FIT5149 S1 2021 Assessment 2 Report Group No. 2

RESIDENTIAL ENERGY APPLIANCE CLASSIFICATION

Faezeh Fazel (ID: 30399653)
Ricardo Arias (ID: 30550971)
Prashant Jajoria (ID: 31187366)

MAY 30, 2021

Abstract

Load monitoring/ load detection is one big breakthrough in tackling the problem of increasing carbon footprint. It helps to provide detailed electricity consumption information in residential households. This project is dedicated to provide a perfect estimate of the usage of the most common appliances at residential buildings.

Contents

1. Introduction

What is the project about?

Greenhouse gas emission is one of the important problems that we face today. Thus to combat this problem, energy production/consumption has to be monitored. Energy efficiency helps track and cut the rapid demand and growth of global energy demands to decommissioning of fossil-fuel power plants and combat climate change. The growing number of population and increasing residential buildings has led to a significant increase in demand for energy in the last few decades. Thus using technology to build energy-saving techniques will help cut the carbon footprint. Using the right amount of electricity at the right time will increase energy efficiency and reduce emissions.The paper entitled “Performance evaluation in non-intrusive load monitoring:Datasets, metrics, and tools-A review", gives a brief overview of Load monitoring for appliances used in residential buildings which are air conditioner, electric vehicle charger, oven, washing machine and dryer. We can infer from the paper that the load for each of the appliances remains high for time when the appliance is in use, and drops significantly when it's off. This pattern of load fluctuation for the appliance can be used to predict the appliance usage given the load of a household.

What is the goal of the project?
In this project, our target is to find the most accurate prediction for each appliance, whether it is on or off given the load value for a residential household and other features like day of the week and hour of the day.

How the project was done in your group?
For all the steps of performing this project, all three team members were involved and contributed together.

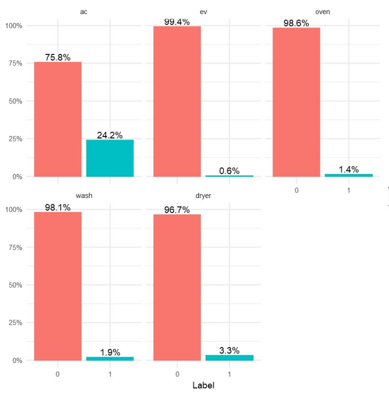
- Data wrangling
- Exploratory Data Analysis
- Feature Extraction and Feature Selection
- Model Training
- Model Comparison
- Conclusion

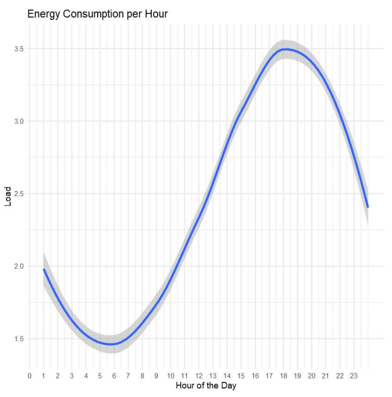
2. Pre-processing of the data and features used

2.1 Steps used to pre-process data
The dataset provided has 8 numerical features, 2 categorical features, and 5 labels, one for each appliance. We perform the following pre-processing for the dataset before the model development phase.

- Converting the categorical variables into one hot values
- Normalizing the data with min-max scaling
- Dividing the data frame into 5 different data frames one for each appliance

In the data exploration, we noticed a very important characteristic of the data which the data is extremely unbalanced which means the proportion of zeros and ones for target variables in each dataset (ac,ev,oven,wash and dryer) follows as: 0.75, 0.99, 0.98, 0.98, 0.96. Besides, there is a clear behaviour for electricity consumption during different hours of the day which leads us to think that people tend to use the appliances more during the time they are at home from 16 to 21.





2.2 Feature Selection

We generate the following 6 new features using the ‘tsfeatures’ package in R, These features are generated by using the concept of sliding window. We use a window of length=30 to extract these features.

crossing_points - number of times the time series had crossed the median value
flat_spots - computed by dividing the sample space of a time series into ten equal-sized intervals, and computing the maximum run length within any single interval.
firstmin_ac - the time of first minimum in the autocorrelation function
firstzero_ac - the first zero crossing of the autocorrelation function.
trev_num - the numerator of the trev function of a time series, a normalized nonlinear autocorrelation
std1st_der - returns the standard deviation of the first derivative of the time series.

<u>What are the feature set(s) used?</u>
Since training all of the possible models 2^N with N= Number of features is time consuming and costly, We apply Fischer and Chi Square feature selection to get the best 5 features from the total set of features that was already provided plus the new extracted features, and train the KNN model with the union of the features from these two methods. But in XGBoost model as we applied the new extracted features in the model and they didn't provide us a good model performance, we just used the all set of features that have been provided in the dataset and since XGBoost has feature selection inside itself we just pass the all features into the algorithm.

As the training of the model with all the features and all the subset features is extremely time consuming and costly, we trained the KNN model with a best strategy of feature selections which is mentioned below:

We select the combination of both feature selection methods. Chi-square method provides us with the top 5 most important features. Also, Fischer method provides us with the 5 most important features. Then, we can use the union of the most important features provided by these two methods.

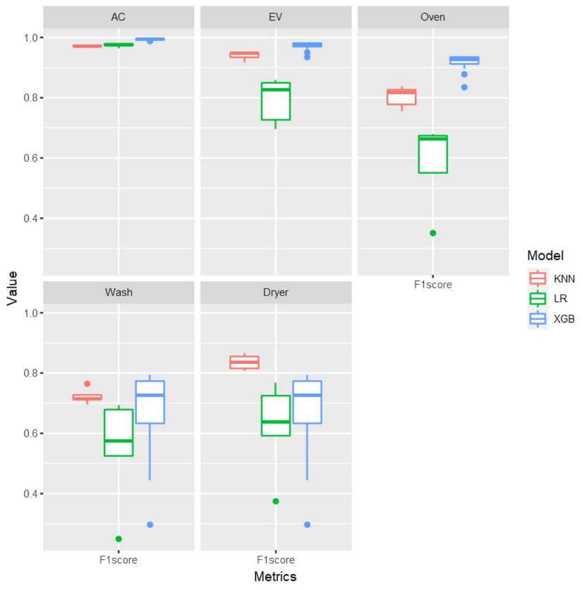
Features Used in XGBoost Model

ALL the Appliances	hourofday, dayofweek, load, dif, absdif, max, var, entropy, nonlinear, hurst

Features Used in KNN Model

Appliance	Chi Square Selection	Fischer Selection	Final features
AC	max, var, trev_num_30, std1st_der_30, absdif	load, max, var, std1st_der_30, absdif	max, var, trev_num_30, std1st_der_30, absdi, load
EV		load, max, nonlinear, hurst, flat_spots_30	max, var, trev_num_30, std1st_der_30, absdif, load, nonlinear, hurst, flat_spots_30
Oven		std1st_der_30, load, absdif, max, firstmin_ac_30	'max', 'var', 'trev_num_30', 'std1st_der_30', 'absdif','load','firstmin_ac_30'
Wash		flat_spots_30,crossing_points_30 , firstmin_ac_30, load entropy	max,var,trev_num_30,std1st_der_30,absdif,flat_spots_30,crossing_points_30,firstmin_ac_30, load , entropy
Dryer		crossing_points, flat_spots_30, load, firstmin_ac_30, max	mamax,var,trev_num_30,std1st_der_30,absdif,crossing_points_30,flat_spots_30,load, firstmin_ac_30

3. Models



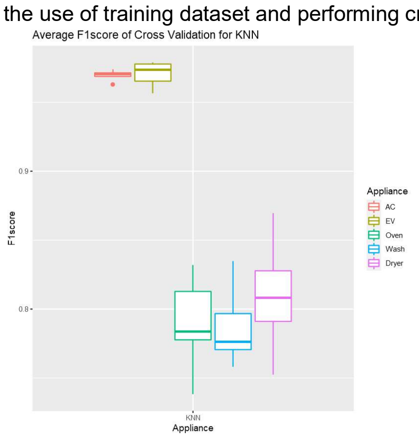
Three different classification algorithms(Logistic Regression, KNN and XGBoost) applied to get a better idea which classification method performs better for this dataset. As can be seen from the below graph, for all the appliances, we see that KNN and XGBoost algorithms perform better than the others in terms of F1score. So these two models have been chosen to be tuned further. Also, a cross-validation method has been applied to get the performance of the best models for each appliance.

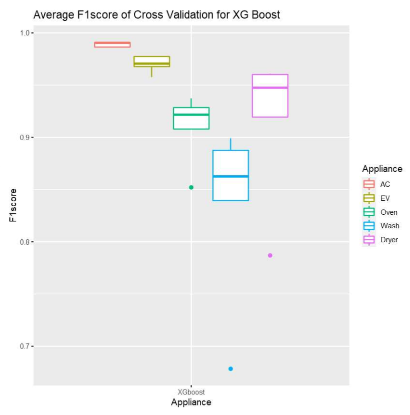
3.1 KNN Model

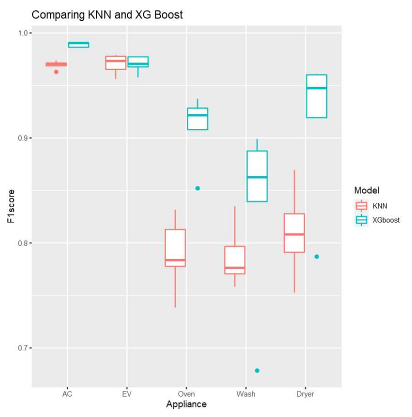
<u>Basic Idea of the KNN model</u>
KNN is a supervised machine learning algorithm that uses labeled data with two or more classes and predicts the class of the new sample point. It also is a non-parametric classification algorithm which means it doesn't make any assumption about the distribution of the population.
Since KNN algorithm is a distance based classification method, one of the most important part of the data characteristics is that data must be in the same scale. Also, the categorical variables need to be hot_coded which have been applied to the data in the early stage.
KNN model is constructed by the 'class' library with the use of the training dataset with selected features and performing cross validation to find the best neighbour.
It calculates the similarity between a data point and all the datapoints of the train dataset using different distances like Euclidean, Manhattan, cosine, Minkowsky,etc. For prediction KNN gets the top K similar datapoints as the given input test data, and gives the average of the K datapoints for Regression. For Classification, it gives the majority vote of the top K similar datapoints.

3.2 Gradient Boosting Model

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modelling problems. Ensembles are constructed from decision tree models. Trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models. This is a type of ensemble machine learning model referred to as boosting. Models are fit using any arbitrary differentiable loss function and gradient descent optimization algorithm. This gives the technique its name, “gradient boosting,” as the loss gradient is minimized as the model is fit, much like a neural network. The XGBoost is an implementation of gradient boosted decision trees which is designed in order to optimise speed and performance. It starts with a base learner which makes the predictions. For this base leaner a loss function is defined and the residuals are calculated. In the subsequent steps, we try to learn from the weak predictions i.e. the residuals. The final prediction is the sum of the base learner and the minimum of the subsequent residuals. Since decision tree based models have feature selection inside themselves, there is no feature selection step for this model training step. XGBoost model is constructed by the xgboost library with the use of training dataset and performing cross validation to find the best iteration.







Discussion of model differences

	KNN	XGBoost
Feature preprocessing	Scaling of numerical features, One hot encoding of categorical values	Scaling of numerical values. It can handle categorical values
Nature	Distance based algorithm	Implementation of Gradient Boosted decision tree
Type	Instance based	Ensemble learner
Parameters	Neighbours, similarity metrics	nrounds, eta, gamma, max_depth, lmbda
Dataset size	Not suitable for large datasets	Can handle large datasets
Space Complexity	O(Features * Rows)	O(Features * Rows * ntrees)
Interpretability	High	Low
Sensitivity to outliers	Sensitive to outliers as it chooses the neighbours on the distance criteria	As the boosting algorithm builds trees on the previous models residuals.
Accuracy	Low	High
Feature selection	Feature engineering needed	XGBoost does the feature selection up to a level
Reason to choose	No prior knowledge about data distribution	Fast Execution speed, High Model performance
Package in R	class, FNN, DMwR2, neighbr, caret	xgboost, caret, gmb, sparkxgb
Advantages	• Can be applied to any kinds of data with any kinds of the relationships • It has very few parameters to tune. • Very easy to implement although the It can be applied to any dataset.	• Important Features can be found out. • It can handle large datasets easily. • It has quite fast training time. • It has pretty good performance.
Disadvantages	• Defining the best k=neighbour is hard. • KNN doesn’t perform well with imbalanced data. • The speed of the algorithm declines monotonically when the number of data grows	• Difficult interpretation. • Overfitting is possible if parameters are not tuned properly.

4. Experiment Setups

4.1 Cross-validation setup

4.1.1 KNN
For getting the best neighbours for each appliance we use Stratified k-fold cross validation because of class imbalance of target variable. This ensures that each fold has the same proportion of class labels in each fold, thus giving a good overview of the model performance. We tuned the neighbours from 2 to 10 (step by 2) for each appliance and got average Accuracy, Recall, Precision and F1score for all the folds. From this CV results we select the best neighbour for each appliance to train the final KNN model for each appliance.

4.1.2 XGBoost
To get the best threshold of XGBoost, first we splitted the data in training dataset (70% of the data) and validation dataset (30%), using stratified sampling to keep the balance of the original data in each dataset. With the training data we performed cross validation on XGBoost to find iteration (nround), that has the best performance. Having found these hyperparameters, we proceed to train the model again with the best iteration.
Once we have the model trained, we tuned the threshold from 0.05 to 0.95 (steps by 0.05) for each appliance and got average Accuracy, Recall, Precision and F1score for all the folds. From this CV results we select the best threshold for each appliance to train the final XGBoost model for each appliance.

4.2 Calculation of metrics

We calculate the following metrics by using **MLmetrics** library from for each of the folds and different parameters to select the best parameters:
1. **Accuracy** - proportion of total number of predictions that were correctly predicted.
2. **Recall** - TP / (TP + FN) A measure of how accurately our model is able to identify the relevant data.
3. **Precision** - proportion of true positives that are correctly classified (TP) of all the positives (TP+FP).
4. **F1Score** - Harmonic mean of precision and recall, which gives a tradeoff between precision and recall.

5. Experiment results

5.1 Comparison of Models results on different metrics

	KNN Algorithm				XGBoost Algorithm			
	Accuracy	Recall	Precision	F1score	Accuracy	Recall	Precision	F1score
AC	0.982	0.972	0.974	0.973	0.997	0.997	0.990	0.994
EV	0.998	0.956	0.956	0.953	0.997	0.995	0.994	0.994
OVEN	0.996	0.845	0.839	0.837	0.997	0.994	0.995	0.994
WASH	0.989	0.757	0.764	0.765	0.997	0.990	0.998	0.994
DRYER	0.995	0.873	0.871	0.866	1.000	0.979	0.986	0.982

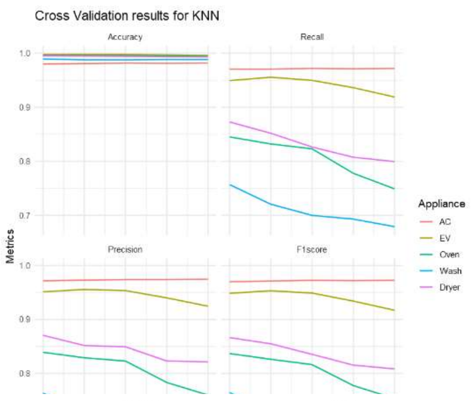
This table shows all the metrics of the chosen models for the best tunes parameters. It can be seen that all of the metrics of the model performance for a classification algorithm including: F1score, Precision, Recall and accuracy for all of the appliances in XGBoost model is higher than KNN model except for dryer which F1score in KNN model is higher than Xgboost.

Parameters of KNN used in the final model:
- **AC:** Neighbors = 2
- **EV:** Neighbors = 4
- **Oven:** Neighbors = 2
- **Wash:** Neighbors = 2
- **Dryer:** Neighbors = 2

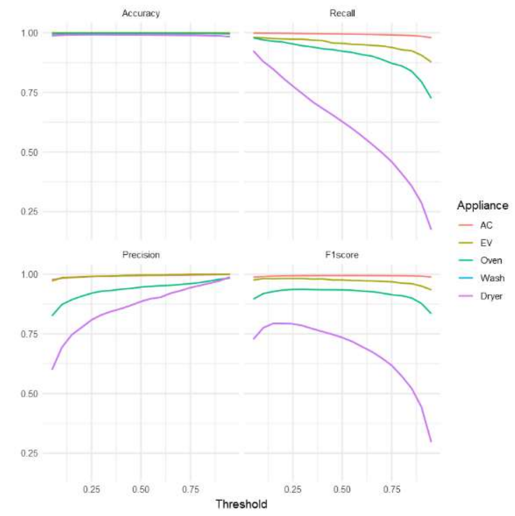
Parameters of XGboost used in the final model:
Since this dataset is imbalanced, we utilized the “scale_pos_weight” parameter in XGBoost algorithm which takes care of the imbalanced datasets and the value depends on the ratio of the number of zeros to the number ones for each appliance. Also, as the prediction depends on the threshold defined for the calculated probabilities, the best F1score along with best mean(aucs) in cross validation step has been achieved by threshold = 1/64.

- **AC:** scale_pos_weight= 3.127709
- **EV:** scale_pos_weight = 177.8951
- **Oven:** scale_pos_weight = 69.13432,
- **Wash:** scale_pos_weight = 51.52358
- **Dryer:** scale_pos_weight = 29.769

Experimental results of Cross validation for KNN



From the KNN experiment we could find out that the best neighbour is the same for all the appliances except for EV. In all the metrics can be seen that when the number of neighbours increases the performance of KNN model decreases. As F1score is the main evaluation metrics, we can select neighbours = 2 as the final parameter after the experimental setup.

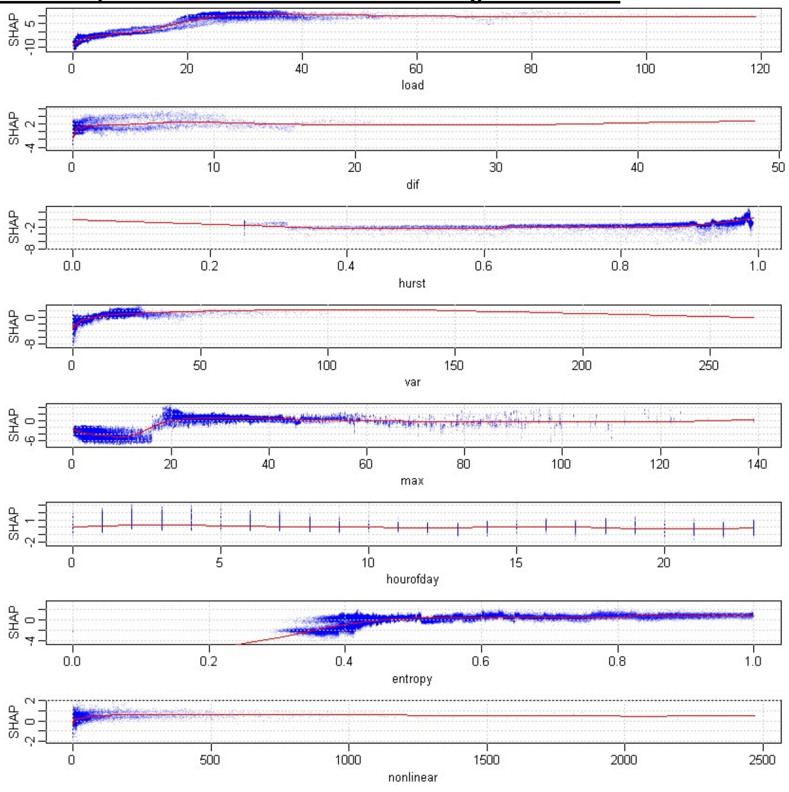


Experimental results of Cross validation for XGboost
From the XGBoost experiment we could find out that the best threshold is different for each appliance, and each metric calculated. For the purpose of this project the most relevant metric is F1 score. Therefore the hyperparameter selection was made based on this metric. It can be seen that when the threshold is too high or too low the performance of the model decreases. For accuracy the threshold seems to be irrelevant because it does not change and that is why the graph is a flat line. Precision and Recall have a completely opposite behavior, if the threshold is low the precision is low too but the recall is pretty high. And when the threshold increases the precision increases too while the recall decreases.

Experimental results of Cross validation for XGboost

From the XGBoost experiment we could find out that the best threshold is different for each appliance, and each metric calculated. For the purpose of this project the most relevant metric is F1 score. Therefore the hyperparameter selection was made based on this metric. It can be seen that when the threshold is too high or too low the performance of the model decreases. For accuracy the threshold seems to be irrelevant because it does not change and that is why the graph is a flat line. Precision and Recall have a completely opposite behavior, if the threshold is low the precision is low too but the recall is pretty high. And when the threshold increases the precision increases too while the recall decreases.

Feature Importance Inference for XGBoost using SHAP values



In order to interpret the importance of different features in the model on the target variable , SHAP VALUES measurement has been applied. It can be seen from the plot, “load”, “dif”, “hurst” and ”var” are the most important features in the xgboost model for ac. In each different feature, data points less than 0 indicate negative contribution toward the target variable which means for example, in “load” data points which are less than 18 increase the likelihood of the air conditioner to be off and data points greater than 18 increase the likelihood of the air conditioner to be on which is sensible. Also, the ‘max’ values smaller than 18 have a negative effect on the ac to be on and the values greater than 18 mostly have a positive effect on the likelihood of the ac to be on.

6. Conclusion

As we discussed in the previous section about the pros and cons of the two chosen algorithms, and the good performance of the XGBoost model in prediction in terms of F1score which is the most important metrics for the stakeholder and also the other metrics for almost all the appliance settings. And also specially because the amount of training, testing and prediction time of the XGboost model is far less than the KNN algorithm. So the final model we choose is XGBoost.

A very important lesson we learned from this project is that how much imbalanced data affects the classification task at hand and also we need to pay more attention to what the stakeholders are more concerned about and change how the model predicts in this regard. Also, we learn that in the real world we will not have the test dataset to see our model performance, hence having a strategy to evaluate the final model using the train data is important. Also, we learn that sometimes having a good prediction for the target variable is what the stakeholders will need, regardless of the data transformations applied, the complexity of the model. We learn that as long as the model gives good predictions we might not care much about the interpretability of the model itself, which makes the model a Black box. Explaining the feature importance of such a model is now becoming a hot topic for research.

Another important lesson learned from this project, is that the experiment set up must be done rigorously, because as the dataframe as that big we cannot afford run the algorithm and that the results are not the one we expected to have , If we spend too much time running a code that at the end does not work it would be a waste of time.

For future similar projects, we learn that having a well-defined strategy to train and validate the model is key to completing the project goals on time. Also, having defined goals and having a schedule of tasks plays a vital role to see the milestones achieved. Defining the limitations and explaining the setup used for the project is important for clear understanding of the results .

7. References

1. Pros and Cons of K-Nearest Neighbors - From The GENESIS. (2021). Retrieved 30 May 2021, from https://www.fromthegenesis.com/pros-and-cons-of-k-nearest-neighbors/
2. Pros and cons of various Classification ML algorithms. (2021). Retrieved 30 May 2021, from https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6
3. (2021). Retrieved 30 May 2021, Retrieved from https://cdn.scribbr.com/wp-content/uploads/2020/09/stratified-sample-7.png
4. Computational complexity of machine learning algorithms. (2021). Retrieved 30 May 2021, fromhttps://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/