Energy Cost and HVAC Optimization in Smart Buildings

Authors

Jonah Bomwell, Alise Bruevich, William Nathan, Esperanza Rozas

Abstract

With mounting concerns around climate change and the rising costs of power, optimizing and reducing energy usage is a growing challenge. In order to provide insight on the opportunity for energy usage and cost optimization, we used energy data from a building on UCSD's campus and auxiliary information about UCSD's pricing model for energy to build a model that predicts future energy spending and attempted reducing predicted energy spending by controlling two user-set variables.

Introduction

Optimizing and reducing energy usage is one of the greatest challenges of the 21st century in regards to climate change and rising costs of power. However, this challenge of optimization and frugality is not elementary. Many factors can change how much energy the same device uses - the time of day, climate trends (which inform heating needs), policy changes that limit energy in specific seasons, and more. Therefore, creating a model that takes into account all these variables and utilizes specific energy sensors available to UC San Diego's researchers to compute energy uses and costs would give users crucial information to plan energy usage long into the future. Our goal in the project is to use energy data from a building on UCSD's campus and auxiliary information about energy costs to build a model to predict future energy spending.

Literature Review

Our work in this project is based on our mentor, Rajesh Gupta, and his work on the Brick schema[1]. The BRICK schema was developed by industry and academic leaders aiming to standardize building metadata. To capture the development process and details of this venture, the researchers published an in-depth scientific paper alongside the Brick schema itself. Titled "Brick: Metadata schema for portable smart building applications"[2], this article goes into great detail about both the vast applications of this project and the functions enabling it to be flexible yet specific. In this research, the creators first explain how many buildings, especially commercial ones, demand a staggering amount of energy for operations. However, much of this demanded power is not properly utilized, with an estimated 30% of all energy being wasted. Losing out on this much power is a major financial and environmental loss, though such a travesty can be remedied by converting these large buildings into so-called smart buildings. By connecting all of the building functions via an internet connection, operations that would normally activate numerous appliances can now save energy by only activating the useful ones. Though the benefits are obvious, conversion to smart buildings is nevertheless painfully slow, primarily because building appliance metadata is neither standardized nor machine-readable.

This major issue is what the Brick schema aims to resolve. By converting building-specific terms with a normalized list of domain terms, machine-readable relationships can be made between building subsystems. These relationships are further established through the setting of classes and hierarchies. For example, sensors can be encapsulated by corresponding devices which themselves are encapsulated by

broad systems. These relationships can then be used to activate and adjust even the most minute processes with ease via SPARQL queries. While Brick is not the first project to tackle metadata standardization, the paper elaborates on how it beats competitors such as IFC and Haystack in both vocabulary and application requirements. To put their project to the test, the researchers applied the Brick schema to HVAC processes in six buildings. Constructed in different locations and during different periods, these buildings put the flexibility of Brick to the test. While some of the older buildings provided a few challenges to overcome, nevertheless Brick was able to be successfully implemented in each location.

Though the system and work behind Brick are highly utilizable, some areas could be expanded upon. Firstly, effective use of Brick requires energy usage information to be known, yet methods to access it are not inherent to Brick. Perhaps there could be a way to package this information alongside the Brick file. Also, Brick does not include real locational information that would allow operators to make informed decisions. Maybe such a quality can be implemented in future ventures so energy usage can be better optimized. Any implementation of Brick will likely be paired with these two types of information, including our initially planned usage of Brick in this paper.

Data

Primary Data & Challenges

Ideally in our project, we would have obtained the most recent HVAC data possible for our desired building. At UCSD, we would have done this by pairing sensor data with a Brick schema mapping for the building in order to query relevant sensors for our calculations, then used UCSD's Brick server to pull the data. Unfortunately, we were unable to obtain access to the Brick server because of data access issues to the Brickserver on campus that affected the four of us, as well as our mentors Rajesh Gupta and Xiaohan Fu.

Thus, we relied on a data pull from a previous project: Hsin-Yu Liu's Batch Reinforcement Learning dataset[3] for the EBU-3B (Computer Science) building. This data represents 15 rooms worth of data on floors 2, 3, and 4 of the building with data spanning from July 2017 to early January 2019. Because of this data's original purpose for batch reinforcement learning, we found that several elements of the data were not conducive to the type of project we created.

Firstly and perhaps most importantly, the 15 rooms in this dataset are un-labeled. There is a secondary index that may hint at which room each of the data points belong to. However, when we tried to use these secondary index values as a basis for the rooms, we found that enough of the data points were ambiguous that it was unclear how much room values would actually help with our model. We also asked the owner of the dataset for the room values - they were not kept. Re-obtaining them would also involve using the Brick server that we cannot obtain access to. This means that our model had to handle data that had multiple different energy variables for the same timestamp and other variables.

Another issue with the dataset was that the dataset was designed to have the same amount of data points per room, but was not locked into representing the same window of time. Some rooms' data cover July 2017-January 2018, while others cover May 2018-January 2019, and other rooms all manner of windows in between. If there was missing data for a room, the time range of data covered by that room was just extended until the number of data points was even. This creates significant missingness in the data that we had to handle.

Finally, the data points were intended to be recorded at regular 5 minute intervals - the data does not match exact 5 minute intervals. The data points range between half a minute to 4 and a half minutes

from the timestamp away from the timestamp that represents an even 5 minutes that they were supposed to represent (discounting timestamps that are entirely missing from the data). Predictions are easier with regular windows, so that was an additional challenge for us to tackle.

Cost Data

Aside from our primary dataset, we collected another dataset that we included in our model. The data that we conclusively will incorporate comes from Keaton Chia of UCSD's DERConnect$_4$ team surrounding UCSD's energy pricing for energy generated by UCSD [1].

UPDATED 5-YEAR RATES PROPOSED PLAN

	FY17/18	FY18/19 Q1-Q3	FY18/19 Q4	FY1920	FY2021	Proposed FY2122	FY2223	FY2324	FY2425	FY2526	FY2627
Rates:											
Electricity	$ 0.07	$ 0.08	$ 0.08	$ 0.10	$ 0.13	$ 0.17	$ 0.21	$ 0.24	$ 0.24	$ 0.25	$ 0.26
Chilled water,$/mmbtu	$ 0.75	$ 0.79	$ 0.10	$ 10.73	$ 11.27	$ 11.83	$ 13.02	$ 14.32	$ 14.75	$ 15.19	$ 15.65
Hi temp water,$/mmbtu	$ 0.75	$ 0.79	$ 0.10	$ 10.73	$ 11.27	$ 11.83	$ 13.02	$ 14.32	$ 14.75	$ 15.19	$ 15.65
natural gas,$/therms	$ 0.72	$ 0.76	$ 0.76	$ 0.80	$ 0.81	$ 0.80	$ 0.84	$ 0.86	$ 0.89	$ 0.92	$ 0.94
potable water	$ 7.86	$ 8.25	$ 9.93	$ 10.23	$ 10.61	$ 10.65	$ 11.18	$ 11.51	$ 11.86	$ 12.22	$ 12.58
reclaimed water	$ 3.93	$ 4.13	$ 2.15	$ 2.21	$ 2.23	$ 2.22	$ 2.33	$ 2.40	$ 2.47	$ 2.54	$ 2.62
% Increase:											
Electricity		14%	0%	30%	30%	30%	30%	10%	3%	3%	3%
Chilled water		5%	32%	3%	5%	5%	10%	10%	3%	3%	3%
Hi temp water		5%	32%	3%	5%	5%	10%	10%	3%	3%	3%
natural gas		5%	0%	5%	5%	-1%	5%	3%	3%	3%	3%
potable water		5%	20%	3%	5%	0.4%	5%	3%	3%	3%	3%
reclaimed water		5%	-48%	3%	5%	-0.4%	5%	3%	3%	3%	3%

Figure 1: UCSD's energy pricing plan for fiscal years 2017/2018 - 2026/2027

UCSD technically uses two sets of rates for the price of electricity - UCSD's rates and San Diego Gas and Electric's (SDGE) rates. We were unable to obtain information about when UCSD used its rates and when it used SDGE's rates, so for consistency for our analysis, we chose to stick exclusively to UCSD's rate values. Our cost estimates for energy usage will likely act as underestimates since it is likely that SDGE's rates are higher because it acts as an outside supplier. That being said, we believe because of the ambiguity, it made the most sense to use the UCSD data because it requires less domain knowledge to understand and reflects a future where UCSD aims to generate more of its own energy.

Because all of our data comes from fiscal years FY17/18 and FY18/19, we used the electricity values $0.07 and $0.08 as a transformation for energy values to get the cost values we used in the final prediction. We had to estimate when UCSD's change of the fiscal years and quarters was, as the values vary by department$_5$ and we did not know for sure which fiscal close date was correct for this scenario. We ended up using June 30th as the annual cutoff date.

Methods

Data Cleaning

As a result of the issues with the dataset that we ran into, we performed several steps to try and address them before performing the cost transformation:

1. We separated our data into training and testing sets based on dates in the original dataset, where approximately 70% of the data is before August 1, 2018 and the rest is August 1, 2018 and onwards.
2. We floored timestamps (with the Pandas Timestamp floor function) in the dataset to the nearest hour. The main goal of this step was to handle the irregularity of the timestamps by ensuring that timestamps were in regular intervals. We also wanted to ensure that there were more data points in each bucket for our next step.
3. We used medians to aggregate data from each floored timestamp into buckets for the training and testing dataset separately. Since the energy values range a lot as a result of the 15 unidentified rooms in the dataset, medians were chosen over means because they would be less susceptible to unlabeled outlier rooms that use less/more energy. This was a problem in early exploratory work on the dataset - some of the energy values in the 4th floor dataset were significantly higher than the rest of the values in the dataset, indicating that there was likely a very large room that acted differently than standard rooms in the building.
4. We imputed the training dataset's missing floored timestamps to have a complete collection of timestamps every hour from July 7, 2017 at 1 pm (UTC) to August 1, 2018 at midnight (UTC). The imputation value was based off of the median value for the hour (ie. if a timestamp was missing on January 1, 2018 at 1:00pm, it was filled with the median energy value for 1:00pm across the whole training dataset). This was chosen because daily patterns for energy were fairly regular. We did not impute the testing dataset to ensure that we did not evaluate our model on predictions of false values.
5. Finally, we performed the cost transformation described in the Introduction.

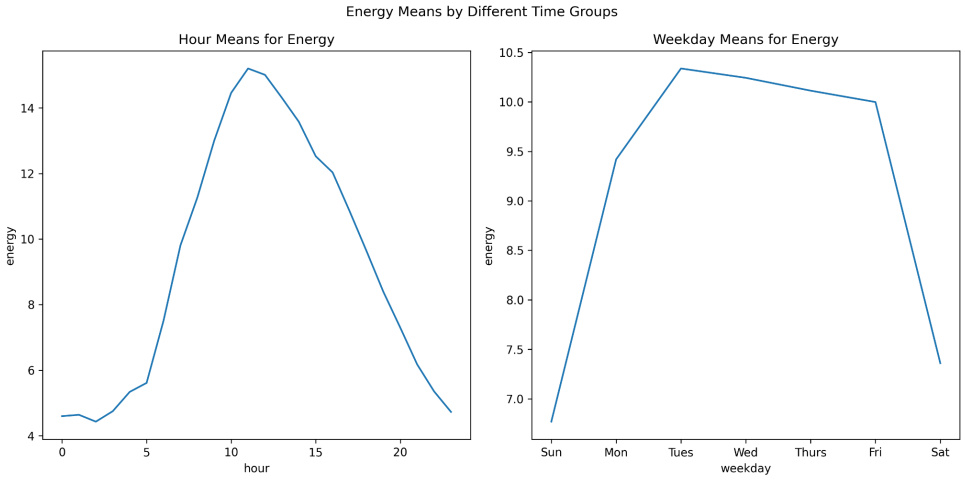

Figure 2: Hour Energy Means (in PST) & Weekday Energy Means for the Original Data

Modeling

Predictive models each have their own strengths and weaknesses. For example, linear models are known for their simplicity in predictions yet run both reliably and quickly. On the other hand, models such as artificial neural networks are hailed for their precise results yet do so at the cost of being operationally complex. A model suited for one predictive effort might struggle when applied to another venture no matter the similarity. To conclusively determine which model to use, our group applied the building data on five different predictive models varying in both complexity and algorithmic principle.

Linear

Our group's initial attempt to predict future energy consumption and spending came about through a linear regression model. This model was specifically created through Scikit-Learn's linear regression function and utilized info that came with the original dataset, apart from the cost values and the floored timestamps, such as humidity, outside air temperature, and setpoint values for temperature and airflow. The floored timestamps were split into different variables for day, hour, minute, second, and weekday. There was insufficient data to use the year value, and month was excluded for the linear model because of an anomaly in the data where October had much higher median values than the other months.

Decision Tree

After testing the performance of linear regression, our group worked with a variety of predictive models. One of these attempts revolved around the decision tree method. Creating this predictive model was fairly straightforward due to the similar usage of Scikit-Learn in implementing the decision tree functionality. This model worked with the same data and transformations analyzed by the linear regressor (including information about zone temperature, outside air temperature, and time). The regressor's max depth was set to 7 and the minimum samples split to 5 after testing different values via cross-validation.

Prophet

The next three models are non-Scikit-Learn based, more complex models we attempted. One of these models is the open source Prophet model[6], developed by Meta's Core Data Science team. The model makes time-series based predictions with several seasonal trends measured: yearly, monthly, weekly, and holidays. (Please note: with the way the data was split when testing this model, there was not enough data to analyze yearly trends).

The Prophet model is a little less diversified than the other models we tried because it uses exclusively time series values (left as a timezone-naive timestamp) and the predictor variable (energy/cost when transformed) - leaving out the other data we have available to us surrounding humidity, temperature setpoints and more. It relies entirely on the timestamps provided which are flawed for this dataset because of the inconsistent time ranges, uneven timestamps, and missingness we noted earlier. Nevertheless, we looked at Prophet as an option because of strengths it has apart from that downside: it is quick to train (2-3 minutes each time we ran it) and can easily generate future predictions with the make_future_dataframe method. This means that the model can quickly be retrained and is easy to evaluate for future data, should we be able to obtain it.

Neural Net

Another model our group created was an artificial neural network (ANN for short). For the ANN model, the uncleaned data and the cleaned data were again both used. We used the TensorFlow and Keras libraries to train the model to predict energy from the other features, and fit the model to optimize for MSE. We attempted to improve this score by iterating over different batch sizes and epochs. Unfortunately, in the end, the model failed to converge, as seen in the output score of "negative infinity", for both the uncleaned and cleaned data, across all combinations of hyperparameters. As such, we opted to proceed with the other models and improve their performance, as reaching an outcome with the neural net model would have taken extra heavy lifting and not guaranteed improved performance.

Autoregression

Here, we attempted to use Vector Autoregression and regular Autoregression on both the uncleaned and cleaned data. Statistical tests and models are provided by the statsmodels library. We performed statistical tests Granger causation, cointegration, and ADF to check if the features would work well with the model. After validating to find the order parameter, we fit the model and plotted the predicted values against actuals. We also modeled a regular autoregressive model, with the same setup as Vector Autoregression.

Optimization

We settled on the Decision Tree model as our final model. Then, we aimed to try and optimize cost by running the model on versions of the training set where the two user-defined values were lowered: the Common (Fahrenheit temperature) Setpoint and the Airflow Setpoint. For the Airflow Setpoint, we had to consider the constraint of minimum airflow for the rooms. Because of the un-labeled nature of the original data points, we did not know the size of the rooms we were studying, so we used two different estimates of the minimum airflow needed for occupancy - a larger estimate based on the average airflow setpoint in all of the data points (100 cubic feet/minute), a smaller estimate based on a known minimum setpoint value for smaller rooms in the building (45 CFM), and the unoccupied state of the room (0 CFM). Optimization sets were generated by looping through changes to temperature setpoint, changes to airflow setpoint, and changes to minimums for airflow setpoint.

With these optimized sets, we ran them through the trained model from earlier and then found the difference between the cost values from the training set and the predicted cost values from the new modified datasets. We stored those differences, along with the aggregates (mean, medium, minimum, maximum) by the hour. We produced visual results as the main method of analyzing the results of the documentation.

Results

Modeling

Model	MSE Data Uncleaned	MSE Data Cleaned
Linear Regression	~5.3	~0.0040
Decision Tree Regressor	~4.4	~0.0040
Meta Prophet Model	Range from 45 - 275+	~44
Neural Network	Failure to Converge	Failure to Converge
Autoregression	81.76	28.51

Figure 3: Mean Squared Errors for 5 Models on Uncleaned and Cleaned Datasets

For the neural net model, it failed to converge on both datasets, even with multiple iterations of various hyperparameters. In the vector autoregression model, the predictions were mostly straight lines with curves in the beginning. Although not terribly off, it did not predict any meaningful patterns visually [4].

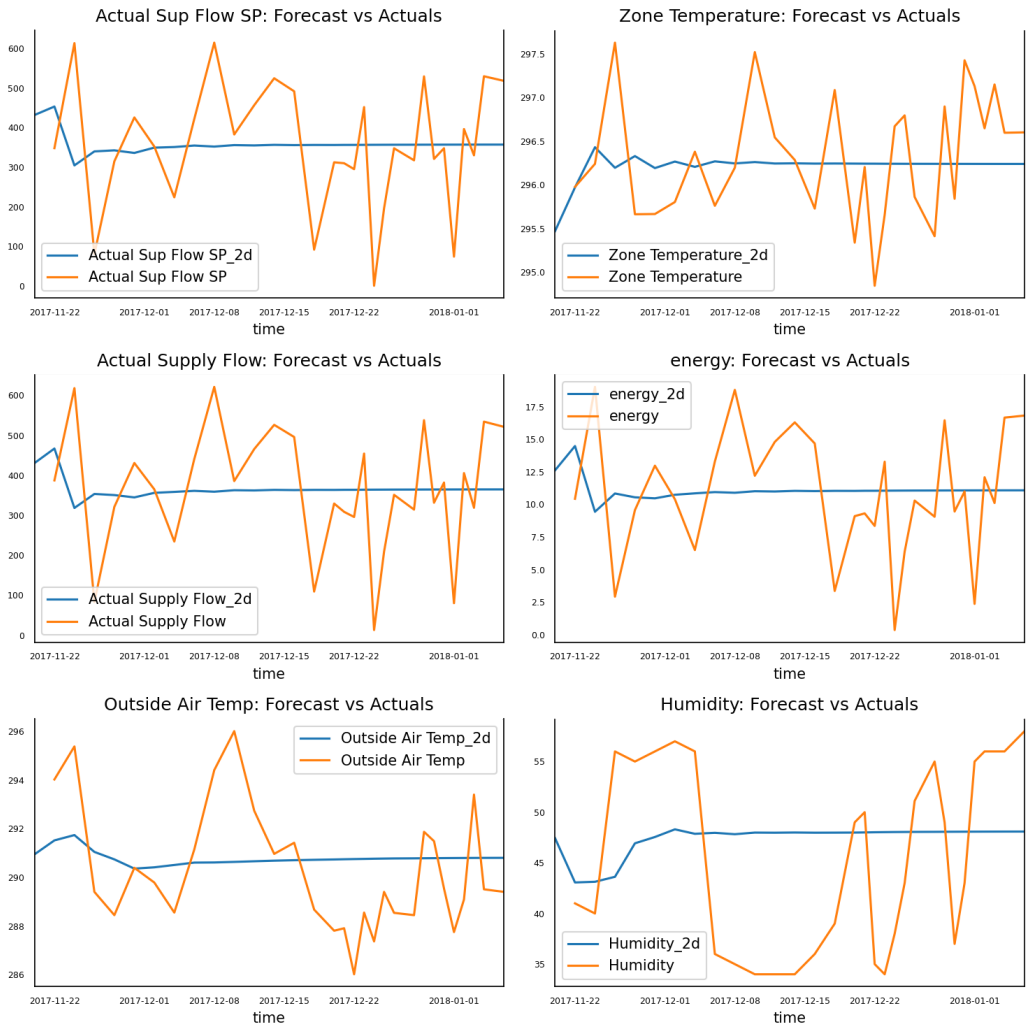

Figure 4: Trends Predicted via Autoregression (Blue) and Actuals (Orange)

Using autoregression instead of vector autoregression made the model more robust against the irregular timestamp intervals, but also diluted the time series patterns. The model was also unable to predict very far into the future.

Optimization

Variable	Feature Importance
Actual Supply Flow	0.8443523212076758
Actual Supply Flow Setpoint	0.15348546003820493
Outside Air Temp	0.0009296463055999001
Zone Temperature	0.00047143445476579686
Humidity	0.000414709750093561

Day	0.0001860857274926939
Hour	0.00012215018639087847
Common (Temperature) Setpoint	2.3290644011251573e-05
Month	1.4901685765064883e-05
Weekday, Year, Bias	0

Figure 5: Feature Importances for Decision Tree Model Ranked

Our optimization attempts were mostly unsuccessful. The largest temperature setpoint change we tried (10 degrees Fahrenheit) resulted in a median change of 0.0030, while the largest airflow setpoint reduction (200 CFM) resulted in a median change of 0.0259.

Discussion

Modeling

Through our group testing a broad range of predictive models, our expectations were met in some ways and thoroughly defied in others. For example, the decision tree model was originally not intended to be the final model. However, its performance and accuracy were much better than initially predicted and outpaced the competing models. The highly complex Prophet model was not nearly as fortunate because of its reliance on using timestamp values as a predictor. The model likely tried to accommodate for large outliers in both the cleaned and uncleaned datasets leading to unrealistic predictions. To make this model viable, our group would have had to do a different kind of data cleaning or use data inaccessible to us students. On another note, the neural network model proved tough to get working and progress remained slow throughout. After multiple unsuccessful attempts to get it running, the neural net model was abandoned due to upcoming deadlines. The autoregression models were also unreliable due to frequent gaps in time for the sensor readings. Although the previous experiment passed the statistical tests, intuitively energy should not affect the power used by the air handling unit. For this reason, the vector autoregression regression especially may not have worked since the features and labels had little to no influence on each other.

As a result of our analysis of all five model types we tried, we decided to use the decision tree as our final predictive model. We needed our model to be reliable, which ruled out the neural network from the beginning. We also wanted the prediction to be as accurate as possible, and our inability to lower the mean squared errors of the Prophet model and autoregression models excluded them. Between the linear regression and decision tree regressor, both had similar mean squared errors and training times (which we wanted to keep low). We ended up selecting the decision tree model over the linear model because we thought it would be more generalizable to future data since the energy values are not strictly linear.

Optimization

Air Setpoint & Occupancy

We expected the airflow setpoint factor to be one of the largest predictors of HVAC cost since the amount of energy an HVAC unit has to use depends on its airflow needs. Our initial results seemed to suggest this was true, as the airflow setpoint values had a wide range of maximum differences that we explored [6].

Examining these maximum differences show that there is a clear pattern of larger cost reductions tied to higher airflow setpoint reductions (which is expected), but also that the most impact is seen in the middle of the day (about 6am - 6pm).

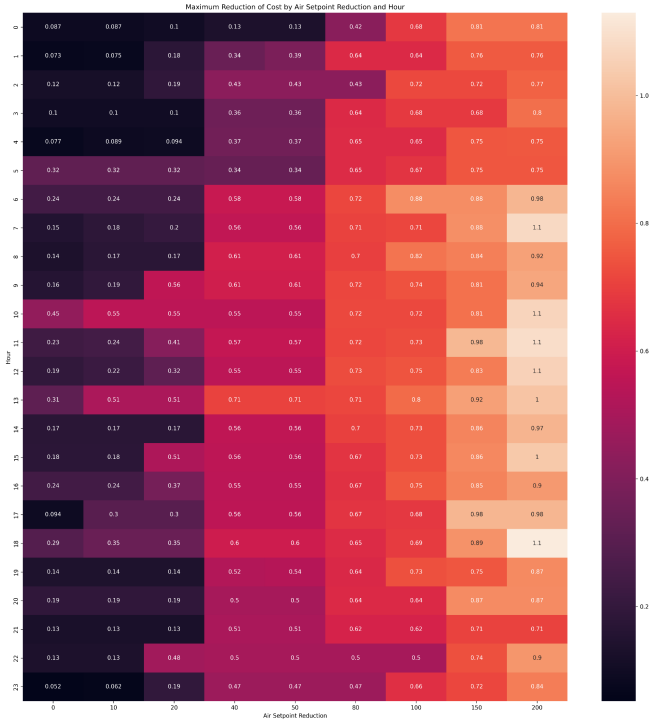

Figure 6: Heatmap of Max. $Cost_{Actual}$ - $Cost_{Predicted}$ Values Based on Hour (PST) and Air Setpoint Reduction

This distribution can partially be explained by another figure, where we tracked the proportion of times the reduction we performed was not fully able to reduce the airflow setpoint to the desired value because it ran into one of the three occupancy lower limits (0, 45, or 100) [7]. As we can see, the proportion of measurements that were unable to be reduced by a given air setpoint reduction value is high from 10 pm - 5 am, and then lower during 6 am - 9 pm. While it's unclear exactly what the cause of this is, our intuition was that it was likely that the HVAC system was completely turned off from 10 pm - 5 am, and was on from 6 am - 9pm (slightly extended business hours). We examined this intuition with a bar graph examining the proportion of zeroes (representing no occupancy and the lowest possible airflow setting) at each time of day [8].

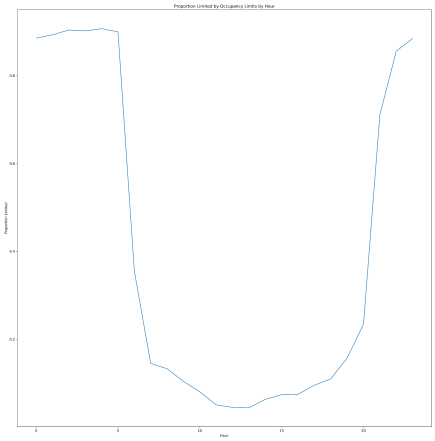

Figure 7: Proportion of Reductions Limited by Hour (PST)

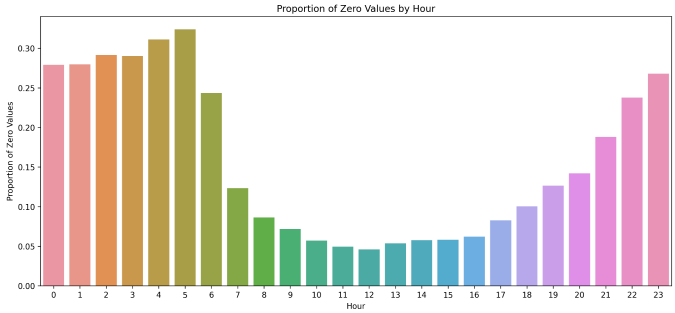

Figure 8: Proportion of Zero Values by Hour (PST)

We then looked more in depth at the occupancy analysis we performed. We created two bar graphs representing the median differences (between training and optimization cost) by hour and airflow setpoint, respectively, segmented by the three different occupancy limits we tried [9, 10].

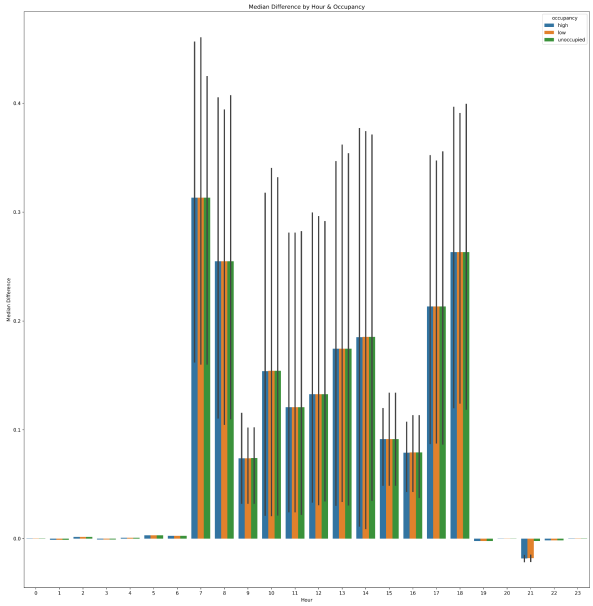

Figure 9: Median Differences by Hour and Occupancy

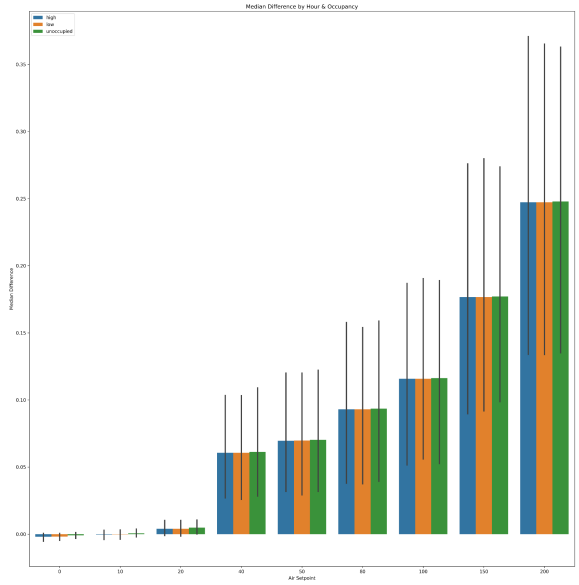

Figure 10: Median Differences by Air Setpoint and Occupancy

What we find is that overall, the different occupancy limits seem to have inconsistent effects on these median differences. Carefully examining the median differences by air setpoint shows that the unoccupied level has a slightly higher difference for each air setpoint difference, but that trend is not reflected for each hour (only reflected during the main workday at 9 am, 10 am, and 2 pm). These results were surprising to us, as we know that occupancy has a significant relationship with airflow because air needs are tied to the number of people in a room, and led us to believe something with our data needed further analysis.

We also saw a large change in the first figure between the median differences at the beginning & end of the work day (7-8 am, 5-6 pm) compared to the rest of the day which was not visible in our heatmap of maximum differences. This also contributed to the idea that this optimization analysis had limits. We believed that there could be an energy toll associated with the start and end of the day - we examine this notion later in the section below.

Exploring Temperature & Flaws with Optimization Process

The other change we made for the sake of optimization was reducing temperature setpoint values, but creating similar visuals to the airflow setpoint changes showed even less of a change. This is when we looked at the model's feature importance to see if we had made a mistake [5].

It turns out that we had made a mistake, albeit not with our visualizations. Rather, the temperature setpoint value was very unimportant to the overall cost calculation, ending up as the fifth least important feature. Examining these model weights demonstrated the actual values (highlighted in orange) instead of the user-set values we changed (highlighted in yellow) were more important in the model. Looking at visualizations for the variables provides some further insight into this [11].

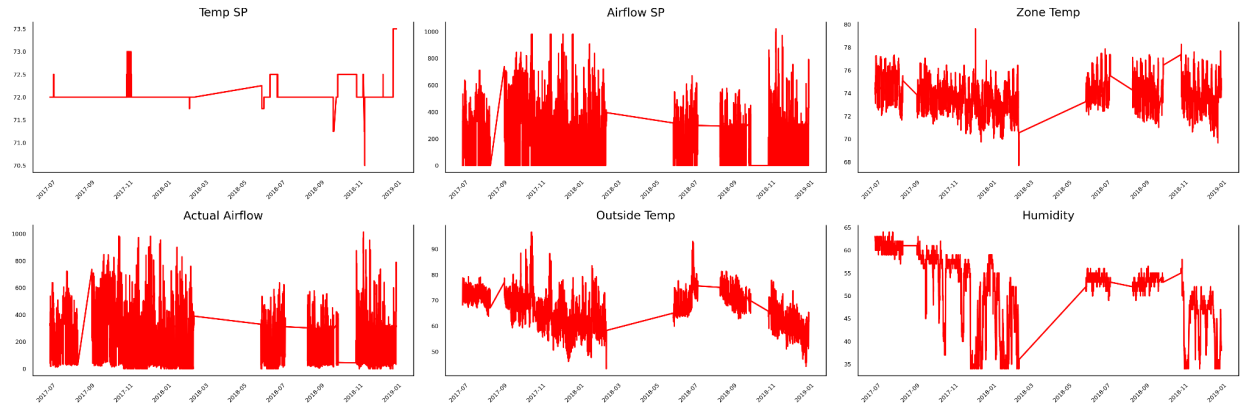

Figure 11: Line Graphs for Variables in the Dataset

Via Figure 11, the outside and zone temperatures both had very different distributions to the temperature setpoint, so those trends are likely more important to monitor than the temperature setpoint values for energy costs. Optimizing temperature likely involves figuring out how temperature setpoint can be made to more closely match zone temperature. More interestingly, the difference in the feature importances for the actual airflow values and the airflow setpoint values ended up surprising us and led us to reexamine our initial optimization work. We plotted a regression of the airflow values against each other to compare the two [12].

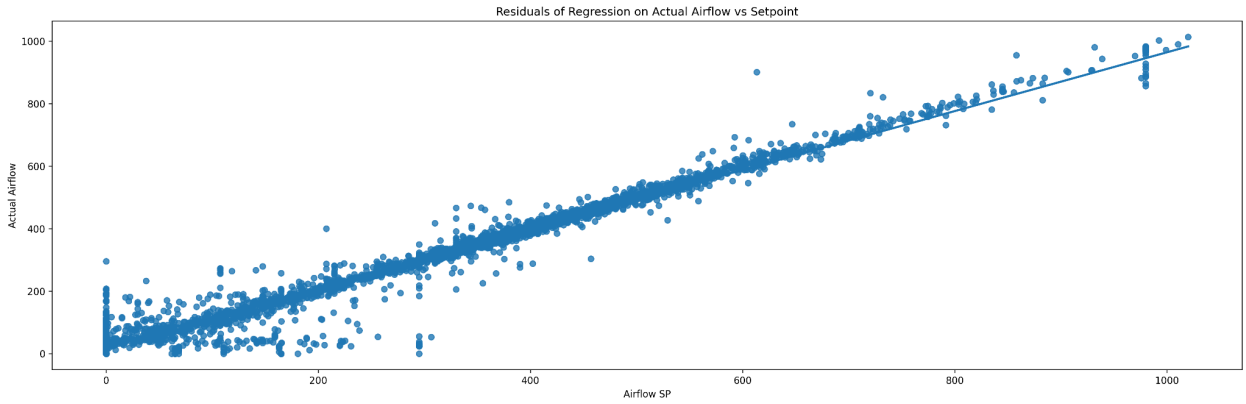

Figure 12: Plot of Residuals of Regression on Actual Airflow vs Setpoint

As seen in the plot, the actual airflow and setpoint values are mostly correlated, except for a set of anomalies where one of the values is set to 0 and the other values are not 0. So instead of optimizing the airflow as we were intending when we reduced airflow setpoint, we were likely optimizing for these cases. This provided a new perspective for our optimization attempts for airflow setpoint: understanding when these differences tend to occur and improving the HVAC system to output airflow to better match the setpoint values would be the key to more successful optimization attempts. Then, airflow and the airflow setpoint could be treated as the same variable and be permuted in the way we attempted to. In a way that we did not anticipate, the difference between the two variables circled back to the idea of occupancy and occupancy limits - when does a user say they are in a space compared to when air actually outputs to a space? This also acts as an explanation of the time periods when our optimization was most successful, between 7-8 am and 5-6 pm: these periods likely represent when a user turns their local HVAC system on and off, which causes larger deviations from the actual supply flow values. For future analysis into optimizing how to limit deviations between actual values and setpoint values and optimize energy values, looking at these periods would likely be the first step.

Conclusions

Our experience was defined by a combination of difficult failures and great successes. For the aspects that did not end up as we hoped: we found from the beginning that obtaining building data proves to be one of the largest challenges. We encourage future building designers at UC San Diego and otherwise to consider how data will be extracted in the design of future (smart) buildings. We also found that attempting to optimize setpoint values is ineffective when setpoints and actual values are separated, so any future optimization attempts on user setpoints need to first reduce differences or somehow devalue the actual values. Part of doing that analysis involves having up to date occupancy information to understand when the actual values or the setpoint values are accurate.

Despite our failures, we did make important discoveries about this data. We found great success in predicting energy and cost usage with simple (linear and decision tree) models. With additional time to train more advanced models, we believe it is possible to improve upon those predictions as well. We also found when there is an energy toll for users to turn on/off their local HVAC setpoint by analyzing output hourly, creating opportunities for future data analysts and building designers to correct for these differences for long-term benefit and energy reduction.

Future Modeling Analysis

While we were successfully able to predict cost values for our time period covered, limitations in our early process lead to multiple avenues in which this venture could be expanded upon. Firstly, this model could be used to analyze HVAC costs during the COVID lockdowns (2020-2021 roughly). COVID fundamentally changed how air circulation worked at UCSD and will need newer data to create a generalizable model. An ambitious project could involve using our predictive model to actively automate HVAC adjustments via BRICK programming. Occupancy, air circulation standards, time of day, and other requirements could be referenced to improve the minute-to-minute HVAC processes. Even small optimizations could save organizations sums of money in the long run. Another large-scale plan could use this predictive model's framework to analyze the cost-benefit of non-HVAC energy uses. Light fixtures, especially those that can shift their brightness, are but one of many smart building areas that stand to benefit greatly. Finally, the broad application of this project could be streamlined by using BRICKschema for deployment. Standardizing smart building asset names would significantly expedite the deployment process and ensure uniform changes for buildings outside of UCSD's scope.

Appendix

Other Data

We also collected extra data surrounding climate and energy from the National Ocean and Atmospheric Administration (NOAA)[7] and the U.S. Energy Information Administration (EIA)[8], respectively. We intended to use these datasets to supplement our work with the original model for optimization, but both due to time constraints and because of our findings that temperature did not have a significant effect on optimization.

Additional Limitations

As with most projects, our group faced various challenges and limitations throughout the development period. One minor example is with regard to our handling of the data timestamps. The dataset itself has timestamps based on the UTC timezone, however, our group's analysis is based on the local PST timezone. Also, daylight savings caused a portion of the time values to be listed in PST, likely causing some small misclassifications in the hour column. Another limitation is that the simple models used (linear and decision tree) are likely overfitting to our data for the air setpoint values. This could be remedied by summing up room usages to examine the building as a whole, however, a lack of accessible sensors made this unfeasible. Connecting Brick schema to these sensors would both fix the aforementioned issue as well as better generalize the process as a whole. Finally, as mentioned earlier, this project could not completely verify energy sourcing due to a lack of data access. Since we defaulted to only using UCSD's energy rates, the cost values are all likely underestimated.

Citations

1. Brick Schema
2. Brick : Metadata schema for portable smart building applications - ScienceDirect
3. https://github.com/HYDesmondLiu/B2RL
4. UCSD DERConnect

5. Fiscal Close: Overview
6. Prophet | Forecasting at scale.
7. NOAA
8. EIA