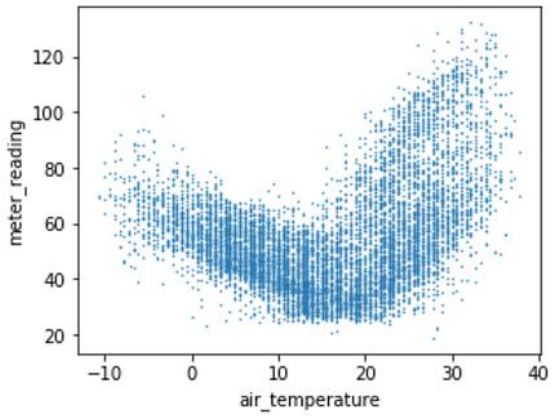


Figure 4. Plot of air temperature vs. meter reading

2.1.3 Processing data & Building variables

Based on the observation from literature review, we decided to apply two prediction methods (Linear regression and neural network) in our model. During the data cleaning and exploration process, we added several variables into the data frame. The first 24 variables are

meter_reading_pre_x', which indicates the previous x-th hour before the timestamp where x is from 1 to 24. For example, meter_reading_pre_1 is the meter reading of the previous one hour. The others are weather-related variables including cloud_coverage, and air_tempertature, etc.. By computing the p-value, we can determine the significant variables with p-value less than 0.05, which is shown in figure 5.

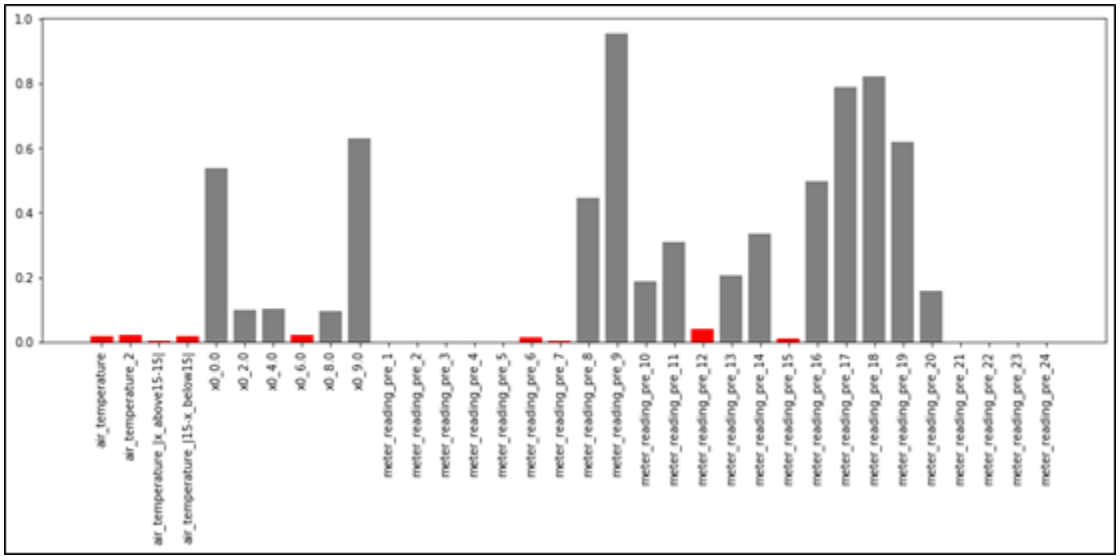


Figure 5. Plot of P-values for each variables

2.2 Optimization

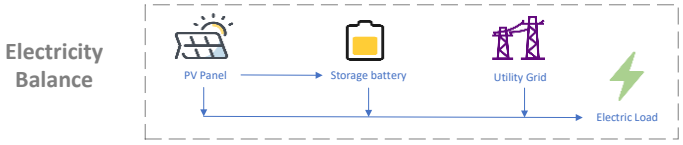


Figure 6. energy supply system

We assumed that the building thermal loads were only supplied by HVAC system that was only driven by electricity (figure 6). The electricity load of the building was from human activity, equipment activity and mostly from HVAC system. Two power sources of electricity were considered. PV array and utility grid help to meet the electrical loads. When the solar radiation was strong enough to serve the electricity demands, the surplus of the energy from PV arrays would be stored in batteries.

2.2.1 Objective function

The overall optimization objective is to minimize both the economic cost in building operation phase and greenhouse effects within a year. To unify two objectives with different units, we apply monetization to measure greenhouse effects by calculating the product of expected carbon dioxide emissions and shadow price of it.

The objective function could be structured as followed:

$$\min f$$

$$f = \sum_{t=1}^{8760} [P_{UG}(t) \times D_{UG}(t) \times \Delta T + E_{CO_2}(t) \times D_{CO_2}(t) \times \Delta T + C_d(t)]$$

2.2.2 Constraints

2.2.2.1 Power balance constraints

The following equation addressed the hourly power balance of the system.

$$\eta_{PV} \times + \eta_{grid} \times P_{grid}(t) = C_{dummy}(t) \times P_{bat}(t) + \eta_{columb} \times (1 - C_{dummy}(t)) \times P_{bat}(t) + P_{load}(t)$$
$$t = 0,1,2,\ldots 8760$$

2.2.2.2 Battery constraints

The hourly SOC of the battery was updated based on the hourly battery charging/discharging power.

$$\big(SOC(t+1) - SOC(t)\big) \times Q_b = P_{bat}(t) \times \Delta T$$
$$t = 0,1,2,\ldots 8760$$

We set the end-of-day SOC to be the same as the initial SOC of the day.

$$SOC(24n) = SOC(\mathbf{0})$$
$$n = 1,2,3,\ldots,365$$

The maximum charging and discharging power of the battery were limited by the battery characteristics.

$$P_{dismax} \leq P_{bat}(t) \leq P_{charmax}$$
$$\theta_{min} \leq SOC_{bat}(t) \leq \theta_{max}$$

2.2.2.3 Power input from the utility grid

We assumed that reverse power flow to the utility grid was not allowed. The maximum power output of the utility grid is constrained by the local transformer, denoted by P_{ugmax}.

$$0 \leq P_{UG}(t) \leq P_{ugmax}$$

2.2.2.4 Carbon dioxide emissions

We expected zero carbon emission from distributed generation from PV arrays. The carbon dioxide emissions from the utility grid were calculated based on International Energy Agency (IEA) dataset [10]. From the publication of IEA, we could get how much magnitude of carbon dioxide is emitted from generating a kilowatt hour electricity.

$$E_{CO_2} = n \times \sum_{t=1}^{8760} P_{UG}(t) \times \Delta T$$

2.2.2.5 Battery degradation

The battery module selected for our systems were 200 Ah, 25.6 Volt LFP batteries by Victron Energy (LFPSmart 25,6/200). Literature identifies an exponential relationship of the Cycle-DOD model [2]. Based on battery specifications from datasheet, we were able to regress a Cycle-DoD function based on three data points provided by the product manufacturers [11].

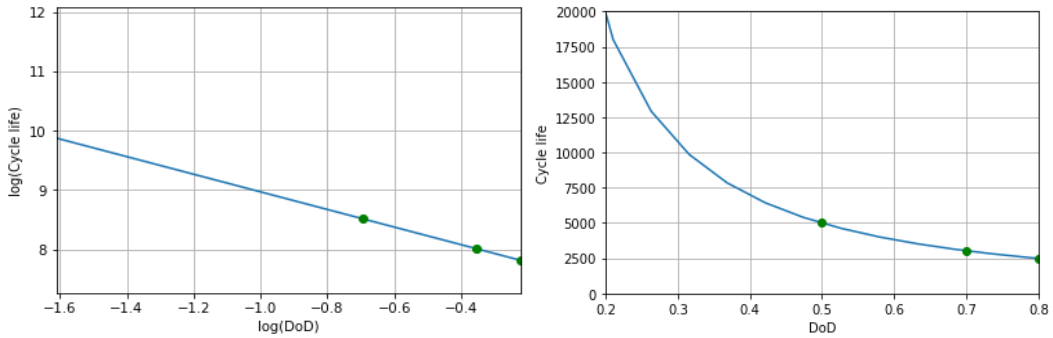


Figure 7. Cycle-life as a function of the DoD for an LFP battery

The degradation cost was defined based on the method in paper [1], [12]. The cost of discharge from a fully charged battery is defined as:

$$C_d = C_b \frac{1}{\Phi(DoD)}$$

where $\Phi(\cdot)$ refers to cycle-life as a function of the DoD, C_b is the battery cost, and C_d is the battery degradation cost.

Likewise, a battery discharges from DoD1 to DoD2 has a degradation cost of:

$$C_d = C_b[\frac{1}{\Phi(DoD1)} - \frac{1}{\Phi(DoD2)}]$$

Two problems remain for this function of battery degradation cost. First, the exponential relationship was not supported in the Gurobi solver. Second, the curve $\frac{1}{\Phi(DoD)}$ was recognized by the solver as a concave function. Due to these limitations of solver, we further simplified the function and regressed a piece-wise linear relationship of battery degradation cost C_d and ΔDoD $(DOD1 - DOD2)$, based on 100 data points in original function. Thus, the battery degradation cost was calculated as part of the optimization framework.

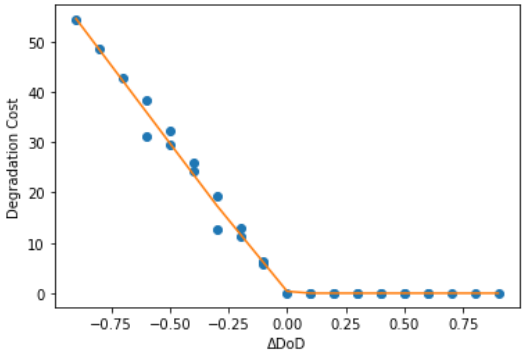


Figure 8. Battery degradation cost as a function of the ΔDoD for an LFP battery

2.2.2 PV and battery capacity

We used PVsyst to estimate the PV power and battery capacity. According to the prediction in the last section, we knew the load of the building, and distribution of the load. In PVsyst, we used these data as a part of input. We have two parameters, Loss of Load Probability and Autonomy, which are significant when we are designing the PV and battery capacity. The desired system

autonomy determines the battery capacity, and the required LOL determines the PV array nominal power. Loss of load probability is the probability that the user's need cannot be satisfied. Autonomy means the time during which the load can be met with the battery alone.
Based on the simulation results, we modify the parameters for the optimization. The PV module area is 1200m², 614.4kWh, which covers about 53% of the load. The battery capacity is about 614.4kWh.

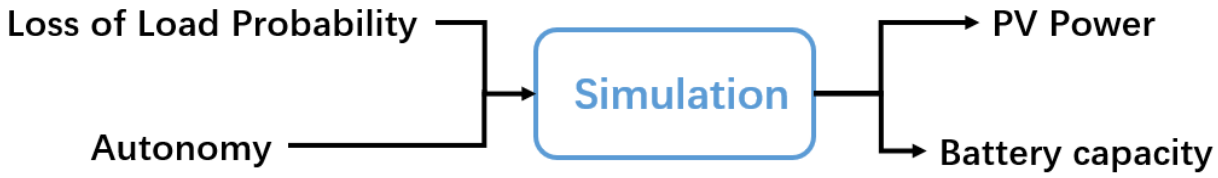


Figure 9. Simulation inputs and outputs

3. Results

3.1 Prediction

In the prediction part, we implemented an autoregressive-based neural network model with exogenous inputs, which integrated the weather information with previous electric loads for training. Combined with exploratory data analysis, we grasped non-linear relationships between electric load and model input (e.g. temperature, cloud coverage) and processed model optimization to keep variables that were significant with p-value less than 0.05. These procedures achieved high-accuracy prediction results as shown in our testing section.

We used the scikit-learn to build two different models, including linear regression and neural networks. The train set was composed of the first 7000 data points. The test set was the last 1760 data points, which is approximately 73 days. The plots below were generated based on the relationship between predicted values and actual values. The closer the point from the red line, the more accurate the prediction is. Both plots in figure 10 demonstrated the high accuracy.

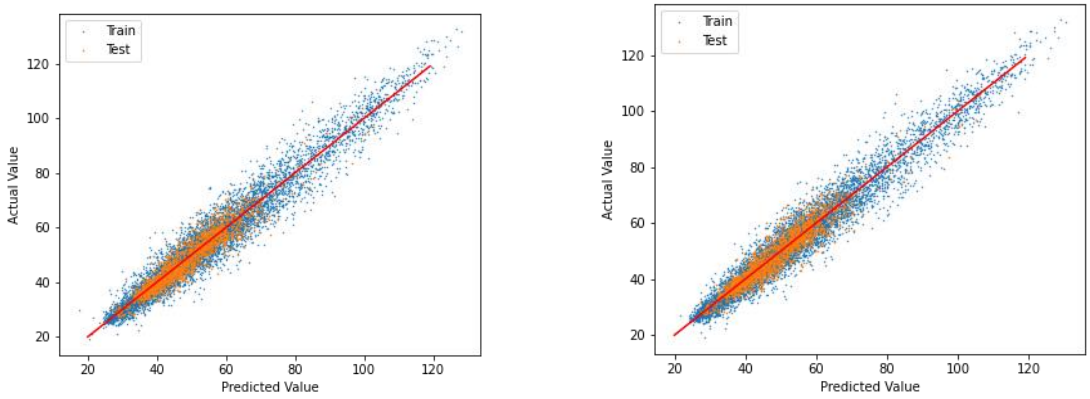


Figure 10. Plot of the predicted value and actual value
(Left grah is from Linear regression model, Right graph is from neural network model)

Figure 11 presented the predicted value and actual values on the time axis, which can give a clearer picture of our models. By looking at both graphs, it could be noticed that the models can catch most of the key points and big trends.

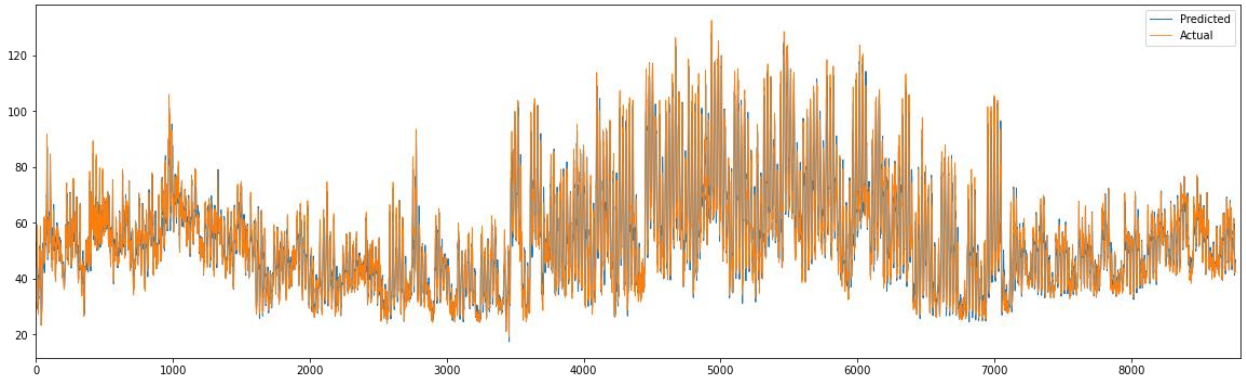


Figure 11-1. Plot of predicted values and actual value from Linear regression model

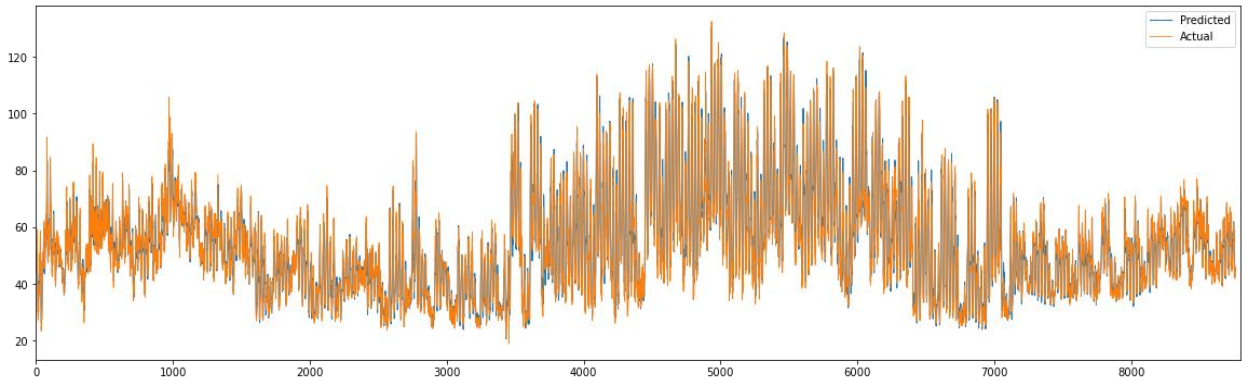


Figure 11-2. Plot of predicted values and actual value from Neural network model

Overall, the prediction results from linear regression and neural network models are acceptable. The test score for linear regression is 0.891. The test score for the neural network is 0.892.

3.2 Optimization

We used a Python extension module of Gurobi optimizer, called “gurobipy”, which offers convenient object-oriented modeling construct to solve our optimization problem [13]. Our model contains 17472 quadratic constraints and 8736 general constraints. In total, we have 43682 continuous variables and 8736 integer variables.

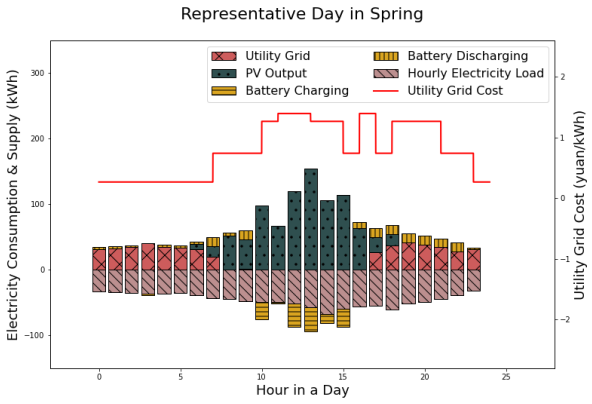
By Gurobi optimizer, battery charging/discharging pattern is given based on minimum economic cost and environmental emissions. Hourly PV output power is optimized based on the predicted electricity loads and insolation given a confirmed peak load and battery capacity. Overall, PV took up 53% of energy supply and improve the environmental performance of this building greatly. The batteries were cycled 106 times in a year under this charging/discharging pattern and the life time

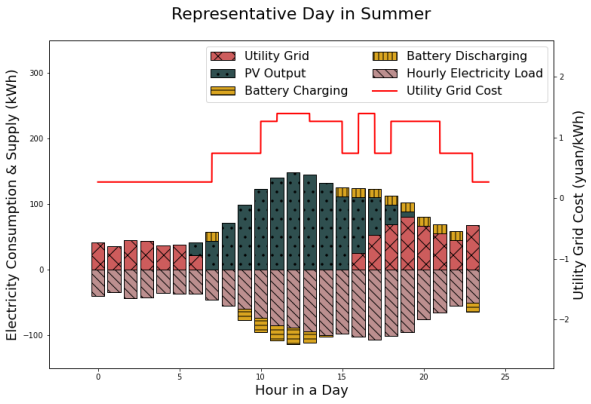
could be 13 years, which was much longer than rating life. Two base models were chosen for comparison and assessment of the economic efficiency of optimization. PV panels and grouped batteries were considered as the main research objective while the circumstance that grouped batteries are eliminated and the circumstance without both PV panels and batteries are two base models. The total cost of the building with both batteries and PV panels is saved 50.15% and 3.78% compared to two base models (table 1).

Table 1. Cost of the three different energy supply systems

Scenario	Operational Cost	Monetization of CO2 Emission	Battery Degradation Cost
w/o PV, w/o Batt	405236	14512	N/A
w/ PV, w/o Batt	208350	9128	N/A
w/ PV, w/ Batt	194933	9566	4763

By looking at the energy cycle in one representative day (figure 12), a symmetrical distribution indicates that energy demand is well satisfied by energy supply. The energy consumption curve lies in different shapes in four seasons. All of four representative days have an electricity consumption peak in the afternoon. The electricity curve of summer fluctuates more than curves of other seasons, while winter curve has the flattest shape. The solar insolation is relatively stronger in spring and summer than in fall and winter, thus it is common that batteries are charged during sufficient insolation period in spring and summer. During the afternoon peak, batteries discharged sometimes to share the pressure of utility grid. In fall and winter, it is more common that batteries are mostly charged by utility grid before dawn when electricity demand is not intense.





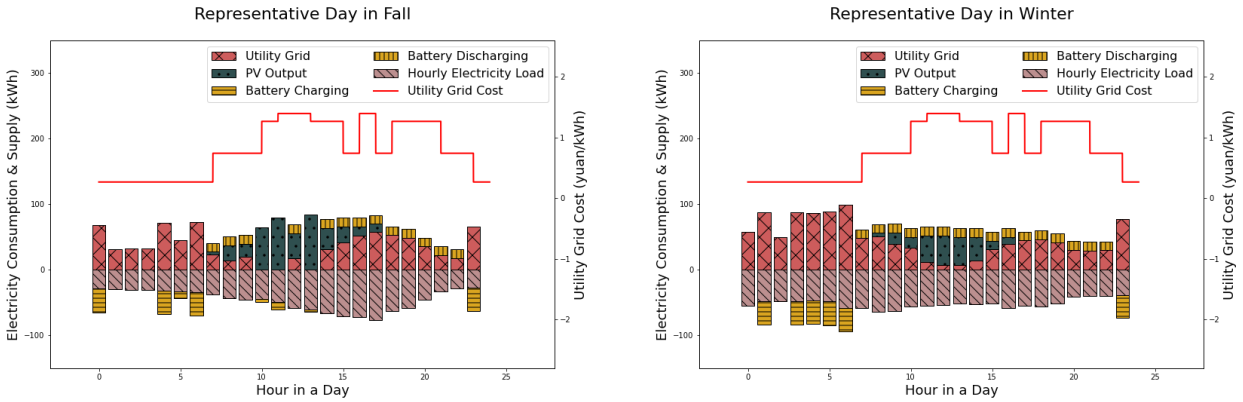


Figure 12. Electricity consumption and supply in a representative day

4. Discussion

Through minimizing the degrading cost of batteries, the depth of battery discharging was controlled in the optimization process. As a result, the battery life time could be near 13 years which is longer than general rating life. Longer life time was derived from the ratio of battery cost over the degradation cost. However, batteries are usually abandoned when the degradation cost is pretty small but not completely zero. When the batteries don't work as well as before, the users tend to replace them which results that the life time from the users' point of view is shorter than the calculated life time. Therefore, 13 years could be a slightly overestimated number.

Battery charging power, as a necessary parameter needed to be decided for optimization process, had an obvious impact on the charging/discharging pattern. Battery charging power was decided once a specified type of batteries was selected for the testing building. We compared two different battery charging power by presenting the representative daily energy supply and demand (figure 13). A high charging power results in a more concentrated charging period (figure 13-1). On the contrary, a low charging power allowed the batteries getting charged more evenly in the time dimension (figure 13-2), which helped stabilize electricity load.

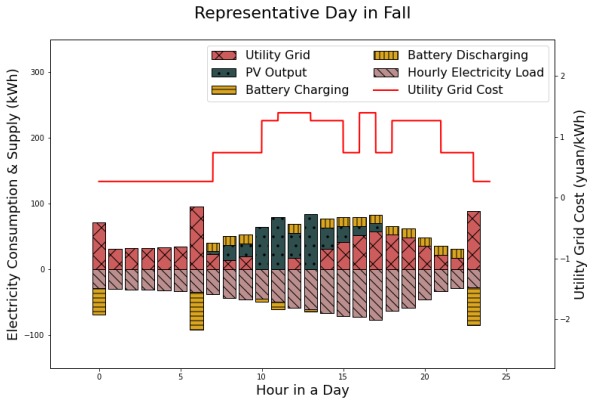


Figure 13-1. High charging power

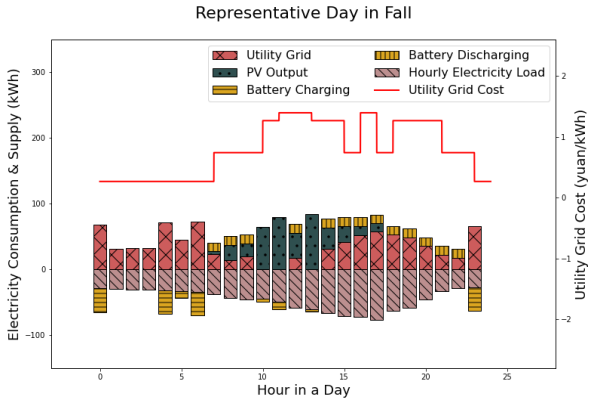


Figure 13-2. Low charging power

5. Summary

Linear regression and neural network were applied to give accurate predictions of hourly electricity load for office buildings based on historical load data and weather information. R-square scores for linear regression and neural network were both over 0.9, which was a pretty high score for predicting. By inputting predicted future loads into optimization model, an energy-supply system is constructed by regulating the battery charging/discharging pattern and PV output power, which helps to save economic cost and to decrease environmental effects.

Reference

[1] J. Schiffer, D. U. Sauer, H. Bindner, T. Cronin, P. Lundsager, and R. Kaiser, “Model prediction for ranking lead-acid battery according to their expected lifetime in renewable energy systems and autonomous power supply systems,” 10th Eur. Lead Batter. Conf., 2006, doi: 10.1016/j.jpowsour.2006.11.092.

[2] C. Zhou, K. Qian, M. Allan, and W. Zhou, “Modeling of the cost of EV battery wear due to V2G application in power systems,” IEEE Trans. Energy Convers., vol. 26, no. 4, pp. 1041–1050, 2011, doi: 10.1109/TEC.2011.2159977.

[3] R. Dufo-López, J. M. Lujano-Rojas, and J. L. Bernal-Agustín, “Comparison of different lead-acid battery lifetime prediction models for use in simulation of stand-alone photovoltaic systems,” Appl. Energy, vol. 115, pp. 242–253, 2014, doi: 10.1016/j.apenergy.2013.11.021.

[4] J. L. Bernal-Agustín and R. Dufo-López, “Simulation and optimization of stand-alone hybrid renewable energy systems,” Renew. Sustain. Energy Rev., vol. 13, no. 8, pp. 2111–2118, 2009, doi: 10.1016/j.rser.2009.01.010.

[5] M. Szypowski, T. Siewierski, and A. Wedzik, “Optimization of Energy-Supply Structure in Residential Premises Using Mixed-Integer Linear Programming,” *IEEE Transactions on Industrial Electronics, Industrial Electronics, IEEE Transactions on, IEEE Trans. Ind. Electron.*, vol. 66, no. 2, pp. 1368–1378, Feb. 2019, doi: 10.1109/TIE.2018.2793276.

[6] K. Shimomachi, R. Hara, H. Kita, M. Noritake, H. Hoshi, and K. Hirose, “Development of energy management system for DC microgrid for office building:-Day Ahead operation scheduling considering weather scenarios-,” *2014 Power Systems Computation Conference, Power Systems Computation Conference (PSCC), 2014*, pp. 1–6, Aug. 2014, doi: 10.1109/PSCC.2014.7038313.

[7] D. Grover, Y. P. Fallah, Q. Zhou, and P. e. Ian LaHiff, “Data-Driven Modeling and Optimization of Building Energy Consumption: a Case Study,” *2020 IEEE Power & Energy Society General Meeting (PESGM), Power & Energy Society General Meeting (PESGM), 2020 IEEE*, pp. 1–5, Aug. 2020, doi: 10.1109/PESGM41954.2020.9281663.

[8] B. A. Davis Noll and B. Unel, “Markets, Externalities, and the Federal Power Act: The Federal Energy Regulatory Commission’s Authority to Price Carbon Dioxide Emissions,” *Environmental Law Reporter*, vol. 50, no. 8, pp. 10629–10634, 2020, Accessed: Apr. 01, 2021. [Online]. Available: https://libproxy.berkeley.edu/login?qurl=https%3a%2f%2fsearch.ebscohost.com%2flogin.aspx%3fdirect%3dtrue%26db%3dedshol%26AN%3dedshol.hein.journals.elrna50.68%26site%3deds-live.

[9] S. Downing and D. Socie, “Simple rainflow counting algorithms,” Int. J. Fatigue, vol. 4, no. 1, pp. 31–40, 1982.

[10] “IEA CO2 Emissions from Fuel Combustion Statistics.” https://www.oecd-ilibrary.org/energy/data/iea-co2-emissions-from-fuel-combustion-statistics_co2-data-en (accessed Apr. 26, 2021).

[11] J. L. Bernal-Agustín and R. Dufo-López, “Simulation and optimization of stand-alone hybrid renewable energy systems,” Renew. Sustain. Energy Rev., vol. 13, no. 8, pp. 2111–2118, 2009, doi: 10.1016/j.rser.2009.01.010.

[12] S. Downing and D. Socie, “Simple rainflow counting algorithms,” Int. J. Fatigue, vol. 4, no. 1, pp. 31–40, 1982.

[13] L. Gurobi Optimization, “Gurobi Optimizer Reference Manual,” 2020, [Online]. Available: http://www.gurobi.com.

Appendices

$\boldsymbol{f}$	The operation cost and environmental effects presented by dollars. [$]
$\boldsymbol{D_{gas}}$	The local marginal cost of gas. [$/kg]
$\boldsymbol{D_{DH}}$	The local marginal cost of district heating. [$/kWh]
$\boldsymbol{D_{UG}}$	The local marginal cost of electricity provided by utility grid. [$/kWh]
$\boldsymbol{D_{CO_2}}$	The local shadow price for carbon dioxide emissions. [$/kg_$CO_2$]
$\boldsymbol{Life_{bat}}$	Battery life time [yr]
$\boldsymbol{M_{gas}(t)}$	The magnitude of gas consumed in the hour t. [kg]
$\boldsymbol{P_{DH}(t)}$	The district heating power in the hour t. [kW]
$\boldsymbol{P_{UG}(t)}$	The power provided by utility grid in the hour t. [kW]
$\boldsymbol{P_{PV}(t)}$	PV panel output power in the hour t. [kW]
$\boldsymbol{P_{bat\_C}(t)}$	Storage battery charging power in the hour t. [kW]
$\boldsymbol{P_{bat\_D}(t)}$	Storage battery discharging power in the hour t. [kW]
$\boldsymbol{P_{bat}^{max}}$	The maximum charging/discharging power of battery. [kW]
$\boldsymbol{E_{CO_2}}$	The expected carbon dioxide emissions in a year. [kg_CO_2]
$\boldsymbol{\Delta T}$	The minimum time interval ($\Delta T = 1hr$)
$\boldsymbol{\eta_{gas}}$	Conversion efficiencies for gas heating network converter.
$\boldsymbol{\eta_{DH}}$	Conversion efficiencies for district heating network converter.
$\boldsymbol{\eta_{PV}}$	Conversion efficiencies for PV panel converter.
$\boldsymbol{\eta_{grid}}$	Conversion efficiencies for utility grid converter.
$\boldsymbol{E}$	Heat productivity of gas. [kWh/kg_gas].
$\boldsymbol{H(t)}$	Heat load in the hour t. [kWh]
$\boldsymbol{E(t)}$	Electricity load in the hour t. [kWh]
$\boldsymbol{SOC_{bat}(t)}$	State of charge of storage battery at the end of the hour t. [kWh]
$\boldsymbol{SOC_{bat}(0)}$	The initial battery SOC.
$\boldsymbol{SOC_{bat}^{max}}$	The maximum SOC.
$\boldsymbol{\theta}$	The minimum percentage of SOC.
$\boldsymbol{D_s}$	Battery self-discharging rate. [h^{-1}]
$\boldsymbol{I_C(t)}$	Battery charging current in the hour t. [A]
$\boldsymbol{V_{oc}}$	Open circuit voltage of storage battery. [V]
$\boldsymbol{C_s}$	Storage battery capacity. [mAh]
$\boldsymbol{P_c}$	Photovoltaic array peak power. [kW]
$\boldsymbol{A}$	The mean daily energy produced from the photovoltaic array to satisfy the load in months with lowest radiation during a year. [kWh]
$\boldsymbol{L}$	Daily load (constant). [kWh]
$\boldsymbol{n_1}$	The magnitude of carbon dioxide emission from gas per kilogram. [kg_CO_2/kg_gas]
$\boldsymbol{n_2}$	The magnitude of carbon dioxide emission from district heating per kWh. [kg_CO_2/kWh_DH]
$\boldsymbol{n_3}$	The magnitude of carbon dioxide emission from utility grid per kWh. [kg_CO_2/kWh_UG]
$\boldsymbol{\delta}$	$\delta = 1000\ mAh/Ah$

Prediction Codes

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from numpy.linalg import matrix_rank
import statsmodels.api as sm
from scipy import stats
```

In [33]:

```
load = pd.read_csv("data/train.csv")

# Electricity meter
load = load.query("meter == 0")
load = load.drop('meter', axis=1)

wea = pd.read_csv("data/weather_train.csv")
# wea['temp_day'] = wea['timestamp'].str[0:10]
# wea.groupby(['site_id','temp_day'], as_index=False)['air_temperature'].mean().groupby('site_id').a
# plt.plot(wea.query("site_id == 2")['air_temperature']); # 11, 7, 2

bui = pd.read_csv("data/building_metadata.csv")
```

In [56]:

```
# We chose 'office' type buildings to do forecast
bui_office = bui.query("primary_use == 'Office'")
bui_office_id = list(bui_office['building_id'])
# site_id 15, 10, 14, 3 have our ideal temperature (hot in summer and cold in winter)
bui_office_site = bui_office.query("site_id == 3")
bui_office_site.head()
```

Out[56]:

	site_id	building_id	primary_use	square_feet	year_built	floor_count
293	3	293	Office	408000	2010.0	NaN
305	3	305	Office	8637	NaN	NaN
317	3	317	Office	16098	NaN	NaN
318	3	318	Office	105000	NaN	NaN
335	3	335	Office	229000	2010.0	NaN

In [57]:

```
wea_ = wea.query("site_id == 3")
# building_id 305 and 563 both in site_id 3 have good "temperature vs. load" relationship (valley sh
load_ = load.query("building_id == 305") # 3-305, 3-563
# merge load table with weather table
load_wea_ = pd.merge(left=load_, right=wea_, left_on='timestamp', right_on='timestamp')

# 'building_id', 'site_id' are useless once we finished mergeing, and 'precip_depth_1_hr' has a poor
load_wea_ = load_wea_.drop(['building_id', 'site_id', 'precip_depth_1_hr'], axis=1)

# Try to project wind to x and y axis, and analyze the relationship between 'wind_speed_x', 'wind_sp
# load_wea_['wind_speed_x'] = load_wea_['wind_speed'] * np.cos(load_wea_['wind_direction'])
# load_wea_['wind_speed_y'] = load_wea_['wind_speed'] * np.sin(load_wea_['wind_direction'])
# 'sea_level_pressure' has a poor relationship with load, so we dropped it
load_wea_ = load_wea_.drop(['wind_speed', 'wind_direction', 'sea_level_pressure'], axis=1)

# # Find missing Timestamp in order to prepare for auto-regressive model
# load_wea_['day'] = load_wea_['timestamp'].str[0:10]
# load_wea_.groupby(['day'], as_index=False).size().sort_values('size').head(10)

# Fill Site3-Building305 missing timestamps with mean values
# ['timestamp', 'meter_reading','air_temperature','cloud_coverage','dew_temperature]
load_wea_missing_305 = pd.DataFrame([["2016-03-13 02:00:00", 30.49, 14.5, np.nan, 8.1],
                                     ["2016-07-10 01:00:00", 45.92, 28.9, np.nan, 16.1],
                                     ["2016-07-10 03:00:00", 46.34, 28.4, np.nan, 16.1],
                                     ["2016-07-10 05:00:00", 45.58, 26.7, np.nan, 16.1],
                                     ["2016-08-16 16:00:00", 128.42, 31.1, np.nan, 24.7],
                                     ["2016-08-16 17:00:00", 128.24, 34.5, np.nan, 24.7],
                                     ["2016-11-06 00:00:00", 35.29, 14.8, np.nan, 5.0],
                                     ["2016-12-03 06:00:00", 42.24, 7.2, np.nan, -2.75]
                                    ], columns=load_wea_.columns)

# # Fill Site3-Building563 missing timestamps with mean values
# # ['timestamp', 'meter_reading','air_temperature','cloud_coverage','dew_temperature]
# load_wea_missing_563 = pd.DataFrame([["2016-03-13 02:00:00", 48.87, 15.52, np.nan, 8.13],
#                                      ["2016-07-10 01:00:00", 48.87, 15.52, np.nan, 8.13],
#                                      ["2016-07-10 03:00:00", 48.87, 15.52, np.nan, 8.13],
#                                      ["2016-07-10 05:00:00", 48.87, 15.52, np.nan, 8.13],
#                                      ["2016-11-06 00:00:00", 48.87, 15.52, np.nan, 8.13],
#                                      ["2016-12-03 06:00:00", 48.87, 15.52, np.nan, 8.13]
#                                     ], columns=load_wea_.columns)

# concate missing-timestamps dataframe with  on-hand timestamps dataframe
load_wea_complete = pd.concat([load_wea_, load_wea_missing_305], axis=0).sort_values(['timestamp']).

# store 'timestamp' for later use
timestamp = load_wea_complete['timestamp']

# drop 'timestamp' from the dataframe
load_wea_complete = load_wea_complete.drop('timestamp', axis=1)

print(np.sum(load_wea_complete.isna()))
# Fill missing 'air_temperature', 'dew_temperature' with mean values
load_wea_complete['air_temperature'] = load_wea_complete['air_temperature'].fillna(np.nanmean(load_w
load_wea_complete['dew_temperature'] = load_wea_complete['dew_temperature'].fillna(np.nanmean(load_w
# Fill missing 'cloud_coverage' with 10.0 in addition to [0.0, 2.0, 4.0, 6.0, 8.0, 9.0], representin
# Please refer to https://en.wikipedia.org/wiki/Okta for the meaning of each value
load_wea_complete['cloud_coverage'] = load_wea_complete['cloud_coverage'].fillna(10.0)
print(np.sum(load_wea_complete.isna()))
```

```
plt.figure(figsize=(16, 8))

plt.subplot(2, 3, 1);
# Plot 'meter_reading' along days in a year
plt.plot(load_wea_complete['meter_reading']);
plt.xlabel("The i-th hour in a year");
plt.ylabel('meter_reading');

plt.subplot(2, 3, 2);
# Plot relationship between 'air_temperature' and 'meter_reading'
plt.scatter(load_wea_complete['air_temperature'], load_wea_complete['meter_reading'], s=0.2);
plt.xlabel('air_temperature');
plt.ylabel('meter_reading');

plt.subplot(2, 3, 3);
# Plot relationship between 'air_temperature' and average 'meter_reading' for each temperature
temp = load_wea_complete.groupby('air_temperature', as_index=False)['meter_reading'].mean()
plt.plot(temp['air_temperature'], temp['meter_reading']);
plt.xlabel('air_temperature');
plt.ylabel('average meter_reading for the specific temperature');

plt.subplot(2, 3, 4);
# Plot relationship between "cloud_coverage" and 'meter_reading'
sns.violinplot(x="cloud_coverage", y="meter_reading", data=load_wea_complete)
plt.ylim([20, 70]);

load_wea_complete
```

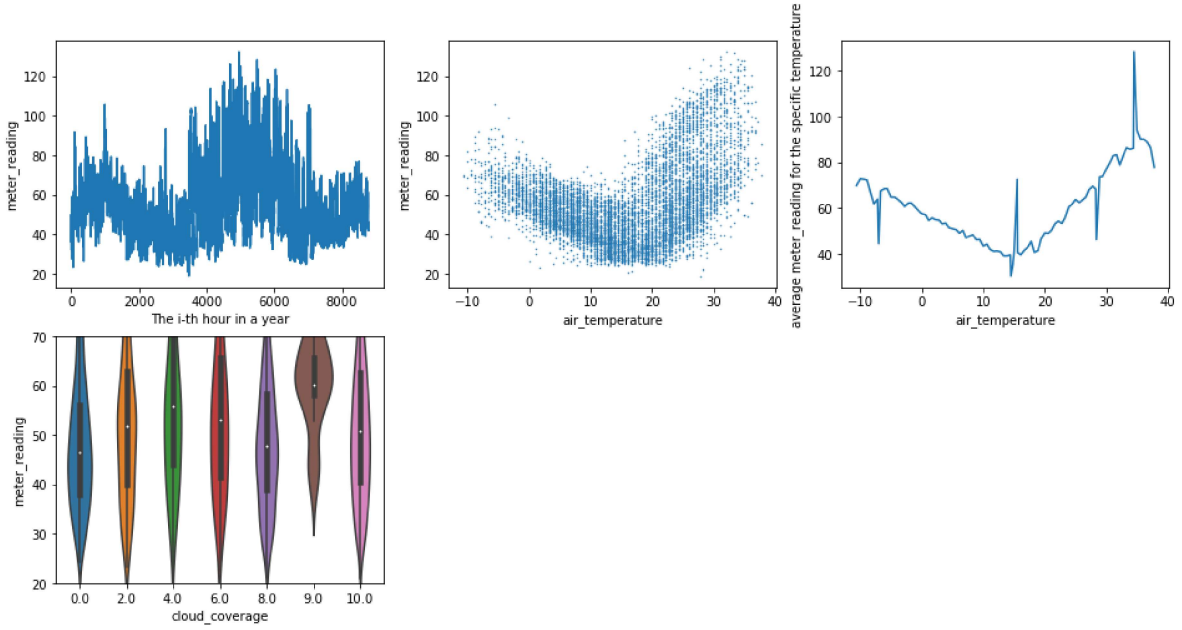
```
meter_reading          0
air_temperature        4
cloud_coverage      3649
dew_temperature        6
dtype: int64
meter_reading      0
air_temperature    0
cloud_coverage     0
dew_temperature    0
dtype: int64
```

Out[57]:

	meter_reading	air_temperature	cloud_coverage	dew_temperature
0	36.37	10.0	8.0	2.2
1	38.65	9.4	10.0	2.8
2	39.93	8.9	10.0	2.2
3	40.55	7.8	8.0	1.1
4	40.91	7.8	10.0	0.6
...	...	...	...	...
8779	42.70	9.4	10.0	-6.7
8780	46.39	8.9	10.0	-6.1
8781	45.36	8.9	6.0	-6.1
8782	45.45	8.9	10.0	-6.1

	meter_reading	air_temperature	cloud_coverage	dew_temperature
8783	42.02	8.9	10.0	-5.6

8784 rows × 4 columns



In [58]:

```
#  As shown in the plot above,
load_wea_complete['air_temperature_2'] = load_wea_complete.loc[:, 'air_temperature'] ** 2
load_wea_complete['air_temperature_|x_above15-15|'] = load_wea_complete.loc[:, 'air_temperature'].ap
load_wea_complete['air_temperature_|15-x_below15|'] = load_wea_complete.loc[:, 'air_temperature'].ap

# 'air_temperature', 'dew_temperature' have high correlation (0.89), so we drop 'dew_temperature'
# load_wea_complete['dew_temperature_2'] = load_wea_complete.loc[:, 'dew_temperature'] ** 2
# load_wea_complete['dew_temperature_|x_above15-15|'] = load_wea_complete.loc[:, 'dew_temperature'].
# load_wea_complete['dew_temperature_|15-x_below15|'] = load_wea_complete.loc[:, 'dew_temperature'].
load_wea_complete = load_wea_complete.drop(['dew_temperature'], axis=1)

load_wea_complete.head()
```

Out[58]:

	meter_reading	air_temperature	cloud_coverage	air_temperature_2	air_temperature_|x_above1 1
0	36.37	10.0	8.0	100.00	(
1	38.65	9.4	10.0	88.36	(
2	39.93	8.9	10.0	79.21	(
3	40.55	7.8	8.0	60.84	(
4	40.91	7.8	10.0	60.84	(

In [59]:

```
# One-hot encode the categorical variable 'cloud_coverage'
enc = OneHotEncoder()
enc.fit(load_wea_complete[['cloud_coverage']])
one_hot_var = pd.DataFrame(enc.transform(load_wea_complete[['cloud_coverage']]).toarray(), columns=e
print("Before OHE DataFrame:")
print(load_wea_complete.shape)
print("OHEed DataFrame:")
print(one_hot_var.shape)
load_wea_complete = pd.concat([load_wea_complete, one_hot_var], axis=1)
load_wea_complete = load_wea_complete.drop(['cloud_coverage'], axis=1)
print("Final DataFrame after OHE:")
print(load_wea_complete.shape)
load_wea_complete
```

```
Before OHE DataFrame:
(8784, 6)
OHEed DataFrame:
(8784, 7)
Final DataFrame after OHE:
(8784, 12)
```

Out[59]:

	meter_reading	air_temperature	air_temperature_2	air_temperature_|x_above15-15|	air_tempe ›
0	36.37	10.0	100.00	0.0	
1	38.65	9.4	88.36	0.0	
2	39.93	8.9	79.21	0.0	
3	40.55	7.8	60.84	0.0	
4	40.91	7.8	60.84	0.0	
...	...	...	...	...	
8779	42.70	9.4	88.36	0.0	
8780	46.39	8.9	79.21	0.0	
8781	45.36	8.9	79.21	0.0	
8782	45.45	8.9	79.21	0.0	
8783	42.02	8.9	79.21	0.0	

8784 rows × 12 columns

In [60]:

```
# Add auto-regressive variables to the dataframe, here we use the previous 24 hours
for hour in range(1, 25):
    load_wea_complete['meter_reading_pre_' + str(hour)] = np.array([np.nan] * hour+ list(load_wea_c

# Drop the first 24 hours, because their auto-regressive variables have null values
load_wea_complete = load_wea_complete.tail(len(load_wea_complete) - 24)

# Drop 'x0_10.0' representing cloud coverage unknown
load_wea_complete = load_wea_complete.drop(['x0_10.0'], axis=1)

print("Shape", load_wea_complete.shape)
print("Rank:", matrix_rank(load_wea_complete))

# The following lines check which columns are not full-rank
# for i in load_wea_complete.columns:
#     for j in load_wea_complete.columns:
#         if matrix_rank(load_wea_complete[[i, j]]) != 2:
#             print(i, j, matrix_rank(load_wea_complete[[i, j]]))

load_wea_complete
```

```
Shape (8760, 35)
Rank: 35
```

Out[60]:

	meter_reading	air_temperature	air_temperature_2	air_temperature_|x_above15-15|	air_tempe ›
24	45.71	4.4	19.36	0.0	
25	44.13	3.9	15.21	0.0	
26	49.89	3.9	15.21	0.0	
27	51.53	3.3	10.89	0.0	
28	45.04	3.3	10.89	0.0	
...	...	...	...	...	
8779	42.70	9.4	88.36	0.0	
8780	46.39	8.9	79.21	0.0	
8781	45.36	8.9	79.21	0.0	
8782	45.45	8.9	79.21	0.0	
8783	42.02	8.9	79.21	0.0	

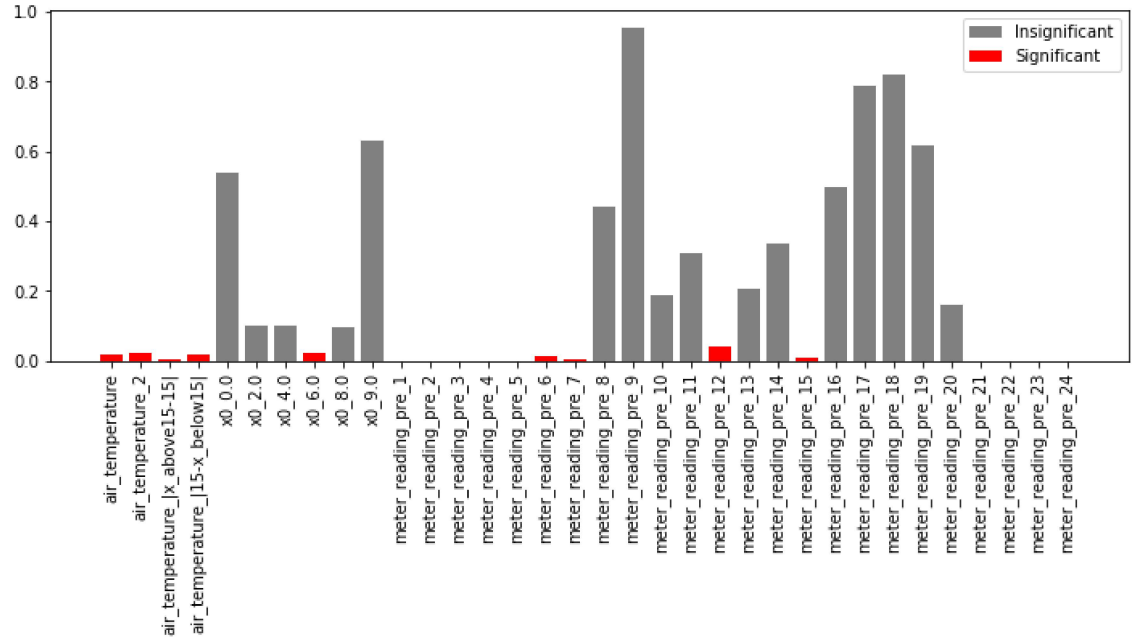
8760 rows × 35 columns

Training using LinearRegression

In [61]:

```
print("Columns in the final dataframe:", load_wea_complete.columns)
print()

X = load_wea_complete.drop(['meter_reading'], axis=1)
y = load_wea_complete['meter_reading']

# Standardize all the variables
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)

# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=0)
X_train = X[:7000]
y_train = y[:7000]
X_test= X[7000:]
y_test= y[7000:]

# reg = LinearRegression().fit(X_train, y_train)

# print("Train Score:", reg.score(X_train, y_train))
# print("Test Score:", reg.score(X_test, y_test))
X2 = sm.add_constant(X)
est = sm.OLS(y, X2)
est2 = est.fit()
# print(est2.summary())
# est2.params
p_without_const = est2.pvalues[1:]
p_without_const.index = load_wea_complete.columns[1:]

plt.figure(figsize=(12,4));
plt.bar(x=p_without_const.index, height=p_without_const, color='gray', label="Insignificant");
significant = p_without_const[p_without_const < 0.05]
plt.bar(x=significant.index, height=significant, color='r', label="Significant");
plt.legend();
plt.xticks(rotation = 90);
```

```
Columns in the final dataframe: Index(['meter_reading', 'air_temperature', 'air_temp
erature_2',
       'air_temperature_|x_above15-15|', 'air_temperature_|15-x_below15|',
       'x0_0.0', 'x0_2.0', 'x0_4.0', 'x0_6.0', 'x0_8.0', 'x0_9.0',
       'meter_reading_pre_1', 'meter_reading_pre_2', 'meter_reading_pre_3',
       'meter_reading_pre_4', 'meter_reading_pre_5', 'meter_reading_pre_6',
       'meter_reading_pre_7', 'meter_reading_pre_8', 'meter_reading_pre_9',
       'meter_reading_pre_10', 'meter_reading_pre_11', 'meter_reading_pre_12',
       'meter_reading_pre_13', 'meter_reading_pre_14', 'meter_reading_pre_15',
       'meter_reading_pre_16', 'meter_reading_pre_17', 'meter_reading_pre_18',
       'meter_reading_pre_19', 'meter_reading_pre_20', 'meter_reading_pre_21',
       'meter_reading_pre_22', 'meter_reading_pre_23', 'meter_reading_pre_24'],
      dtype='object')
```



In [62]:

```
print("Significant variables:", significant.index)
```

```
Significant variables: Index(['air_temperature', 'air_temperature_2',
       'air_temperature_|x_above15-15|', 'air_temperature_|15-x_below15|',
       'x0_6.0', 'meter_reading_pre_1', 'meter_reading_pre_2',
       'meter_reading_pre_3', 'meter_reading_pre_4', 'meter_reading_pre_5',
       'meter_reading_pre_6', 'meter_reading_pre_7', 'meter_reading_pre_12',
       'meter_reading_pre_15', 'meter_reading_pre_21', 'meter_reading_pre_22',
       'meter_reading_pre_23', 'meter_reading_pre_24'],
      dtype='object')
```

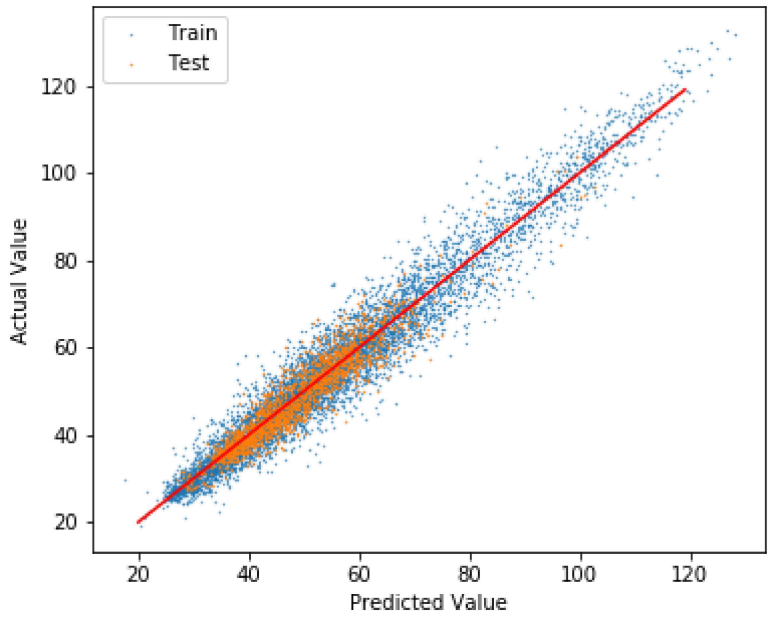
In [63]:

```
print("Columns in the final dataframe:", load_wea_complete.columns)
print()

X = load_wea_complete[significant.index]
y = load_wea_complete['meter_reading']

# Standardize all the variables
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)

# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=0)
X_train = X[:7000]
y_train = y[:7000]
X_test= X[7000:]
y_test= y[7000:]


reg = LinearRegression().fit(X_train, y_train)

print("Train Score:", reg.score(X_train, y_train))
print("Test Score:", reg.score(X_test, y_test))
```

```
Columns in the final dataframe: Index(['meter_reading', 'air_temperature', 'air_temp
erature_2',
       'air_temperature_|x_above15-15|', 'air_temperature_|15-x_below15|',
       'x0_0.0', 'x0_2.0', 'x0_4.0', 'x0_6.0', 'x0_8.0', 'x0_9.0',
       'meter_reading_pre_1', 'meter_reading_pre_2', 'meter_reading_pre_3',
       'meter_reading_pre_4', 'meter_reading_pre_5', 'meter_reading_pre_6',
       'meter_reading_pre_7', 'meter_reading_pre_8', 'meter_reading_pre_9',
       'meter_reading_pre_10', 'meter_reading_pre_11', 'meter_reading_pre_12',
       'meter_reading_pre_13', 'meter_reading_pre_14', 'meter_reading_pre_15',
       'meter_reading_pre_16', 'meter_reading_pre_17', 'meter_reading_pre_18',
       'meter_reading_pre_19', 'meter_reading_pre_20', 'meter_reading_pre_21',
       'meter_reading_pre_22', 'meter_reading_pre_23', 'meter_reading_pre_24'],
      dtype='object')

Train Score: 0.9455532019330204
Test Score: 0.8910442465435017
```

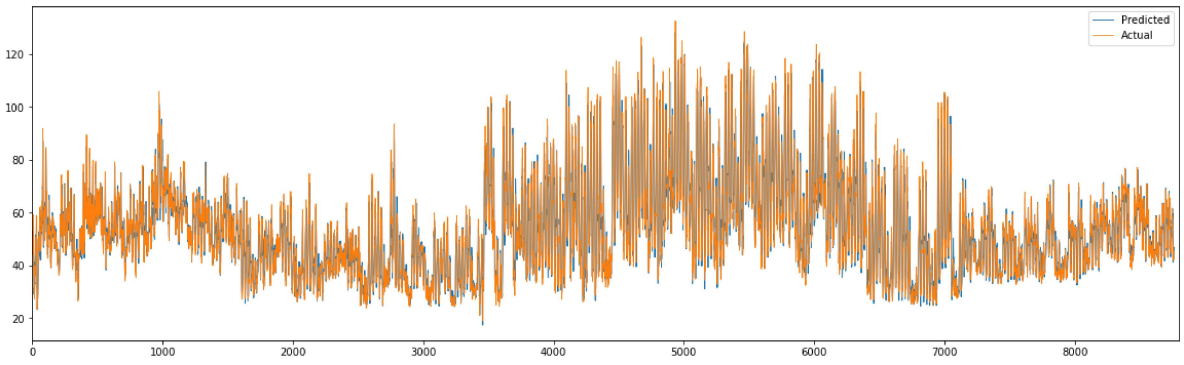
In [64]:

```
# Plot the relationship between "Predicted Value" and "Actual Value". Ideally, they should lie on th
plt.figure(figsize=(6, 5));
plt.scatter(reg.predict(X_train), y_train, s=0.1, label="Train");
plt.scatter(reg.predict(X_test), y_test, s=0.2, label="Test");
plt.plot(range(20, 120), range(20, 120), c='r');
plt.legend();
plt.xlabel("Predicted Value");
plt.ylabel("Actual Value");
```



In [65]:

```
# Plot predicted load and actual load
plt.figure(figsize=(20, 6));
plt.plot(reg.predict(X), label="Predicted", linewidth=1);
plt.plot(np.array(y), label="Actual", linewidth=0.8);
plt.xlim([0, 8800]);
plt.legend();
```



Optimize the model based on LinearRegression

In [66]:

```
X = load_wea_complete[significant.index] # Only use variables significant at 10% level
y = load_wea_complete['meter_reading']

# Standardize all the variables
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)

# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=0)
X_train = X[:7000]
y_train = y[:7000]
X_test= X[7000:]
y_test= y[7000:]


reg = LinearRegression().fit(X_train, y_train)

print("Train Score:", reg.score(X_train, y_train))
print("Test Score:", reg.score(X_test, y_test))
```

```
Train Score: 0.9455532019330204
Test Score: 0.8910442465435017
```

In [67]:

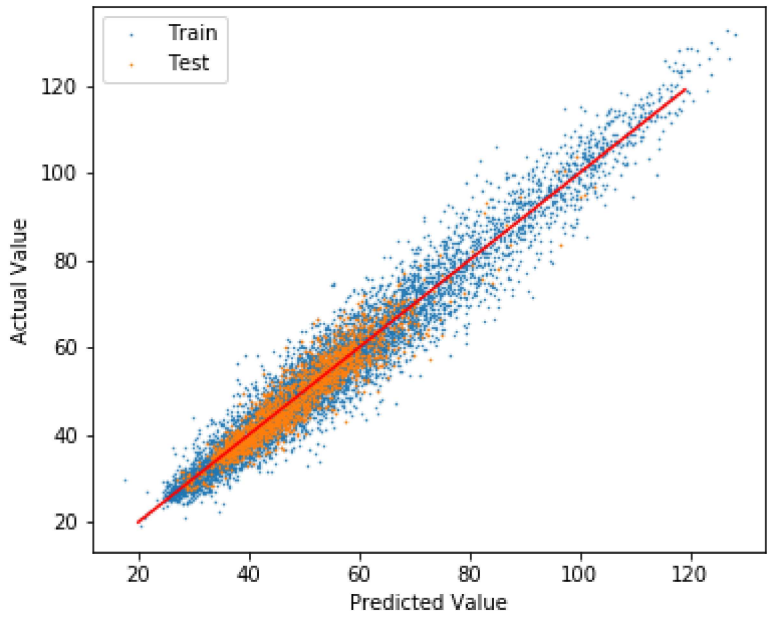
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
mean_squared_error(reg.predict(X_test), y_test), mean_absolute_error(reg.predict(X_test), y_test)
```

Out[67]:

```
(12.55398523312569, 2.657246663424488)
```

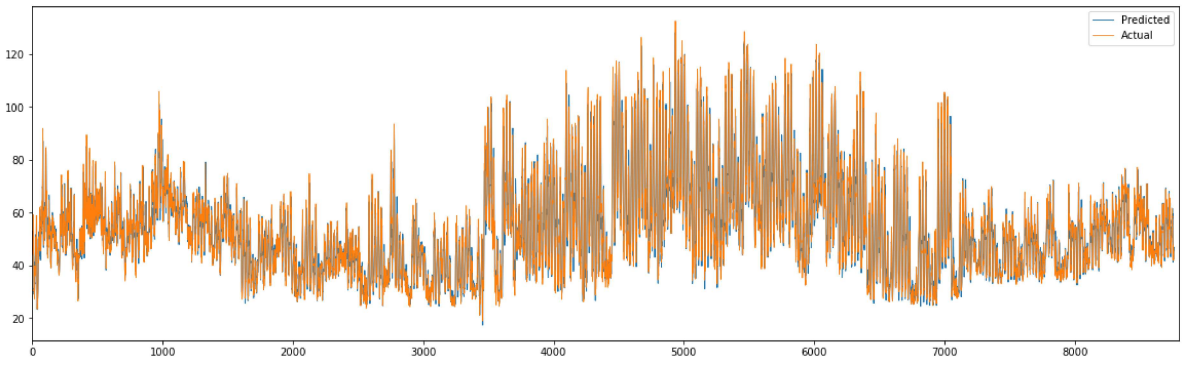
In [68]:

```
# Plot the relationship between "Predicted Value" and "Actual Value". Ideally, they should lie on th
plt.figure(figsize=(6, 5));
plt.scatter(reg.predict(X_train), y_train, s=0.2, label="Train");
plt.scatter(reg.predict(X_test), y_test, s=0.4, label="Test");
plt.plot(range(20, 120), range(20, 120), c='r');
plt.legend();
plt.xlabel("Predicted Value");
plt.ylabel("Actual Value");
```



In [69]:

```
# Plot predicted load and actual load
plt.figure(figsize=(20, 6));
plt.plot(reg.predict(X), label="Predicted", linewidth=1);
plt.plot(np.array(y), label="Actual", linewidth=0.8);
plt.xlim([0, 8800]);
plt.legend();
```



Training using MLPRegressor

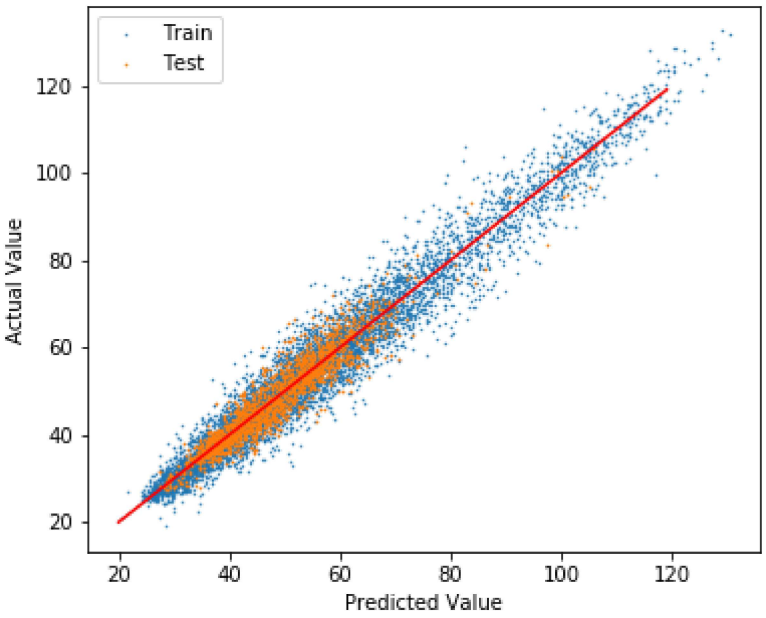
In [70]:

```
print("Columns in the final dataframe:", load_wea_complete.columns)
print()

X = load_wea_complete[significant.index]
y = load_wea_complete['meter_reading']

# Standardize all the variables
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)

# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=0)
X_train = X[:7000]
y_train = y[:7000]
X_test= X[7000:]
y_test= y[7000:]


reg = MLPRegressor(random_state=0, max_iter=1000, hidden_layer_sizes=(10, 6, 3)).fit(X_train, np.ra

print("Train Score:", reg.score(X_train, y_train))
print("Test Score:", reg.score(X_test, y_test))
```

```
Columns in the final dataframe: Index(['meter_reading', 'air_temperature', 'air_temp
erature_2',
       'air_temperature_|x_above15-15|', 'air_temperature_|15-x_below15|',
       'x0_0.0', 'x0_2.0', 'x0_4.0', 'x0_6.0', 'x0_8.0', 'x0_9.0',
       'meter_reading_pre_1', 'meter_reading_pre_2', 'meter_reading_pre_3',
       'meter_reading_pre_4', 'meter_reading_pre_5', 'meter_reading_pre_6',
       'meter_reading_pre_7', 'meter_reading_pre_8', 'meter_reading_pre_9',
       'meter_reading_pre_10', 'meter_reading_pre_11', 'meter_reading_pre_12',
       'meter_reading_pre_13', 'meter_reading_pre_14', 'meter_reading_pre_15',
       'meter_reading_pre_16', 'meter_reading_pre_17', 'meter_reading_pre_18',
       'meter_reading_pre_19', 'meter_reading_pre_20', 'meter_reading_pre_21',
       'meter_reading_pre_22', 'meter_reading_pre_23', 'meter_reading_pre_24'],
      dtype='object')

Train Score: 0.9504703134146729
Test Score: 0.892548101228349
```

In [71]:

```
# Plot the relationship between "Predicted Value" and "Actual Value". Ideally, they should lie on th
plt.figure(figsize=(6, 5));
plt.scatter(reg.predict(X_train), y_train, s=0.2, label="Train");
plt.scatter(reg.predict(X_test), y_test, s=0.4, label="Test");
plt.plot(range(20, 120), range(20, 120), c='r');
plt.legend();
plt.xlabel("Predicted Value");
plt.ylabel("Actual Value");
```



In [72]:

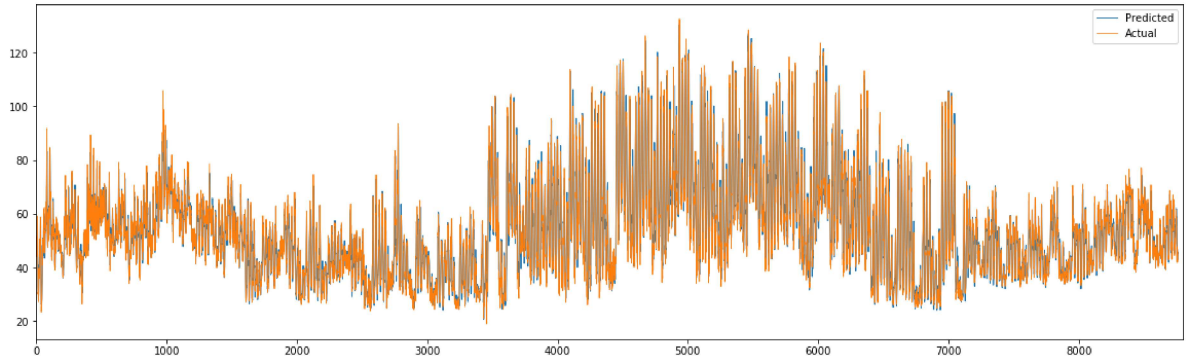
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
mean_squared_error(reg.predict(X_test), y_test), mean_absolute_error(reg.predict(X_test), y_test)
```

Out[72]:

(12.380709670271852, 2.6298407433848143)

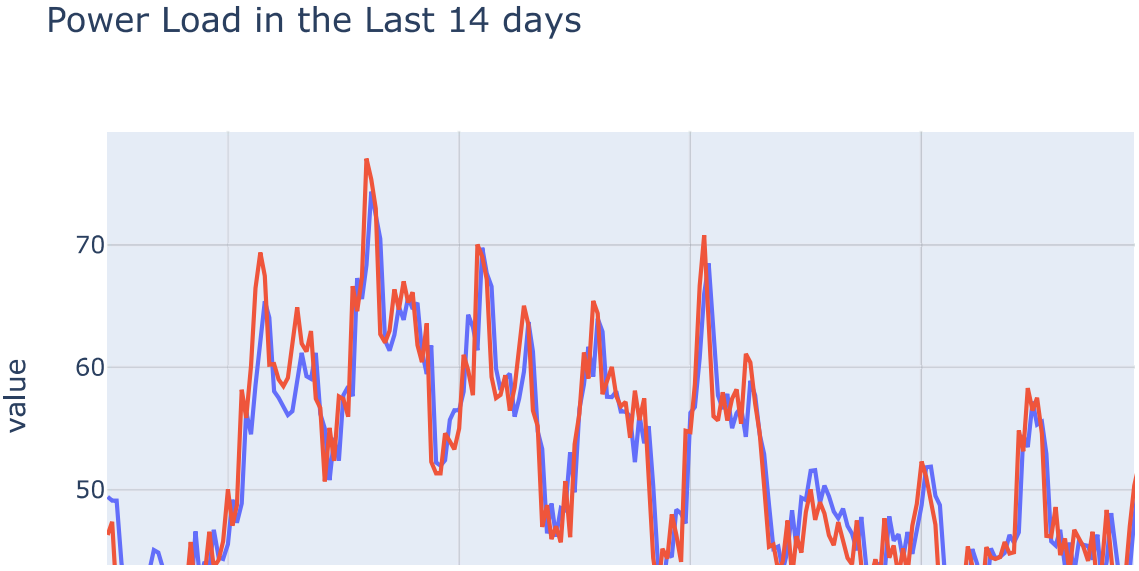
In [73]:

```
# Plot predicted load and actual load
plt.figure(figsize=(20, 6));
plt.plot(reg.predict(X), label="Predicted", linewidth=1);
plt.plot(np.array(y), label="Actual", linewidth=0.8);
plt.xlim([0, 8800]);
plt.legend();
```



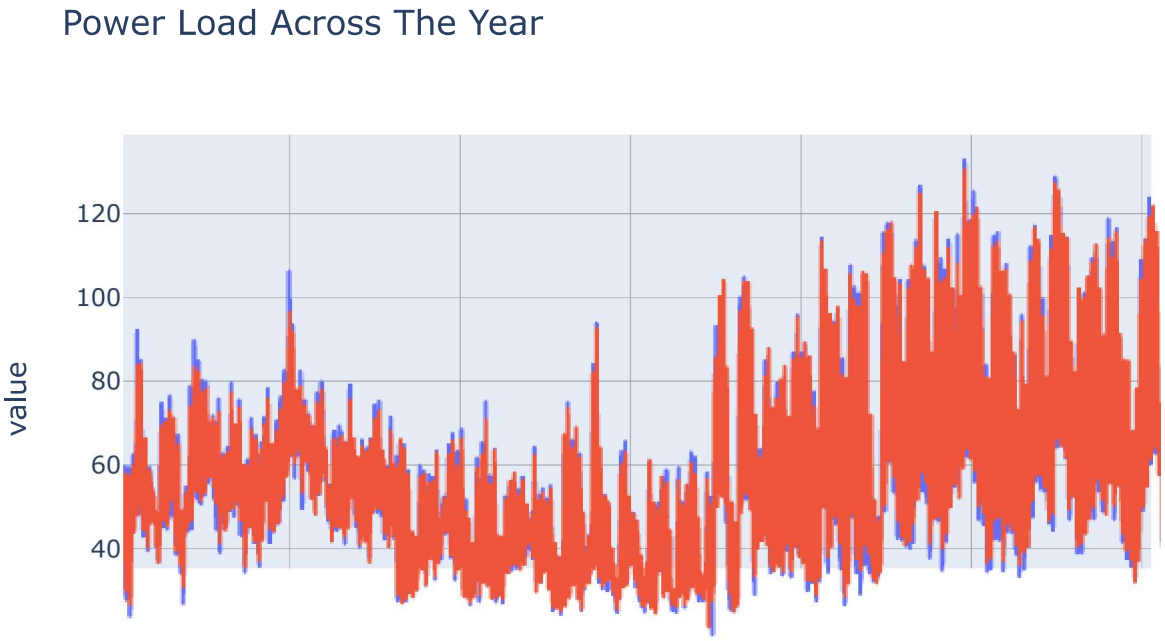
In [74]:

```
df = pd.DataFrame(data=[list(reg.predict(X)), list(np.array(y))], index=['Prediction', 'Actual']).tr
fig = px.line(df, y=['Prediction', 'Actual'], title="Power Load in the Last 14 days")
fig.update_xaxes(rangeslider_visible=True)
# df = pd.DataFrame(data=[list(reg.predict(X)), list(np.array(y))], index=['Prediction', 'Actual']).
# fig = px.line(df, y=['Prediction', 'Actual'])
# df = pd.DataFrame(data=[list(reg.predict(X)), list(np.array(y))], index=['Prediction', 'Actual']).
# fig = px.line(df, y=['Prediction', 'Actual'])
# df = pd.DataFrame(data=[list(reg.predict(X)), list(np.array(y))], index=['Prediction', 'Actual']).
# fig = px.line(df, y=['Prediction', 'Actual'])
fig.show()
```



In [75]:

```
df = load_wea_complete

df['prediction'] = reg.predict(X)
df['hour'] = df.index

fig = px.line(df, x='hour', y=['meter_reading', 'prediction'], title='Power Load Across The Year')
fig.update_xaxes(rangeslider_visible=True)

fig.show()
```



In [76]:

```
# Output weather information for Xinyu
load_wea_complete['timestamp'] = np.array(timestamp[24:])
load_wea_complete['meter_reading_prediction'] = np.array(reg.predict(X))
temp = load_wea_complete[['timestamp', 'meter_reading', 'meter_reading_prediction']]
temp.to_csv("305.csv")
temp
```

Out[76]:

	timestamp	meter_reading	meter_reading_prediction
24	2016-01-02 00:00:00	45.71	44.951156
25	2016-01-02 01:00:00	44.13	45.435744
26	2016-01-02 02:00:00	49.89	44.081044
27	2016-01-02 03:00:00	51.53	49.479615
28	2016-01-02 04:00:00	45.04	51.086487
...	...	...	...
8779	2016-12-31 19:00:00	42.70	46.849017
8780	2016-12-31 20:00:00	46.39	42.631869
8781	2016-12-31 21:00:00	45.36	45.238999
8782	2016-12-31 22:00:00	45.45	45.581927
8783	2016-12-31 23:00:00	42.02	45.289690

8760 rows × 3 columns

Optimization Codes

In [1]:
```
import gurobipy as gp
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from gurobipy import GRB

step = 24 #daily time intervel
C_b = 100000

# TOU
def D_ug():
    # the unit electricity fee (may vary during the day)
    a=np.ones((7))*0.2671
    b=np.ones((3))*0.7405
    c=np.ones((1))*1.2662
    d=np.ones((2))*1.3955
    e=np.ones((2))*1.2662
    f=np.ones((1))*0.7405
    g=np.ones((1))*1.3955
    h=np.ones((1))*0.7405
    i=np.ones((3))*1.2662
    j=np.ones((2))*0.7405
    k=np.ones((1))*0.2671
    D_ug=np.hstack((a,b,c,d,e,f,g,h,i,j,k))
    return D_ug

    return D_ug
def D_CO2():
    # the unit shadow price of CO2
    D_CO2 = pd.read_csv('shadow_price_hours.csv')
    D_CO2 = D_CO2.drop(columns =['Unnamed: 0'])
    D_CO2 = D_CO2.drop(columns =['day'])
    return D_CO2

D_ug = D_ug() #electricity fee
D_CO2 = D_CO2() #cost of CO2
```

In [2]:
```
solar = pd.read_csv('Beijing_Radiation.csv')
solar = solar.drop(columns =['Unnamed: 0'])
solar = solar.rename(columns={'Unnamed: 1':'hours'})
solar.head()
```

Out[2]:

	hours	HORI_TOTAL_RAD
0	0	0.0
1	1	0.0
2	2	0.0
3	3	0.0
4	4	0.0

In [3]:
```
data = pd.read_csv('305.csv')
data = data.rename(columns={'Unnamed: 0':'hours'})
data = pd.merge(data,D_CO2, how = 'left', on = ['hours'])
data = pd.merge(data,solar, how = 'left', on = ['hours'])
data = data.rename(columns = {'HORI_TOTAL_RAD':'insolation(W/m2)'})
data = data.rename(columns = {'meter_reading':'energy_con(kWh)'})
data = data.rename(columns = {'meter_reading_prediction':'energy_pred(kWh)'})
data = data.rename(columns = {'shadow_price':'shadow_price(yuan/ton)'})

data = data.drop(data.index[data['insolation(W/m2)'].isnull()])
```

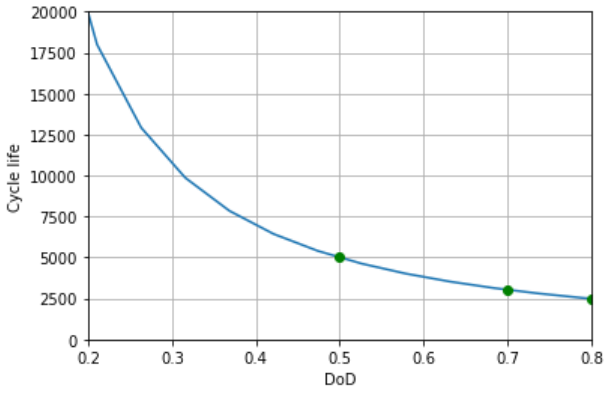
In [4]:
```
days = len(data['hours'])/24
days = int(days)
```

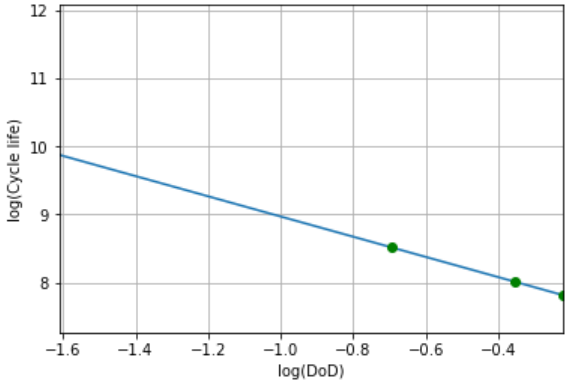
In [5]:
```
# data['PV'] = data['insolation(W/m2)'] * 1000 * 0.15 /1000
# plt.figure(figsize=(20,6))
# plt.plot(data['hours'],data['PV'])
# plt.plot(data['hours'],data['energy_con(kWh)'])
# plt.show()
data['energy_con(kWh)'].max()
```

Out[5]: 132.48

In [6]:
```
# battery degredation
# three DOD-Cycle points from datasheet
x = [0.8, 0.7, 0.5]
y = [2500,3000,5000]
logy = np.log(y)
logx = np.log(x)

# literature identifies an exponential relationship of the Cycle-DOD model
# Calculate the coefficients.

mm, b = np.polyfit(logx, logy, 1)
# Print the findings
print ('slope =', mm,'intercept =', b)
print ('log(Cycle life) =', mm,'* log(DOD) +',b)

xx = np.linspace(0,1,20)
logxx = np.log(xx)
logCycle =  mm*logxx + b
Cycle = np.exp(logCycle)

#visualization
plt.plot(xx, Cycle)
plt.plot( x[0], y[0], 'go' )
plt.plot( x[1], y[1], 'go' )
plt.plot( x[2], y[2], 'go' )
# plt.yscale('log')
plt.xlabel('DoD')
plt.ylabel('Cycle life')
plt.xlim(0.2,0.8)
plt.ylim(0,20000)
plt.grid('on')
plt.show()

plt.plot(logxx, logCycle)
plt.plot( logx[0], logy[0], 'go' )
plt.plot( logx[1], logy[1], 'go' )
plt.plot( logx[2], logy[2], 'go' )
# plt.yscale('log')
plt.xlabel('log(DoD)')
plt.ylabel('log(Cycle life)')
plt.xlim(np.log(0.2),np.log(0.8))
# plt.ylim(0,20000)
plt.grid('on')
plt.show()
```

```
slope = -1.483192289615199 intercept = 7.486517964870765
log(Cycle life) = -1.483192289615199 * log(DOD) + 7.486517964870765

<ipython-input-6-4ecb7750070d>:17: RuntimeWarning: divide by zero encountered in log
  logxx = np.log(xx)
```

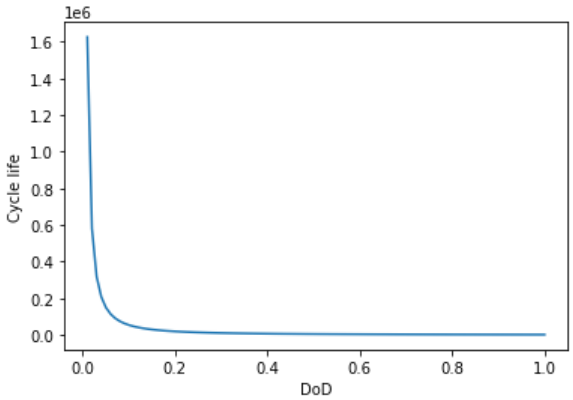




In [7]:
```
def cycle(DOD):
    logCycle =  mm*np.log(DOD) + b
    cycle = np.exp(logCycle)
    # gurobipy.LinExpr has no callable exp method
    return cycle

def degra_cost(DOD1,DOD2,C_b):
    C_d = max(C_b* (1/cycle(DOD2)-1/cycle(DOD1)),0)
    return C_d

xx = np.linspace(0,1,100)
plt.plot(xx, cycle(xx))
plt.xlabel('DoD')
plt.ylabel('Cycle life')
```

```
<ipython-input-7-cd66cd774ee0>:2: RuntimeWarning: divide by zero encountered in log
  logCycle =  mm*np.log(DOD) + b
```

Out[7]: Text(0, 0.5, 'Cycle life')



Problem is:

- gurobipy.LinExpr has no callable exp method
- gurobipy.LinExpr has no callable rint method
- MIP cannot deal with 1/cycle (maybe??)

Solution

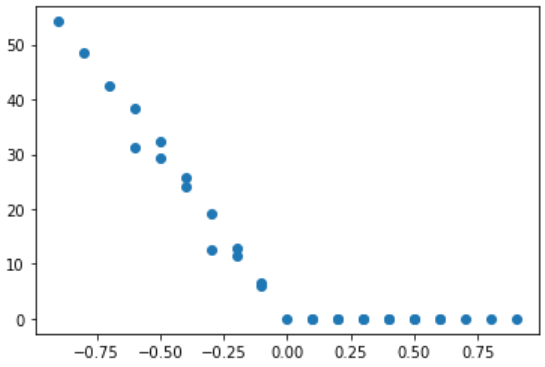
- find (dod1-dod2)-cycle (mean value)
- fit a 10-segment piecewise linear function based on the 10 datapoints

In [8]:
```
cost_dod = pd.DataFrame(0, index=np.arange(100),columns=['dod1','dod2','cost'])

k=0
for soc1 in np.arange(0,1,0.1):
    dod1 = 1-soc1
    for soc2 in np.arange(0,1,0.1):
        dod2 = 1-soc2
        cost_dod.iloc[k,:] = [dod1,dod2,degra_cost(dod1,dod2,C_b)]
        k=k+1
# cost_dod = cost_dod.set_index(['dod1', 'dod2'])
cost_dod['dod1-dod2'] = cost_dod['dod1']- cost_dod['dod2']
cost_dod1 = cost_dod.drop(['dod1','dod2'],axis=1)
cost_dod1 = cost_dod.groupby('dod1-dod2',as_index=False).mean()
xx = cost_dod1['dod1-dod2']
yy = cost_dod1['cost']
plt.plot(xx, yy, 'o')
cost_dod
```

Out[8]:

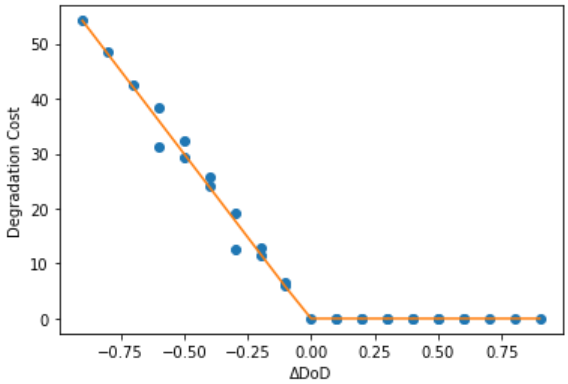
	dod1	dod2	cost	dod1-dod2
0	1.0	1.0	0.000000	0.0
1	1.0	0.9	0.000000	0.1
2	1.0	0.8	0.000000	0.2
3	1.0	0.7	0.000000	0.3
4	1.0	0.6	0.000000	0.4
...	...	...	...	...
95	0.1	0.5	18.209463	-0.4
96	0.1	0.4	12.559375	-0.3
97	0.1	0.3	7.557063	-0.2
98	0.1	0.2	3.308872	-0.1
99	0.1	0.1	0.000000	0.0

100 rows × 4 columns



In [9]:
```
import pwlf

pw_fit = pwlf.PiecewiseLinFit(xx, yy)
breaks = pw_fit.fit(2)
print(breaks)
# pw_fit.calc_slopes()
inter = pw_fit.intercepts
slope = pw_fit.slopes

# predict for the determined points
yHat = pw_fit.predict(xx)

# plot the results
plt.figure()
plt.plot(xx, yy, 'o')
plt.plot(xx, yHat, '-')
plt.xlabel('ΔDoD')
plt.ylabel('Degradation Cost')
plt.show()
```

```
[-0.9        -0.00532236  0.9        ]
```



In [10]:
```
def cost1(delta_dod):
    cost1 = slope[0] * delta_dod +inter[0]
    return cost1

def cost2(delta_dod):
    cost2 = 0
    return cost2
```

In [11]:
```
data[data.index>6923]
```

Out[11]:

	hours	timestamp	energy_con(kWh)	energy_pred(kWh)	shadow_price(yuan/ton)	insolation(W/m2)
6924	6948	2016-10-16 12:00:00	30.87	29.396892	52.9375	701.530029
6925	6949	2016-10-16 13:00:00	35.06	33.602610	52.9375	680.580017
6926	6950	2016-10-16 14:00:00	43.42	37.600703	52.9375	586.710022
6927	6951	2016-10-16 15:00:00	51.01	44.861102	52.9375	438.329987
6928	6952	2016-10-16 16:00:00	52.85	51.165843	52.9375	263.609985
...	...	...	...	...	...	...
8731	8755	2016-12-30 19:00:00	57.70	62.914775	55.4000	0.000000
8732	8756	2016-12-30 20:00:00	59.29	55.380193	55.4000	0.000000
8733	8757	2016-12-30 21:00:00	52.43	56.878605	55.4000	0.000000
8734	8758	2016-12-30 22:00:00	44.41	49.681997	55.4000	0.000000
8735	8759	2016-12-30 23:00:00	49.26	43.772856	55.4000	0.000000

1812 rows × 6 columns

In [13]:
```
# PV - 1200 m2, 20% peak effi (240 kW capacity)
Ppv = np.array(data['insolation(W/m2)'] * 1200 * 0.15 /1000)
```

In [14]:
```
p=6
s=20
# s=30
# s=20
# s=10

# LFP battery module 200 Ah, 25.6 V
Q_b = p*s*200*25.6/1000 #battery capacity kWh
```

In [15]:
```
Q_b
```

Out[15]: 614.4

In [33]:
```
Pload = np.array(data['energy_con(kWh)'])
D_CO2 = np.array(data['shadow_price(yuan/ton)'])

C_b = 100*Q_b   #battery cost

# max current when contant charging/discharging
Icharmax = 50*p #A
Idismax = 200*p  #A
V_bat = 25.6*s  #V

Pcharmax = Icharmax*120/1000  #kW
Pdismax = -Idismax*120/1000   #kW

# efficiency
yita_pv_derating = 0.85 #PV derating
yita_pv_inv = 0.9  #PV inverter
yita_pv = yita_pv_derating * yita_pv_inv
yita_columb = 0.9  #Columb efficiency (assume when charging)
yita_ug = 0.95     #grid efficiency (due to in building line losses)
```

In [34]:
```
Pcharmax
```

Out[34]: 36.0

In [35]:
```
D_ug=np.tile(D_ug, (364))
```

In [36]:
```
m = gp.Model("Opration")
bigM=10000000 # a large enough number

P_bat = m.addVars(days*step+1,lb=Pdismax,ub=Pcharmax ,name ='P_bat') # if charging, + ; if discharging, -
P_ug = m.addVars(days*step,lb=0,ub=160,name="P_ug")
SOC = m.addVars(days*step+1,lb=0.1,ub=1,name="SOC")
C_d = m.addVars(days*(step),lb=0,ub=1,name="bat_deg_cost") #cost of battery degradation
C_dummy = m.addVars(days*step,vtype=GRB.BINARY,name="Charging_dummy")
# w = m.addVars(3,days*step,lb=0,ub=1,vtype=GRB.BINARY,name="piecewise")
E_CO2 = m.addVars(days*step,lb=0,ub=160,name="E_CO2")
# CO2 Emission of UG in China: 632.104 g_CO2/kWh (2015)

# Set objective
m.setObjective((gp.quicksum(P_ug[i]*D_ug[i] for i in range (days*step)) \
              + gp.quicksum(E_CO2[i]*D_CO2[i]+C_d[i] for i in range (days*step))), GRB.MINIMIZE)


# add indicator constraint
# c_dummy = 1 -> P_bat > 0
for i in range (days*step):
    m.addGenConstrIndicator(C_dummy[i], 1, P_bat[i] >= 0)

# power balance
m.addConstrs((P_ug[i] * yita_ug + Ppv[i] * yita_pv >= \
              C_dummy[i] * P_bat[i] + yita_columb * (1-C_dummy[i]) * P_bat[i] + Pload[i] \
              for i in range (step*days)),"c1")

# battery SOC update
m.addConstrs(((SOC[i+1]-SOC[i])*Q_b == P_bat[i] for i in range (step*days)),"c2")

# battery degredation
m.addConstrs((cost1(SOC[i+1]-SOC[i])*(1-C_dummy[i])+\
              cost2(SOC[i+1]-SOC[i])*C_dummy[i] == C_d[i] for i in range (days*step)),"c4.1")

# m.addConstrs((gp.quicksum(w[j,i] for j in range(0,3))==1 for i in range (days*step)),"c4.2")

# for i in range (days*step):
#     m.addGenConstrIndicator(w[0,i], 1, (SOC[i+1]-SOC[i]) <= -0.6)
#     m.addGenConstrIndicator(w[2,i], 1, (SOC[i+1]-SOC[i]) >=0)

# battry constraint at the end of day
for k in range (days):
    m.addConstr(SOC[0] == SOC[k*step])
    # m.addConstr((SOC[0] == 0.5), "c6")

# CO2 emission
m.addConstrs((E_CO2[i] == P_ug[i] * 632.104/10**6 for i in range (days*step)), "c5")
```

Out[36]:
```
{0: <gurobi.Constr *Awaiting Model Update*>,
 1: <gurobi.Constr *Awaiting Model Update*>,
 2: <gurobi.Constr *Awaiting Model Update*>,
 3: <gurobi.Constr *Awaiting Model Update*>,
 4: <gurobi.Constr *Awaiting Model Update*>,
 5: <gurobi.Constr *Awaiting Model Update*>,
 6: <gurobi.Constr *Awaiting Model Update*>,
 7: <gurobi.Constr *Awaiting Model Update*>,
 8: <gurobi.Constr *Awaiting Model Update*>,
 9: <gurobi.Constr *Awaiting Model Update*>,
 10: <gurobi.Constr *Awaiting Model Update*>,
 11: <gurobi.Constr *Awaiting Model Update*>,
 12: <gurobi.Constr *Awaiting Model Update*>,
 13: <gurobi.Constr *Awaiting Model Update*>,
 14: <gurobi.Constr *Awaiting Model Update*>,
 15: <gurobi.Constr *Awaiting Model Update*>,
 16: <gurobi.Constr *Awaiting Model Update*>,
 17: <gurobi.Constr *Awaiting Model Update*>,
 18: <gurobi.Constr *Awaiting Model Update*>,
```

In [37]:

```
def printSolution():
    if m.status == GRB.OPTIMAL:
        print('\nCost: %g' % m.objVal)
        #print('Q: %g' % Q)
        #print('Pcharmax: %g' % Pcharmax)

        P_ugx = m.getAttr('x', P_ug)
        P_batx = m.getAttr('x', P_bat)
        SOCx = m.getAttr('x', SOC)
        C_dx = m.getAttr('x', C_d)
        C_dummyx = m.getAttr('x', C_dummy)

    else:
        print('No solution')
    return P_ugx,P_batx,SOCx,C_dx,C_dummyx
```

In [38]:
```
# Solve
m.optimize()
P_ugx,P_batx,SOCx,C_dx,C_dummyx = printSolution()

if m.solCount == 0:
    print("Model is infeasible")
    m.computeIIS()
    m.write("model_iis.ilp")

if m.status == GRB.INFEASIBLE:
    vars = m.getVars()
    ubpen = [1.0]*m.numVars
    m.feasRelax(1, False, vars, None, ubpen, None, None)
    m.optimize()
    printSolution()

vars = m.getVars()
constrs = m.getConstrs()
```

```
Gurobi Optimizer version 9.1.1 build v9.1.1rc0 (win64)
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads
Optimize a model with 17836 rows, 52418 columns and 44406 nonzeros
Model fingerprint: 0x5aaf85ae
Model has 17472 quadratic constraints
Model has 8736 general constraints
Variable types: 43682 continuous, 8736 integer (8736 binary)
Coefficient statistics:
  Matrix range     [6e-04, 6e+02]
  QMatrix range    [1e-01, 6e+01]
  QLMatrix range   [3e-01, 6e+01]
  Objective range  [3e-01, 7e+01]
  Bounds range     [1e-01, 2e+02]
  RHS range        [0e+00, 0e+00]
  QRHS range       [3e-02, 1e+02]
Presolve added 42433 rows and 0 columns
Presolve removed 0 rows and 9516 columns
Presolve time: 0.39s
Presolved: 60269 rows, 42902 columns, 180813 nonzeros
Variable types: 34166 continuous, 8736 integer (8736 binary)

Deterministic concurrent LP optimizer: primal and dual simplex
Showing first log only...

Concurrent spin time: 0.00s

Solved with dual simplex

Root relaxation: objective 1.307024e+05, 35930 iterations, 2.08 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 130702.444    0 8322          - 130702.444      -     -    6s
H    0     0                    243768.76425 130702.444  46.4%     -    7s
H    0     0                    243670.95706 130702.444  46.4%     -    7s
H    0     0                    243618.82370 130702.444  46.3%     -    8s
     0     0 194660.508    0 5826 243618.824 194660.508  20.1%     -   18s
H    0     0                    221346.08660 194660.508  12.1%     -   19s
     0     0 195702.908    0 4266 221346.087 195702.908  11.6%     -   23s
     0     0 195865.056    0 3606 221346.087 195865.056  11.5%     -   23s
     0     0 195865.190    0 3606 221346.087 195865.190  11.5%     -   24s
     0     0 207342.048    0 1918 221346.087 207342.048  6.33%     -   25s
H    0     0                    213329.51309 207342.048  2.81%     -   25s
     0     0 207629.801    0 1688 213329.513 207629.801  2.67%     -   25s
     0     0 207631.209    0 1688 213329.513 207631.209  2.67%     -   26s
     0     0 209417.739    0  528 213329.513 209417.739  1.83%     -   29s
H    0     0                    209920.62879 209417.739  0.24%     -   29s
     0     0 209436.267    0  508 209920.629 209436.267  0.23%     -   29s
     0     0 209730.036    0  193 209920.629 209730.036  0.09%     -   30s
H    0     0                    209920.30636 209730.036  0.09%     -   31s
     0     0 209734.688    0  182 209920.306 209734.688  0.09%     -   31s
     0     0 209821.616    0   83 209920.306 209821.616  0.05%     -   31s
     0     0 209821.649    0   83 209920.306 209821.649  0.05%     -   31s
     0     0 209850.489    0   62 209920.306 209850.489  0.03%     -   32s
H    0     0                    209913.49540 209850.489  0.03%     -   32s
H    0     0                    209913.19720 209850.489  0.03%     -   32s
     0     0 209850.537    0   63 209913.197 209850.537  0.03%     -   32s
     0     0 209853.898    0   44 209913.197 209853.898  0.03%     -   33s
     0     0 209853.902    0   45 209913.197 209853.902  0.03%     -   33s
     0     0 209854.625    0   44 209913.197 209854.625  0.03%     -   33s
```

```
     0     0 209854.625    0   41 209913.197 209854.625  0.03%     -   37s
H    0     0                     209901.46641 209854.625  0.02%     -   38s
H    0     0                     209898.48896 209854.625  0.02%     -   38s
H    0     0                     209896.91138 209854.625  0.02%     -   40s
H    0     0                     209896.80847 209854.625  0.02%     -   40s
H    0     2                     209896.16362 209854.625  0.02%     -   41s
     0     2 209854.625    0   41 209896.164 209854.625  0.02%     -   41s
H   77   146                     209895.84119 209859.033  0.02%   6.9   42s
H  126   146                     209894.99229 209859.033  0.02%   5.0   42s
H  136   146                     209894.52634 209859.033  0.02%   4.7   42s
H  162   334                     209893.55906 209859.033  0.02%   4.3   44s
H  189   334                     209892.91420 209859.033  0.02%   4.1   44s
H  219   334                     209890.25664 209859.033  0.01%   3.9   44s
   333   714 209862.809   52   18 209890.257 209859.033  0.01%   3.7   46s
H  346   714                     209888.25664 209859.033  0.01%   3.7   46s
H  713  1043                     209873.06899 209859.033  0.01%   3.9   49s
*  761  1043             150    209867.20545 209859.033  0.00%   3.8   49s

Cutting planes:
  Gomory: 129
  Implied bound: 508
  MIR: 12715
  RLT: 9
  Relax-and-lift: 79

Explored 1081 nodes (77400 simplex iterations) in 49.17 seconds
Thread count was 12 (of 12 available processors)

Solution count 10: 209867 209873 209888 ... 209896

Optimal solution found (tolerance 1.00e-04)
Best objective 2.098672054477e+05, best bound 2.098593593764e+05, gap 0.0037%

Cost: 209867
```

In [39]:
```
P_ugx = pd.Series(P_ugx)
P_batx = pd.Series(P_batx)
C_dx = pd.Series(C_dx)
SOCx = pd.Series(SOCx)
C_dummyx = pd.Series(C_dummyx)
```

In [40]:
```
# percentage of energy provided by PV
sum(Ppv)/sum(Pload)
```

Out[40]:
```
0.5319512810824824
```

In [41]:
```
# the ratio of annual battery degradation cost versus battery cost
# indicates the lifetime of battery
C_b/sum(C_dx)
```

Out[41]:
```
13.178044329302608
```

In [42]:
```
# equivalent cycles of battery
sum(abs(P_batx)/2)/Q_b/0.9,sum(P_batx)
```

Out[42]:
```
(116.12599349757393, -447.3388641323139)
```

In [43]:
```
# w bat and PV
a= sum(P_ugx[i]*D_ug[i] for i in range (days*step))+\
sum(P_ugx[i] * 632.104/10**6 * D_CO2[i] for i in range (days*step))+\
sum(C_dx)
```

In [44]:
```
# sum(((Pload-Ppv)<0).astype(int))
```

In [45]:
```
# w/o bat, w/ PV
# P = np.zeros((len(Pload)))
# for i in range (days*step):
#     P[i] = max((Pload[i]-Ppv[i]),0)
P = Pload-Ppv
P[P < 0] = 0
b = sum(P[i]*D_ug[i] for i in range (days*step))+\
sum(P[i] * 632.104/10**6 * D_CO2[i] for i in range (days*step))
```

In [46]:
```
# w/o bat and PV
c = sum(Pload[i]*D_ug[i] for i in range (days*step))+\
sum(Pload[i] * 632.104/10**6 * D_CO2[i] for i in range (days*step))
```

In [47]:
```
(c-a)/c,(b-a)/b
```

Out[47]: (0.5000172412929382, 0.03500244317684181)

Visualization

May 8, 2021

[1]:
```
import pandas as pd
import numpy as np
data1 = pd.read_csv('result1-2.csv')
data2 = pd.read_csv('result2-2.csv')
```

[2]:
```
data1
```

[2]:

	Unnamed: 0	index	P_ug	C_d	C_dummy	Pload	Ppv
0	0	0	86.010526	0.0	1.0	45.71	0.0
1	1	1	46.452632	0.0	1.0	44.13	0.0
2	2	2	52.515789	0.0	1.0	49.89	0.0
3	3	3	54.242105	0.0	1.0	51.53	0.0
4	4	4	47.410526	0.0	1.0	45.04	0.0
…	…	…	…	…	…	…	…
8731	8731	8731	48.030667	1.0	0.0	57.70	0.0
8732	8732	8732	49.704351	1.0	0.0	59.29	0.0
8733	8733	8733	42.483299	1.0	0.0	52.43	0.0
8734	8734	8734	34.041193	1.0	0.0	44.41	0.0
8735	8735	8735	39.146456	1.0	0.0	49.26	0.0

[8736 rows x 7 columns]

[3]:
```
data2
```

[3]:

	Unnamed: 0	index	P_bat	SOC
0	0	0	36.000000	0.707031
1	1	1	0.000000	0.765625
2	2	2	0.000000	0.765625
3	3	3	0.000000	0.765625
4	4	4	0.000000	0.765625
…	…	…	…	…
8731	8731	8731	-13.412074	0.322463
8732	8732	8732	-13.412074	0.300634
8733	8733	8733	-13.412074	0.278804
8734	8734	8734	-13.412074	0.256975
8735	8735	8735	-13.412074	0.235145

[8736 rows x 4 columns]

[4]:
```
data1 = data1.drop(columns = {'Unnamed: 0'})
data1.rename(columns = {'index':'hour'})
```

[4]:

	hour	P_ug	C_d	C_dummy	Pload	Ppv
0	0	86.010526	0.0	1.0	45.71	0.0
1	1	46.452632	0.0	1.0	44.13	0.0
2	2	52.515789	0.0	1.0	49.89	0.0
3	3	54.242105	0.0	1.0	51.53	0.0
4	4	47.410526	0.0	1.0	45.04	0.0
…	…	…	…	…	…	…
8731	8731	48.030667	1.0	0.0	57.70	0.0
8732	8732	49.704351	1.0	0.0	59.29	0.0
8733	8733	42.483299	1.0	0.0	52.43	0.0
8734	8734	34.041193	1.0	0.0	44.41	0.0
8735	8735	39.146456	1.0	0.0	49.26	0.0

[8736 rows x 6 columns]

[5]:
```
data2 = data2.drop(columns = {'Unnamed: 0'})
data2.rename(columns = {'index':'hour'})
```

[5]:

	hour	P_bat	SOC
0	0	36.000000	0.707031
1	1	0.000000	0.765625
2	2	0.000000	0.765625
3	3	0.000000	0.765625
4	4	0.000000	0.765625
…	…	…	…
8731	8731	-13.412074	0.322463
8732	8732	-13.412074	0.300634
8733	8733	-13.412074	0.278804
8734	8734	-13.412074	0.256975
8735	8735	-13.412074	0.235145

[8736 rows x 3 columns]

[6]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[7]:
```
data = pd.merge(data1,data2)
data['P_BATC'] = 0.00
data['P_BATD'] = 0.00
for i in range(0,8735):
    if data['P_bat'][i] > 0:
        data['P_BATC'][i] = -data['P_bat'][i]
    if data['P_bat'][i] < 0:
        data['P_BATD'][i] = -data['P_bat'][i]
```

```
data['Pload'] = -data['Pload']
data
```

```
<ipython-input-7-305141713cbd>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['P_BATC'][i] = -data['P_bat'][i]
<ipython-input-7-305141713cbd>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['P_BATD'][i] = -data['P_bat'][i]
```

[7]:

	index	P_ug	C_d	C_dummy	Pload	Ppv	P_bat	SOC	P_BATC	P_BATD
0	0	86.010526	0.0	1.0	-45.71	0.0	36.000000	0.707031	-36.0	0.000000
1	1	46.452632	0.0	1.0	-44.13	0.0	0.000000	0.765625	0.0	0.000000
2	2	52.515789	0.0	1.0	-49.89	0.0	0.000000	0.765625	0.0	0.000000
3	3	54.242105	0.0	1.0	-51.53	0.0	0.000000	0.765625	0.0	0.000000
4	4	47.410526	0.0	1.0	-45.04	0.0	0.000000	0.765625	0.0	0.000000
…	…	…	…	…	…	…	…	…	…	…
8731	8731	48.030667	1.0	0.0	-57.70	0.0	-13.412074	0.322463	0.0	13.412074
8732	8732	49.704351	1.0	0.0	-59.29	0.0	-13.412074	0.300634	0.0	13.412074
8733	8733	42.483299	1.0	0.0	-52.43	0.0	-13.412074	0.278804	0.0	13.412074
8734	8734	34.041193	1.0	0.0	-44.41	0.0	-13.412074	0.256975	0.0	13.412074
8735	8735	39.146456	1.0	0.0	-49.26	0.0	-13.412074	0.235145	0.0	0.000000

8736 rows x 10 columns

[8]:
```
def D_ug():
    # the unit electricity fee (may vary during the day)
    a=np.ones((7))*0.2671
    b=np.ones((3))*0.7405
```

```
    c=np.ones((1))*1.2662
    d=np.ones((2))*1.3955
    e=np.ones((2))*1.2662
    f=np.ones((1))*0.7405
    g=np.ones((1))*1.3955
    h=np.ones((1))*0.7405
    i=np.ones((3))*1.2662
    j=np.ones((2))*0.7405
    k=np.ones((1))*0.2671
    D_ug=np.hstack((a,b,c,d,e,f,g,h,i,j,k))
    return D_ug

D_ug = D_ug() #electricity fee
```

[9]:
```
ug = D_ug
ug = np.mat(ug)
ug = ug.T
ug[7]
```

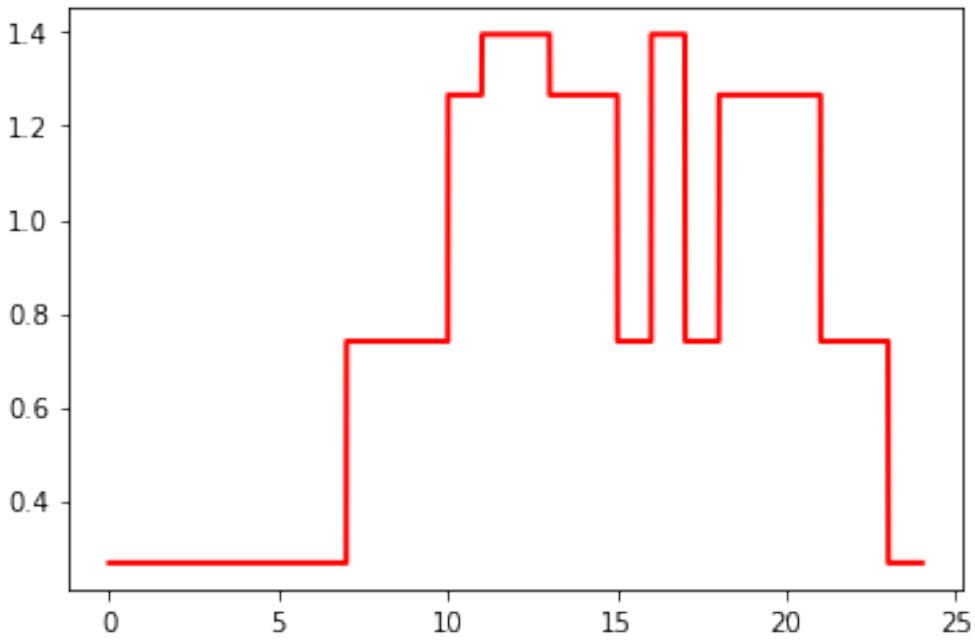
[9]:
```
matrix([[0.7405]])
```

[10]:
```
# Select a representative day in summer (someday in July,    whose index is␣
 ↪in 200*24 201*24-1 )
# Select a representative day in winter (someday in January, whose index is␣
 ↪in 20*24  21*24-1  )
# Select a representative day in spring (someday in April,   whose index is␣
 ↪in 110*24 111*24-1 )
# Select a representative day in fall   (someday in October, whose index is␣
 ↪in 280*24 281*24-1 )

s1 = 200*24
s2 = s1+24

w1 = 5*24
w2 = w1+24

sp1 = 110*24
sp2 = sp1+24

f1 = 285*24
f2 = f1+24

summer = np.arange(s1,s2)
winter = np.arange(w1,w2)
spring = np.arange(sp1,sp2)
fall = np.arange(f1,f2)
```

```
day_s = data[data['index'].isin(summer)]
day_w = data[data['index'].isin(winter)]
day_sp = data[data['index'].isin(spring)]
day_f = data[data['index'].isin(fall)]
```

[11]:
```
interval = np.linspace(0,24,10000)
interval0 = [1 if ((i>=0 and i<=7) or (i>23 and i<=24)) else 0 for i in␣
↪interval]
interval1 = [1 if ((i>7 and i<=10) or (i>15 and i<=16) or (i>17 and i<=18) or␣
↪(i>21 and i<=23)) else 0 for i in interval]
interval2 = [1 if ((i>10 and i<=11) or (i>13 and i<=15) or (i>18 and i<=21))␣
↪else 0 for i in interval]
interval3 = [1 if ((i>11 and i<=13) or (i>16 and i<=17)) else 0 for i in␣
↪interval]
y = ug[0]* interval0 + ug[7] * interval1 + ug[10]*interval2 + ug[11]*interval3
y = y.T
```

[12]:
```
plt.plot(interval,y,'r-', linewidth = 2, label = 'Utility Grid Electricity␣
↪Price' )
plt.show()
```

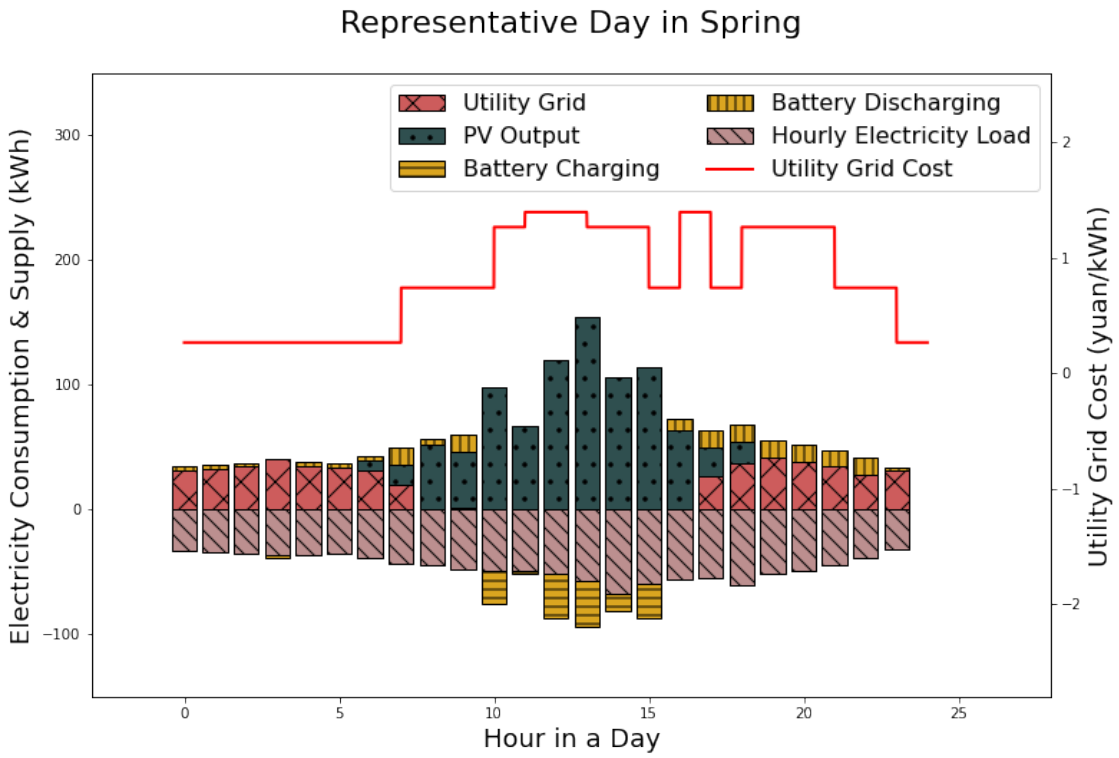


[13]:
```
x = np.arange(0,24)
```

[14]:
```
fig = plt.figure(figsize = (12,8))
ax1 = fig.add_subplot(111)
```

```
ax1.bar(x,height = day_sp['P_ug'],   color = 'indianred', label = 'Utility␣
 ↪Grid', alpha = 1, hatch = 'x', edgecolor = 'k')
ax1.bar(x,height = day_sp['Ppv'],    color = 'darkslategray', label = 'PV␣
 ↪Output', alpha = 1, hatch = '.', edgecolor = 'k', bottom=day_sp['P_ug'])
ax1.bar(x,height = day_sp['P_BATC'], color = 'goldenrod', label = 'Battery␣
 ↪Charging', alpha = 1, hatch = '--', edgecolor = 'k', bottom= day_sp['Pload'])
ax1.bar(x,height = day_sp['P_BATD'], color = 'goldenrod', label = 'Battery␣
 ↪Discharging', alpha = 1, hatch = '||', edgecolor = 'k', bottom=␣
 ↪day_sp['P_ug']+day_sp['Ppv'])
ax1.bar(x,height = day_sp['Pload'],  color = 'rosybrown', label = 'Hourly␣
 ↪Electricity Load', alpha = 1, hatch = '\\\\', edgecolor = 'k')

ax2 = ax1.twinx()
ax2.plot(interval,y,'r-', linewidth = 2, label = 'Utility Grid Cost' )

fig.legend(ncol = 2,fontsize = 16, bbox_to_anchor=(1,1), bbox_transform=ax1.
 ↪transAxes)
ax1.set_xlabel('Hour in a Day', fontsize = 18)
ax1.set_ylabel('Electricity Consumption & Supply (kWh)', fontsize = 18)
ax2.set_ylabel('Utility Grid Cost (yuan/kWh)', fontsize = 18)
ax1.set_ylim(-150,350)
ax2.set_ylim(-2.8,2.6)
ax1.set_xlim(-3,28)
ax1.set_title('Representative Day in Spring',y = 1.05, fontsize = 22)

fig.show()
```

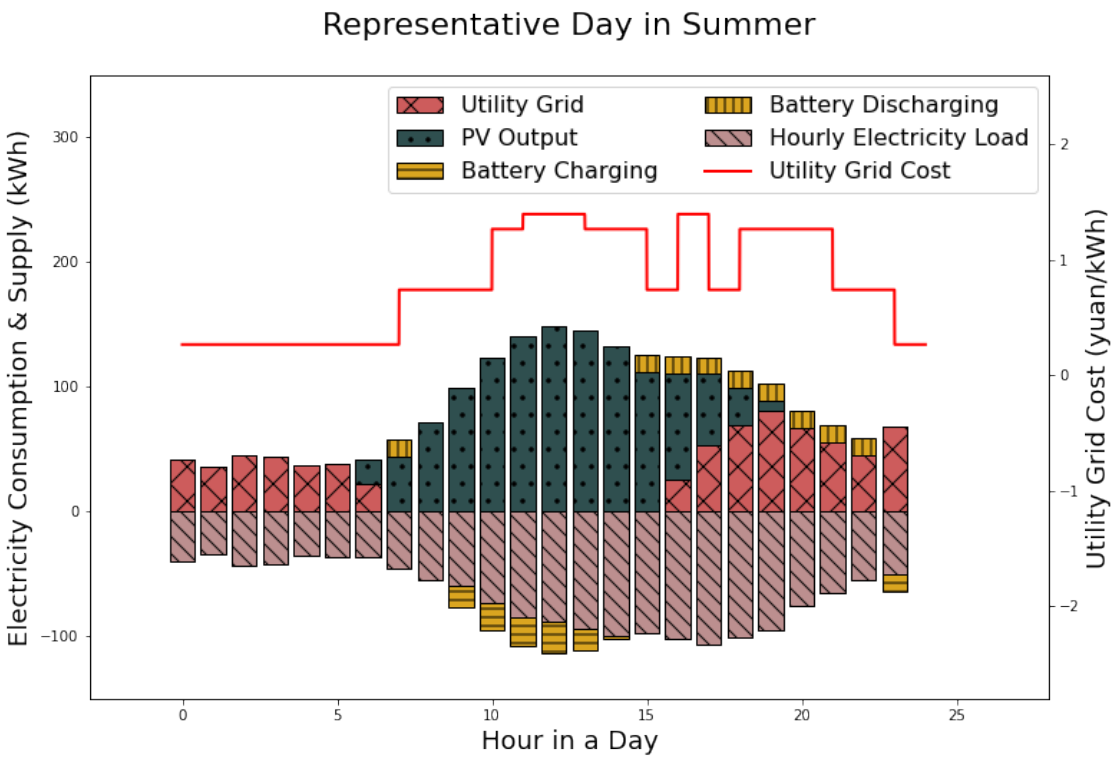
```
<ipython-input-14-e3a44ba8a9b5>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



[15]:
```
fig = plt.figure(figsize = (12,8))
ax1 = fig.add_subplot(111)

ax1.bar(x,height = day_s['P_ug'],   color = 'indianred', label = 'Utility␣
 ↪Grid', alpha = 1, hatch = 'x', edgecolor = 'k')
ax1.bar(x,height = day_s['Ppv'],    color = 'darkslategray', label = 'PV␣
 ↪Output', alpha = 1, hatch = '.', edgecolor = 'k', bottom=day_s['P_ug'])
ax1.bar(x,height = day_s['P_BATC'], color = 'goldenrod', label = 'Battery␣
 ↪Charging', alpha = 1, hatch = '--', edgecolor = 'k', bottom= day_s['Pload'])
ax1.bar(x,height = day_s['P_BATD'], color = 'goldenrod', label = 'Battery␣
 ↪Discharging', alpha = 1, hatch = '||', edgecolor = 'k', bottom=␣
 ↪day_s['P_ug']+day_s['Ppv'])
ax1.bar(x,height = day_s['Pload'],  color = 'rosybrown', label = 'Hourly␣
 ↪Electricity Load', alpha = 1, hatch = '\\\\', edgecolor = 'k')

ax2 = ax1.twinx()
ax2.plot(interval,y,'r-', linewidth = 2, label = 'Utility Grid Cost' )

fig.legend(ncol = 2,fontsize = 16, bbox_to_anchor=(1,1), bbox_transform=ax1.
 ↪transAxes)
ax1.set_xlabel('Hour in a Day',fontsize = 18)
ax1.set_ylabel('Electricity Consumption & Supply (kWh)',fontsize = 18)
```

```
ax2.set_ylabel('Utility Grid Cost (yuan/kWh)', fontsize = 18)
ax1.set_ylim(-150,350)
ax2.set_ylim(-2.8,2.6)
ax1.set_xlim(-3,28)
ax1.set_title('Representative Day in Summer',y = 1.05, fontsize = 22)

fig.show()
```

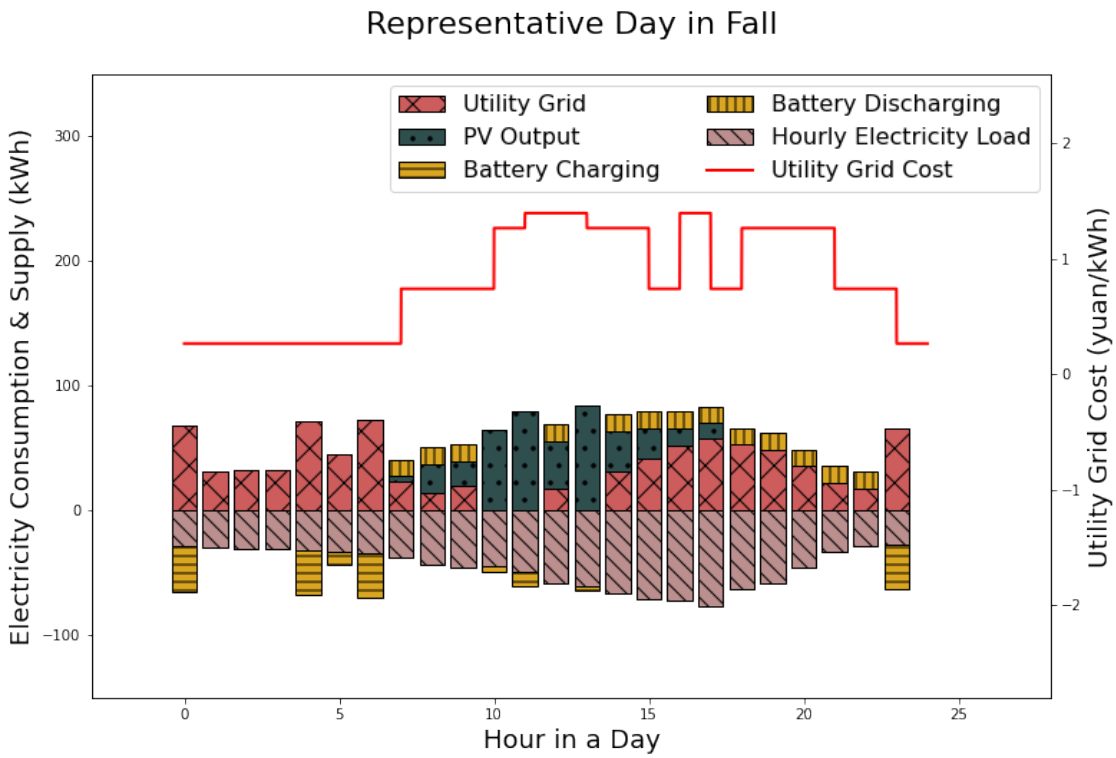
```
<ipython-input-15-4b81eb4cda2b>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



[16]:
```
fig = plt.figure(figsize = (12,8))
ax1 = fig.add_subplot(111)

ax1.bar(x,height = day_f['P_ug'],   color = 'indianred', label = 'Utility␣
 ↪Grid', alpha = 1, hatch = 'x', edgecolor = 'k')
ax1.bar(x,height = day_f['Ppv'],    color = 'darkslategray', label = 'PV␣
 ↪Output', alpha = 1, hatch = '.', edgecolor = 'k', bottom=day_f['P_ug'])
ax1.bar(x,height = day_f['P_BATC'], color = 'goldenrod', label = 'Battery␣
 ↪Charging', alpha = 1, hatch = '--', edgecolor = 'k', bottom= day_f['Pload'])
```

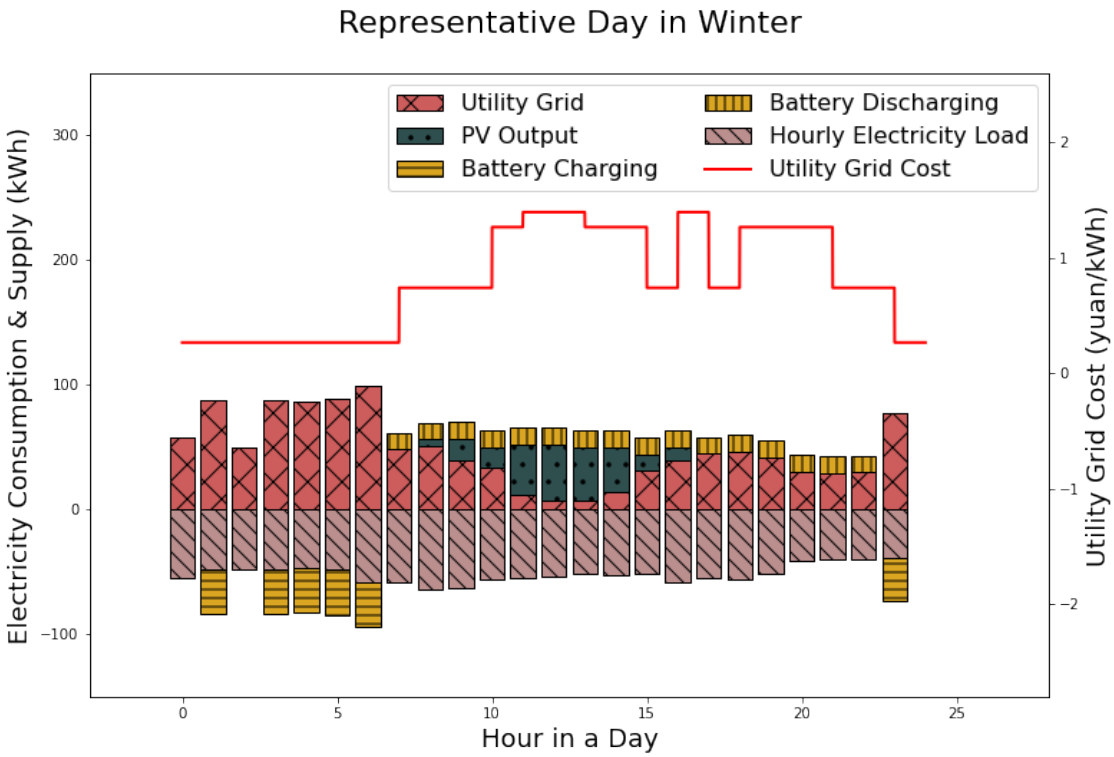
```
ax1.bar(x,height = day_f['P_BATD'], color = 'goldenrod', label = 'Battery␣
 ↪Discharging', alpha = 1, hatch = '||', edgecolor = 'k', bottom=␣
 ↪day_f['P_ug']+day_f['Ppv'])
ax1.bar(x,height = day_f['Pload'],  color = 'rosybrown', label = 'Hourly␣
 ↪Electricity Load', alpha = 1, hatch = '\\\\', edgecolor = 'k')

ax2 = ax1.twinx()
ax2.plot(interval,y,'r-', linewidth = 2, label = 'Utility Grid Cost' )

fig.legend(ncol = 2,fontsize = 16, bbox_to_anchor=(1,1), bbox_transform=ax1.
 ↪transAxes)
ax1.set_xlabel('Hour in a Day',fontsize = 18)
ax1.set_ylabel('Electricity Consumption & Supply (kWh)',fontsize = 18)
ax2.set_ylabel('Utility Grid Cost (yuan/kWh)', fontsize = 18)
ax1.set_ylim(-150,350)
ax2.set_ylim(-2.8,2.6)
ax1.set_xlim(-3,28)
ax1.set_title('Representative Day in Fall',fontsize = 22,y = 1.05)

fig.show()
```

```
<ipython-input-16-76098936e8fa>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



[17]:
```
fig = plt.figure(figsize = (12,8))
ax1 = fig.add_subplot(111)

ax1.bar(x,height = day_w['P_ug'],   color = 'indianred', label = 'Utility␣
 ↪Grid', alpha = 1, hatch = 'x', edgecolor = 'k')
ax1.bar(x,height = day_w['Ppv'],    color = 'darkslategray', label = 'PV␣
 ↪Output', alpha = 1, hatch = '.', edgecolor = 'k', bottom=day_w['P_ug'])
ax1.bar(x,height = day_w['P_BATC'], color = 'goldenrod', label = 'Battery␣
 ↪Charging', alpha = 1, hatch = '--', edgecolor = 'k', bottom= day_w['Pload'])
ax1.bar(x,height = day_w['P_BATD'], color = 'goldenrod', label = 'Battery␣
 ↪Discharging', alpha = 1, hatch = '||', edgecolor = 'k', bottom=␣
 ↪day_w['P_ug']+day_w['Ppv'])
ax1.bar(x,height = day_w['Pload'],  color = 'rosybrown', label = 'Hourly␣
 ↪Electricity Load', alpha = 1, hatch = '\\\\', edgecolor = 'k')

ax2 = ax1.twinx()
ax2.plot(interval,y,'r-', linewidth = 2, label = 'Utility Grid Cost' )

fig.legend(ncol = 2,fontsize = 16, bbox_to_anchor=(1,1), bbox_transform=ax1.
 ↪transAxes)
ax1.set_xlabel('Hour in a Day',fontsize = 18)
ax1.set_ylabel('Electricity Consumption & Supply (kWh)',fontsize = 18)
ax2.set_ylabel('Utility Grid Cost (yuan/kWh)', fontsize = 18)
ax1.set_ylim(-150,350)
ax2.set_ylim(-2.8,2.6)
ax1.set_xlim(-3,28)
ax1.set_title('Representative Day in Winter',fontsize = 22,y = 1.05)

fig.show()
```

```
<ipython-input-17-c8f37c9a8fce>:22: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```


Representative Day in Winter
Utility Grid
PV Output
Battery Charging
Battery Discharging
Hourly Electricity Load
Utility Grid Cost
Electricity Consumption & Supply (kWh)
Utility Grid Cost (yuan/kWh)
Hour in a Day
300
200
100
0
−100
2
1
0
−1
−2
0
5
10
15
20
25

Modeling, optimization, and optimal control of renewable energy battery storage on the California electric grid

Elliott Suen, Hardi Sura, Sydney Holgado, Tianchen Xu, Xinwei Zhuang, Yalu Guo

CEE 295: Data Science for Energy, Spring 2021

Abstract

The levelized cost of electricity (LCOE) of renewable energy has reached price parity with conventional fossil fuel energy sources. However, most renewable sources, such as solar and wind, are intermittent, and are not dispatchable; there is often a time mismatch between peak supply and peak demand of electricity, leading to massive solar curtailments in the afternoon and insufficient supply in the evenings. One way to address this problem is by implementing battery storage on the grid. This project aims to use existing weather (irradiance and wind speed) data and historical electricity demand data to optimize the amount of storage that would need to be available in order to store the curtailed renewable energy to meet demand in hours without renewables generation that would otherwise be met by burning fossil fuels. We specifically aim to optimize system costs, thus providing a way to cost-effectively decarbonize the grid.

Introduction

Motivation and Background

The primary energy consumption source in the US by 2019 is petroleum at 37%, followed by natural gas at 32%. If we specifically focus on electricity generation, renewable energy accounts for just 11% of total generation (US EIA). The electricity sector is responsible for 26.9% of total US greenhouse gas (GHG) emissions annually, while transportation, industry, and residential sectors are responsible for 29%, 22%, and 12.3% respectively. The Biden Administration has set ambitious goals of reducing carbon emissions by 50% by 2030, and has proposed a $2 trillion infrastructure bill, part of which will go towards grid upgrade and modernization, in order to achieve that goal. Many individual states have similar, if not even stricter, emissions cutting initiatives, such as California's SB100, which requires the state to go 100% renewable or zero-carbon by 2045. It is clear, therefore, that widespread electrification of transportation (e.g. battery electric vehicles), industry (e.g. electric arc furnaces), and residential sectors (e.g. induction stovetops), combined with decarbonizing the electricity sector, will be necessary to achieve our emission reduction goals, but what is less clear is how to do so rapidly and cost effectively.

The electric grid is a system for electric power transmission, distribution, and demand-driven control (Hossian, 2016). The grid was traditionally understood from a demand perspective: electricity usage served as inputs to the grid, and supply was closely matched to and anticipated demand in order to prevent mismatch. Furthermore, the grid traditionally relied on consistent and dispatchable baseload power provided by fossil fuel plants. This has remained largely unchanged since the first transmission networks were installed in 1886, and in fact, the vast majority of the US grid is over 25 years old, with some components over a century old, and are still able to fulfill their function, albeit with increasing unreliability. Renewables, specifically the highly variable and intermittent solar and wind resources, bring

about a paradigm shift to our understanding of the grid, where supply now becomes an additional input rather than a variable that can be controlled. If we are to deploy 100% renewable energy on the grid as of today, there will be a gap where renewables supply and electricity demand are not synchronized. The duck curve will also lead to Overgeneration (figure 1). This will require the development of novel monitoring and control techniques. Furthermore, this paradigm shift requires not just a change in our grid control algorithms, but also a fundamental change in our physical infrastructure - we will need energy storage devices, be it chemical batteries, pumped hydro storage, or hydrogen fuel cell technologies, in order to balance electricity supply and demand; we will need greater high voltage direct current (HVDC) transmission lines to better integrate solar, wind, and batteries into the grid; and our traditional substations, inverters, and transformers will need to be upgraded to better handle distributed flow of electricity.

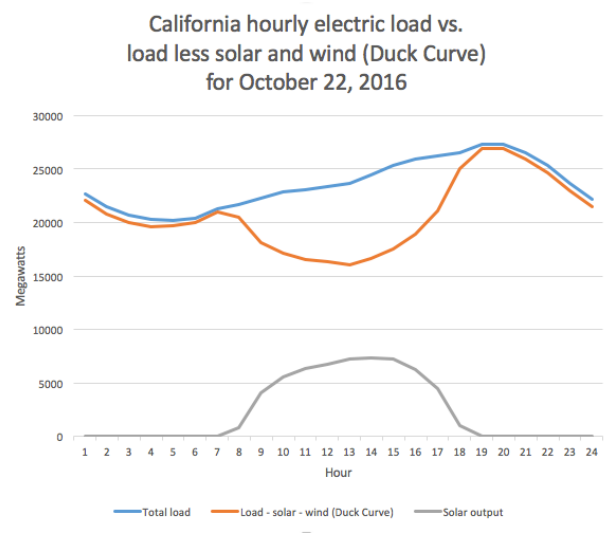


Figure 1. California hourly electric load vs load less solar and wind, CAISO, 2016

When considering the integration of traditional energy and renewable energy such as solar and wind energy, there are several challenges to integrating and increasing renewables within the grid system. While renewables are preferable to fossil fuel generators from an emissions standpoint, power output from renewable sources depends on variable natural resources, which makes these plants more difficult to control and presents challenges for grid operators. The fluctuation in output can lead to instability in the grid due to the lack of storage in our current grid system, which consequently causes demand-supply mismatch. Including battery storage systems and energy management systems can close this gap. Storing energy provides benefits such as reducing and smoothing peak energy demands (Sufyan *et al.* 2019), and by modelling, optimizing and controlling the integration of renewable energy and smart grid systems, we can minimize the effect of ramping events brought about by renewable energy, and consequently, enhance the stability of the grid system. However, deciding the amount of storage on the grid is a complex problem, which involves not only the properties of the battery, but also the cost and ecological impact. Existing batteries are expensive, and although process optimization and economies of scale have brought down the price of batteries, they are still relatively costly as an infrastructure investment; furthermore, lithium and cobalt mining are associated with a litany of ecological impacts and human rights violations, so we must seek to minimize the amount of batteries deployed while still ensuring grid load-supply balancing capabilities.

Our workflow is, collecting data in previous years in the wind speed, solar intensity for potential electricity supply and load demand, then we did data analysis to identify the scale of the supply & demand gap. We then optimize this gap in the most extreme months under power balance constraints, battery constraints and ramping constraints, and finally we provide optimized battery storage for the renewables, and the consequent change in the supply-demand gap. The project pipeline is shown below.

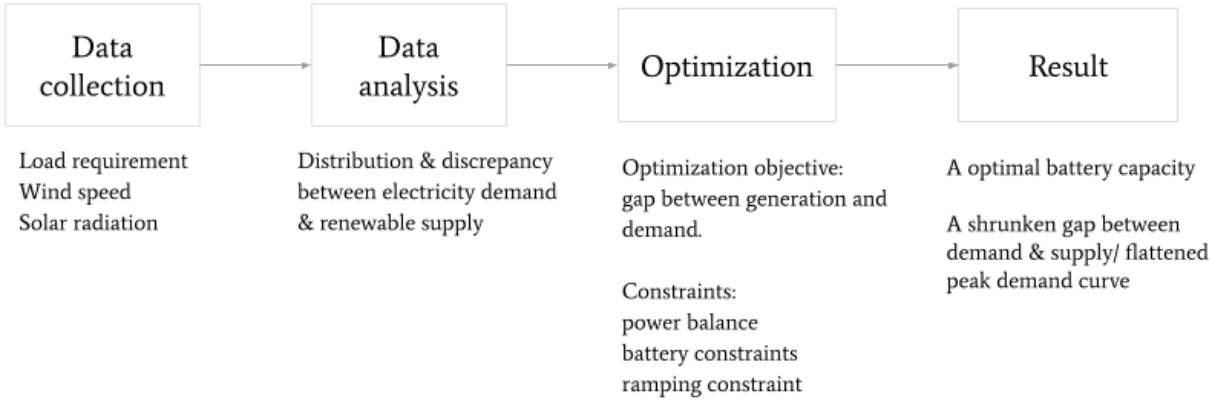


Figure 2. Project pipeline.

Relevant Literature

Current studies on modeling and optimization of renewable energy mainly propose model predictive control, dynamic programming or mixed-integer linear programming for energy microgrids under operation constraints. Sufyan *et al.* (2019) presented a review on different types of battery energy storage technologies, and the results show that using electromechanical batteries can provide benefits such as peak shaving, load levelling, power fluctuations regulation and reliability improvement. To decide the size of the storage, studies include probabilistic and stochastic approaches. Greenwood *et al.* (2017) investigated the optimal sizing of the energy storage system by using real-time thermal rating and reliability metrics. Bludszuweit and Domınguez-Navarro (2011) and Zhao *et al.* (2018) take the uncertainty in the weather forecast data into consideration. They integrated the Monte Carlo Method and support vector machine to process weather data and perform load prediction. Stochastic methods such as the particle swarm optimization, the genetic algorithm are also studied (Sakawa & Matsui, 2003, Fossati *et al.*, 2015, Sukumar *et al.*, 2017). In the very recent years, artificial intelligence and machine learning has also been used as a tool to find the optimal energy response (Antonopoulos *et al.*, 2020). However, these artificial intelligence based approaches for renewable energy have disadvantages such as sensitivity to the input dataset and fuzzy rules, lack of theoretical foundations and high in computational complexity (Zor *et al.*, 2017).

Monte Carlo simulation is a widely used method to evaluate the availability of the power grid, and has demonstrated its effectiveness on risk analysis on the grid systems. Pereira *et al.* (2014) used the Monte Carlo method to estimate the behaviour of the economic parameters, and presented the feasibility in a case study of a grid-connected photovoltaic system. Wadi and Baysal (2018) provided a modified Monte Carlo simulation method where the total loss of continuity of a smart grid system is integrated with the original

Monte Carlo simulation, which enables the method to access not only the open-ring but also the closed-ring networks.

Focus of this study

In this project, we focus on existing electricity demand data to model the distributions of solar and wind energy needed to completely offset the energy supplied by fossil fuel sources on the grid. We plan to model battery storage to meet the electricity demand for at least 90% of the day in addition to solar and wind based energy supplied from the grid.

Technical Description

For this project, the optimization approach is studied with several assumptions. Firstly, the battery type is assumed to be lithium-ion for its promising storage characteristics and economic feasibility (Diouf and Pode, 2015). We also assume a nameplate capacity of 15,000 MW for the solar plant and 10,000 MW for the wind plant based on California's current estimates. The discharge duration corresponding to battery size is assumed to be 2-5 hours, and the life cycle is larger than 4000, with 15-20 years life time (Eyer and Corey, 2010).

Probability distribution of weather and load data

In order to generate our model, we collected hourly electricity demand and weather data. We extracted hourly irradiance and wind speed data using NREL's Highly Scalable Data Service API, and downloaded hourly electricity demand data from the CAISO Historical Load archives. Because we conduct a grid scale optimization in this project, we selected one location each in which to model solar and wind plants. The dataset used in this project contains hourly irradiance in watts per square meter measured in Los Angeles County, wind speed in meters per second measured in Kern County and adjusted for wind turbine height, and electricity demand in megawatts for the whole state of California.

We firstly looked at the probability distribution of weather and load data. It can be seen that the solar energy is roughly in line with the electricity demand throughout the month, whereas the wind energy peaks at winter but drops in the summer when the demand is high. On a daily basis, the energy demand peaks at around 21:00 and reaches lowest at 5:00. Both solar and wind follow a distribution pattern where it is higher during the daytime, peaking at noon and drops in the evening. Solar energy can cover the energy demand in most of the daytime, but it needs supplement during the night time (from 6pm to 8am). Outside winter time, the solar power can approximately cover the electricity demand, but the discrepancy is prominent in months between October and March. Both imply the necessity of the storage of the solar and wind energy during the midday to meet the energy demand in the evening.

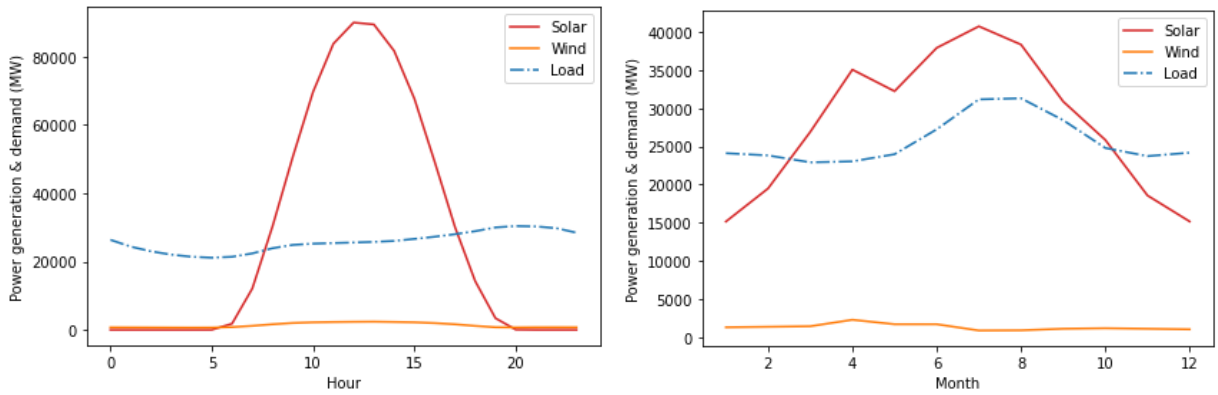


Figure 3.Average hourly data (left) and monthly data (right).

Statistical models using the Monte Carlo method

The intermittent nature of wind and solar energy poses operational and planning challenges to the electricity market. These challenges can be addressed with accurate power forecasting methods. To determine the hourly uncertainty in the power generation from renewables, we would build our statistical model with the Monte Carlo Simulation (MCS). MCS converts uncertainty in the power generation into an approximate probability distribution of the outputs. It entails determining system response to a finite number of initial conditions and random input functions generated according to their specified statistics.

Our current model did not use this method. But we will propose how to use it to improve our future model in the Discussion section.

Battery storage optimization

We aim to estimate the required battery storage capacity to meet the peak demand with renewable energy by formulating an optimization problem, as follows:

Objective function: minimize the gap between load and renewable energy:

$$Minimize \sum_{t=0}^{N} (P_{load} - P_{solar} - P_{wind} - P_{battery} * EF_{battery})^2$$

The goal of our optimization is to determine the level of battery storage needed to ensure that demand is met 90% using just renewables and batteries. Thus, our objective function is to minimize the squared difference between load and the power generation, where P_{load} is the load, P_{solar} and P_{wind} are power generated by solar and wind energy, $P_{battery}$ is the power that can be charged/discharged from the battery system, and $EF_{battery}$ is the conversion factor of input/output power of the battery system.

Subject to constraints: renewable and battery storage must meet load at least 90% of each time step, power generation by renewables must stay at or below the nameplate capacity of power plants, and batteries only able to charge and discharge at certain rates:

$$P_{solar} \leq P_{solar\ capacity}$$

$$P_{wind} \leq P_{wind\ capacity}$$

$$P_{battery\ min} \leq P_{battery} \leq P_{battery\ max}$$

$$P_{solar} + P_{wind} + P_{battery} * EF_{battery} \geq 0.9 * P_{load}$$

Take weather and load data in July 2019 as an example:

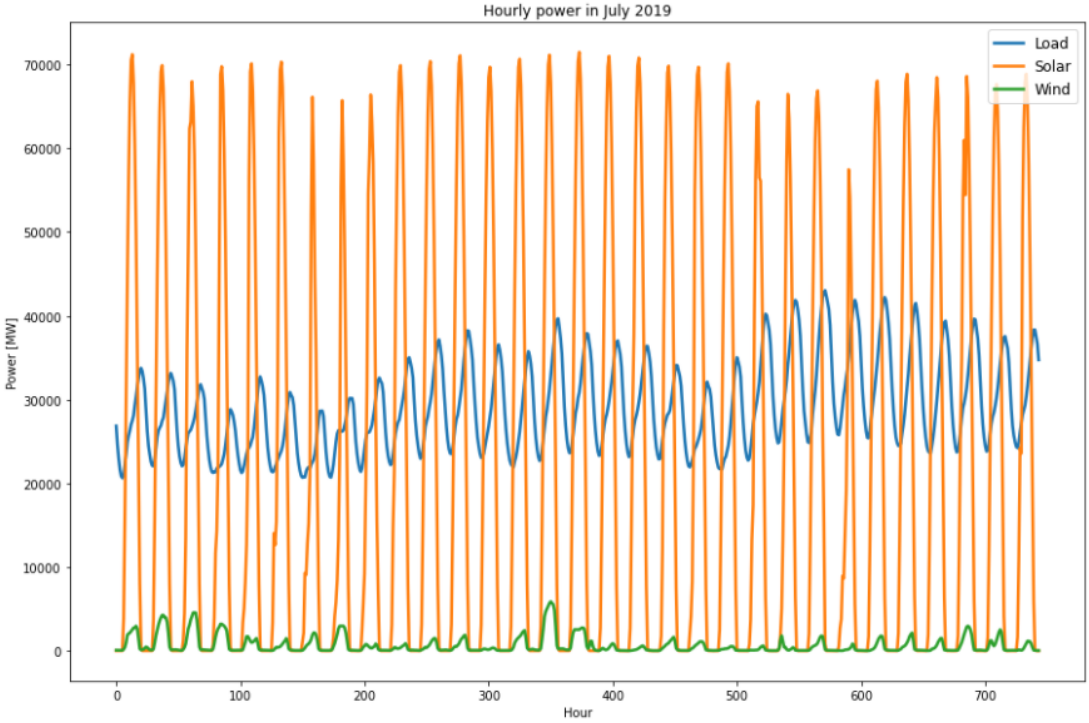


Figure 4. Distribution of hourly solar, wind, and load data in July

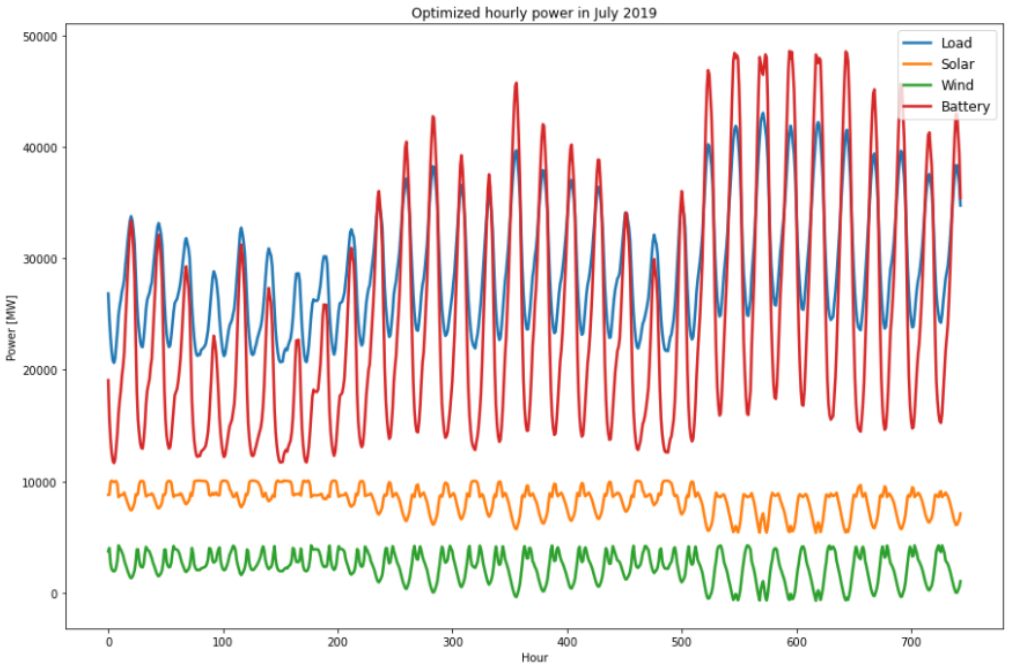


Figure 5. Distribution of hourly solar, wind, and load data in July after optimization

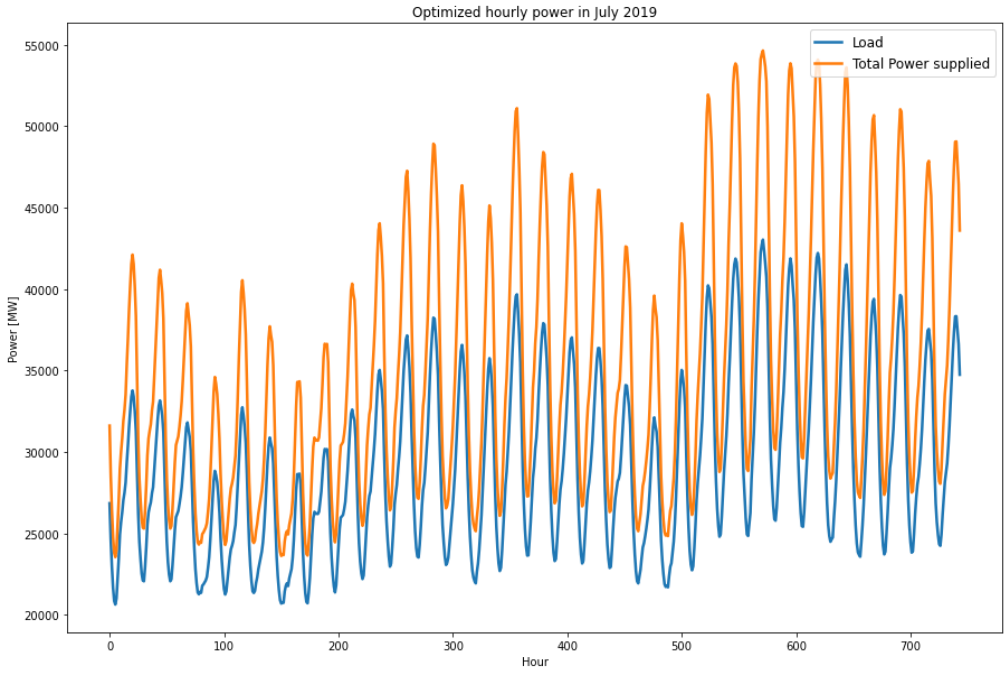


Figure 6. Distribution of hourly solar + wind + battery and load data in July after optimization

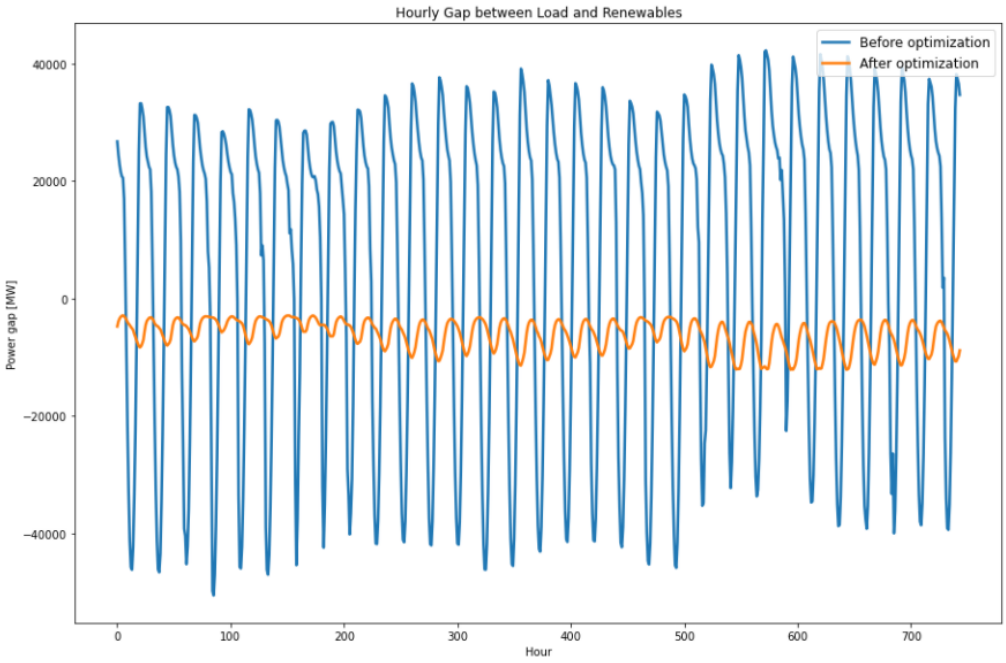


Figure 7. Hourly gap between load and renewables before and after optimization

Along with the battery storage optimization, the battery charge state is determined using the load gap for the month of July, with a maximum charge capacity of 50,000 MW. Maximum charge and discharge rates of 25% are used to determine the charge in battery state every hour as the battery is charged or discharged as per the load gap (Eyer and Corey, 2010). It is further assumed that the battery would only be able to charge up to 80% and discharge upto 20% of its maximum capacity. This analysis renders a distribution of the battery state, load gap and renewable power (solar + wind) available for the month of July in Figure 8. As shown in the figure, the daily peaks in renewable energy supply are enough to charge the battery to compensate for at least 90% of the load gap. There are 5 days in July (around the 600 hour mark) where the maximum attainable battery capacity is less than the required amount to meet the load gap. This occurs because of the uncertainties in solar and wind power supply due to various reasons including weather, and we describe in the Discussion section how these uncertainties can be accounted for using Monte Carlo simulations.

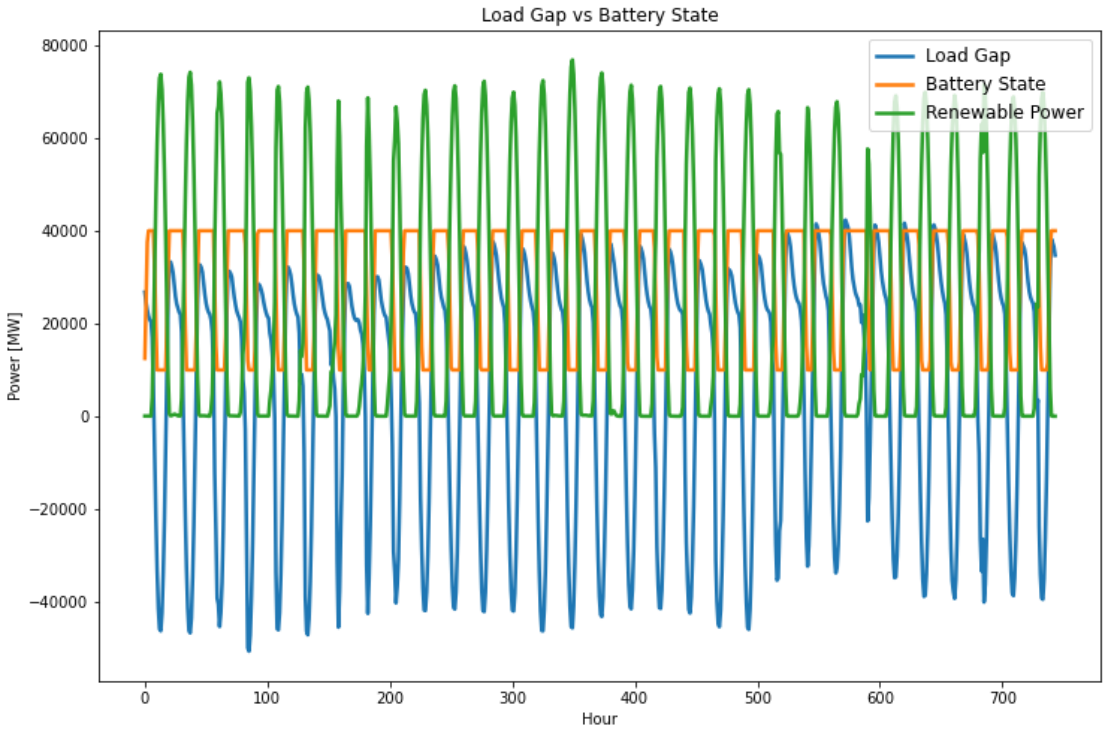


Figure 8. Hourly distribution of battery state, renewable power supply and load gap in July

System costs

We also considered the costs associated with deploying more renewables and storage on the grid, and we evaluated the system costs of the hypothetical grid proposed in the storage optimization scenarios. We assume that the infrastructure is financed using loans with a 5% interest rate and roughly 20 year lifetime, and financing costs are calculated using a standard PMT function. This monthly loan payment is then added to the operations and maintenance costs per megawatt for solar and wind (Lawrence Berkeley National Laboratory 2020, National Renewable Energy Laboratory 2016). The total monthly operating costs are given by the equations below:

$$Solar\ cost\ =\ PMT(0.05,\ 20,\ Cost_{install}\ *\ Solar\ Capacity)\ +\ Cost_{O\&M}\ *\ E_{solar}$$

$$Wind\ cost\ =\ PMT(0.05,\ 20,\ Cost_{install}\ *\ Wind\ Capacity\ +\ Cost_{O\&M}\ *\ E_{wind}$$

$$Battery\ cost\ =\ PMT(0.05,\ 20,\ Cost_{install}\ *\ max(E_{battery}))$$

$$Total\ system\ cost\ =\ Solar\ cost\ +\ wind\ cost\ +\ battery\ cost$$

For July 2019, the monthly system cost is $322 million based on the optimized power of renewable energy and battery storage.

Discussion

As described in the previous sections, our model uses excess solar and wind power supplied during the day to charge the batteries, which would then be used as a power source during the night. Such a process

aims to minimize load gap while maximizing zero carbon renewable energy percentage. This process was defined as a quadratic optimization problem described in the 'Battery Storage Optimization' section. From Figure 5, we can see that the currently available solar and wind power charges the battery enough to meet 90% of the load in the month of July. However, there still exists a non-zero load gap (load - battery) in the first few days of July. As shown in Figure 6, the load is fully met using a combination of power supply from solar, wind and battery sources. Furthermore, Figure 7 shows that the load gap (load - solar + wind + battery power) is significantly decreased after optimization. Hence, from the results of the optimization problem using data from the month of July, we can conclude that batteries are a feasible method for significant decarbonization.

However, further investigation is needed to determine the feasibility of batteries throughout the year, using data from multiple years and across different geographical locations; less daylight hours in winter months may necessitate longer-term seasonal storage or additional storage options, and this seasonal shift is compounded in geographies situated at higher latitudes. Moreover, the analysis of battery states in Figure 8 does not take the battery lifetime into account, which could lead to deterioration of maximum battery capacity and charging rate over years of usage. Hence, a future work for this project would warrant deeper analysis into battery states along with the optimal battery power requirements. Finally, we would also like to incorporate uncertainties in estimation of solar and wind power in the optimization problem (described below).

Uncertainty Estimation in Power Forecasts Using Monte Carlo Simulations

As mentioned before, wind power and solar power have very large uncertainties in their power generation. Therefore, it is very difficult to accurately predict its contribution to electricity in a certain period. In the previous analysis, we used a three-year average to consider the impact of the error on our model. But this approach is obviously relatively primitive and rough, without a very comprehensive consideration of the error of power generation. In fact, we can treat them as random processes. A commonly used method is to perform Monte Carlo Simulation (MCS) to quantify the risk in the power prediction, predict the wind power and solar power in a certain period more accurately. The quantification of uncertainty will permit assessing the risk of relying on the power prediction forecasts and then improve our model.

To obtain wind power generation predictions with MCS, typically, hourly wind-speed forecast error distributions can be assumed to be approximately normal, then the random process can be generated as a Gauss-Markov Process. And suppose that is a First-Order Markov (FOM) process, which can describe the dynamics of the prediction error over time (24 hour). The state space formulation of the FOM model can be given as $de/dt = u(t) - \beta e$. e is the wind prediction error and β is the sum of a white noise process and a deterministic term. The input $u(t)$ is determined by requiring that the statistics (the mean and the standard deviation) of e match empirical values of the hourly forecast-error data. This process is simplified by separating e into its random component and deterministic part: $e = e_r + \widehat{\mu_r}(t)$for $(r-1) < t < r,\ r = 1, 2, \ldots, 24$. The random component of e then satisfies $de/dt = w_n(t) - \beta e_r$.

Given the system model, initial condition statistics and random input statistic, the MCS technique

generates an ensemble or large number n of the system responses to wind speed realizations. The ensemble of system responses is generated by performing the following procedure n times: choose a random initial condition ($e_r(0)$). Then select a random input vector $w_n(mh)$; the value of white noise with spectral density $Q_{r(t)}$ can be simulated by a random number generator to obtain a sequence of random values. The input vector is then passed to the Euler integration technique to propagate the solution from $t = 0$ to t = h, and so on until the final time $t_f = 23:59:59$ hours is reached. Performing n independent trials yields an ensemble of n prediction trajectories or realizations of e_r.

Adding the above realizations of e_r to the deterministic components of the prediction error gives n realizations of the prediction error. Adding the above error realizations to the forecasted wind speeds v_f will give the wind speed realizations and power predictions will then be obtained by passing wind speed realizations to the wind power generator. Each realized power prediction is determined by inputting each realized wind speed $v_r^i(t)$ to the wind power generation model. Generating n realizations of forecast error gives n statistically meaningful realizations of wind speed, which yields an ensemble of n realized power predictions.

Generally, to obtain the wind power predictions, we first determine the wind speed realization and pass that to the wind power generator. Since there are uncertainties in the realized wind speeds, the uncertainties get transferred to the power prediction as well. And the similar method and process can be applied to the solar power prediction as well.

The MCS essentially gives the variability in the wind power and solar power production in terms of statistics which can be used for assessing the risk involved in power production forecasts.

Summary

In summary, this project used principles of mathematical modeling and optimization in order to estimate what amount of battery storage would be needed on the California electric grid. We took real historical weather and electricity demand data to conduct an optimization that determines the level of storage needed to ensure that demand is met at least 90% of the day using just renewables and batteries. We also conducted a cost evaluation. Based on our findings and assumptions made in designing the controller, we find that a combination of solar (15,000 MW capacity), wind (10,000 MW capacity) and battery storage (50,000 MW maximum capacity, with 80% attainable capacity at 25% charging/discharging rate) was able to meet at least 90% of the estimated load gap based on historical data for the month of July in California. Based on the installation costs for solar plants, wind plants and batteries and the optimal hourly power supplied by the solar and wind energy sources, we estimated a total monthly cost of $322 million for California. We also describe how Monte Carlo simulations can be used to further reduce the uncertainties in estimations of solar and wind power and further increase the accuracy of battery storage forecasts.

References

1. Hossain MS, Madlool NA, Rahim NA, Selvaraj J, Pandey AK, Khan AF. Role of smart grid in renewable energy: An overview. Renewable and Sustainable Energy Reviews. 2016 Jul 1;60:1168–84.
2. Wadi M, Baysal M, Shobole A, Tur MR. Reliability Evaluation in Smart Grids via Modified Monte Carlo Simulation Method. 2018. 841 p.
3. Celli G, Ghiani E, Pilo F, Soma GG. Reliability assessment in smart distribution networks. Electric Power Systems Research. 2013 Nov 1;104:164–75.
4. Feng C, Zhang J. Reinforcement Learning based Dynamic Model Selection for Short-Term Load Forecasting. arXiv:181101846 . 2018 Nov 5.
5. Gioutsos DM, Blok K, van Velzen L, Moorman S. Cost-optimal electricity systems with increasing renewable energy penetration for islands across the globe. Applied Energy. 2018 Sep 15;226:437–49.
6. Cui M, Zhang J, Feng C, Florita AR, Sun Y, Hodge B-M. Characterizing and analyzing ramping events in wind power, solar power, load, and netload. Renewable Energy. 2017 Oct 1;111:227–44.
7. Antonopoulos I, Robu V, Couraud B, Kirli D, Norbu S, Kiprakis A, et al. Artificial intelligence and machine learning approaches to energy demand-side response: A systematic review. Renewable and Sustainable Energy Reviews. 2020 Sep 1;130:109899.
8. Pereira EJ da S, Pinho JT, Galhardo MAB, Macêdo WN. Methodology of risk analysis by Monte Carlo Method applied to power generation with renewable energy. Renewable Energy. 2014 Sep 1;69:347–55.
9. Zame KK, Brehm CA, Nitica AT, Richard CL, Schweitzer III GD. Smart grid and energy storage: Policy recommendations. Renewable and Sustainable Energy Reviews. 2018 Feb 1;82:1646–54.
10. Sufyan M, Rahim NA, Aman MM, Tan CK, Raihan SRS. Sizing and applications of battery energy storage technologies in smart grid system: A review. Journal of Renewable and Sustainable Energy. 2019 Jan 1;11(1):014105.
11. Rubinstein RY, Kroese DP. Simulation and the Monte Carlo method. John Wiley & Sons; 2016 Oct 20.
12. Zhao J, Duan Y, Liu X. Uncertainty Analysis of Weather Forecast Data for Cooling Load Forecasting Based on the Monte Carlo Method. *Energies*. 2018 Jul 20;11:1900.
13. Zhuo W. Control Strategies for Microgrids with Renewable Energy Generation and Battery Energy Storage Systems. arXiv:191102126
14. D. M. Greenwood, N. S. Wade, P. C. Taylor, P. Papadopoulos, and N. Heyward, "A probabilistic method combining electrical energy storage and real-time thermal ratings to defer network reinforcement," *IEEE Trans. Sustainable Energy* 8(1), 374–384 (2017).
15. H. Bludszuweit and J. A. Domınguez-Navarro, "A probabilistic method for energy storage sizing based on wind power forecast uncertainty," IEEE Trans. Power Syst. 26(3), 1651–1658 (2011).
16. S. Sukumar, H. Mokhlis, S. Mekhilef, K. Naidu, and M. Karimi, "Mix-mode energy management strategy and battery sizing for economic operation of grid-tied microgrid," Energy 118, 1322–1333 (2017).
17. J. P. Fossati, A. Galarza, A. Martın-Villate, and L. Fontan, "A method for optimal sizing energy storage systems for microgrids," Renewable Energy 77, 539–549 (2015).

18. Zor K, Timur O, Teke A. A state-of-the-art review of artificial intelligence techniques for short-term electric load forecasting. In: *2017 6th International Youth Conference on Energy (IYCE)*. 2017. p. 1–7.
19. Sakawa M, Matsui T. Fuzzy multiobjective nonlinear operation planning in district heating and cooling plants. Fuzzy Sets Syst 2013;231:58–69. doi:10.1016/j.fss.2011.10.020.
20. Diouf B, Pode R. Potential of lithium-ion batteries in renewable energy. *Renewable Energy*. 2015 Apr 1;76:375-80.
21. Eyer J, Corey G. Energy storage for the electricity grid: Benefits and market potential assessment guide. Sandia National Laboratories. 2010 Feb;20(10):5.
22. Lawrence Berkeley National Laboratory. Utility Scale Solar. *Electricity Markets & Policy.* 2020.
23. National Renewable Energy Laboratory, Stehly T, Heimiller D, Scott G. 2016 Cost of Wind Energy Review. 2016.