Summer 2020

Smart Meters in London: Exploring the effects of surge pricing schemes on household energy usage

INF2179 – Project Report

Group 25: A Mahfouz & Thomas Rosenthal

Introduction

The Smart Meters of London data consists of energy consumption readings from a sample of 5,566 London households participating in the Low Carbon London (LCL) project between November 2011 and February 2014. The LCL project tested if notifying households that their energy usage would cost more (“surge pricing”) during certain hours of the following day would lead to lower energy usage and reduce stress on local distribution grids throughout 2013. As per the London Datastore, from which the Kaggle data is derived, households were categorized according to the CACI Acorn classification scheme; trial customers are meant to be representative of the Greater London Area according to these categories. LCL customers were further divided into two subgroups. The first, consisting of 1123 households, were subjected to Dynamic Time of Use energy prices and were informed of prices ahead of time via an in-home smart display or text message. The remaining customers paid a flat rate for energy. The London data is supplemented in Kaggle by daily and hourly weather readings taken from Dark Sky. The original Tariff information from the London Datastore will be included (not provided within the Kaggle dataset).

Data Description

The data consists of 340 files containing 167,817,021 records, amounting to 9.3 GB of data. Entity relationships and join keys are summarized in the diagram below (Figure 0.0).

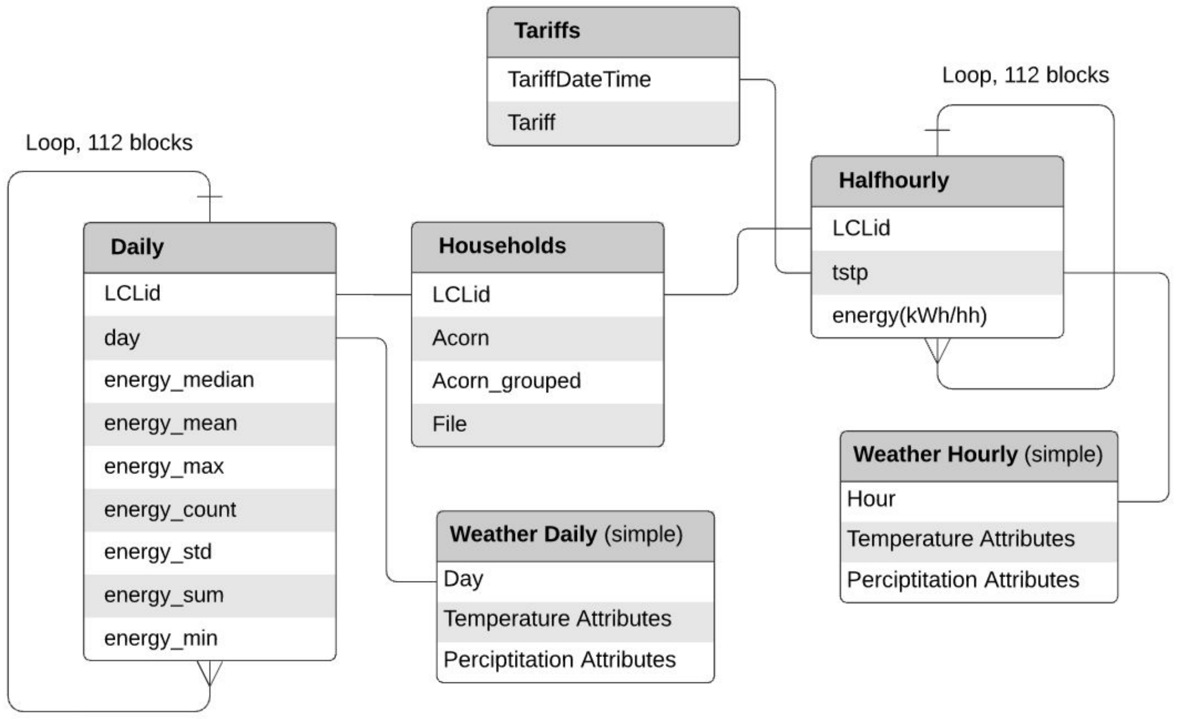


Figure 0.0

The definition for each dataset is as follows (Table 0.0):

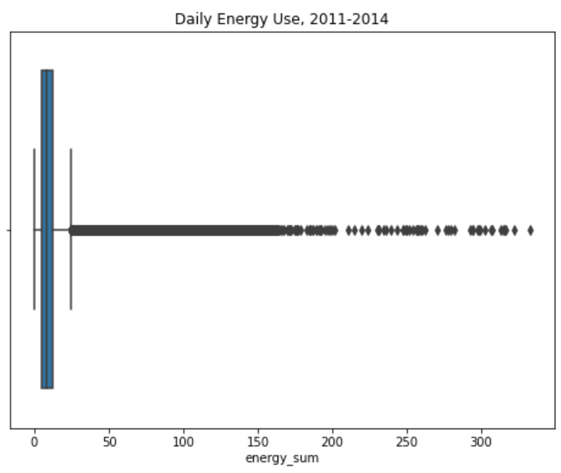
Tariffs	“High”, “Low”, or “Normal” tariffs, per half-hour for 2013.
Households	This dataset contains characteristics of participating households, including block assignments and ACORN group, a socioeconomic indicator derived from customer segmentation research. Households are categorized under one of nineteen classes, which then roll up to one of three larger ACORN groups: “Adversity”, “Comfortable”, or “Affluent”. 51 households are categorized under undocumented groups (“ACORN-U” and “ACORN-”), suggesting data errors.
Daily	Each row contains meter reading summary statistics for a given household and day. Statistics include energy sum and count of readings. Not every household has a record for each day.
Halfhourly	Each row is a meter reading for one customer at one half-hour interval. Not every household has a reading for each interval.
Weather Hourly	Hourly weather for London from 2011-11-01 to 2014-03-31. Fields include temperature variables, wind speed and direction, atmospheric pressure, visibility, humidity, precipitation type, and a weather summary.
Weather Daily	Daily weather for London from 2011-11-01 to 2014-03-31. Fields include those in the hourly readings, as well as summary information like daily high and low temperatures, sunrise and sunset times.

Table 0.0

Exploratory Data Analysis

Preliminary exploratory data analysis reveals a few trends worth investigating, as well as characteristics in the data that should be accounted for in an analysis.

Daily household energy usage is highly right-skewed, (Figure 0.1, Figure 0.2). While the median total energy use is 7.8 kWh, the mean is 10.12 kWh, with a standard deviation of 9.1; daily energy use levels go as high as 332.6 kWh, suggesting either outsize houses or faulty smart meters.



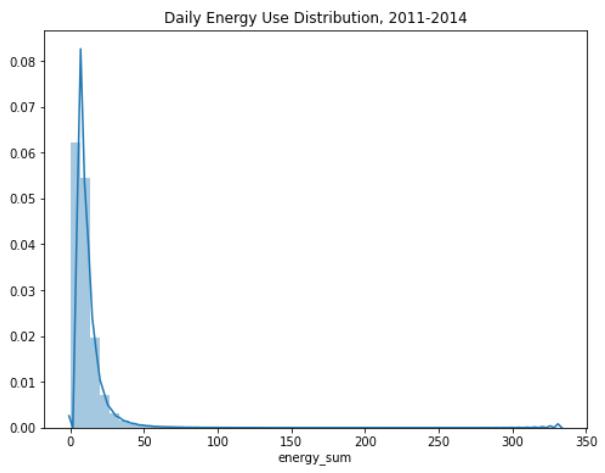


Figure 0.1

Figure 0.2

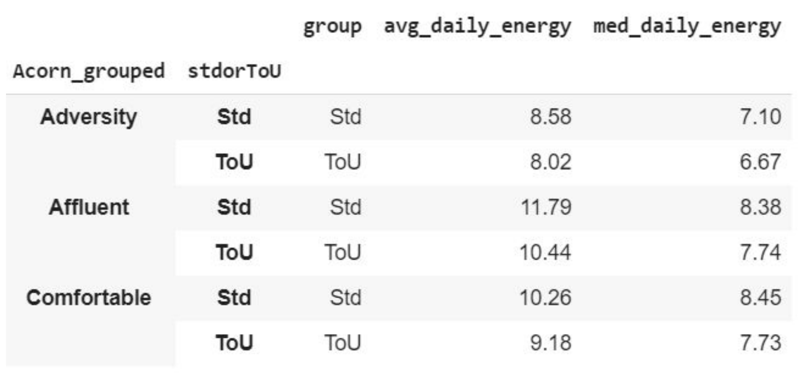
Daily energy use statistics are calculated based on half-hourly readings. A household with complete readings for the day should have an energy_count value of 48. As seen in Table 0.1, most records do consist of 48 readings, but almost 1% (34,059) of all records have fewer.

energy_count	**# of records**
48	3469352
47	21209
1	11301
46	1005
29	544

Table 0.1

The household information dataset shows that customers were grouped into 111 blocks of 50 households and a final overflow block of 16 households. Within a block, households tended to belong to the same ACORN group (adversity, comfortable, or affluent).

More affluent households tend to use more energy. Within socioeconomic class, households on the dynamic pricing scheme tended to use less power, as shown in Table 0.2.



Acorn_grouped	stdorToU	group	avg_daily_energy	med_daily_energy
Adversity	**Std**	Std	8.58	7.10
	ToU	ToU	8.02	6.67
Affluent	**Std**	Std	11.79	8.38
	ToU	ToU	10.44	7.74
Comfortable	**Std**	Std	10.26	8.45
	ToU	ToU	9.18	7.73

Table 0.2

Research Questions

1. Given household type and weather information, can we predict daily household energy use?
2. Can we predict the dynamic pricing tariff class label (high/normal/low) based on energy usage patterns, ACORN class, half-hour increment, and month?

Question 1

Motivation

This question was motivated by an interest in predicting aggregate energy demand, which is useful for infrastructure and resiliency planning. Understanding the extent to which weather can be used to predict demand can also help design interventions like surge pricing -- tariffs can be raised during expected high-demand periods to discourage elective appliance use.

Methods and Analysis

Preprocessing

Loading and Type Conversion

Three datasets were used for this analysis: the *daily household energy* readings, *household information*, and *daily weather*. Because the 29 months in this data include 12 months of the dynamic pricing pilot, which likely affected some households' behaviours, analysis was conducted for both the full 29 months (2011-2014) as well as for 2013 only. Unless stated otherwise, all methods described here were applied to both the 2011-2014 and 2013-only data.

Upon loading all daily energy and daily weather data, date variables were transformed from objects or strings to datetimes. In the daily energy data, only the date field was transformed this way. In the weather readings, a “day” datetime field was created from sunset times. Sunrise and sunset time features were also converted to datetimes.

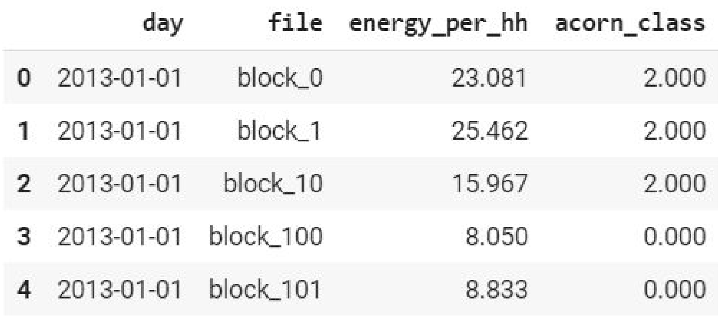
Aggregating, Joining, and Feature Engineering

The exploratory data analysis revealed considerable variation and right-skewness in daily energy use among households, even within an ACORN group. In addition, some records are incomplete and do not reflect a full day’s energy use. As the motivation behind the research question is not to predict individual household energy use, but rather overall energy demand, records with an energy_count value less than 48 (indicating the full day’s energy usage was not recorded) were dropped, and the remaining energy readings were aggregated up to the block level and averaged, producing mean household energy use values (energy_by_hh) by block and day. Aggregation by block mitigated some of the outlier effects and reduced the number of records, while still retaining the variation in energy readings. Smoothing outliers this way was preferable to dropping outlier records outright. Without information on whether the outlier or legitimate or mismeasurements, it makes more sense to treat them as legitimate for demand planning purposes -- the human cost of overestimating demand on a given day is less than the cost of underestimating demand and risking power grid strain. To do the aggregation, block information was joined to energy readings by household identifiers, then energy_sum values were summed by block and those sums were divided by the number of households in a block.

Although ACORN class is a property of individual houses, houses in a block tend to belong to the same ACORN group. In order to include socioeconomic indicators in the analysis, blocks were assigned the ACORN group to which most of their constituent households belonged. For example, if a block consisted of 48 affluent households and 2 comfortable households, that block was categorized as affluent. Households that were categorized as “ACORN-” or “ACORN-U” were excluded from the analysis, as there was no documentation available about those categories. Readings from 51 such households were removed for this reason.

ACORN groups are ordinal, with “adversity” being the poorest category and “affluent” being the richest; “comfortable” falls in between. Therefore, blocks were given ordered numeric codes: majority-adversity blocks were assigned an acorn_class code of 0, majority-comfortable blocks an acorn_class code of 1, and majority-affluent blocks an acorn_class code of 2.

These transformations resulted in a data frame of average household energy use and socioeconomic class, where every record was unique by block (file) and day, as shown in Table 1.0 below. For the 2013-only data, the data frame consists of 40,880 records; for the 2011-2014 data, the data frame consists of 88,860 records.



	day	file	energy_per_hh	acorn_class
0	2013-01-01	block_0	23.081	2.000
1	2013-01-01	block_1	25.462	2.000
2	2013-01-01	block_10	15.967	2.000
3	2013-01-01	block_100	8.050	0.000
4	2013-01-01	block_101	8.833	0.000

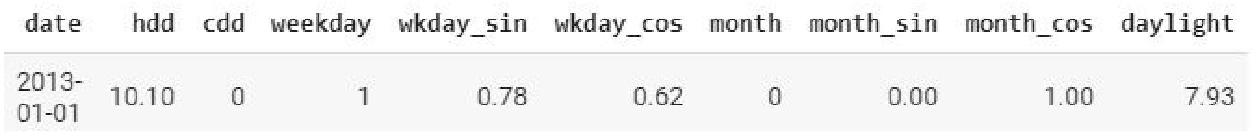
Table 1.0

Daily weather data was then joined in by day. To improve modeling, heating degree day (‘hdd’) and cooling degree day (‘cdd’) features were added. Heating degrees and cooling degrees are both measurements designed to quantify energy demand for indoors climate control. Heating degrees are the number of degrees below 15.5 C the average temperature was on a given day. Cooling degrees are the number of degrees over 25 C the average temperature was on a given day. For example, if a day’s high was 13 C, and the low was 7 C, then the average temperature was 10 C, and the heating degrees were 5.5. If the day’s average temperature was 20 C, then both heating degrees and cooling degrees for the day were 0.

Daylight times were also calculated, since households use more energy to light their homes when there is less natural light. This calculation is simply the amount of time in hours between sunrise and sunset.

Finally, day of week (‘weekday’) and month were both extracted as integers from the dates to model cyclical trends in energy use, such as increased weekend usage or decreased summer usage. These values were zero-indexed to ease computations. For day of week, values range from 0 for Mondays to 6 for Sundays. For month, values range from 0 for January to 11 for December.

Of course, energy use patterns in January (month 0) are more similar to those in December (month 11) than they are to patterns in May (month 4), although 0 is numerically closer to 4 than it is to 11. To capture this similarity, month and day of week variables were transformed into sine and cosine components. Examples of these engineered features, showing the similarity between sine and cosine variables in January and December, is shown below.



date	hdd	cdd	weekday	wkday_sin	wkday_cos	month	month_sin	month_cos	daylight
2013-01-01	10.10	0	1	0.78	0.62	0	0.00	1.00	7.93

Table 1.1

date	hdd	cdd	weekday	wkday_sin	wkday_cos	month	month_sin	month_cos	daylight
2013-12-31	7.44	0	1	0.78	0.62	11	-0.50	0.87	7.91

Table 1.2

Feature Selection

Feature selection occurred through an interactive, iterative process, and was informed by lived experience as well as exploratory data visualization. After the data was prepped, correlation matrices were generated to identify promising features as well as confounding variables to avoid. Due to the number of variables, correlation coefficients were left out of Figure 1.0 and Figure 1.1, but they are available in the associated Jupyter notebook.

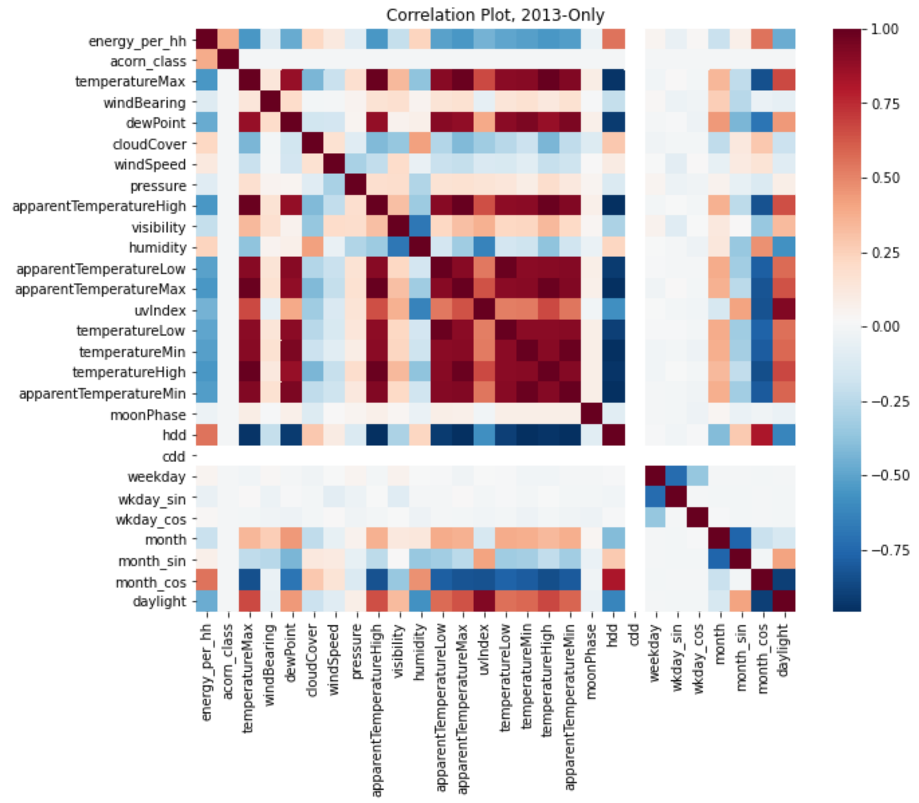


Figure 1.0

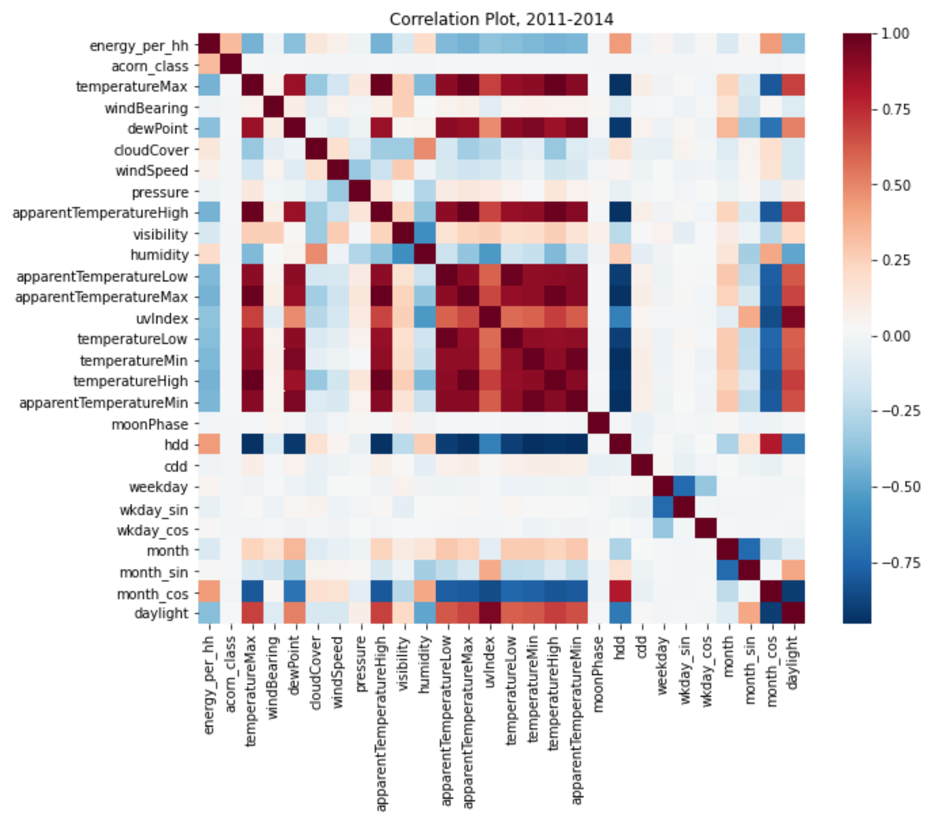


Figure 1.1

The matrices show that while most temperature variables are highly correlated with average household energy use and with each other, there is little relationship between cooling degrees ('cdd') and the target variable ('energy_per_hh'). The matrices also highlighted some less obvious confounds to consider when selecting features. Daylight, for example, is highly correlated with uv indices; dew point is correlated with temperature.

While correlation plots are useful for finding linear relationships, they do not reveal non-linear relationships. A pairwise plot was generated to check for non-linear relationships and to confirm the distributions of individual features. As can be seen in Figure 1.2, some variables, like cloud cover, are imperfectly normally distributed, while others like day of week are not normally distributed at all. Looking at the month column confirms the cyclical nature of many weather variables. The energy_per_hh column furthest on the right and its corresponding row at the bottom show that the correlation plot did not mask any polynomial relationships and further suggests linear relationships between energy usage and daylight, weather, and socioeconomic class.

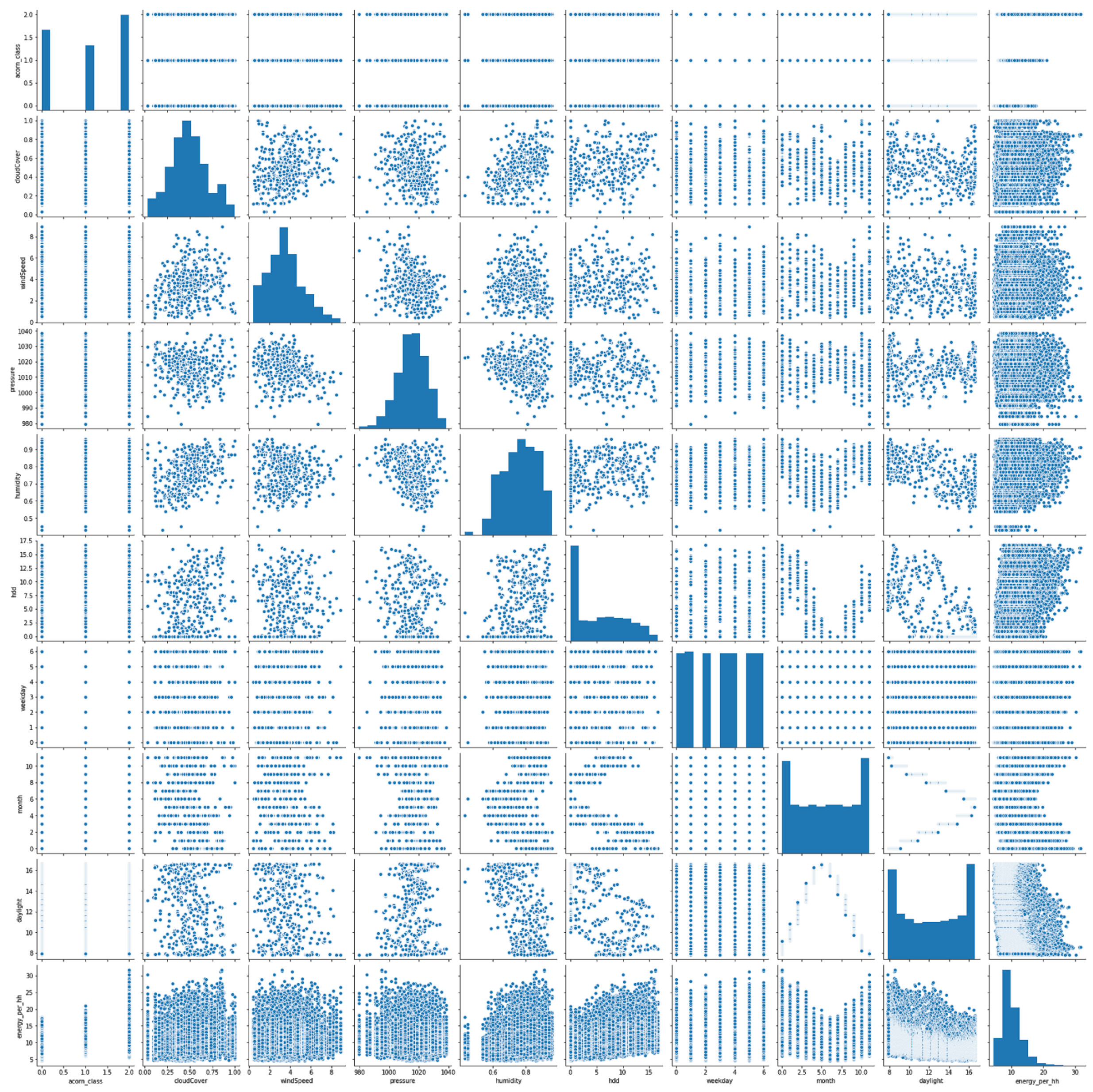


Figure 1.2

Ultimately, a reduced set of features was included in the model, as they helped predict energy use without overfitting. The features prior to scaling are shown in Table 1.3.

	acorn_class	cloudCover	windSpeed	pressure	humidity	hdd	wkday_sin	wkday_cos	month_sin	month_cos	daylight
0	1.000	0.412	0.423	0.631	0.736	0.605	0.901	0.802	0.500	1.000	0.012
1	1.000	0.412	0.423	0.631	0.736	0.605	0.901	0.802	0.500	1.000	0.012
2	1.000	0.412	0.423	0.631	0.736	0.605	0.901	0.802	0.500	1.000	0.012
3	0.000	0.412	0.423	0.631	0.736	0.605	0.901	0.802	0.500	1.000	0.012
4	0.000	0.412	0.423	0.631	0.736	0.605	0.901	0.802	0.500	1.000	0.012

Table 1.3

Feature Scaling

The features were scaled using Min-Max scaling. Min-Max scaling adjusts values in a field so that they all fall in the [0, 1] range, while maintaining the original shape of the distribution. The features in the data have different scales. Humidity, for example, ranges from zero to one, while atmospheric pressure values have a broader range centering on 1014 millibars. Such differences in scales can distort the perceived impact of a feature. For example, a change in one unit of humidity is much more dramatic than a change of one unit of atmospheric pressure, and regression coefficients for a model using non-standardized humidity and pressure features will reflect that difference.

Scaling has some other advantages, such as speeding up model computation -- the methods used here work best with scaled features. This step is also key for methods such as ridge and lasso regression that seek to minimize coefficients.

Modeling

Regression methods were used to model the data, as the target variable was continuous, quantitative, and more or less normally distributed. Unless stated otherwise, the features in the model were **ACORN class, cloud cover, wind speed, pressure, humidity, heating degree days, daylight,**, and the cyclical **sine and cosine features for day of week and month. For 2011-2014 models, year was also included as a feature.**

Train/Test Split

The data was initially randomly split into train and test data, with 80% in train and the other 20% in test. However, because a time series aspect is involved, the data was re-divided into train and test splits in a more time-aware manner. Various splits were tried, such as seasonal splits (with February, May, August, and November in the test set and all other records in train), and epochal splits (i.e., every day before a cutoff in the training set, and records after that date in test). These splits are discussed in more detail in the Modeling subsection of Question Two.

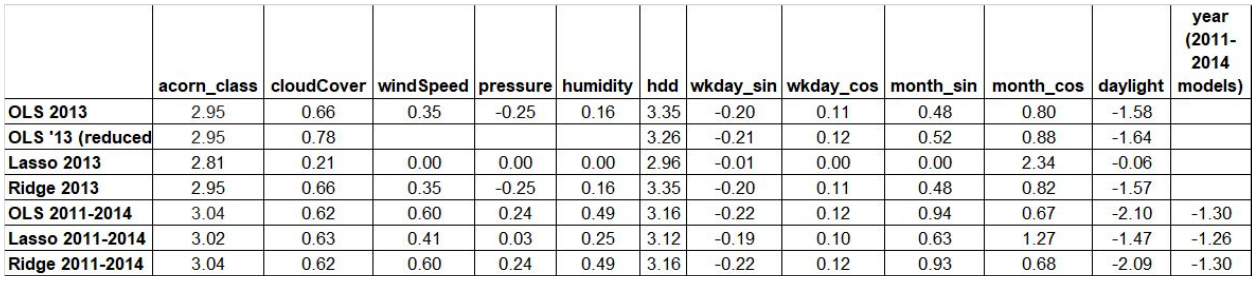
Ultimately, both Tuesdays and Sundays were placed in the test set and all other days were put in train for both the 2013-only data and the 2011-2014 data. This amounted to a roughly 71/29 train/test split.

Ordinary Least Squares

Ordinary least squares regression models were constructed for both the 2013-only and 2011-2014 data. Scikit-learn's default linear regression parameters were used here. All attempts to improve OLS model performance involved feature selection and engineering, rather than hyperparameter tuning.

Using all the features listed above, the 2013-only OLS model yielded an R2 of 0.507 on the training data. After performing the lasso regression, wind speed, pressure, and humidity were removed from the model, and this new model had a training R2 of 0.506. Using all the above-listed features in the 2011-2014 OLS model yielded a training R2 of 0.354.

Due to the min-max scaling, coefficients are unintuitive to interpret. Nonetheless, they are shown for all models in Table 1.4 below. ACORN class and heating degrees consistently had the highest coefficients, indicating that as observations moved from the low end of their respective ranges to the high end, average household energy use increased most dramatically. Increased daylight was associated with reduced energy use, though the lasso regression model for 2013-only suggests that this effect is relatively unimportant.



	acorn_class	cloudCover	windSpeed	pressure	humidity	hdd	wkday_sin	wkday_cos	month_sin	month_cos	daylight	year (2011-2014 models)
OLS 2013	2.95	0.66	0.35	-0.25	0.16	3.35	-0.20	0.11	0.48	0.80	-1.58	
OLS '13 (reduced	2.95	0.78				3.26	-0.21	0.12	0.52	0.88	-1.64	
Lasso 2013	2.81	0.21	0.00	0.00	0.00	2.96	-0.01	0.00	0.00	2.34	-0.06	
Ridge 2013	2.95	0.66	0.35	-0.25	0.16	3.35	-0.20	0.11	0.48	0.82	-1.57	
OLS 2011-2014	3.04	0.62	0.60	0.24	0.49	3.16	-0.22	0.12	0.94	0.67	-2.10	-1.30
Lasso 2011-2014	3.02	0.63	0.41	0.03	0.25	3.12	-0.19	0.10	0.63	1.27	-1.47	-1.26
Ridge 2011-2014	3.04	0.62	0.60	0.24	0.49	3.16	-0.22	0.12	0.93	0.68	-2.09	-1.30

Table 1.4

Lasso Regression

Lasso models were also created for both the 2013-only and 2011-2014 data. These models performed no better than the OLS models, though they helped identify features that explained little variance in the data. With the default alpha values of 1, the lasso models returned a poorly fit line with coefficients of zero for all features.

Alpha values were then tuned using grid search. For the 2013-only data, the optimal alpha value was found to be 0.025; for the 2011-2014 data, an even smaller alpha value of 0.0036 was optimal. Training metrics were only slightly worse than the OLS and ridge models, even with the coefficients for several variables -- wind speed, pressure, humidity, weekday sine, and month cosine -- reduced to zero in the 2013-only model, as seen in Table 1.4 above. The zeroed coefficients informed a second iteration of the 2013-only OLS model.

Ridge Regression

Ridge regression models, like lasso, did not show notable improvements over the OLS models. The ridge models returned lines of best fit that matched their OLS counterparts, with equivalent R2s. Attempts to tune the alpha values did not produce better results. These ridge regression results, coupled with the small optimal alpha values for the lasso regression models, suggests that the original OLS models were not overfitting the data, and that increasing model bias does little to improve variance.

The ridge regression models were identical to their OLS counterparts.

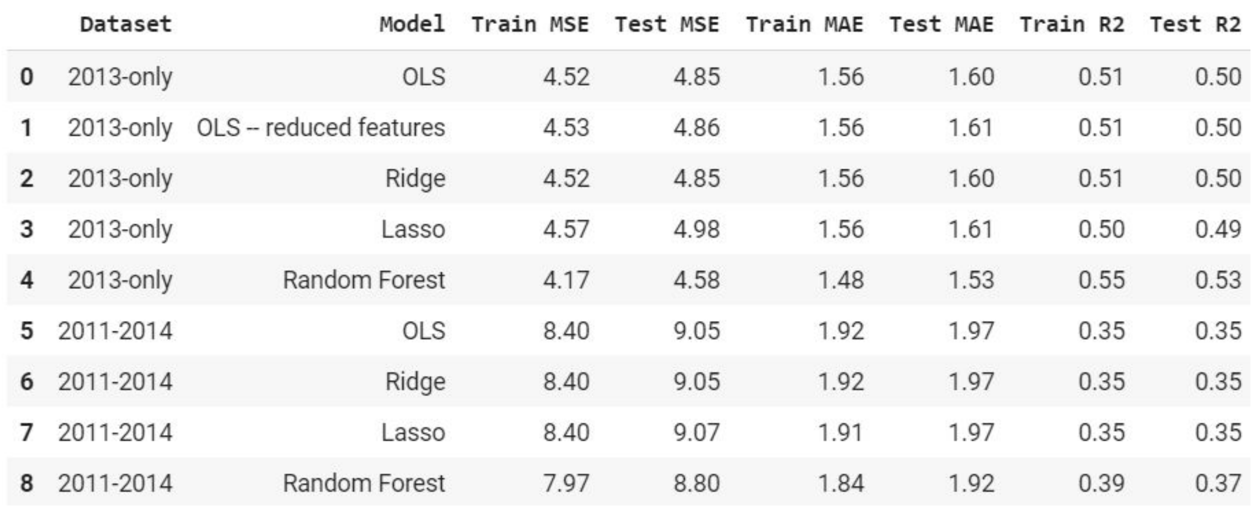
Random Forest Regression

Finally, random forest regression models were created to predict average daily household energy use. Random forests have two advantages over the other regression models used. First, they do not assume linearity or normality as the other methods do. This is useful, as not all the variables in the data are normally distributed. Second, decision trees can be visualized and more easily explained than multiple regressions. We can, for example, see that the month and ACORN class drive many of the early splits, implying that those two features are important to predicting energy use.

Grid search was used to help tune the maximum tree depth and the number of estimators in both the 2013-only and 2011-2014 models. As this method does not take into account the time series aspect of the data, some manual tuning was also performed. In the end, both models had 30 estimators. The 2013-only model had a maximum depth of 10, while the 2011-2014 model had a maximum depth of 12. These models performed marginally better than their OLS, lasso, and ridge counterparts, as can be seen in the model evaluation section. The improvement was more pronounced in the 2011-2014 models, possibly because the full dataset is characterized by less linearity in relationships between features and the target variable.

Model Evaluation

Models were evaluated on the test data described in the train/test section. Performance metrics for the various models are shown in the table below.



	Dataset	Model	Train MSE	Test MSE	Train MAE	Test MAE	Train R2	Test R2
0	2013-only	OLS	4.52	4.85	1.56	1.60	0.51	0.50
1	2013-only	OLS -- reduced features	4.53	4.86	1.56	1.61	0.51	0.50
2	2013-only	Ridge	4.52	4.85	1.56	1.60	0.51	0.50
3	2013-only	Lasso	4.57	4.98	1.56	1.61	0.50	0.49
4	2013-only	Random Forest	4.17	4.58	1.48	1.53	0.55	0.53
5	2011-2014	OLS	8.40	9.05	1.92	1.97	0.35	0.35
6	2011-2014	Ridge	8.40	9.05	1.92	1.97	0.35	0.35
7	2011-2014	Lasso	8.40	9.07	1.91	1.97	0.35	0.35
8	2011-2014	Random Forest	7.97	8.80	1.84	1.92	0.39	0.37

Table 1.5

Test metrics are fairly close to train metrics, particularly for the 2013-only models. The R2 values around 0.5 indicate that these models explain roughly half of the variation in average daily household energy use. However, the mean squared error and mean absolute error values could be lower -- the mean daily energy use per household is around 9.5 kWh, so the error values are comparatively large. The errors are even bigger in the 2011-2014 data; the much higher MSE values suggest that the 2011-2014 data has a more extreme range of target values.

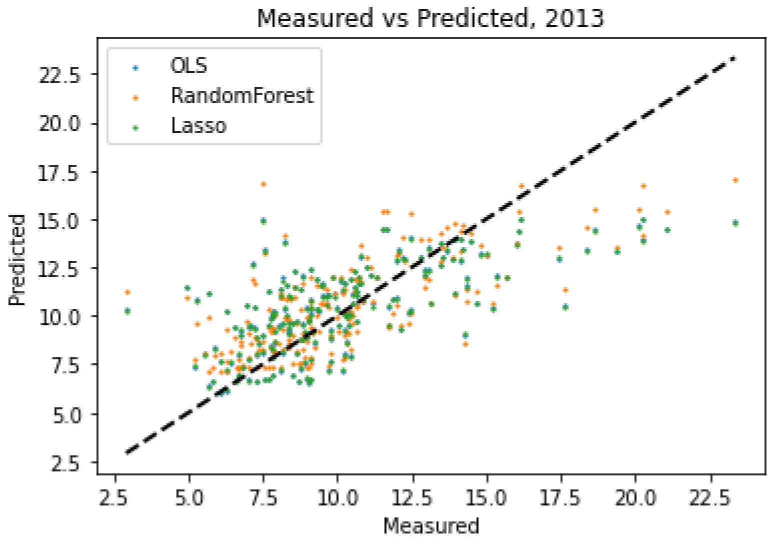


Figure 1.3

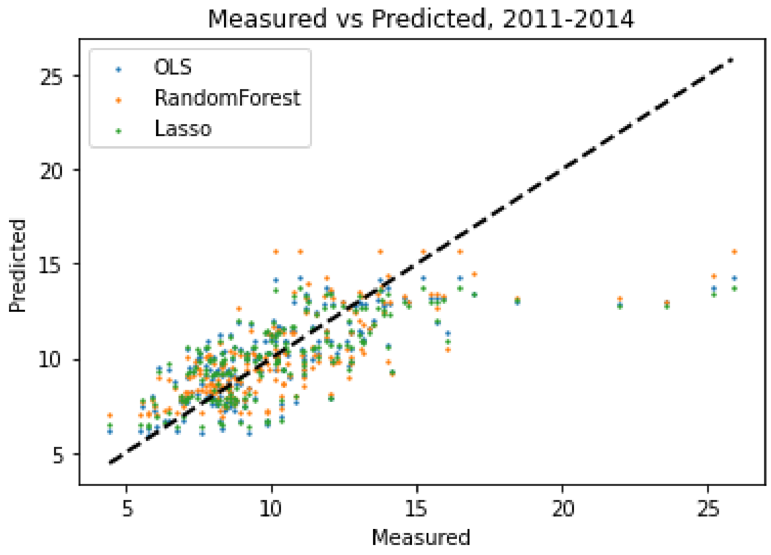


Figure 1.4

Graphing the measured versus predicted energy use values shows that the models tend to under-predict the actual observed value. This effect is more pronounced with the 2011-2014 data, as can be seen in Figures 1.3 and 1.4.

Discussion and Limitations

Prediction quality for the models was middling verging on poor, as indicated by the R2 values in Table 1.5. At best, models explained just over half of the variation in average household daily energy use; at worst, they explained just over a third of the variation. This suggests that energy use is significantly affected by variables besides weather and socioeconomic class. Built environment factors, such as the number of rooms, age and size of housing units could be key in more accurately predicting energy demand.

The differences in predictive power between the 2013-only and 2011-2014 models also suggest that there were 2013-specific differences in household energy use habits. Both models learned these habits, but energy usage outside of 2013 was different enough that the 2011-2014 models made worse predictions. Optimistically, this could be because the dynamic pricing pilot encouraged more thoughtful energy use among participating households. Other possible explanations for the reduced predictive power of the 2011-2014 models could be that the full dataset included more extreme outliers to begin with. This intuition is supported by the prediction versus measurement graphs above showing that the models tended to under-predict energy use.

Of all the methods used, regression forests were the most effective, and heating degree days and socioeconomic class were the most important predictors throughout. Both of these findings make sense. The relationship between energy use and variables like month are cyclical, not linear, and some explanatory variables, like heating degrees, are not normally distributed. A method that does not assume normal distributions or linearity is therefore well-suited to this data.

Ultimately, average household daily energy usage predictions would likely be improved if averages were taken per ACORN class and day, rather than per block and day. This would have further smoothed outlier effects, though it would also have dramatically reduced the number of observations. This could still be acceptable for predicting aggregate daily energy demand. Removing energy use outliers would also improve model metrics, though possibly at the cost of practical use.

Question 2

Motivation

Initially, I proposed this question as a supplement to our third group member's question aiming to predict energy usage based on the half-hour dataset. Having clear delineations of usage per ACORN group, per month, per half-hour would not only help to determine the effectiveness of a pilot, but also allow thresholds to be established in the broader sense of when price tariffs might be most effective in behavioural modification. The ideal model would allow decision makers to explore the cusp of mean energy usage under each tariff label and test the thresholds that cause a consumer to stratify their energy usage.

The pilot uses cost as a deterrent for energy usage during tariffs, but if tariff labels are unknowable by machine learning models, this might otherwise suggest that the energy usage is determined much more by external factors (such as weather, household wealth, culture, and many more) than the pilot. For example, a household might ignore price tariffs to keep their house warm on a cold day and simply endure the extra cost.

Methods and Analysis

Loading

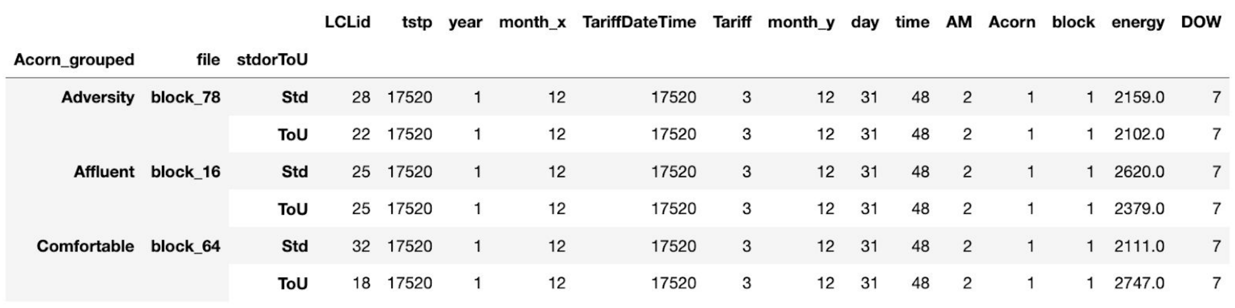
Using a for-loop in conjunction with a file walk, the *Halfhourly* dataset was loaded. This dataset was split into 112 distinct csv files, named by their Block № (e.g. `Block_0, Block_10, Block_100`). Each csv file was loaded and concatenated (vertically added) to an output `pandas` dataframe. Due to the enormous size of this data, three blocks were selected (see §Preparation), although the entire dataset was loaded during initial exploration using this process. The blocks not selected for this project were moved to another subdirectory (`'/overflow/'`) and can be examined at any point by changing the for-loop's subdirectory condition. The dataframe was modified to include year, month, and day of week (DOW). The loaded dataframe was filtered to include only data from 2013 (the year the pilot ran).

Additionally, two other datasets were loaded: *Tariffs* and *Households*. These were loaded as dataframes by use of read_excel and read_csv respectively. *Tariffs* was modified to include basic date parts: month, day, time, and AM/PM. *Households* was modified to include the Block № as an independent column from the filename (e.g. `Block_0 = 0`).

Wrangling

This question required each dataframe to be merged: 1) *Tariffs* was merged with *Halfhourly*, to append the Tariff value for each half-hour to its respective timestamp (e.g. `2/1/2013 11:30:00, "High"`); and, 2) *Households* was merged with the new *Tariffs-Halfhourly* dataframe to append the ACORN group and pricing scheme for each household, as well as the household’s Block №.

Aggregating this final dataset by ACORN group and pricing scheme demonstrates the expected similarity of each file, where only ACORN group, energy usage, and the number of households on each price scheme is variable (Table 2.0).



			LCLid	tstp	year	month_x	TariffDateTime	Tariff	month_y	day	time	AM	Acorn	block	energy	DOW
Acorn_grouped	**file**	**stdorToU**														
Adversity	**block_78**	**Std**	28	17520	1	12	17520	3	12	31	48	2	1	1	2159.0	7
		ToU	22	17520	1	12	17520	3	12	31	48	2	1	1	2102.0	7
Affluent	**block_16**	**Std**	25	17520	1	12	17520	3	12	31	48	2	1	1	2620.0	7
		ToU	25	17520	1	12	17520	3	12	31	48	2	1	1	2379.0	7
Comfortable	**block_64**	**Std**	32	17520	1	12	17520	3	12	31	48	2	1	1	2111.0	7
		ToU	18	17520	1	12	17520	3	12	31	48	2	1	1	2747.0	7

Table 2.0

Model Preparation

Determining which blocks to select for analysis became increasingly important as even basic transformations took an exceptionally long time due to the data’s size. The ideal block would be an equal distribution of ACORN groups *and* price schemes; however, few blocks contained mixed ACORN groups. Instead, one block for each ACORN group was selected with a relatively even number of households per price scheme. Because the pilot contained 5566 households, with only 1123 on the Dynamic Time of Use (ToU) pricing scheme, few blocks were ideal candidates, and of those selected, the blocks were slightly weighted towards Standard (std) pricing schemes. The chosen blocks were as follows: 1) Block 78: Adversity - 28 std 22 ToU; 2) Block 16: Affluent - 25 std, 25 ToU; and 3) Block 64: Comfortable - 32 std, 18 ToU.

Sampling

The resulting three-block dataframe contained 2,541,570 rows. This continued to remain problematic for the models to be run locally (see §Limitations) and thus further limits on the dataframe were required. Using `pandas v1.1.0 df.sample(frac = .25)`, the dataframe was randomly sampled within the following grouping constants: ACORN groups, price schemes, every day within the year, and every half-hour within those days. These groupings create even distributions of these values, thus reducing the presence of which households’ energy readings were selected for each time period. The resulting data sample was 625,729 rows (Table 2.1).

			LCLid	tstp	year	month_x	TariffDateTime	Tariff	month_y	day	time	AM	Acorn	block	energy	DOW
Acorn_grouped	**file**	**stdorToU**														
Adversity	**block_78**	**Std**	28	17520	1	12	17520	3	12	31	48	2	1	1	1717.0	7
		ToU	22	17520	1	12	17520	3	12	31	48	2	1	1	1723.0	7
Affluent	**block_16**	**Std**	25	17520	1	12	17520	3	12	31	48	2	1	1	1976.0	7
		ToU	25	17520	1	12	17520	3	12	31	48	2	1	1	1933.0	7
Comfortable	**block_64**	**Std**	32	17520	1	12	17520	3	12	31	48	2	1	1	1754.0	7
		ToU	18	17520	1	12	17520	3	12	31	48	2	1	1	1913.0	7

Table 2.1

Dimensional Reduction

The resulting sample dataframe contained 17 columns, several of them irrelevant and/or duplicated, following `pandas` default merge syntax. Ten of these columns were removed, including LCLid, several datepart derived columns, block, and filename. Block was initially believed to be important, but due to dataset size stratification (each block representing a single ACORN group), it was removed. Block might be a worthwhile consideration if the entire dataset was analysed.

To further simplify datetime metrics, timestamps were converted into half-hour increments. A function 'halfhourinc' divides timestamps into three dateparts (hour, minute, second). Because energy readings were taken every half-hour and already in 24-hour time, hour dateparts could simply be multiplied by two, with minute dateparts equalling 30 adding 1. For example, 14:00:00 (2pm) would equal 28 `(14*2)` and 14:30:00 would equal 29 `(14*2+1)`.

Relationships

Intermittent Exploratory Data Analysis (EDA) further examined the relationships between ACORN group, price scheme, and energy usage.

Two graphs were produced (Figure 2.0, Figure 2.1):

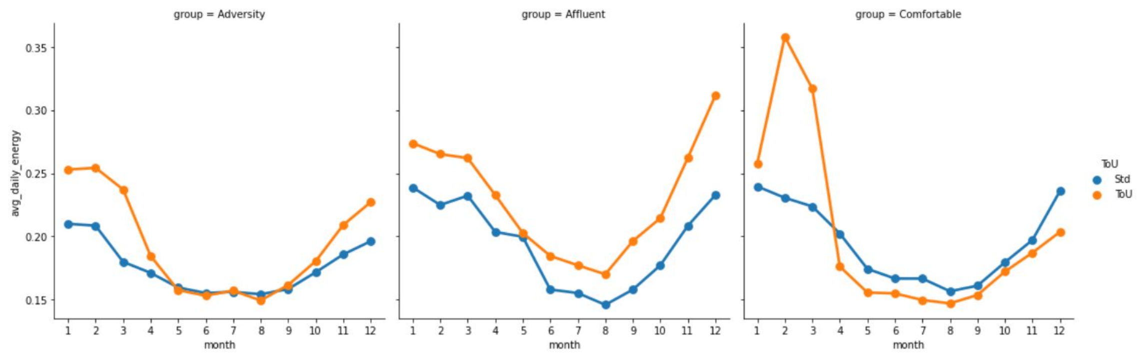


Figure 2.0

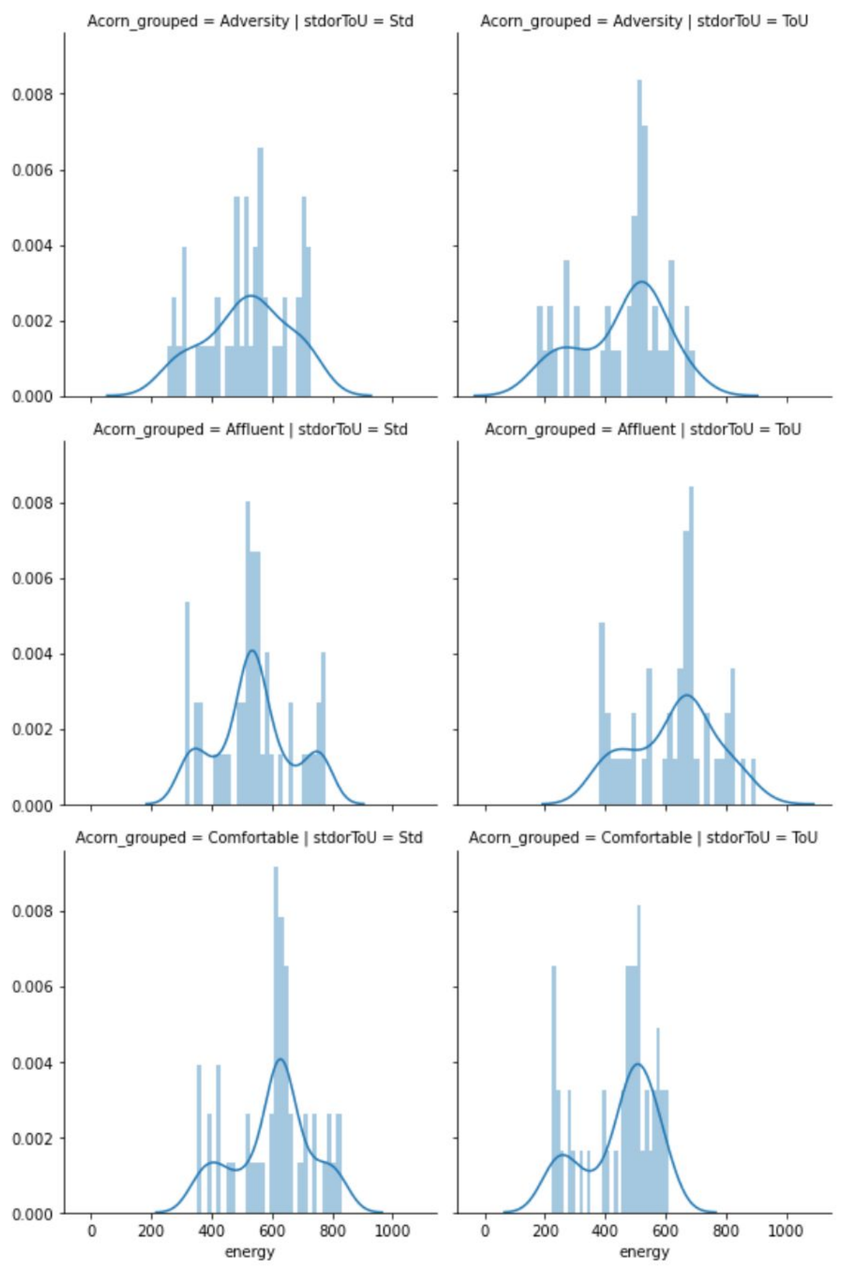


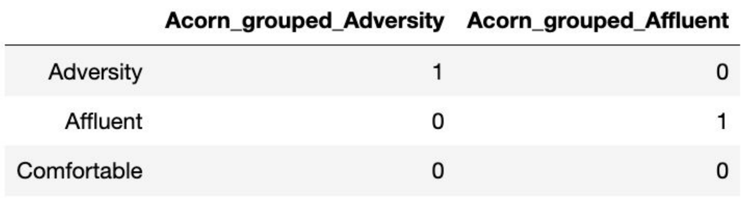
Figure 2.1

From this, it is possible to generalize that Affluent ACORN group households use more energy regardless of price scheme. Adversity ACORN group households use the least amount of energy, including households in the lowest energy bins (<200), and no households in the highest energy bins (>800). All ACORN groups show good normality within the selected blocks.

Additionally, it is notable that dynamic price scheme households use more energy overall (although Comfortable ACORN group households are much closer in usage across both price schemes). This may seem counterintuitive to the pilot’s purpose; however, Low tariffs occur at a much greater frequency than High tariffs and are intended to encourage energy usage at non-peak times.

Encoding

Using `pandas get_dummies`, non-numeric values (pricing schemes and ACORN groups) were encoded. No ordinal encoding was required for these variables. Pricing schemes were encoded as binary variables in a single column where ToU = 1. ACORN groups were encoded as binary variables into two columns (3 groups - 1) following the mapping below (Table 2.2).



	Acorn_grouped_Adversity	Acorn_grouped_Affluent
Adversity	1	0
Affluent	0	1
Comfortable	0	0

Table 2.2

Correlation

Using `seaborn clustermap` no correlation was found amongst variables (Figure 2.2).

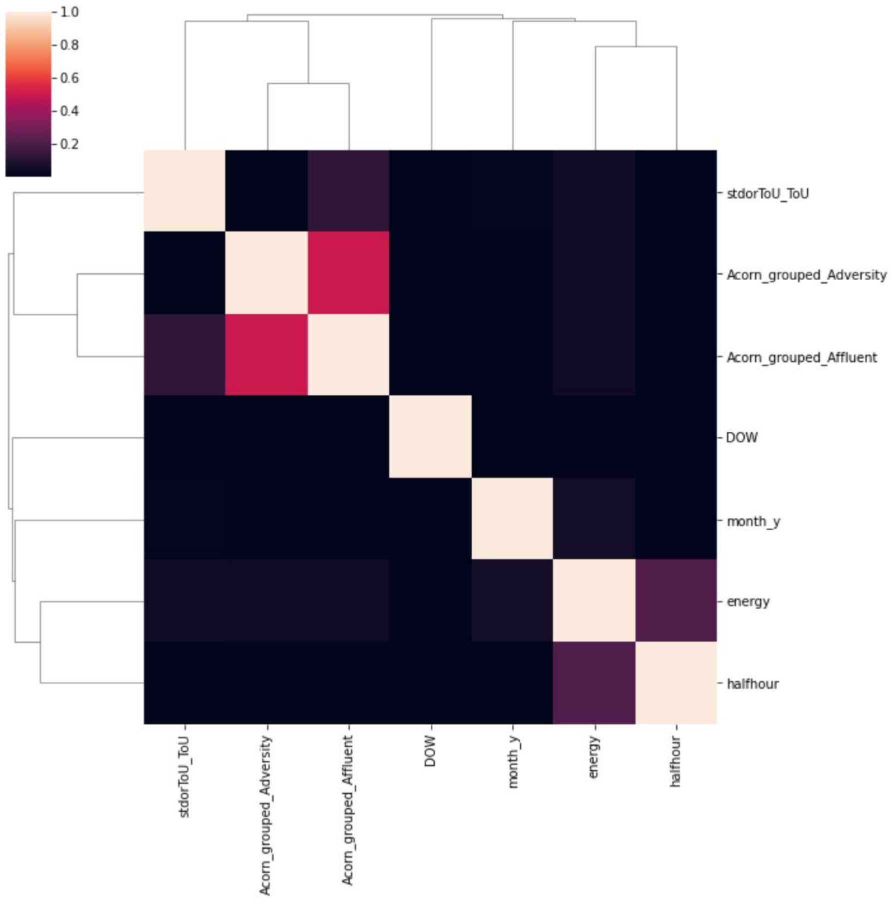


Figure 2.2

It should be noted that despite higher correlation scores, Acorn_group_Adversity and Acorn_group_Affluent are not corollary by definition (no household is in more than one group, and the existence of any one household is unrelated to the existence of another). Their Boolean encoding simply notes that there is an obvious relationship where Adversity = 1, Affluent always = 0, and vice versa.

Normalization and Cyclical Transformations

Energy was normalized using `scikit-learn preprocessing MinMaxScalar`. Normalization scales a variable's values (x) to be between 0 and 1 as
$x = (x - x_{min}) / (x_{max} - x_{min})$.

Further research into feature scaling suggested that cyclical sine/cosine transformations were appropriate for the remaining time features (month, DOW, halfhour). The transformation is intended to create a relationship between the start and end of cycles, where sine values (0 at the start of a cycle, 1 at the end of a cycle) are reciprocated by their cosine values (1 at the start of a cycle, 0 at the end of a cycle). This relationship helps models realize these inherent relationships, such as Sunday (DOW:6) sine values are cosine values for Monday (DOW:0) and vice versa.

The generalized transformation is `sin(x)*2*π/n` where n represents the number of increments required to complete a cycle (e.g. for DOW: 7, for Month: 12). It is expected that the starting value for x is 0 (i.e. `x-1` will adjust Month values to 0-11 rather than 1-12).

Modelling

Epoch Splitting

Standard train:test splits are inadvisable for models consisting of time series data. In order to avoid randomly selecting months which may have significantly different energy usage (as noted by Figure 2.0), Epoch splitting instead splits the data on specific date constructs.

Three options were explored:

1. Days, where Tuesday and Sunday were placed into the test dataset. The desired effect was to have one weekday and one weekend day. Tuesday was determined to be the most average weekday, having an on-par average energy usage (Figure 2.3) and a reasonable amount of each Tariff class (Figure 2.4). This represents a 71.5:28.5 train:test split.

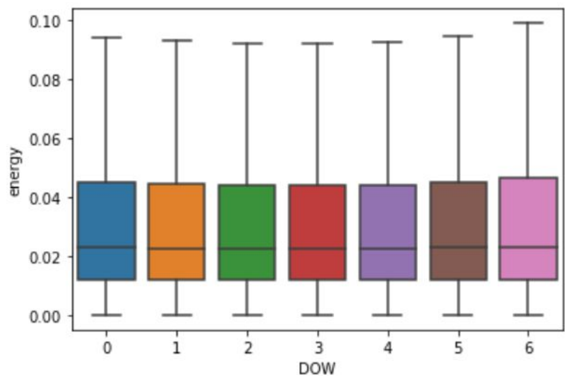


Figure 2.3

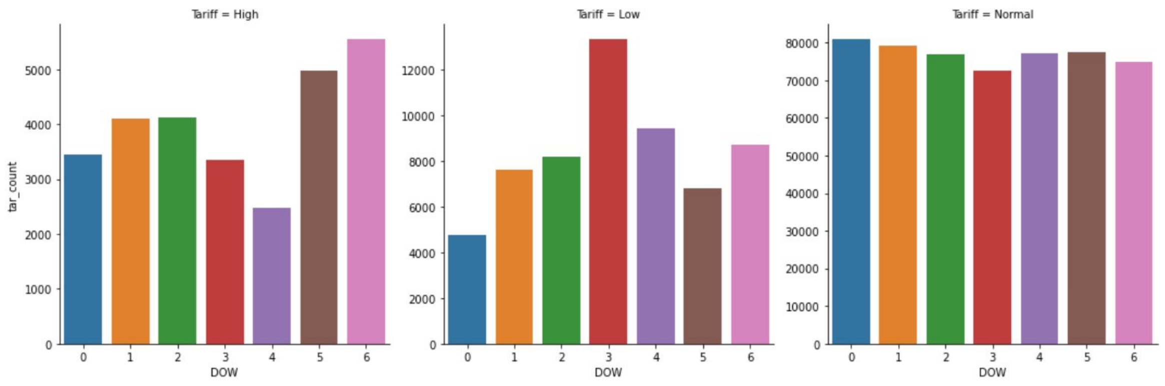


Figure 2.4 (note independent y-axes due to tariff frequency imbalance)

2. Seasonal, where January (representing winter), April (representing spring), July (representing summer), and October (representing autumn) were placed into the test dataset. These months were likewise considered average amongst their respective seasonal months (e.g. January, selected from December through February) (Figure 2.5). This represents a 66.6:33.3 train:test split.

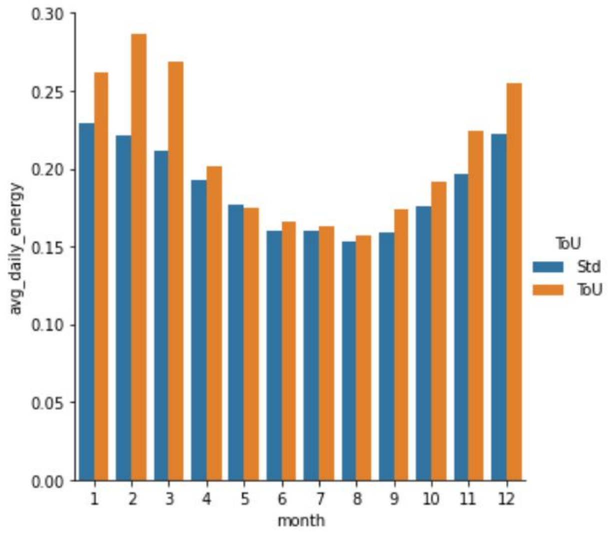


Figure 2.5

3. By temperature groupings (Cold, Mild, Hot) in an attempt to lower the seasonal train:test split to 75:25. May (representing mild), July (representing hot), and December (representing cold) were selected based on horizontal lines drawn on Figure 2.6 dividing each sector into three distinct periods of four months and then selecting the average month.

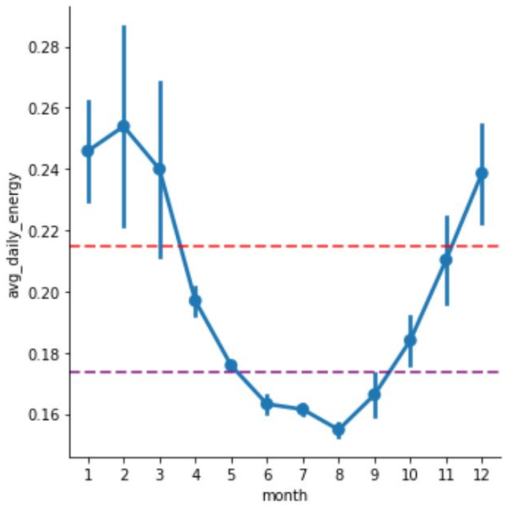


Figure 2.6

K-Nearest Neighbors

K-Nearest Neighbors (KNN) algorithms using `scikit-learn neighbors KNeighborsClassifier` were tested with default and hypertuned parameters. Hypertuned parameters were run using a for-loop rather than `scikit-learn model_selection GridSearchCV` in order to avoid cross-validation. (This is true of all hypertuned models in this project; train:test datasets cannot be randomized during by cross-validation when stratified by Epoch splitting, thus disallowing the use of `GridSearchCV`.)

KNN was selected as an initial model due to its simplicity and the dataset's relatively few features. All features were numeric and scaled, meeting the KNN's model assumption.

156[1] KNN models were run, exploring three parameters:

1. `metric`, the distance measurement
    - euclidean
    - manhattan
    - minkowski (generalization distance metric)
2. `weights`, means by which point nearness is measured
    - uniform (where nearness to neighbors does not determine the label)
    - distance

1 (3 metrics * 2 weights * 26 n_neighbors)

3. `n_neighbors`, the number of neighbors (K)
    - odd integers between 1 and 52
    - model accuracy threshold checks at 101, 501, and 1001

No KNN model performed notably. Model accuracy scores needed to be greater than ~85% which represented an entirely Normal class label (i.e. Normal Recall = 1.0). Thus worse performing models actually represent a better distribution of class labels, even if incorrectly labeled. This challenge remained true for all models. “Best” models exclude models with Normal Recall = 1.0 (see also §Limitations, with regards to balanced accuracy scores, which would have better accounted for this).

The best KNN model (`metric='manhattan', weights='distance', n_neighbors=3`) produced the following confusion matrix (Table 2.3) and classification report (Table 2.4) and an overall accuracy rate of 80%.

	High	Normal	Low
High	774	8344	536
Normal	5133	141316	7518
Low	559	14672	1117

Table 2.3

	Precision	Recall	F1	Support
High	.12	.08	.10	9654
Normal	.86	.92	.89	153967
Low	.12	.07	.09	16348

Table 2.4

Random Forest

Random Forest (RF) algorithms using `scikit-learn ensemble RandomForestClassifier` were tested with default and hypertuned parameters. Hypertuned parameters were also run using a for-loop.

Following the motivation of this research question, the ideal model to determine Tariff class labels might resemble a decision tree, but with many significant variables possibly affecting the starting split for a decision tree, RF algorithms seemed logical to pick the best possible tree.

720[2] RF models were run, exploring three parameters:

1. `n_estimators`, how many trees
    - 5, 25, 50, 100, 250, 500
2. `min_samples_split`, number of samples required to split a node
    - 2, 5, 10, 15, 100
3. `min_samples_leaf`, number of samples required to create a leaf
    - 1, 2, 5, 10
4. `max_depth`, the nodes per tree
    - 2, 4, 8, 16, 32, None

This model performed poorly but was moderately successful in further tests (see §Testing on New Data).

The best RF model (`n_estimators = 5, min_samples_split = 2, min_samples_leaf = 1, max_depth = 32`) produced the following confusion matrix (Table 2.5) and classification report (Table 2.6) and an overall accuracy rate of 81%.

	High	Normal	Low
High	740	8573	341
Normal	5424	143460	5083
Low	581	5083	825

Table 2.5

	Precision	Recall	F1	Support
High	.11	.08	.09	9654
Normal	.86	.03	.89	153967
Low	.13	.05	.07	16348

Table 2.6

Gradient Boosting

Gradient Boosting (GB) algorithms using `scikit-learn ensemble GradientBoostingClassifier` were tested with default and hypertuned parameters. Hypertuned parameters were also run using a for-loop.

[2] (6 n_estimators * 5 min_samples_split * 4 min_samples_leaf * 6 max_depth)

Following disappointing RF results, GB was considered in hopes that the model's punishing mechanism might create clearer boundaries for each Tariff class label to be appropriately predicted.

48[3] GB models were run, exploring three parameters:

1. `max_depth`, the nodes per tree
    - 1, 3, 5, 7, 9, 11
2. `n_estimators`, the number of boosting stages to perform
    - 5, 50, 250, 500
3. `learning_rate`, the speed at which the curve is descended
    - 0.01
    - 0.1
    - faster rates, which were discarded due to routinely poor performance

It may have been desirable to explore more potential parameters, but GB performance was exceptionally slow. Increased `learning_rates` resulted in model accuracies well below 50% and were thus discarded in later runs.

Similar to the other models, GB did not perform notably. Hypertuned GB models generally produced Normal Recall scores close to 1.0.

The best GB model (`max_depth= 7, n_estimators = 500, learning_rate = .01`) produced the following confusion matrix (Table 2.7) and classification report (Table 2.8) and an overall accuracy rate of 83%.

	High	Normal	Low
High	39	9275	340
Normal	1032	150230	2705
Low	201	15803	344

Table 2.7

	Precision	Recall	F1	Support
High	.03	.00	.01	9654
Normal	.86	.98	.91	153967
Low	.10	.02	.03	16348

Table 2.8

[3] (4 n_estimators * 2 learning_rates * 6 max_depths)

Support Vector Machine, *maybe in the future?*

Support Vector Machine (SVM) algorithms using `scikit-learn svm SVC` were tested with default parameters, but were unbearably slow. Resulting confusion matrices with DOW Epoch stratification resulted in all class labels as "Normal" (Table 2.9).

	High	Normal	Low
High	0	9654	0
Normal	0	153967	0
Low	0	16348	0

Table 2.9

Conceptually, `SVC` hypertuning requires:
Either `kernel = 'rbf'` or `kernel = Linear`
with `C = [1, 10, 100, 1000]` and `gamma = [1e-3, 1e-4]` when the kernel is rbf. Additionally, there may be some future justification to test `scikit-learn svm SVCLinear` instead, which would require scalar transformation rather than normalization. This was simply not tested due to time constraints and because SVM was not covered in class.

Testing on New Data

Despite poor model performance, I wanted to explore the concept of applying a model on a new dataset. For this process, I selected three additional blocks: 1) Block 79: Adversity - 31 std 19 ToU; 2) Block 37: Affluent - 25 std, 25 ToU; and 3) Block 63: Comfortable - 27 std, 23 ToU to create a new dataset, `ml_testing`.

This data was identically wrangled, sampled, reduced, encoded, and scaled (however, it was not split into train:test datasets). The prepared `ml_testing` dataset was subsequently modelled against the highest performing RF. However, the algorithm of choice was ultimately irrelevant due to poor model performance and could easily be swapped for another model (i.e. the RF was stored within a variable: `best_rf`, which could be switched to `best_knn` to run KNN instead).

Using the `scikit-learn predict` function, the `ml_testing` dataset's class label predictions were compared to its actual class labels. The resulting confusion matrix is unsurprising, due to poor model performance initially, but this process was worthwhile from a personal learning objective.

The results of the highest performing RF produced the following confusion matrix (Table 2.10) and classification report (Table 2.11), and an overall accuracy rate of 82%.

	High	Normal	Low
High	6287	21478	1446
Normal	16579	510249	29877
Low	1801	45604	13884

Table 2.10

	Precision	Recall	F1	Support
High	.25	.22	.23	29211
Normal	.88	.92	.90	556705
Low	.31	.23	.26	45604

Table 2.11

Discussion

Unfortunately, despite many different data preparations, the selected algorithms did not yield any ideal results. Ideal results needed to appropriately classify High and Low Tariff class labels with enough Precision and Recall that the model had moved beyond simply classifying everything as Normal. A lower overall Accuracy rate would thus be deemed acceptable if generally drawn away from the Normal class.

Judging model quality was difficult after hypertuning because Accuracy score is generally a good guideline for picking the best parameter set. My judgements generally placed greater emphasis on Recall for High and Low class labels than on other metrics, such as Precision or F1 scores, because it indicates that the model produces more false negatives than false positives. While both scores are ideal, retrieving actual High and Low class labels from the dataset suggests that Tariff classification resulted in actual energy usage change (with respect to the other variables).

The best model selected was a hypertuned Random Forest Classifier (`n_estimators = 5, min_samples_split = 2, min_samples_leaf = 1, max_depth = 32`) with a marginally successful classification against the `ml_testing` dataset.

One accidental experiment[4] placed Sunday in both train and test datasets (and discarded Tuesday from both datasets) and used the `ml_testing` dataset (blocks 37, 63, and 79) to train the RF. This resulted in spectacular results, as one might expect. However, it was incredibly curious when this RF was predicted against the original blocks (16, 64, 78). This resulted in a

[4] Unintentionally created by a syntactical mistake

model scoring 91% with excellent distribution of class labels across the resultant confusion matrix. It should be noted that this was not selected as a model outcome for this project, as I am unable to adequately describe the model. Considering that data was highly stratified in its block selection and then sampled, this may ultimately represent an interesting exploration, especially if tested against the larger dataset.

Limitations

As mentioned, this dataset was exceptionally large. Even with a powerful local machine, the speed at which many algorithms ran was well over an hour for each hypertuning for-loop (the entire script for .25% of three blocks, totaling less than 1% of the entire possible dataset, runs for roughly eight hours). In general, this meant rerunning different scenarios was difficult and did not encourage rethinking different feature combinations (nonetheless, several of these were tested fairly rigorously). A bigger dataset may have helped to generate useful results—I would have preferred to avoid sampling outright and believe that having an imbalance of pricing schemes was inherently problematic. A better (but larger) subset may have been two blocks per ACORN group (totalling 50 ToU and 50 std households, although this combination would not have been possible for the Adversity ACORN group) to ensure each variable was equally represented.

Additionally, the resulting algorithms were significantly affected by the Tariff class label imbalance. It may have also been beneficial to use `scikit-learn metrics balanced_accuracy_score` within for-loops, but this was not realized until models could not be rerun before the deadline. Balance accuracy scores assess the accuracy rate as `(True Positive Rate + True Negative Rate)/Total Classes`. For example, a balanced accuracy score on the SVM confusion matrix (Table 2.9) would have produced a model accuracy score of 33% rather than 85.5%.

This project, like all machine learning projects, would benefit from additional Epoch splitting, feature engineering, and more robust algorithm hypertuning for the myriad of variables possible.

Self-Assessment

A: I was responsible for the work in the question one section, though Thomas and I coordinated on some methods, such as coming up with a train/test split and the sine/cosine cyclical transformations. I also contributed to the EDA and introduction sections.

Thomas: All of question two is entirely my work with the exception of some reused code from A (Loading: for-loops and file walk) and some adaptations from code segments (e.g. sine/cosine cyclical transformations). I contributed to the ERD and wrote the data description sections. The nature of our questions led to independent work and we instead discussed concepts together and then made similar implementations into our own code.

We both feel we have contributed equally to this project and both made necessary adjustments to account for our third member dropping the class in the sixth week.

Appendices

Code for question one is available at
https://colab.research.google.com/drive/1G8yaUhW7QTcf4CL1x-pFuLj147Z7V0_L?usp=sharing

Code for question two has been uploaded as an .ipynb to quercus. This notebook has been documented explaining this process with similar depth and mirrors the headings within this document for convenience. Model scores have been uploaded as an .xlsx to quercus. This is intended to allow review of hypertuning parameters without having to rerun the notebook, due to the hours required to compile.

References

Data References:

https://www.kaggle.com/jeanmidev/smart-meters-in-london
https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households
https://acorn.caci.co.uk/downloads/Acorn-User-guide.pdf

Methods References:

http://blog.davidkaleko.com/feature-engineering-cyclical-features.html