Machine Learning Engineer Nanodegree
Capstone Project Report

Salvador Joel Núñez Gastélum
May 2, 2018

Table of Contents

I. Definition

Project Overview

Advancements in the electric automobile industry are contributing to the disruption of the energy industry. Electric utilities are increasingly interested in identifying the load from electric vehicles (EVs) in order to develop a better-informed investment strategy and create incentives for residential customers to charge their EVs during certain times in the day. This project analyzes the electricity consumption of houses to identify a) if there is an electric vehicle (EV) charging at each house and b) during which 30-minute time intervals the EV is charging at the house.

A data set consisting of 1,590 houses has been downloaded from GridCure's website[1], containing labels to which houses are charging their EVs and when. The website also contains a separate unlabeled test dataset with an additional 699 houses. The goal of this project is to create a series of data transformations and machine learning models that correctly predict (a) which houses have EVs and (b) when the EVs are charging at each house. The performance of the models is benchmarked against a validation data set separated from the original labeled data set. Finally, these transformations and models are used to submit predictions for the 699 test houses.

Problem Statement

This project will be approached as a supervised learning classification problem. The first strategy used to solve this project is to find ways to decrease the imbalance of the classes. To help with this, first houses that have EVs are predicted. Then, using only that subset, time intervals when EVs are charging are predicted. By doing so, this will increase the prevalence of a positive label, making the models easier to train.

The second strategy used is to augment the raw data with implicit temporal information. Adding cyclical temporal features in the data such as the day and the time of day will enhance the data set with much more information.

Using the daily cycles in energy consumption, daily consumption archetypes can be defined. Companies like Opower Inc, have developed load curve archetypes[2] by analyzing the cyclical daily energy load curves through unsupervised learning techniques like K-Means clustering. Load curve archetypes are applied here with the objective of creating clusters that effectively discriminate between EV houses and non-EV houses. K-Nearest Neighbors is subsequently used

[1] Optional Predictive Modeling Challenge. GridCure, https://www.gridcure.com/contact/

[2] Fischer, Barry. "We plotted 812,000 energy usage curves on top of each other. This is the powerful insight we discovered." *Opower, Inc.* October 13, 2014. https://blogs.oracle.com/utilities/load-curve-archetypes

to assign a cluster to the validation data set, since these data were not included in the original k-means clustering.

Finally, a Gradient Boosting Decision Tree (GBDT) classifier is used to classify the houses and the time intervals. Other models were also attempted, but these were discarded since they demonstrated lower performance. The goal will be for the models to outperform the benchmark metric, the F_1 score, for both house and time interval classification.

Metrics

The performance of the models developed for this project will be evaluated based on the F_1 score of the trained model on the validation dataset. The F_1 score is a better evaluation metric than accuracy when considering a model trained on uneven class distribution. The F_1 score is calculated as the harmonic mean of precision and recall.

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

High recall attempts to minimize false negatives, whereas precision tries to minimize false positives. Accuracy works best if false positives and false negatives have similar cost. In this case, a false negative (incorrectly predicting a house or time interval is not charging an EV) is much more expensive than a false positive (incorrectly predicting that a house or time interval is charging an EV when it isn't).

II. Analysis

Data Exploration

The raw data consists of 1590 rows and 2881 columns. It contains one categorical variable for the 1590 “House IDs” and 2880 continuous variables consisting of two months of energy (kWh) data taken at 30 minute intervals (i.e. 2x24x60 + 1 =2,881).

This is a sample of the raw training data:

	House ID	Interval_1	Interval_2	Interval_3	Interval_4	Interval_5	Interval_6	Interval_7	Interval_8	Interval_9	...
0	11655099	0.950	0.826	0.361	0.238	0.342	0.233000	0.351000	0.194000	0.292000	...
1	11633257	0.353	0.327	0.358	0.292	0.285	0.304000	0.361000	0.342000	0.355000	...
2	11651552	0.150	0.181	0.150	0.150	0.131	0.125000	0.088000	0.106000	0.094000	...
3	11636092	2.088	2.075	2.121	2.098	2.046	2.081000	1.847000	0.420000	0.399000	...
4	11647239	1.416	1.250	1.270	1.258	1.239	1.753105	4.609256	4.619256	4.075151	...

The raw labels for this data set similarly contains 1590 rows and 2881, but the continuous variables become Boolean variables indicating whether or not the house is charging and EV during that time interval.

This is a sample of the raw labels:

	House ID	Interval_1	Interval_2	Interval_3	Interval_4	Interval_5	Interval_6	Interval_7	Interval_8	Interval_9	...
0	11655099	0	0	0	0	0	0	0	0	0	...
1	11633257	0	0	0	0	0	0	0	0	0	...
2	11651552	0	0	0	0	0	0	0	0	0	...
3	11636092	0	0	0	0	0	0	0	0	0	...
4	11647239	0	0	0	0	0	1	1	1	1	...

While 31% of the houses in the training dataset are labeled at charging during at least 1 time interval, that corresponds to only 2% of the total 30 minute time intervals or data points yielding a positive label. Training a model to accurately predict a positive label when it only occurs 2% of the time in the training data set will be very difficult. To help with this, first houses that have EVs are predicted. Then, using only that subset, time intervals when EVs are charging are predicted. By doing so, the prevalence of a positive label is increased from 2% to 8%. This four-fold increase in a positive label will make the training much easier.

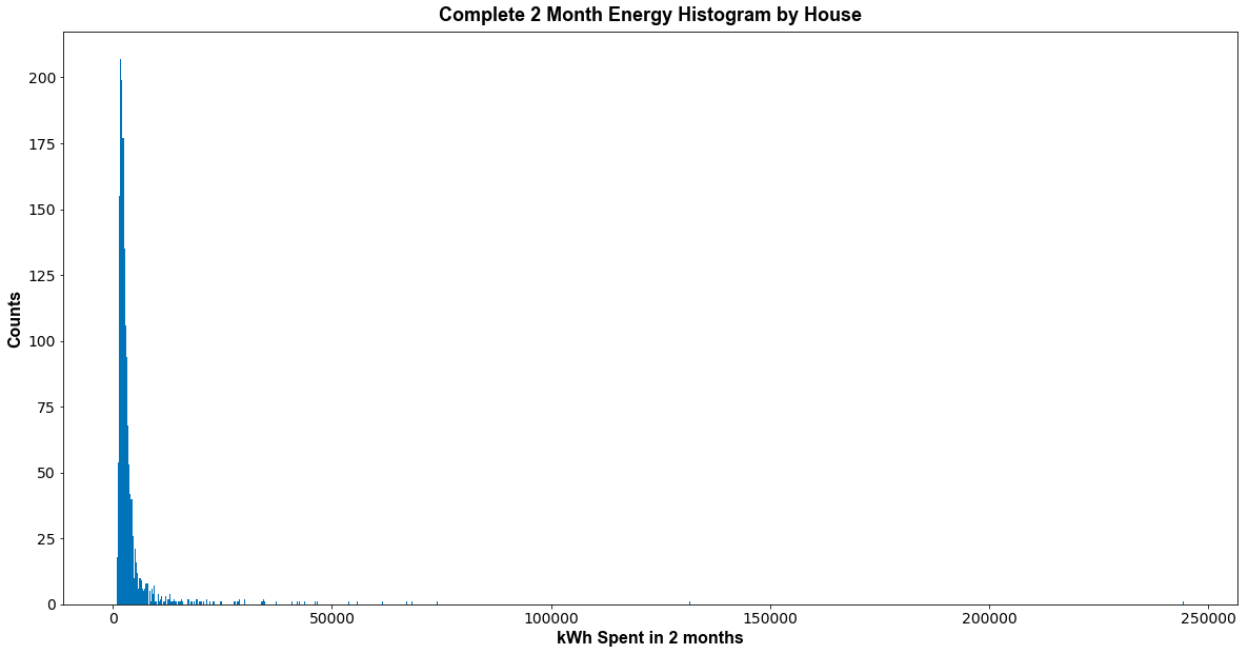
There are also 60 24-hour cycles in the data, but those cycles are not represented in the dataset. Therefore, the day and the time of day corresponding to each interval are explicitly added to cyclical features in the data.

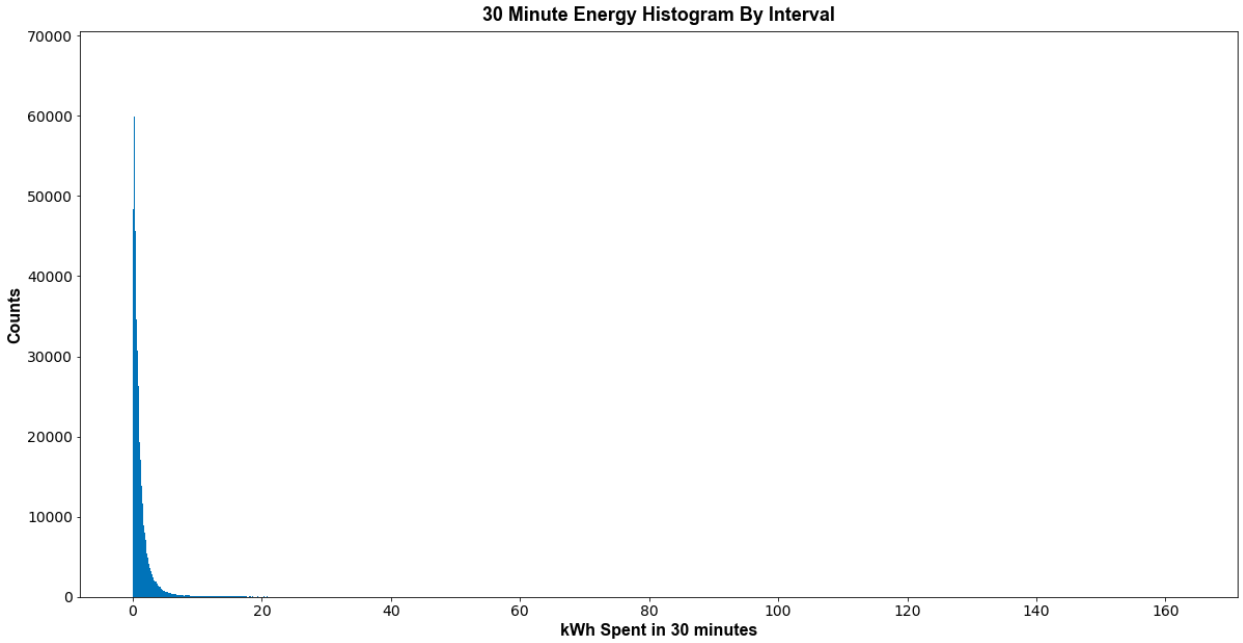
Furthermore, the “wide” data format for the data set does not conform to the traditional format for scikit learn with a matrix **X** and a target vector **y** corresponding to exactly 1

prediction per row. There are also missing data points in the data set, 4 houses are missing energy readings for anywhere between 48 and 144 intervals.

Exploratory Visualizations

Preliminary exploration of the raw dataset shows a long tail distribution of the energy consumption of the 30-minute interval, as well as the distribution of energy consumption for houses over the entire 2 months. This indicates that there may be some outliers in the data, but the data doesn't seem to be multimodal.





Algorithms and Techniques

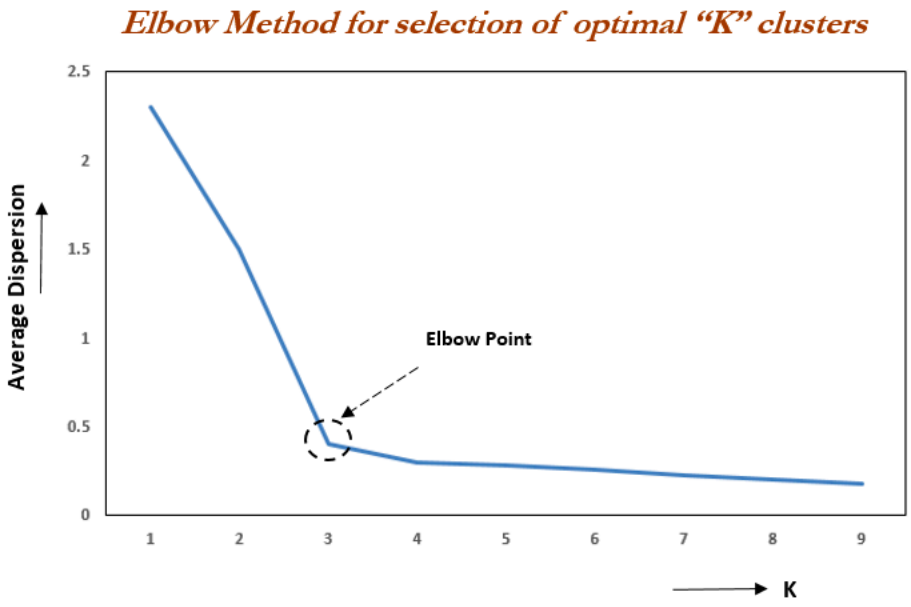
K-Means Clustering

The hypothesis is that there are hidden clusters in the data that can help identify EV houses and EV time intervals, which can be revealed through unsupervised learning techniques. It is further hypothesized that the shape of the daily load curves for each house will be an important feature in classifying EV houses and may even be helpful in classifying time intervals as well. This means that the clustering will be performed on 48-dimensional data corresponding to each of the half hour intervals. The most popular clustering techniques are mixture models, k-means, and hierarchical clustering. Not only is k-means the most popular technique, but these data does not seem to have any hidden hierarchy and its dimensionality is probably too high for a Gaussian mixture model. Moreover, Opower, an industry leader in identifying energy load curve archetypes[3], had successfully used K-Means clustering as an unsupervised learning technique to determine these archetypes or clusters. Therefore, K-Means clustering was used as the unsupervised learning model.

The data will be clustered on the percent of average daily consumption during each interval in order to characterize the shape of the load curve, irrespective of the magnitude of the load. By iterating over cluster assignment and centroid update steps, K-Means provides a simple clustering algorithm. The trickier step can often be selecting an optimal value of k. One

[3] Fischer, Barry. "We plotted 812,000 energy usage curves on top of each other. This is the powerful insight we discovered." *Opower, Inc.* October 13, 2014. https://blogs.oracle.com/utilities/load-curve-archetypes

approach is to use the Elbow Method, which graphically depicts the optimal value of k as the that looks like an arm's elbow.



[4]

The Elbow method identifies the point at which the explained variance shows the greatest increase and after which incremental values of k produce diminishing returns. Instead of optimizing for explained variance, this project should optimize for the ability to discriminate EV-houses from non-EV houses. Therefore, the Elbow method is inadequate. Instead, a custom algorithm is developed.

K- Nearest Neighbors

A validation data set is separated from the original file and is not exposed to any training or optimization. However, when the trained model is tested on the validation data, the training data set and the validation data set must have the same features. In this case, the training data set will have a cluster assignment obtained through k-means whereas the validation data set will not.

A supervised learning technique can be used to predict the cluster assignment for new data. Since, k-means assigns clusters by calculating the Euclidian distance to each centroid, a supervised learning technique that also uses Euclidian distance would be a sound choice. One of the simplest algorithms to use, which also has a similar name to K-Means, is K-Nearest

[4] Dangeti, Pratap. Statistics for Machine Learning. O'Reilly. https://www.safaribooksonline.com/library/view/statistics-for-machine/9781788295758/c71ea970-0f3c-4973-8d3a-b09a7a6553c1.xhtml

Neighbors (KNN). KNN is a simple algorithm that classifies a new data point by selecting the most common class amongst the k nearest known data points. It uses Euclidian distance within the feature space in a similar way to K-Means. This makes it a good method for providing the validation data set with a cluster without directly including it in the training process. While K-Nearest Neighbors is a supervised learning algorithm that can be used to identify the closest load curve archetype or "cluster" in 48-dimensional space, it is not a robust enough algorithm to identify EV houses or EV time intervals.

Gradient Boosted Decision Tees

A robust classifier that performs well in predicting imbalanced classes is needed to accurately predict EV houses and EV time intervals. Several classifiers were originally attempted with the default parameters. Based on the F1-Score on the validation data set, Gradient Boosted Decision Trees (GBDT) or the GradientBoostingClassifier was the best performing classifier without any fine tuning. This is true for both the house predictions and the time interval predictions. This confirms the notion that GBDT models are often a good choice[5] when training on imbalanced datasets.

A decision tree is built from recursive partitioning of the dataset in order to yield relatively uniform or homogenous groups at the end of the tree in the leaf nodes. Ensemble learners use several learners together. Boosting is an ensemble method where each new learner (in this case a decision tree) that is added to the ensemble is forced to focus on misclassified samples in the training set from previous learners. At a high level, gradient boosting is a combination of gradient descent and boosting, where the shortcomings of the additive models are identified as gradients. While it can be a very good algorithm, especially for imbalanced data sets, it may also be too sensitive to faulty labeled training data. It also tends to have a larger memory footprint than other classifiers. To start, the default parameters will be used. These may have to be tweaked to optimize the validation F_1 score, and this project will focus on fine-tuning the learning rate and the maximum depth.

Benchmark

A naïve predictor is used to benchmark both the house classification and the time classification. For the house classification, a naïve predictor that assumes that there are no EV owners in the dataset did not make too much sense because this benchmark yields 0.00.

[5] 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset. https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

In fact, this resulted in the following error:

```
/Users/salvadornunez/anaconda/envs/ev-capstone/lib/python3.6/site-packages/sklearn/metrics/cl
assification.py:1113: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due
to no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Therefore, two more naïve predictors were used: (a) a naïve predictor that sequentially alternates between predicting 0 and 1, and; (b) naïve predictor that predicts that assumes that all houses are EV houses (all 1). These scores were 0.367 and 0.487, respectively.

Similarly, for the time interval classification, the corresponding F_1 benchmark scores are 0.000, 0.048, 0.050 for all 0, alternating 01, and all 1, respectively.

III. Methodology

Data Preprocessing

All training data set and the training labels are combined and pivoted to a “long” format. Then, the implicit temporal dimensions are explicitly added with the assumption that sequence in the interval column names can be translated to 60 cycles of 48 30-minute intervals.

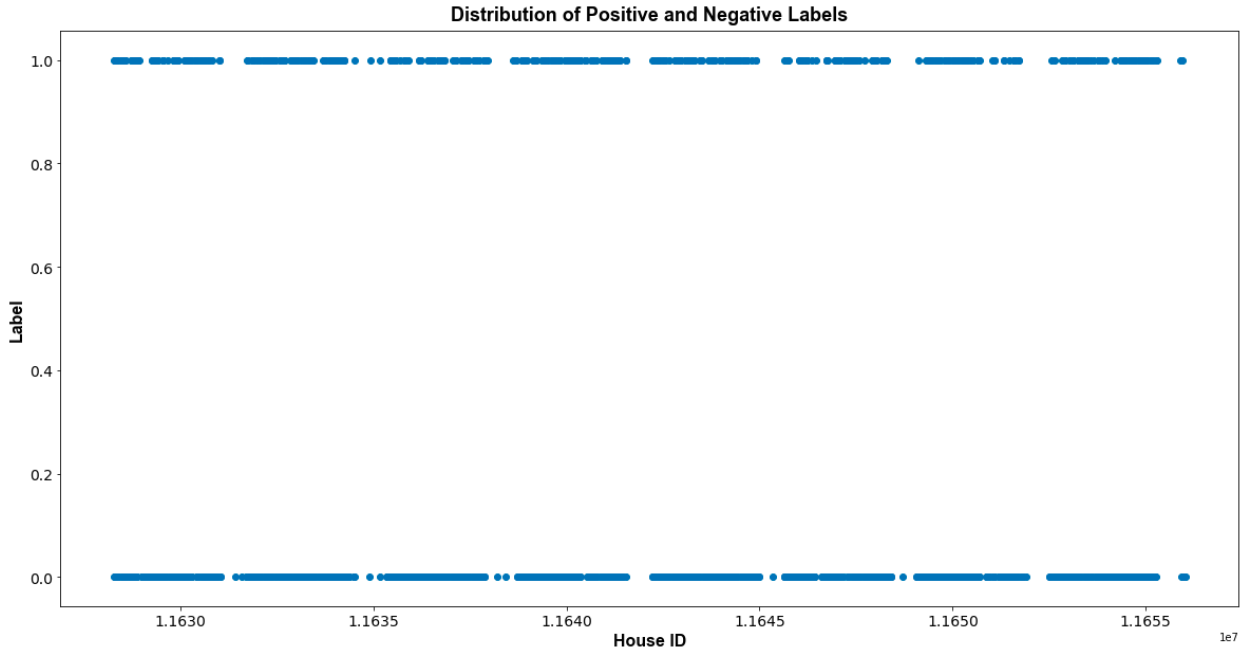
The first benefit of this temporal dimension is that it facilitates better data imputation for the missing values identified in the data. The missing values for the 4 houses are filled with the average energy value for that house during that time of day.

This results in the following data set without any null values, in the traditional Xy format:

	House ID	Day	Hour	Half Hour	Interval	kWh	Label
0	11628280	1	1	1	1	1.114	0
1	11628280	1	1	2	2	0.845	0
2	11628280	1	2	3	3	0.463	0
3	11628280	1	2	4	4	0.453	0
4	11628280	1	3	5	5	0.610	0

House Processing

The data is later aggregated at the house level and new labels are created at the house level. If a house has **any** time interval positively labeled for charging EVs, it is a positively labeled “EV house”. If not, it is a “non-EV house”. As we can see below, the distribution of labels is relatively uniform and there is no correlation between the House ID value and the House Label.



These house aggregations are further described with the mean and standard deviation of the kWh the house consumes every half hour. The means are also divided by the average total daily consumption to produce a metric for the average *percent of daily energy* spent in that half hour. This last summary statistic, the percent of daily energy spent, is particularly helpful in isolating the load shape of the house, irrespective of the total amount of energy the house may consume. This helps neutralize the effect of outliers with extremely large energy loads. These columns are named with the following convention, where "i" is an interval between 1 and 48, corresponding to all the half hour intervals in a day:

- u_i – mean during the i^{th} interval for that house
- s_i – standard deviation value during the i^{th} interval for that house
- p_i – average percent of the daily consumption during the i^{th} interval for that house

Overall, this results in the following house-level data set:

	House ID	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	...	p_39
0	11628280	1.034950	0.990733	0.904383	0.940583	0.958350	0.940750	0.936317	0.919983	0.894133	...	0.021406
1	11628291	0.742283	0.743917	0.742250	0.740300	0.743651	0.838403	0.771068	0.797265	0.762628	...	0.020703
2	11628301	0.236132	0.273704	0.255804	0.248360	0.191183	0.202845	0.314648	0.314576	0.251421	...	0.063615
3	11628319	0.779583	0.766733	0.660650	0.591733	0.600967	0.578200	0.581083	0.595033	0.615067	...	0.030123
4	11628335	0.299017	0.301467	0.297050	0.286950	0.283717	0.293583	0.790850	0.914083	0.929167	...	0.037180

Before any additional processing, this data set aggregated at the house level is split between training and validation groups, using a test size of 0.25.

Time Processing

In order to classify the time intervals when the EVs are charging, the data set at the "interval" level is also augmented with additional features while keeping the same training and validation split used for house classification.

Only the houses which are positively classified as EV houses are subsequently used to train the classifier at the interval level. A negatively labeled house implies that all the time intervals for that house are not EV-charging intervals. Furthermore, as discussed earlier, this increases the prevalence of positively labeled time intervals in the data set, making it easier to train the classification model.

The features added to the interval-level data set are meant to capture information on each house, as well as information that describes differences in energy consumption at that time for that house compared to other days. Existing models to detect EV ownership, like the one describes by Fischer et al[6]., have used features in the data such as increases/decreases in power load by certain pre-determined amounts (e.g. 1-2 kWh), a particular frequency of such increases/decreases, and temporal spacing between such events.

Therefore, the following features were added to the interval dataset:

a) the difference in energy consumption with respect to each of the clusters defined by the house classification, and;
b) (b) the difference in energy consumption compared to the same house at the same time in the past 7 days.

To compare the house's energy consumption with each cluster, the unit for the former is energy in kWh whereas the latter is the % of daily energy and does not have any units. Thus, to compare them, the percent daily energy use for each cluster in that half hour interval is scaled to the total daily energy consumption for that day. Then, the difference is taken, and the columns are labeled c_i through c_k, corresponding to each of the *k* clusters.

Comparing the energy values for the same house in the past 7 days is more straight forward. This comparison is done for 7 days to capture any weekly cycles hidden within the data. These columns are labeled 1d_diff to 7d_diff. However, the first 7 days for each house will be missing some values. Since the classification models do not accept missing or NA values in the training data, these missing values are filled with zero.

Overall, this results in the following time interval data set:

[6] Fischer, et al. Identifying Electric Vehicle Owners. United States Patent US 9.576,245 B2. United States Patent and Trademark Office. Feb. 21, 2017.

	House ID	Day	Hour	Half Hour	Interval	kWh	Day_kWh	k_9	c1	c2	...	c7	c8	c9	1d_diff	2d_di
0	11628297	1	1	1	1	0.815	53.133	2	0.456112	0.471126	...	1.856814	0.204553	0.521030	0.0	0.
1	11628297	1	1	2	2	0.743	53.133	2	0.439292	0.473900	...	1.829126	0.207639	0.488270	0.0	0.
2	11628297	1	2	3	3	0.832	53.133	2	0.429735	0.473533	...	1.730825	0.207550	0.476077	0.0	0.
3	11628297	1	2	4	4	0.880	53.133	2	0.425891	0.468220	...	1.708864	0.208038	0.465067	0.0	0.
4	11628297	1	3	5	5	0.909	53.133	2	0.419292	0.485443	...	1.711497	0.207016	0.449702	0.0	0.

Compared to the raw data set that essentially would have only contained **2** columns, a House ID and a kWh value, this data set contains **24** columns, resulting in *significantly* more information that can be used to train a classification model.

Implementation

House Classification

An additional column is added to the separated training data set: a cluster assignment. The data set is clustered through k-means using only the columns describing the percent of daily consumption during the i^{th} interval (i.e. all the p_i's). Multiple values of k were used in the k-means clustering.

The ability for K-Means to discriminate between EV houses and non-EV houses is defined as the ratio comparing the cluster with the highest percent of positive labels with the cluster with the lowest percent of negative labels. The algorithm then performs k-means clustering with the same random seed (42) for each value of k and chooses the value of k that maximizes the following ratio:

$$Ratio = \frac{max(\%\ positive\ labels\ in\ cluster)}{min(\%\ positive\ labels\ in\ cluster)}$$

The "optimal" k was selected based on the k that would maximize the ratio comparing the cluster with the highest percent of positive labels with the cluster with the lowest percent of negative labels, for each set of clusters produced by k ranging from 2 to 10.

K	Ratio	Max %	Min %
2	1.446	32.3%	22.3%
3	1.509	32.6%	21.6%
4	1.920	40.5%	21.1%
5	2.062	46.5%	22.6%
6	2.564	48.1%	18.8%
7	3.322	57.0%	17.1%
8	2.908	54.3%	18.7%

9	***5.069***	***61.4%***	***12.1%***
10	2.965	58.3%	19.7%

In this case, the optimal "k" was 9 and the column "k_9" was added to the aforementioned dataset, assigning a cluster from 0 to 8 to each house (since python starts its indices at 0).

...	p_40	p_41	p_42	p_43	p_44	p_45	p_46	p_47	p_48	k_9
...	0.035223	0.033186	0.030959	0.029389	0.027297	0.016718	0.012797	0.012377	0.011757	5
...	0.021319	0.021060	0.021830	0.020230	0.019202	0.015799	0.013511	0.011439	0.010201	0
...	0.051914	0.053570	0.045476	0.019140	0.014406	0.011500	0.010308	0.008121	0.006240	8
...	0.023128	0.025465	0.026261	0.030501	0.032329	0.027500	0.027435	0.029378	0.029757	3
...	0.053381	0.053078	0.046755	0.038444	0.024479	0.015984	0.010947	0.008577	0.007237	8

This modeling exercise also yield the centers for each of the 9 clusters in 48 dimensions, corresponding to each of the 48 intervals. As mentioned in the Algorithms and Techniques section, several classifiers were attempted to identify and classify EV houses. Each model is tested on the validation data. A K-Nearest Neighbors (KNN) algorithm is trained on the training data set and used to predict a cluster for the validation data set. This is because the validation data set was excluded from the k-means exercise in order keep the validation data set unexposed to any supervised or unsupervised training technique. A KNN model with k = 5 is fitted to the training data set on the mean (u), standard deviation (s), and percentage (p) columns to predict one of the 9 clusters. The fitted model is then used to predict a cluster for the validation data set. Once column "k_9" (for the k-means clusters) is appended to the validation dataset using KNN, each model is used to predict EV houses in the validation data set.

Once the house-level data set had all the features, several models were evaluated as classifiers. By using the default parameters of the GradientBoostingClassifier object, using the GBDT model, clearly performed best in terms of the validation F1-Score.

House Classification Model Summary

Model	Training Time (sec)	Training F1-Score	Validation F1-Score
GradientBoostingClassifier	**1.280**	**0.981**	**0.753**
LogisticRegression	0.016	0.000	0.000
SGDClassifier	0.012	0.000	0.000
SVC	0.240	0.813	0.092
GaussianNB	0.004	0.464	0.491
MLPClassifier	0.025	0.461	0.487

Time Classification

A separate set of models are trained on the dataset at the time interval level, also with the default parameters. However, only the time intervals from houses that were predicted as having EV are passed into the model. Only during training, the *true* values of the house classifications are used to subset the intervals that are passed into the model for classification. If the house is not charging an EV that implies that all of the time interval records for that house are not charging EV. Once the time intervals are classified for EV-house subset, the remaining records in the training data (all of which are zero) are combined into the data set.

The same data augmentation, classification, and recombination that was performed on the training data set is later performed on the validation data set. This involves the preprocessing steps of creating 24 columns described above. However, in this case the subset of time intervals that are passed into the classification model are based on the *predicted* values, not the true values. This will lower the resulting F1 score but accurately reflects the predictive power of the model, end-to-end, when it is exposed to entirely new data.

A complication arose when training the SVC object, which uses the Support Vector Machine (SVM) model. Despite training the model for 48 hours without interruptions, the training process had not concluded. Since the default SVC classifier did not perform well for house classification and refining a model that takes that long to train would be impractical, training was abandoned for that classifier.

Due to this complication, since it became apparent that training models could take a long time, the pickle package was used for model persistence in order to save and load trained classifiers without having to re-run the entire code. The use of pickle files was applied retroactively to the house classification and was later used for model refinement. Again, by using the default parameters of the GradientBoostingClassifier object, the GBDT model clearly performed best in terms of the validation F1-Score.

Time Interval Classification Model Summary

Model	Training Time (sec)	Training F1-Score	Validation F1-Score
GradientBoostingClassifier	**343.450**	**0.724**	**0.565**
LogisticRegression	2.086	0.000	0.000
SGDClassifier	1.563	0.000	0.000
SVC	abandoned after 48 hours!	NA	NA
GaussianNB	1.009	0.577	0.437
MLPClassifier	40.783	0.000	0.000

Refinement

To refine and improve the results, grid search is used to optimize the parameters. The GradientBoostingClassifier object has over a dozen parameters, but this project focused on 5 of them:

1) loss
    a. The default value is 'deviance'. This is the loss function to be optimized. The deviance refers to logistic regression for classification with probabilistic outputs. Another loss function option is the 'exponential' function which uses the AdaBoost algorithm.
2) learning_rate
    a. The default value is 0.1. The learning rate shrinks the contribution of each tree by the learning rate. It controls the magnitude of the change in the estimates. Lower values would require a higher number of trees to model all the relations and therefore take the algorithm longer to converge to an answer. On the other hand, a large learning rate generally takes less time to converge but may yield a suboptimal answer.
3) n_estimators
    a. The default value is 100. This is the number of sequential trees to be modeled.
4) max_depth
    a. The default value is 3. The maximum depth of the tree. Used to control over-fitting as higher depth will allow the model to learn very specific relations to a particular sample.
5) min_samples_leaf
    a. The default value is 1. This defines the minimum observations required in a terminal node or leaf. It is used control over-fitting. Generally, lower values should be chosen for imbalanced class problems.

Furthermore, grid search was used in combination with shuffle spiting and cross-validation. Using shuffle split and cross validation will already increase the training time for a single set of parameters, and the training cost becomes exacerbated by searching for the optimal parameters in the parameter grid. Using this approach to refine the models led to complications, as this process was *extremely* time consuming.

For the house classification, shuffle split cross validation was used with an F_1 scorer iterating over the 2*3^4 (162) parameter grid. The time classification, however, is a much larger dataset that takes even longer to train. While the house training set consisted of 1192 rows and 146 columns, the time training data set consisted of 1,028,160 rows and 24 columns. Since the size of the training data set is several orders of magnitude greater than the house training data set, it will take even longer to train. The complexity of solving this grid search would quickly increase as the complexity and the number of operations would increase by O(n^2). Therefore, a smaller parameter grid was used to perform grid search: only 3^2=9 combinations. Several

random seeds and several parameter grids were used to evaluate the model's robustness and mitigate any bias in the original selection of the parameters in the grid.

As mentioned before, pickle was used to save and load trained models in order to perform the training process in pieces and saving the progress of this project. This still had its complications. One grid search took over a day to train on a laptop. During this time, the laptop may have had to be occasionally closed for 30 minutes or so, causing the training to sleep until the laptop was opened and the processed resumed. This may have led to slightly inaccurate estimate of the true training for Time Grid Search 6.

IV. Results

Model Evaluation and Validation

House Model Refinement Summary

Grid Search	Seed	Parameter Grid	Best Parameters	Grid Train Time (min)	Best Training CV F1-Score	F1-Score On Validation Data
House Grid Search 1	**42**	**'loss': ['deviance','exponential'] 'learning_rate': [0.09, 0.1, 0.11] 'n_estimators': [90, 100, 110] 'max_depth': [2, 3, 4] 'min_samples_leaf': [1, 2, 3]**	**'loss'='deviance' 'learning_rate'=0.09 'n_estimators'= 100, 'max_depth'= 4 'min_samples_leaf': 3**	**27.386**	**0.754**	**0.770**
House Grid Search 2	94	'loss': ['deviance','exponential'] 'learning_rate': [0.09, 0.1, 0.11] 'n_estimators': [90, 100, 110] 'max_depth': [2, 3, 4] 'min_samples_leaf': [1, 2, 3]	'loss'='deviance' 'learning_rate'=0.09 'n_estimators'= 110, 'max_depth'= 3 'min_samples_leaf': 1	31.020	0.750	0.761
House Grid Search 3	672	'loss': ['deviance','exponential'] 'learning_rate': [0.09, 0.1, 0.11] 'n_estimators': [90, 100, 110] 'max_depth': [2, 3, 4] 'min_samples_leaf': [1, 2, 3]	'loss'='exponential' 'learning_rate'=0.11 'n_estimators'= 110, 'max_depth'= 4 'min_samples_leaf': 3	27.730	0.758	0.760
House Grid Search 4	42	'loss': ['deviance','exponential'] 'learning_rate': [0.08, 0.09, 0.10] 'n_estimators': [90, 100, 110] 'max_depth': [3, 4, 5] 'min_samples_leaf': [2, 3, 4]	'loss'='deviance' 'learning_rate'=0.10 'n_estimators'= 110, 'max_depth'= 5 'min_samples_leaf': 4	40.448	0.756	0.737
House Grid Search 5	42	'loss': ['deviance','exponential'] 'learning_rate': [0.09, 0.10, 0.11] 'n_estimators': [100, 110, 120] 'max_depth': [4, 5, 6] 'min_samples_leaf': [3, 4, 5]	'loss'='deviance' 'learning_rate'=0.10 'n_estimators'= 120, 'max_depth'= 6 'min_samples_leaf': 5	55.117	0.765	0.737
House Grid Search 6	42	'loss': ['deviance','exponential'] 'learning_rate': [0.09, 0.1, 0.11] 'n_estimators': [110, 120, 130] 'max_depth': [5, 6, 7] 'min_samples_leaf': [4, 5, 6]	'loss'=exponential 'learning_rate'=0.11 'n_estimators'= 110, 'max_depth'= 7 'min_samples_leaf': 6	74.719	0.767	0. 736

Overall, the best classifier came from House Grid Search 1. The F_1 score on the validation set for the refined or optimized model on the house data was 0.770, which does a much better job at predicting EV houses than the benchmarks which were 0.000, 0.367, 0.487 for all 0, alternating 01, and all 1, respectively.

The standard deviation for the grid training time, best training cross-validation F1-Score, and F1-Score on the validation data were 18.865, 0.007, and 0.015, respectively. The training time was most sensitive to the parameter grid and random seed. The training cross-validation F1-score was very consistent. The F1-Score on the validation data also showed to be sensitive to the model. The training cross-validation score and the final validation score are close, indicating that the model at the house level is not doing too much overfitting. Interestingly, a couple of House Grid Searches showed the validation F1-Score to be higher than the training cross-validation F1-Score. This may be caused by chance or may be the result of shuffle-splitting.

Time Interval Model Refinement Summary

Grid Search	Seed	Parameter Grid	Best Parameters	Grid Train Time (hours!)	Best Training CV F1-Score	F1-Score On Validation Data
Time Grid Search 1	42	'learning_rate': [0.09, 0.1, 0.11] 'max_depth': [2, 3, 4]	'learning_rate'=0.11 'max_depth'= 4	9.560	0.742	0.586
Time Grid Search 2	266	'learning_rate': [0.09, 0.1, 0.11] 'max_depth': [2, 3, 4]	'learning_rate'=0.11 'max_depth'= 4	9.768	0.742	0.586
Time Grid Search 3	737	'learning_rate': [0.09, 0.1, 0.11] 'max_depth': [2, 3, 4]	'learning_rate'=0.11 'max_depth'= 4	9.540	0.742	0. 586
Time Grid Search 4	42	'learning_rate': [0.10, 0.11, 0.12] 'max_depth': [3, 4, 5]	'learning_rate'=0.12 'max_depth'= 5	14.537	0.760	0. 589
Time Grid Search 5	**42**	**'learning_rate': [0.11, 0.12, 0.13] 'max_depth': [4, 5, 6]**	**'learning_rate'=0.13 'max_depth'= 6**	**20.276**	**0.775**	**0.592**
Time Grid Search 6	42	'learning_rate': [0.12, 0.13, 0.14] 'max_depth': [5, 6, 7]	'learning_rate'=0.14 'max_depth'= 7	30.500	0.787	0.587

Overall, the best classifier came from Time Grid Search 5. The F_1 score on the validation set for the refined or optimized model on the time interval data was 0.592, which does a much better job at predicting EV time intervals than the benchmarks which were 0.000, 0.048, 0.050 for all 0, alternating 01, and all 1, respectively.

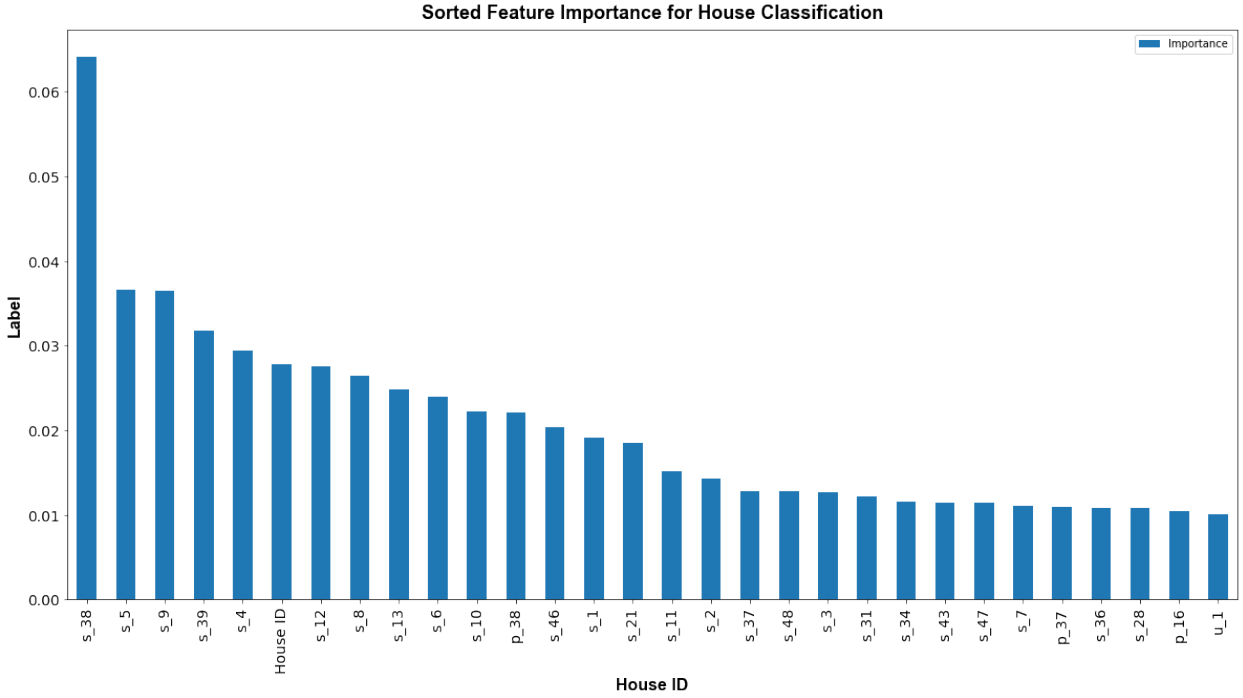
The standard deviation for the grid training time, training cross-validation F1-Score, and F1-Score on the validation data were 8.392, 0.020, and 0.003, respectively. As the learning rate and the max_depth increased, so did the grid training time and the training cross-validation score. However, the final validation score (to which none of the modeling has been exposed) did not change significantly. Despite the fact that Time Grid Search 6 yielded a 0.787 as the training cross-validation score after over 30 hours of training, the corresponding F1-Score on the validation data actually decreased. This is a clear signal of overfitting resulting from the increased model complexity and depth. Therefore, selecting the best classifier from Time Grid Search 5 is a wise choice since it is the best model before the training and validation scores begin to diverge.

Justification

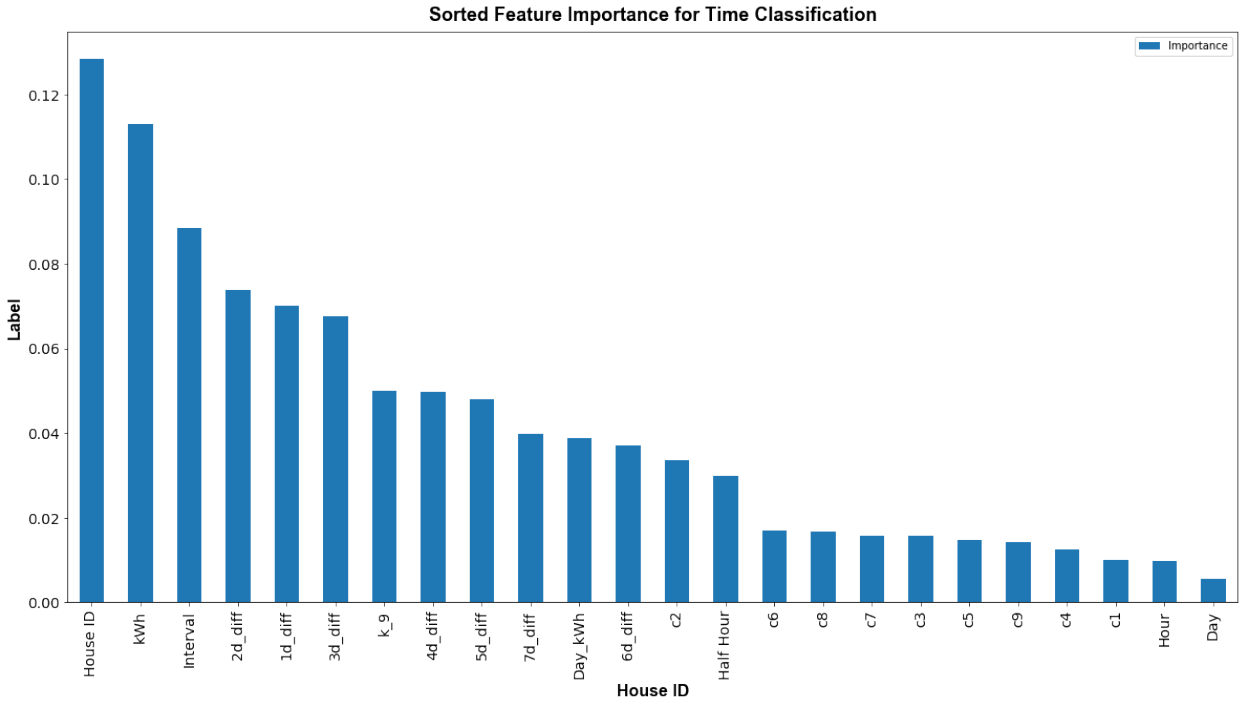
By separating and sequencing the classification of EV houses and then EV time intervals, revealing implicit temporal data, using K-Means clustering, applying K-Nearest Neighbors, and training Gradient Boosting Decision Trees, the time intervals were successfully classified for EV charging. The house classification F_1 score was 0.770, 58% higher than 0.487, the highest house-level benchmark. The time classification F_1 score was 0.592, 12x higher than 0.050, the highest time-level benchmark. Such a dramatic difference between the model's F_1 score and the benchmarks, particularly at the time interval level, make this a satisfactory model in solving this problem.

V Conclusion

Free-Form Visualization



According to the ranked feature importance for the trained house classifier, understanding the variance or standard deviation of the % of daily energy consumed per half hour (interval) were among the most important features. The most important feature by far was the standard deviation at the 38th half-hour, which roughly corresponds to 7PM, which is when residential energy demand peaks. EV owners charging their car after work would most likely begin charging at this time. Other EV owners may program their vehicles to charge in the middle of the night, which may explain why the 4th, 5th and 9th interval would also be highly predictive. These time intervals would occur shortly after midnight. The House ID number itself was not excluded from the model and ranked as being the 6th most important feature. The cluster assignment, column k_9, is not within the top 30 features for the house classification and actually ranked last.



According to the ranked feature importance for the trained time interval classifier, the House ID was the most important feature, followed by the absolute value of KWh, and the Interval. This makes sense since House ID indicates that the power values came from the same house and EV charging occurs at certain step load changes in kWh values. It's surprising that the Interval is so much more important than the Half Hour or the Hour, which are all features that signal the time of day. The relevant time of day information may already be provided by the Interval features. Similarly, while the cluster assignment --labeled k_9-- is relatively important, the distances from each of the 9 cluster centers had less importance (c1-c9). The difference in energy compared to the past 3 days (1d_diff, 2d_diff, and 3d_diff) do seem to have been important though.

Reflection

This project proved more difficult than originally expected. The hypothesis was that data augmentation via the implicit time dimension and identifying load curve archetypes would sufficiently augment the data such that training any model would be relatively straight forward. Initial attempts to train a classifier either took too long or simply predicted that all the time intervals were not intervals with a charging EV. The impact of the class imbalance was underestimated, especially at the time interval level. Once the process was changed to classifying time intervals only from EV houses, the imbalance of classes was mitigated and the training process was much easier. However, the time it took to train all these models on a laptop became a much bigger issue than I anticipated! Using pickle helped to perform the training in chunks, but it still was problematic.

In looking at ranked feature importance we can see that the House ID is an important feature to ensure that time interval records can be related to one another. In this context, the House ID is important for the model to understand that all those readings came from the same location. However, if the model uses the House ID in other ways, it may lead to overfitting. For example, if the House ID value is correlated with the likelihood that it is an EV house, this relationship may not hold for new data sets. This was not the case since the correlation between the House ID and the House Label is -0.055.

Finally, using unsupervised learning, specifically k-means, as a vehicle for data augmentation was an interesting element of the classification problem. However, k-means is stochastic, so it may be possible to obtain slightly better or worse results depending on the random seed. Optimizing for a random seed seems like an obfuscated way of performing subtle overfitting. The method for determining the best value of *k* was also an interesting part of the implementation. The motivation behind it was to determine if there is a mathematical way to describe wanting at least one cluster to be purely non-EV and another cluster to be purely EV. In this project, this was eventually described through the "optimal k ratio". However, this was all based on the hypothesis that clustering the houses was going to be an important feature in classifying whether or not a house is an EV house. As it turns out, the cluster assignment ended up being the least important feature for house classification. Fortunately, it was relatively important for the time interval classification.

Improvement

The current model allows inconsistencies between the house and the time classification. Even if the house is classified as an EV house, there is no guarantee that at least 1 time interval for each house will be classified as an EV time interval. A forcing mechanism, could be developed to enforce consistency between these steps.

Using K-Nearest Neighbors with k=5 to assign clusters to the validation data set was somewhat arbitrary and may have added some level of bias. Sensitivity analysis to different values of k could be performed, or perhaps better yet, the validation data can be assigned a cluster directly by calculating the distance to each centroid. Only the centroid re-centering step is what should be avoided since it would include the validation data in the training process.

One of the more obvious and significant improvements would be to better leverage cloud computing services. Performing all this training on a laptop (3.1 GHz Intel Core i7, 16 GB 2133 MHz) is feasible but takes too much time! Training in the cloud could be performed through services such as AWS, floydhub, or paperspace. Another way to solve for the computational

expense would be to use Beam search or RandomizedSearchCV instead of GridSearchCV to improve the high parameter tuning for the GradientBoostingClassifier.

Although the energy consumption (kWh) was the most important feature in classifying the time intervals, step changes in power are also very important and these were not explicitly added as features to this data set. After all, the size of EV loads are relatively consistent and power delivery rates typically range from 1.4 kW to 7.7kW. The exact load acceptance rate depends on the amperage that the EV manages and voltage from the outlet which is typically either 120V and 240V. Acceptance rates for the most common EV vehicles can be found online and the most common energy load sizes could be codified as features for the decision tree.

Finally, using GBDT to classify highly imbalanced time intervals seems to be a rather unorthodox way of analyzing time series data. This become particularly problematic when using shuffle splitting and cross-validation which samples time intervals independently instead of as a part of a larger time series. Appending engineered time series features to each time interval record was meant to mitigate this issue. Alternatively, a completely different modelling approach could be pursued which disaggregates time series through harmonic feature analysis and multiple-class support vector machines[7] or using hidden Markov modeling (HMM) and residual analysis.[8] However, these alternative modeling approaches are advanced topics outside of the scope of Udacity's nanodegree program.

[7] Jiang et al. "An Approach of Household Power Appliance Monitoring Based on Machine Learning". Intelligent Computation Technology and Automation (ICICTA) – IEEE. Jan 14,2012.

[8] Pattem, S. "Unsupervised Disaggregation for Non-intrusive Load Monitoring ". Machine Learning and Applications (ICMLA) –IEEE. Dec 12,2012.