RESIDENTIAL ENERGY APPLIANCE CLASSIFICATION

MILINDA ABAYAWARDANA

PRASHANT ARORA

VISHAL PATTABIRAMAN

MAY 30, 2021

ABSTRACT

Smart metre readings data for five electrical appliances were assessed to determine the model that provides the best F1 Score. Two metrics, ROC and accuracy, were used to evaluate the model performance before selecting three models xgbTree, GLM and GBM to continue our analysis.

The data presented were time-series in nature with a class imbalance. Caret's train control method time-slice was used to fit models respecting the time series nature of the data. Due to performance issues, we were unable to accommodate the full dataset. The issue of class imbalance was addressed successfully using the ROSE packages 'ovun' function. However, we were unable to address both attributes of the full dataset.

Multiple model runs were performed using smaller balanced datasets and the best combination of appliance and model were found.

TABLE OF CONTENTS

INTRODUCTION

In this project we are looking at the Non-intrusive load monitoring data for five appliances, these include, oven, air conditioner (AC), electronic vehicle (EV), Wash and a dryer. The overall goal of this project is to create classifiers that can accurately detect whether target appliances (i.e are being used in a particular time interval (e.g., 1-minute interval). Each appliance's prediction value, i.e., whether they are up or not will be assessed using a unique collection of parameters for each appliance. Several models were considered of which three models were chosen which includes the Generalised Linear Model (Logistic Regression), Gradient Boosting Algorithm and eXtreme Gradient Boosting Trees (xgbTree).

Every piece of work completed for this assignment was a collaborative effort. The tasks were systematically assigned to each team member. In the table below, the percentage of work contributed by each team member on various modules is depicted.

Modules	Milinda	Prashant	Vishal
EDA	30%	40%	30%
Model building	40%	30%	30%
Unit testing	30%	30%	40%
Report writing	30%	35%	35%

PREPROCESSING AND FEATURE GENERATION

The table (Table 1) below shows the summarised results of the on/off status of each appliance, with their relative percentages.

	0 (Off)		1 (On)	
	Frequency	Percentage	Frequency	Percentage
ac	316521	75.77	101199	24.23
dryer	404144	96.75	13576	3.25
ev	415385	99.44	2335	0.56
oven	411764	98.57	5956	1.43
wash	409767	98.1	7953	1.9

Table 1: Shows the class imbalance of frequency and percentage distribution of 0's and 1's for all the appliances in the training dataset.

Pre Processing

1. Both data sets combined have 523260 rows, train_data_withlabels.csv (417720 rows or 290 days) and test_data_nolabels.csv (105540 rows or 73 days) with readings at approximately one-minute intervals.
2. When checking for seasonality it was observed the AC is 1 (on) in the first half of the train data set and hardly used during the second. This indicated that we should create a balanced validation (hold-out) data set from around row 244460 to 294460 from the train data.
3. Following some preliminary exploratory data analysis (EDA), it became apparent that there is a class imbalance for all appliances, requiring the need to build classifiers to determine the operations status of the appliance. Table1 shows more appliances are not being used (i.e., value of 0s) as opposed to appliances being used (i.e., values of 1s) in the training dataset.
4. EDA also revealed that the load is highly related to the air conditioner. This can easily be explained by the previously mentioned class disparity, where the ac has a 75:25 split for 0s and 1s, while other appliances are truly imbalanced, with more than 90% of entries being 0s. This finding was considered when creating the classifier for the various appliances.

Feature Generation

1. We could extract new features that provide further insight into the time series that help define the model for predicting not only in the unlabelled test data but also in future datasets that will emerge using the time series tsfeatures package available in the R CRAN library[1]. The following derived features are currently available in the labelled train and unlabelled test data: dif, absdiff, max, var, entropy, non-linear, and hurst.
2. We observed the id column does not agree with the dayofweek and hourofday columns further feature extraction was not performed from the data because it is not a complete continuously ordered time series. This was later confirmed in the ED forum.
3. Furthermore, we discovered that the features provided lacked continuity. Variance (var) column calculations did not agree with load number. Nevertheless, we chose to build models using the available feature set as these discrepancies were sparse.
4. We chose to convert the 'dayofweek' into separate days using one-hot encoding, i.e., converting categorical data to integer data. As we are aware that the load consumption is not the same for all days of the week and is different on weekends than on weekdays.
5. To solve this classification problem, we translated the predictors of different types into factors. This was done to make it easier for the classifiers to learn because all the predictors were categorical in nature. In using them as factors, we would aid in model learning and improve overall performance.
6. Due to model complexities in handling numerical data for classification(mainly xGB) we had to convert 0 and 1 to on and off respectively.

MODELS

We used a variety of models, including logistic regression (GLM) , support vector machines (SVM), Naive Bayes, CNN, RNN, SGD, Gradient Boost Method (GBM), Random Forest and eXtreme

Gradient Boosting Trees (xgbTrees). Then, based on the accuracy values of our forecasts, we tried to predict the values of labels corresponding to each appliance, using the following three classifiers:

1. GLM (Generalized linear model - Logistic Regression)
2. Gradient Boost Machines (GBM)
3. eXtreme Gradient Boost Tree (xgbTree)

Common Assumptions For All The Models Developed

- All the observations both in test and train datasets were assumed to be linear and sequential with respect to time.
- All the features provided in the dataset such as Entropy, Dif, AbsDif, etc. were assumed to be correct, even though the values of these features did not match those produced by tsFeatures package.

Models Development and Construction

- To predict the labels for each appliance, five different classifiers were developed.
- The baseline function which was discussed earlier uses all the predictors that were provided in the training data and were used for all the appliances.
- All three classification models indicated above were constructed for all appliances, and the best model for each appliance was picked after tuning..

Model 1 : GLM (General Linear Model / Logistic Regression)

Basic Idea behind the Model

- Logistic function is used to model a binary dependent variable.
- Supervised classification algorithm is used in this model.
- Relationship between one dependent binary variable and one or more independent variables is described using logistic regression.

Advantages

- Logistic regression is one of the most basic machine learning algorithms. It is simple to implement and, in some situations, offers excellent insights. Because of these factors, training a model with this algorithm does not necessitate a lot of computing power.
- Logistic regression proves to be very efficient when the dataset has linearly separable features.
- Since it is relatively fast and simple to implement, logistic regression is often used as a benchmark model to measure performance rather than a complex model.

Disadvantages

- Since logistic regression has a linear decision surface, it can't solve nonlinear problems, and linearly separable data is uncommon in real-world scenarios.
- Complex relationships are difficult to capture using logistic regression. Algorithms that are more efficient and complex, such as boosting algorithms, can easily outperform this model.

Model 2 : Gradient Boost Algorithm

Basic Idea behind the Model

- Although most winning models in Kaggle competitions are ensembles of advanced machine learning algorithms, the Gradient Boosting Machines (GBM) is a common component of such groups.
- This model essentially operates on the concept of transforming weak learners to strong learners, i.e., it identifies the flaws of weak learners (in this case, decision trees) by using gradients in the loss function, which is a user-specified cost function that corresponds to real-world applications [2].
- The fact that GBM has many tuning parameters is both a benefit and a challenge. The beauty of this is that GBMs are extremely adaptable. However, the deficiency of this model is the time required to find the best combination of hyperparameters.

Advantages

- Frequently has unrivalled statistical precision.
- Lots of versatility - can optimize on a variety of loss functions and has many hyperparameter tuning choices, making the feature fit extremely versatile.
- There is no need to pre-process data; it always works well with categorical and numerical values as is.
- Imputation is not necessary to handle missing data.

Disadvantages

- GBMs will continue to improve to reduce all errors to a minimum. This can lead to overfitting by exaggerating outliers. To neutralize this error cross-validation must be used.
- GBMs also need many trees (>1000), which can be time and memory intensive.
- Less interpretable as it is mostly used as a black box by many developers.

Model 3 : eXtreme Gradient Boosting Tree Algorithm

Basic Idea behind the Model

- XgbTree is a machine learning algorithm that has recently dominated Kaggle competitions for structured or tabular results.
- As a high-speed and high-performance implementation of gradient boosted decision trees, XgbTree is preferable in this case. Furthermore, in XgbTree, complex models can be penalised using both L1 and L2 regularisation, preventing overfitting.
- Data is sparse when it contains missing values or has been processed in a particular way, such as one-hot encoding. To deal with different types of sparsity patterns in the data, XgbTree includes a sparsity-aware split finding algorithm and since the data provided to us has all these features, it was decided that this would be the best model to use.

Advantages

- Since XgbTree makes use of **parallel processing**, it is significantly faster than other available models. The model is executed on several CPU cores.

- **Missing values are handled** automatically by XgbTree. When XgbTree encounters a missing value at a node, it attempts both the left and right-hand splits and learns which route results in the highest loss for each node. When working with the testing results, it repeats the process.

Disadvantages

- In XgbTree, you must manually create dummy variable/ label encoding for categorical features before feeding them into the models.
- Training time is high for larger datasets.

EXPERIMENTAL SETUPS

We used R's Caret package extensively with varying implementations of the train and trainControl functions. Other packages were used in data reshaping, wrangling and parallel processing, namely:- **doParallel, data.table, tidyverse, xgbTree, ggplot2, ROSE.**

1. Models

We used three models xgbTree, gbm,glm in combination with varying trainControl function configurations for both the time series data and class imbalance runs.

2. Configuration of trainControl

There were two main configurations, one for time-series and the other for class imbalance.

For time series data we adopted the following and sampled different values for initialWindow, horizon and skip. A sample of values used is shown in red.
trainControl(method = "timeslice", initialWindow = (160,300,540) horizon = (50,120,288) fixedWindow = T, skip = (219,349,599), verboseIter = TRUE, savePredictions = "final", seeds = NULL, allowParallel = TRUE, classProbs = TRUE, summaryFunction = twoClassSummary)

For class imbalance, we adopted the following trainControl fundction configuration. We sampled different configurations of the method, number and search. A sample of values used is shown in red.
trainControl(method = ("repeatedcv","cv"), number = (5,7,10), search = ("random","grid"), verboseIter = TRUE, savePredictions = T, seeds = NULL, allowParallel = TRUE, classProbs = TRUE, summaryFunction = twoClassSummary).
The 5-fold cross validation (CV) method entails randomly partitioning the training dataset into five parts and using each of the five parts as a testing dataset for the model learned on the other four. The average of the five error terms thus obtained is used to evaluate the performances of the model.

3. Rebalance of class distribution

We decided to deal with the class imbalance problem by using the 'ovun' function which is available in the Rose package, which does over-sampling for minority class and under-sampling for majority class. This is achieved by using the parameters 'method' and 'p' and equating them to "both" and 0.5, respectively. We further sampled multiple 'N' values which correspond to several rows to train the model. The range of values for 'N' ranged from 1000 to 100,000 for different models including GLM, GBM and xgbTree. The method for the trainControl was set to 'repeatedcv' with the number of folds set to 5.

Shown below is an example of the configuration for the air conditioner.
ovun.sample(ac ~ ., data = app_bal$ac, method = "both", p = 0.5, N = 10000, seed = 1)$data

4. F1 calculation

Our experimental setup consisted of five bespoke classification models for each appliance. Hence the F1 score was calculated per appliance and tabulated. The confusion matrix was calculated using the hold-out validation data. Also, the matrix prediction was configured to 'on'. Finally, each run's F1 along with the appliance name, the model used, and other metrics were written to a CSV. After tabulating for all combinations of N values, we were able to identify the best model for each appliance and the rebalance N value.

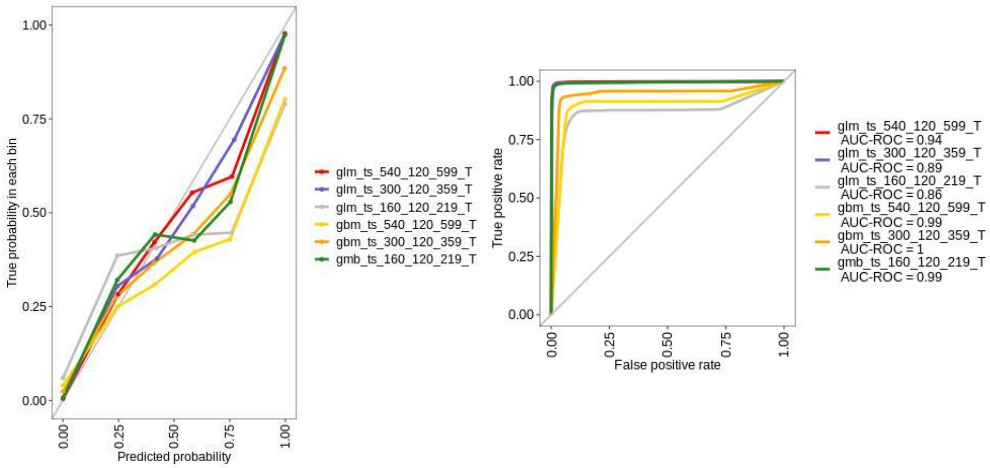
EXPERIMENTAL RESULTS

Firstly, basic preprocessing was performed to understand the relationships between the predictors and outcome. Multiple models were applied to the training data to check for accuracy. Two metrics, ROC and accuracy, were used to evaluate the model performance before selecting three models xgbTree, GLM and GBM to continue our analysis.

After selecting the three models, due to the class imbalance and this being a binary classification problem ROC was selected as the key metric in Caret's train function.

Finally, to progress, the team needed to answer a two-fold problem. One of the time series data and the other of class imbalance. A review of the literature indicated that no approach would accommodate both attributes. However, during our review, one possible solution did present itself. Fast Fourier transform appeared to be ideal for this problem, but we were unable to implement it with any success.

During model development, we decided to use the first 50000 rows. This enabled rapid prototyping of ideas, methods and models. One pathway was to respect the time-series nature of the data. We implemented Caret's trainControl with method="timeslice" and various combinations of the initialWindow, horizon, fixedWindow and skip as detailed [3]. We were quite successful while in development with the first 50K rows. However, we were not able to successfully deploy this on the full data set.



Calibration Curve Plot ROC curve Plot

Figure 1:The Calibration Curve and ROC curve for different models with different time slices for the AC

Figure 1 shows the Calibration Curve and ROC curve for different models with different time slices for the AC. Other appliances were also subjected to the same method. This aided us in developing a model that provided better outcomes and had a time slice that was suited for each appliance. The ROC curve and Calibration Curve obtained from different models after training data was used to measure the accuracy of the test data set.

While addressing the class imbalance aspect of this problem we had to hypertune the value of N which corresponds to training sample dimension. Other metrics such as F1 score, Kappa, Sensitivity, Specificity and Accuracy were also used to assess how well the different models performed.

Number	app	model	Sensitivity	Specificity	Pos.Pred.Value	Neg.Pred.Value	Precision	Recall	F1	Prevalence	Detection.Rate	Detection.Prevalence	Balanced.Accuracy
N70000	**ac**	**xgb**	**0.943**	**0.997**	**0.875**	**0.999**	**0.875**	**0.943**	**0.908**	**0.021**	**0.020**	**0.023**	**0.970**
N80000	ac	xgb	0.922	0.998	0.893	0.998	0.893	0.922	0.907	0.021	0.019	0.022	0.960
N2000	**dryer**	**gbm**	**0.461**	**0.900**	**0.135**	**0.980**	**0.135**	**0.461**	**0.209**	**0.033**	**0.015**	**0.111**	**0.681**
N13000	dryer	xgb	0.313	0.935	0.141	0.976	0.141	0.313	0.194	0.033	0.010	0.073	0.624
N9000	**ev**	**glm**	**0.700**	**0.973**	**0.303**	**0.995**	**0.303**	**0.700**	**0.423**	**0.017**	**0.012**	**0.039**	**0.836**
N10000	ev	glm	0.688	0.972	0.291	0.995	0.291	0.688	0.409	0.017	0.011	0.039	0.830
N90000	**oven**	**gbm**	**0.759**	**0.996**	**0.778**	**0.996**	**0.778**	**0.759**	**0.768**	**0.018**	**0.014**	**0.017**	**0.877**
N40000	oven	gbm	0.775	0.995	0.758	0.996	0.758	0.775	0.766	0.018	0.014	0.018	0.885
N15000	**wash**	**xgb**	**0.176**	**0.973**	**0.130**	**0.981**	**0.130**	**0.176**	**0.150**	**0.022**	**0.004**	**0.030**	**0.575**
N4000	wash	gbm	0.174	0.967	0.107	0.981	0.107	0.174	0.133	0.022	0.004	0.036	0.571

Table 2: Results showing the optimum combination of model *vs* N that gives the best F1 score.

Table 2 shows the optimum combination of model *vs* N that gives the best F1 score. As F1 score is the statistic we are looking for when it comes to the accuracy of the results given by the models. From Table 2, it is evident that very low scores are being obtained for the washer and dryer. The fundamental reason for this is because both appliances have low 'Precision' scores. Precision is a measure of the spread of repeated measurement results that are only based on the distribution of random errors; it does not indicate how close the results are to the true value.

CONCLUSION

This project looked at Smart metre readings data for five electrical appliances to create classifiers that can accurately detect whether the target appliances are being used in a particular time .

Basic preprocessing was performed to understand the relationships between the predictors and outcome before multiple models were applied to the training data to check for accuracy. Two metrics, ROC and accuracy, were used to evaluate the model performance. The data was tested on three models xgbTree, GLM and GBM to continue further analysis.

The data presented was both time series and class imbalanced. Caret's train control method time-slice was used to fit models, however due to performance issues we were unable to accommodate the full dataset. The issue of class imbalance was addressed successfully using the Rose packages 'ovun' function. This paper described the best model to use on a limited dataset. However, we were unable to address both attributes in the full dataset. These findings can be further improved by using Fourier transform to convert the time series data into frequency data.

REFERENCES

1. Yangzhuoran Yang and Rob J Hyndman. (2020, June 07). Retrieved from https://cran.r-project.org/web/packages/tsfeatures/vignettes/tsfeatures.html#hurst
2. Singh, H. (2018, November 04). Understanding Gradient Boosting Machines. Retrieved from https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab
3. Kuhn, M. (2019, March 27). The caret Package. Retrieved from http://topepo.github.io/caret/data-splitting.html#data-splitting-for-time-series