RESIDENTIAL ENERGY APPLIANCE CLASSIFICATION

Jirarote Jirasirikul - 31334679
Mahesh Pandharinath Hase - 31094821
Pichaphop Sunthornjittanon - 31258301

MAY 28, 2021

Abstract

This task aims to develop multiple classifiers for each target appliance that can detect whether target appliances are being used in each time interval as correctly as possible. The report includes exploratory data analysis to better understand the data and feature extraction to get more meaningful variables out of the data. We developed five independent classifiers to detect five appliances i.e. Air Conditioner (AC), Electric Vehicle (EV), Oven, Washing Machine (Wash), Dryer as a result of our exploratory.

Table of Contents

1 Introduction

Load monitoring is a promising technique to provide detailed electricity consumption information and usage of individual appliances in residential buildings. In theory, it could be collected by setting up measurement meters for appliances, however, in practice, this approach can be considered too intrusive.

This challenge aims to develop multiple classifiers for each target appliance that can detect whether target appliances are being used in each time interval as correctly as possible using only load data that was aggregate of the whole residential building. We will develop five independent classifiers to detect five appliances (AC, EV, Oven, Wash, Dryer) being used as accurately as possible and explain the outcomes of the classifiers.

We have been provided with the following datasets:

- **train_data_withlabels.csv** contains 417,720 instances, each of which has 15 columns consist of load value, two time-related variables, seven attributes/features derived from the load using the tsfeatures library, and five appliance labels.
- **test_data_nolabels.csv**: It contains 105,540 instances with all features & no labels.

Initially, we individually perform exploratory data analysis on the given datasets. This gave us a better understanding of the data to conduct a discussion. Each of us has been tasked to applied different algorithms as a baseline model (i.e. XGBoost, Random Forest, SVM, Logistic). Other tasks such as "Data Preprocessing (Jirarote)", "Data Modeling (Pichaphop)", and "Reporting (Mahesh)" have been assigned to each of us as lead and support to take responsibility. These are not subjected to perform such tasks alone, but to keep work progressing and bring the issue into group discussions.

2 Pre-processing and features used

2.1 Data Preprocessing

1. Data quality check has been performed and no missing values in training and testing datasets detected.
2. The training data contains time series information of the load from the meter in a sequential manner (each row represents a record at a specific minute). Subsequently, more features can be extracted i.e. lag, rolling mean, rolling max and rolling variance.
3. A simple check regarding continuity of "hourofday" in testing data mentioned from https://edstem.org/courses/5384/discussion/481603 revealed that testing data has an issue of anomalies as shown in Figure 1.

	load	hourofday	dayofweek	dif	absdif	max	var	entropy	nonlinear	hurst
118	1.378	1	Mon	-0.001	0.001	6.124	1.188513137	0.723010807	0.491220066	0.986880852
119	1.38	1	Mon	0.002	0.002	6.124	1.194384385	0.727525277	0.447320211	0.986619676
120	1.965	18	Mon	0.585	0.585	6.124	1.188462813	0.743644512	0.373256698	0.986225604
121	1.725	18	Mon	-0.24	0.24	6.124	1.177170019	0.754315334	0.272917264	0.985979867
122	1.849	18	Mon	0.124	0.124	6.124	1.165268328	0.752902641	0.507364521	0.985614662

Figure 1: anomalies time series of test data

4. Our group is aware of this discontinuity of time series data and understands that feature extraction is no longer a requirement. However, after additional investigation on the test dataset, these anomalies only occur in a small portion of the test dataset.

5. We consider this as Noise-to-Signal with a very low ratio and believe that it wouldn't affect much on the model performance. Moreover, we reckon that additional information that could gain from extracting more features is worth the trade-off.
6. We exclude attribute X in the dataset as it was a repetition to indices of the data frame.
7. We converted all 5 target attributes to categorical format using as.factor() command, for the classification task.
8. Data has been converted into DMatrix for the XGBoost algorithm before applying the model on the training, validation and testing dataset.

2.2 Data Exploration

Explore Load

1. As we know that "Load" is the Main predictor in our focus, We chose to explore the load first and separately. This should make us understand the relationship between predictor and target.
2. Each target device shows different patterns when the load is plotted along the timeline.

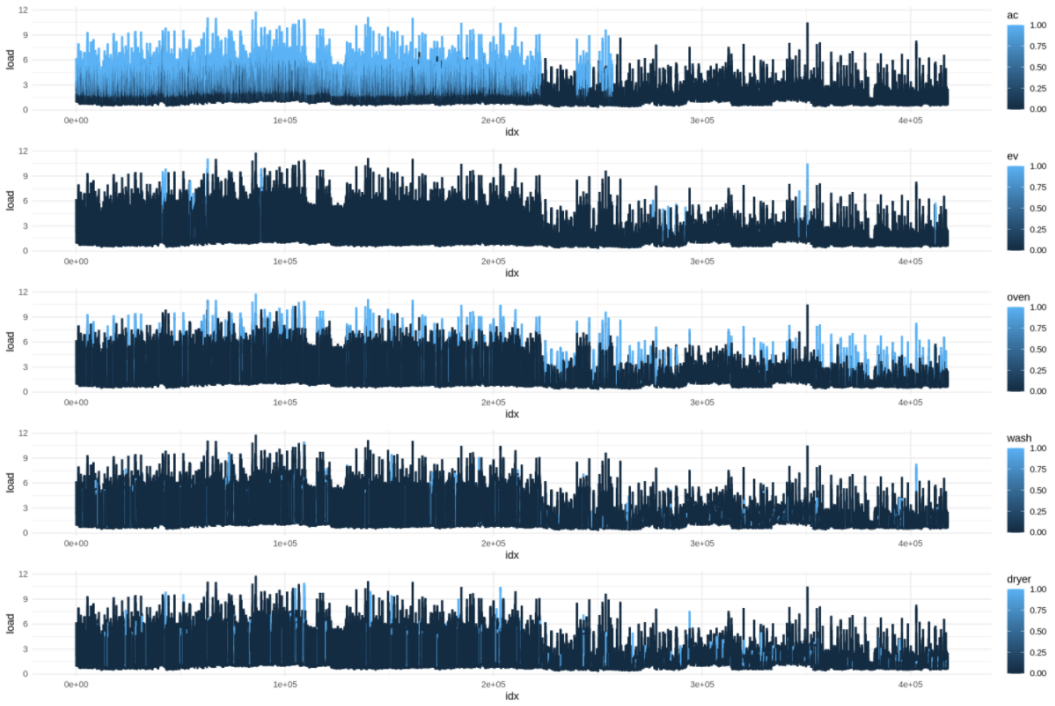

Figure 2: Plot of load-timeline in train dataset

3. Some observation on load-timeline plotted
   a. 'AC' has been turned on during the first half of our data.
   b. 'EV' and 'Oven' do have some patterns that might be able to be extracted.
   c. 'Wash' and 'Dryer' are intermittently ON, but their patterns are similar.
4. Due to an imbalanced training target labeled dataset, this shows the necessity of stratified sampling to create validation dataset. This sampling helped us maintain the proportion of the ON/OFF status between the train and a validation dataset for each target appliance.
5. During EDA, we acknowledge that **load** shows significance as the most important feature. The box plot and density plot of load for ON and OFF status for all target appliances varied significantly.

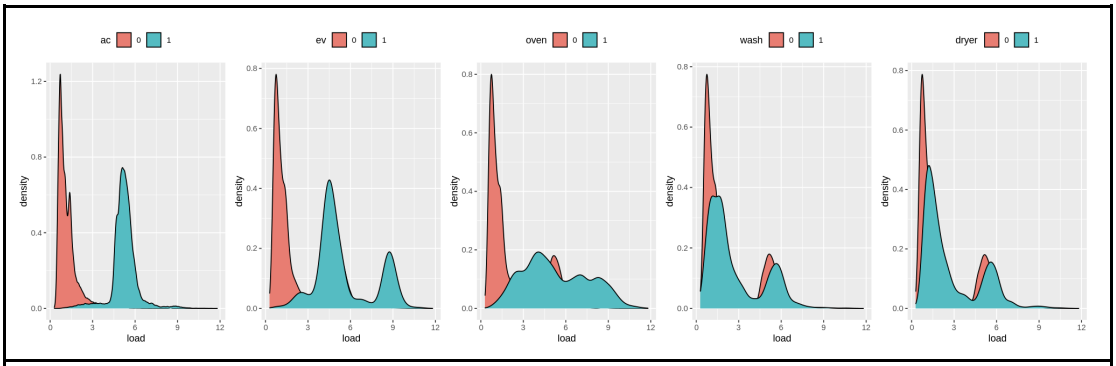

Figure 3: plot load-density of each appliance

Explore Target

6. After exploring the basic structure of the training data using boxplot and histogram, we observed the skewed distribution of ON and OFF status of all target variables. Using an Accuracy matrix will lead to incorrect interpretation of high performance. We decided to go ahead with using the F1 score and AUC metric that considers precision and recall instead.

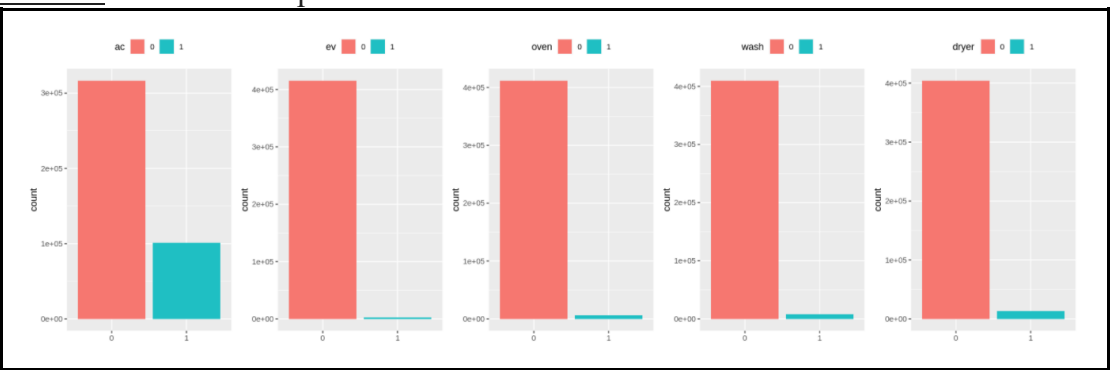

Figure 4: frequency boxplot of value in each appliance within train dataset

Explore Predictors - Qualitative

7. We further took into consideration the seasonal variation of the different target appliances to understand the effect of the hour of day and day of the week. For example, during the day, on average, the peak time for the oven is between 4 PM to 7 PM with Tuesday representing the day with maximum load.

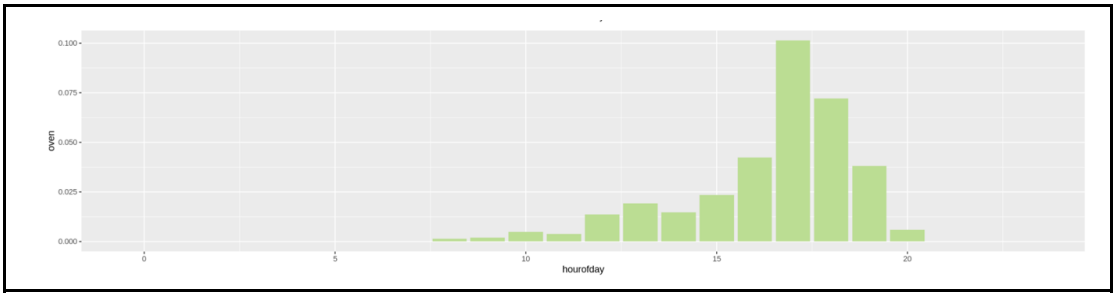

Figure 5: Histogram of the oven is ON at the different time of the day

8. We performed one-hot encoding for all categorical variables. This generates additional features for the 'dayofweek' attribute where each day is now considered as a separate attribute e.g. 'dayofweek_mon'

Explore Predictors - Quantitative

9. We perform boxplot and histogram on quantitative predictors. We know that all predictors will have positive correlation with load as they are generated by feature extraction from 'load' predictors.
10. We understand the behaviour of `max`, `diff`, `absdiff`, `var`, `entropy`, `nonlinear` and `hurst`. We decided from the relationship of the 'load' feature with each target that more features extracted from the time series can capture more aspects of its relationship.

3 Models

After pre-processing and sampling of the training dataset, we tried four different baseline models such as Logistic regression, Support Vector Machine, Random Forest, and Extreme Gradient Boosting on all target variables to analyze their performance.

3.1 Model 1: Logistic Regression

Basic Idea:
We started with the implementation of the basic model for the classification task which is Logistic Regression since it models the probability that the target variable belongs to a particular category.
Assumption:
We assumed that the true decision boundaries for most of the target variables are linear. In this case, the logistic regression approach will tend to perform well.
How the model works:
We loaded the train and validation set for each target variable. Then, we applied logistic regression using glm() and using the family as binomial (ON/OFF) status.
Advantages:
- Logistic regression works well with the large sample size data.
- This method works well for a binary (two-level) qualitative response.

Disadvantages:
When the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. (An Introduction to Statistical Learning, 2013)

3.2 Model 2: Support Vector Machine (SVM)

Basic Idea:
Since the results of the logistic regression model were not great, to accommodate non-linear class boundaries, we implemented SVM. SVM uses a decision boundary to separate Support vectors (data points closest to this decision boundary) using kernels.
Assumption:
We assumed that decision boundaries are non-linear.
How the model works:
SVM is a generalization of a simple and intuitive classifier called the maximal margin classifier. In a p-dimensional space, SVM generates multiple hyperplanes with (p-1) dimensions. SVM uses a kernel trick for data transformation so that support vectors will be mapped to a higher dimension. (An Introduction to Statistical Learning, 2013)
Advantages:
- SVM works best when support vectors are easily separable with margins.
- SVM is memory efficient since it uses support vectors (subset) rather than the entire data.

Disadvantages:
SVM requires more training time (1.5 - 2.5 hours) compared to other models depending on the selection of the kernel (linear, radial, polynomial or sigmoid).

3.3 Model 3: Random Forest

Basic Idea:
Since results of both Logistic Regression and SVM were not impressive (except AC), we thought of using an ensemble approach that combines different decision tree models to achieve a better result using bagging technique i.e. Random forest. Ensemble approach works on the principle that the combination of predictions of many models will be more robust than those produced by individual models.
Assumption:
Random Forest is a non-parametric model, which has no formal distributional assumptions saying that this model can handle skewed data.
How the model works:
Each of the trees in the Random Forest learns something new about the dataset and combining this collection into an ensemble makes the random forest much more robust than a single decision tree. Each tree can be built independently of the other tree. Within each level of the tree, we can parallelize the work to find the optimal split.
Each time we make a split, we only consider a random subset of the features. Due to this randomness, each tree is not too deep i.e. not with an increased complexity to avoid overfitting.
For a baseline model, we choose the number of trees (ntree) as 50 and the number of variables available for splitting at each tree node (mtry) as 6.
Advantages
- It is a highly flexible and powerful model that has no restriction on data distribution
- Variance and overfitting problems can be reduced because random forest uses ensemble learning techniques called bagging.
- This algorithm is robust to outlier
- Tuning the random forest is easier than boosting algorithm

Disadvantages:
- It is more complex and requires a longer training period than other simple models such as decision tree or logistic regression

3.4 Model 4: Extreme Gradient Boosting (XGBoost)

Basic Idea:
XGBoost is another ensemble approach using a gradient boosting framework. Gradient Boosting is a sequential learning algorithm creating weak learners that learn from previous learning mistakes by giving larger weight on misclassified samples to next predictions. Also, XGBoost is an optimised scalable version of gradient boosting, which is highly efficient, flexible and portable.
Assumption:
XGBoost is a non-parametric model, which has no formal distributional assumptions saying that this model can handle skewed data.
How the model works:
The multiple decision trees are built sequentially by minimizing error from previous models using a gradient descent algorithm. XGBoost implementation allows gradient boosting to process in parallel increasing efficiency and flexibility. For our baseline model, we use early stopping to find the best number of rounds that decision trees were built (stop training models when no improvement on the validation set). We also assign a scale of positive weight suggested by the XGBoost document to deal with imbalanced data in the. Other hyperparameters were set as default.
Advantages
- It is a highly flexible and powerful model that has no restriction on data distribution, especially when dealing with imbalanced data.
- XGBoost has an efficient implementation, which is faster than gradient boosting.

- It is an ensemble algorithm, in which multiple trees are built sequentially by learning from previous tree mistakes. By doing that, it potentially leads to better performance.

Disadvantages

- The model can potentially overfit if parameters are not properly tuned
- There are many hyperparameters to be tuned in XGBoost

3.5 Discussion of model difference(s)

Logistic Reg	Works only with linearly separable data, hence, provide poor predictions.
SVM	Requires more training time compared to other models.
Random Forest	Uses ensemble learning techniques called bagging to avoid overfitting.
XGBoost	Uses boosting approach, which is flexible and powerful.

4 Experiment setups

Before the model development part, experiment setups should be appropriately designed. In our case, we deal with the classification of imbalanced target variables. Only a few records have on-status in each appliance (only 24 % for AC, 0.6 % for EV, 1.4 % for Oven, 1.9 % for Wash and 4.8 % for Dryer). This required the specialised approaches for experiments setup mainly divided into three sections, which are:

4.1 Metric Selection

As discussed above, choosing an evaluation metric is important especially when dealing with imbalanced data because the selected metric will be used for the model selection part. Due to the requirement, the F1 score is selected as an evaluation matrix considering not only accuracy but also precision and recall. These measurements can be obtained from a confusion matrix looking at the proportion of positive identifications that are actually correct (Precision) and the proportion of actual positives that were identified correctly (Recall). As expected, our imbalance dataset usually gives us a high accuracy regardless of any predicted models so using F1 gives us more information about model performance. In our setting, we create a function that generates a confusion matrix and measurements that can be extracted from the table namely accuracy, recall, precision, F1 score and AUC (Example is shown on the right). By doing this, we can analyse how model performance and which part of the models need to be improved.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

```
 ev
[1] "Accuracy :  99.98 %"
[1] "Precision :  98.07 %"
[1] "Recall :  98.92  %"
[1] "F1 Score :  98.49 %"
[1] "AUC :  99.03 %"
           valid.label
pred.label     0     1
         0 83067     9
         1     5   458
```

4.2 Validation approach

In our experiment setup, we split the given training dataset into training and validation data (80/20) using stratified sampling to ensure that both have a similar distribution. The models were selected and evaluated based on performance on the validation dataset. In other words, the model comparison uses the F1 score on the validation set to select the best model as a final predictor.

4.3 Hyperparameter Tuning

In a random forest, we tuned the parameters mtry and ntree. In XGBoost, we tuned several parameters, namely eta, colsample_bylevel, subsample, max_depth, min_child_weight, gamma, nrounds using early stopping and scale_pos_weight for imbalanced classes. By selecting the best parameters, we used iterative

methods to choose each parameter given other parameters constant, which provided the best F1 score in the validation dataset.

After experiment setup, we built baseline models for logistic regression, SVM, random forest and XGBoost, then evaluated their performance using the F1 score on the validation dataset. Overall, random forest and XGBoost performed better than Logistic regression and SVM so later on, we mainly focus on improving those two models by tuning parameters and include more extracted features.

5 Experimental results

As a result of fitting several models, F1 scores are reported in the table below. For AC, every model can perform very well because the load was significantly different when AC was turned on (we learn this from EDA). In other words, every model can mostly capture the AC trend. However, other appliances cannot simply rely on the load feature but need some complex models with several features to capture the pattern. That's why logistic regression and SVM have very low F1 scores, unlike random forest and XGBoost. Moreover, when we add more extracted features (rolling mean, variance and maximum with window size equal to 2-30) in XGBoost, it gives a higher F1 score because there are more features to learn and the model can do the feature selection. In conclusion, XGBoost models with extra features are selected as final models to predict five appliances' status.

Table 1: Model comparison

	Logistic Regression	SVM	Random Forest (RF)	RF (Extra features)	XGBoost	XGBoost (Extra features)
F1 Score of different models on the validation dataset						
Air Conditioner (AC)	96.79 %	96.72 %	98.97 %	99.21 %	98.45 %	**99.44 %**
Electric Vehicle (EV)	14.56 %	0.00 %	96.25 %	94.49 %	98.62 %	**98.61 %**
Oven	34.82 %	0.00 %	87.74 %	87.67 %	90.88 %	**93.79 %**
Washing M/c (Wash)	0.00 %	0.00 %	60.36 %	35.05 %	70.76 %	**81.57 %**
Dryer	0.00 %	0.00 %	88.40 %	77.04 %	88.05 %	**97.09 %**
Other Parameters for comparison						
Interpretability	High	Low	Medium	Medium	Medium	Medium
Training time	Low	High	Medium	Medium	Medium	Medium
Prediction Power	Low	Low	Medium	Medium	**High**	**High**
Memory Efficiency	High	Low	Medium	Medium	Medium	Medium

6 Conclusion

The classifier which delivers the best results (F1 scores) is Extreme Gradient Boosting (XGBoost) due to a boosting approach that allows models to learn slowly by using information from multiple sequential trees. Another advantage is that it can do feature selection identifying important features used for the models shown in figure 6 below. To discuss a few, Load and its derived rolling features contribute a lot to the models especially in predicting AC. Rolling max (window size =2) alone had around 80% importance for AC, while other models used combinations including load and time-related features. An alternative way to calculate this is the use of SHAP values, which also identify the importance of each feature in terms of SHAP values which correlates with the Importance values of different features.

Intuitively, AC patterns can be easily captured by any models using load, while others require more complexity to create accurate models. That's why XGBoost performed very well in this case.

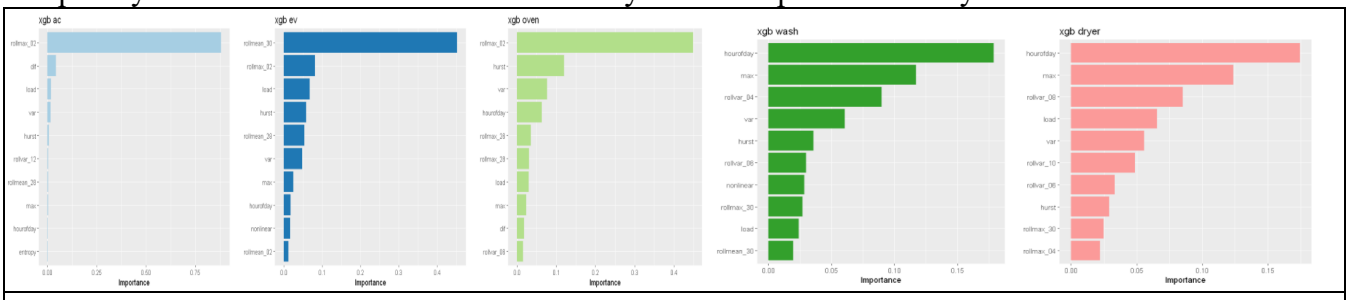

Figure 6: The best set of features used by XGBoost for different target variables

One lesson learned from this assignment is that real world data is not always perfect. We accept real dataset along with anomalies (i.e. broken time series data) and devise a way around to apply theoretical concepts to handle these anomalies. In this case, we used knowledge of noise/signal ratio to handle anomalies in the test data. We observed that since noise (anomalies) in the test data is very low, hence we can still apply time-series analysis by way of imputing the default values at the start where anomalies exist. This also takes care of null values of tsfeatures for the first few values.

We also learned about new techniques such as stratified sampling to handle imbalanced data. In this way, we can verify results of the trained model on the validation dataset more accurately, since stratified sampling divides both training and validation data in proportion to imbalanced data. This also rules out the application of cross-validation techniques which are more complex and time-consuming since each train and validation dataset is unique for each target variable as per the distribution of ON/OFF status.

In future, we would like to explore more about parameter tuning for different baseline models. We would like to explore a methodical approach for parameter tuning rather than trying different values to validate model performance on the validation dataset. For example, in a random forest, which parameter should we tune first? What is the optimum size of the number of trees for a given model for tuning purpose? etc. We would also like to explore more about the interpretability of the ensemble models such as XGBoost.

References

- Brownlee, J. (2020, March 9). Step-By-Step Framework for Imbalanced Classification Projects. Machine Learning Mastery. https://machinelearningmastery.com/framework-for-imbalanced-classification-projects/
- Notes on Parameter Tuning — xgboost 1.5.0-SNAPSHOT documentation. (2020). https://xgboost.readthedocs.io/en/latest/tutorials/param_tuning.html