RESIDENTIAL ENERGY APPLIANCE CLASSIFICATION

Srinithi Punniamurthy, 31284868
Chaitanya Srinevas, 31216072
Alonso Henry Gomez Zeballos, 31399835

MAY 30, 2021

Abstract

Energy efficiency plays a crucial role in the transformation of future energy systems and cutting the rapid growth of global energy demand to enable early decommissioning of fossil-fuel power plants and combat climate change. Electricity consumption in residential sectors accounts for more than 20% of total consumption, and thus energy-saving technology for residential buildings is of vital importance.

With the advent of the latest machine learning technologies, researchers have tried to apply this to study Non-Intrusive Load Monitoring (NILM). The underlying motive of the project was motivated by the energy efficiency concerns and the need to reduce the carbon footprint of buildings and households with the help of advanced metering technologies such as smart grids.

Non-intrusive load monitoring is defined as the process of estimating the energy consumption of individual appliances from electric power measurements This approach reduces sensing infrastructure costs by relying on machine learning techniques to monitor electric loads.

Through this project, we develop and present a comprehensive set of different machine learning classification algorithms that best performs on the appliance usage and generates the best F1 score by selecting the best model for individual appliances. Using the dataset provided, we tried to analyse the data based on time series analysis to study trends in the appliance usage and understand the patterns. Based on the exploratory data analysis, suitable classification algorithms were applied for each appliance. Subsequently the inference task established to focus was on identifying features from the dataset that exert strong influence on the prediction results and indicating strong drivers in the usage of the appliance.

Table of Contents

1 Introduction

Non-intrusive load monitoring is the process of estimating the energy consumption of individual appliances from electric power measurements as the approach reduces sensing infrastructure costs by relying on machine learning techniques to monitor electric loads. The project aims at systematic performance evaluation of classification algorithms blending a mix of traditional and ensemble models for determining the appliance usage and generating the best F1.

While Exploratory Analysis was conducted to generate useful insights on usage pattern, suitable classification models were adopted based on significance of features, trends and f1 scores, lastly feature importance was implemented to understand the most influential parameters for the model.

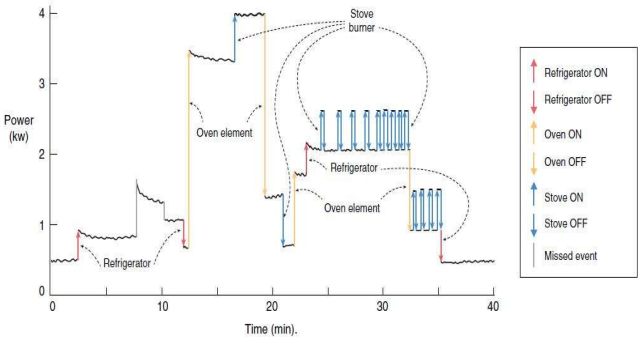

Fig1: Load Monitoring, an example.

The goal of the project is to develop classifiers for each appliance and select the best classifier which gives the most accurate prediction. The empirical analysis between models based on the f1 score would help us decide the best classifier for the appliance. F1 score metric is defined as the harmonic mean of the Precision and Recall and as in most real-world applications, there are class imbalance distributions. F1 score is a better metric to assess our model.

The second task of the project deals with studying the importance of the features which are indication of strong drivers in appliance usage. For this task, various system defined functions for feature importance were used from the libraries in the classification models we used. The plots reveal the most important features for the models.

The work was split into EDA, Feature Selection and Engineering and Model building and lastly Feature Importance. We all worked collaboratively together to develop and optimise the code with modular programming, redesign and debug the issues whenever encountered. Understanding, the working of Tsfresh package was done collectively by all the team members. The agile methodology was helpful, because this practice avoided the confusion of adopting different variable, function names and helped in maintaining the readability of the flow from the beginning.

2 Pre-processing and features used

Data pre-processing is an extremely crucial task imperative to every data science problem. It transforms the collected raw data into a comprehensible and efficient format. While the assignment had real world data, we followed the following steps to pre-process the data and make it ready to be analysed.

1. Checking for missing, null values, unique values and to verify if there were any redundancies and data inconsistencies.
2. Removing unnecessary columns such as 'Unnamed:0' which was not required for further analysis.
3. Encoding categorical variables
4. Since the data was time series data, the dataset had to be ordered by timestamps.
5. For the same, we were instructed to use the Tsfresh package to extract characteristics from time series. The package provides functions to extract a comprehensive set of features through functions such as extract_features() .
6. Introducing Rolling timestamp for multiple time series to reshape data for feature extraction.

Identification of Class Imbalance problem and Sampling as a solution

The class imbalance problem is defined as a classification problem where the classes are not distributed equally. In our case, the problem was encountered while checking the distribution of the values in the appliances where it had many values for 0 and very less values for 1. In this case the models that we build will be biased towards the majority class, resulting in poor predictive performance for the minority class.

Sampling of the data provides a solution to the problem of imbalance data. UpSampling is done where the data points are synthetically generated, and minority and majority class counts are equalised. and usually implemented via SMOTE. On the contrary Downsampling reduces the data points from the majority class. For Sampling we have created our own function get_balance_df() which downsamples or up samples based on the parameter given.

Exploratory analysis

After building a comprehensive exploratory analysis through univariate, bivariate, multivariate, and trend analysis of the time series data, we concluded that the features given in the dataset should be used for building the model. We derived that the appliances exhibit a trend towards the second half of the day and more towards the weekends rather than weekdays. Amongst all, 'ac' was the most used appliance. The below figure demonstrates the variation of the difference of two sequential load points over the hour of the day for each day of the week.

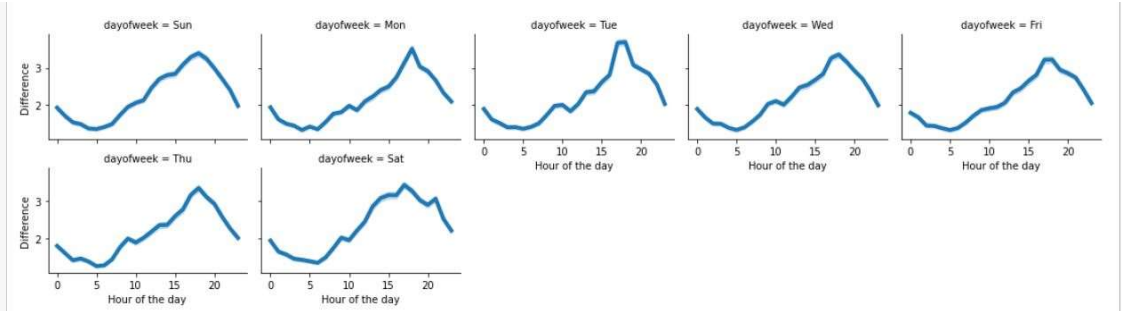

Fig2: Trend Analysis depicting the usage of appliances over the week

3 Models

All the models were presented in the form of a table including model construction, differences between the models,advantages and disadvantages, and practical applications.

3.1 Models (Logistic Regression, Random Forest, SVM, XGBoost, Decision Tree)

	Introduction to the model	Model Construction and Assumptions	Advantages	Disadvantages	Other Aspects - Applications
Logistic Regression	Logistic regression is a classification algorithm. It is used to predict a binary outcome based on a set of independent variables.	The logistic function, also called the sigmoid function used to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It is an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits. 1 / (1 + e^-value). Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. **Assumptions:** It assumes that there is minimal or no multicollinearity among the independent variables. It usually requires a large sample size to predict properly. It assumes the observations to be independent of each other.	Easy to interpret, implement and train. Does not require too much computational power, makes no assumption of the class distribution, fast in classifying unknown records, can easily accommodate new data points, it is very efficient when features are linearly separable.	Tries to predict precise probabilistic outcomes, which leads to overfitting in high dimensions, since it has a linear decision surface, it can't solve non-linear problems, Tough to obtain complex relations other than linear relations, requires very little or no multicollinearity, Needs a large dataset and sufficient training examples for all the categories to make correct predictions.	In gaming industry, the collaborative system predicts what the user would like to buy based on ratings from users with similar preferences in previous purchases, and other activity. A content-based algorithm makes its decision based on properties specified in the item description and what the user indicated as interests in her profile. The third type is the hybrid, and it is a combination of two previous types. In hotel booking, they try to predict users' intentions and recognize entities. Where will you go, where do you prefer to stop, what are you planning to do? Used in text editing.
Random Forest	It assembles randomized decisions based on several decisions and makes the final decision based on the majority. It does not search for the	The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. Many relatively uncorrelated models (trees) operating as a committee will	Powerful and highly accurate, Robust to outliers, Works well for non-linear data, Low risk of overfitting, Runs	They are biased to certain features sometimes, Slow training, cannot be used for linear methods, Worse for high dimensional data	In Banking Sector, to determine whether the customer is a loyal or fraud, to detect and predict the drug sensitivity of a medicine,

	best prediction. Instead, it makes multiple random predictions. Thus, more diversity is attached, and prediction becomes much smoother.	outperform any of the individual constituent models. The low correlation between models is the key. Uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors. **Pre-requisites:** There needs to be some actual signal in our features so that models built using those features do better than random guessing. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.	efficiently on large datasets.		the behaviour of the stock market can be analysed. Also, it can show the expected loss or profit which can be produced while purchasing a particular stock, In e-commerce, it suggests what type of products customers should see.
XG Boost	XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. The implementation of the algorithm was engineered for efficiency of compute time and memory resources. A design goal was to make the best use of available resources to train the model	Each regression tree maps an input data point to one of its leaves that contains a continuous score. XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity (in other words, the regression tree functions). The training proceeds iteratively, adding new trees that predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction. It's called gradient boosting	Handling Missing Values, tree pruning, can work in parallel, no need for scaling or normalizing data, Fast to interpret, Great execution speed. Another advantage is that sometimes a split of negative loss say -2 may be followed by a split of positive loss +10. GBM would stop as it encounters -2. But XGBoost will go deeper, and it will see a combined	Can easily overfit if parameters are not tuned properly, Hard to tune.	XGBoost is used for supervised learning problems, where we use the training data (with multiple features) to predict a target variable

		because it uses a gradient descent algorithm to minimize the loss when adding new models. **Assumptions**: It may have an assumption that encoded integer value for each variable has ordinal relation	effect of +8 of the splits and keep both.		
Decision Tree	A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.	what is actually going on in the background? Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, you will need to trim it down. **Assumptions:** Initially, whole training data is considered as root. Records are distributed recursively based on the attribute value.	Compared to other algorithms, data preparation requires less time, doesn't require data to be normalized, Missing values, to an extent, don't affect its performance much, It is very intuitive as can be explained as if-else conditions.	Needs a lot of time to train the model, A small change in data can cause a considerably large change in the Decision Tree structure, comparatively expensive to train, Not good for regression tasks.	Assessing prospective growth opportunities, using demographic data to find prospective clients, Serving as a support tool in several fields
Support vector Machine (SVM)	The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.	The binary SVM constructs a set of hyperplanes in an infinite dimensional space, which can then be divided into two kinds of representations, such as the linear and nonlinear SVM. **Assumptions**: It assumes data is independent and identically distributed.	Works well on high dimensional data. Memory efficient. Effective in cases where the number of dimensions is greater than the number of samples.	Not suitable for large datasets. Does not work well when the dataset has noise, i.e., the target classes are overlapping. Slow to train. No probabilistic explanation for classification.	Inverse Geosounding Problem, Seismic Liquefaction Potential, Protein Fold and Remote Homology Detection, Data Classification using SSVM, Facial Expression Classification, Texture Classification using SVM, Text Classification, Speech Recognition, Stenography Detection in Digital Images, Cancer Diagnosis and Prognosis

3.2 Discussion of model difference(s)

A plot is used for showing how the features are considered as important or not in each model.

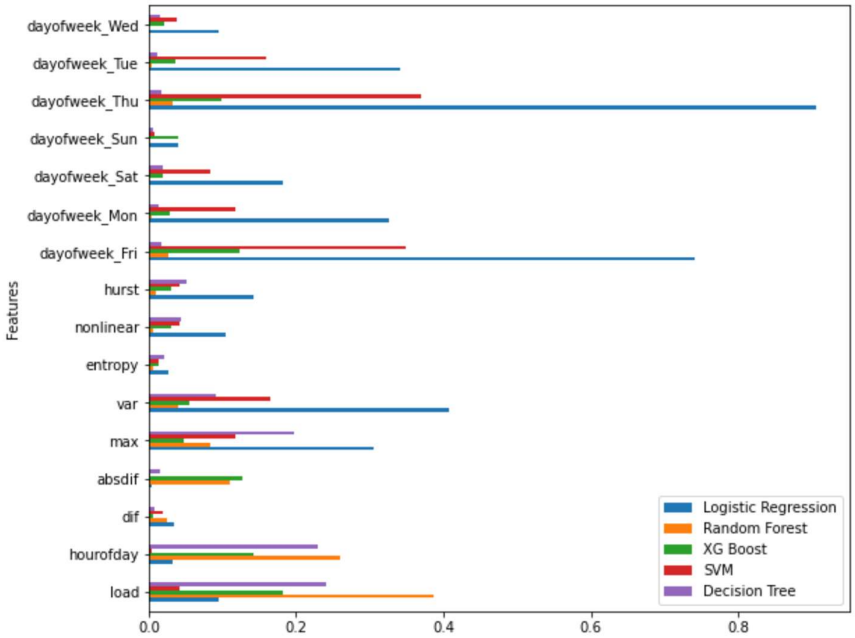

Fig3: Feature Importance or each Model

From the previous plot, the models that were built present differences feature importance. At the very beginning, the assumption was `load` would be as part of the most important feature for all models, but at this stage, `dayofweek_Thu` and `dayofweek_Fri` appear to be more representative. `dif` and `entropy` seem to have very low impact when building those models.

4 Experiment setups

Cross-Validation is used for comparing different machine learning algorithms or models. The basic idea is to split the data frame into folds where one of the folds is used for testing or evaluation of the models while the rest of folds are used for training the model itself. By doing this split, the bias is reduced. Here we have used 10-Fold Cross-Validation. We have used a user defined function to embed the implementation of cross validation, model building and sampling under one function. The scikit-learn library provides an implementation that will split a given data sample up. The KFold() scikit-learn class can be used. It takes as arguments the number of splits, whether to shuffle the sample, and the seed for the pseudorandom number generator used prior to the shuffle.

F1- score is one of the measures used for testing or measuring the classification model accuracy. This measure takes into account two separate measurements, which are recall and precision. The range of this indicator is from 0 to 1 where 1 is the highest value. For computing F1- score, the following expression can be used.

$$F1 = 2\,x\,\frac{Precision\ x\ Recall}{Precision\ +\ Recall}$$

where

$$Precision = \frac{Number\ of\ True\ Positive}{Number\ of\ True\ Positive\ and\ False\ Positive}$$

$$Recall = \frac{Number\ of\ True\ Positive}{Number\ of\ True\ Positive\ and\ False\ Negative}$$

Once all models are built, the F1 score is calculated for all of them and recorded in a table for future comparison. The idea is to choose the best model which has the highest F1 score for a given appliance. It is expected to have different F1 scores for each model in each appliance so the final prediction might be done for different models for each appliance.

A function was created for getting the balance data frame, building the model and performing the 10-fold cross-Validation for all appliances at the same time. By doing this, the execution time is reduced and the comparison between F1 score for a given model for all appliances was much easier to understand.

5 Experimental results

Models such as Logistic Regression, Random Forest, XGBoost, SVM and Decision Tree have different F1 scores when training those models. The following table is the final result, and it summarizes the results for each model on the F1 score when predicting the usages of appliance. using the Testing Data Frame.

	model	ac	ev	oven	wash	dryer
0	Logistic Regression	0.983722	0.901676	0.901976	0.667117	0.657201
1	Random Forest	0.986374	0.945304	0.948649	0.780140	0.812590
2	XG Boost	0.993643	0.996313	0.987974	0.889447	0.896151
3	SVM	0.980766	0.846637	0.856929	0.692638	0.508757
4	Decision Tree	0.997275	0.999751	0.998868	0.996456	0.997550

Table 1: Model Score Summary

In general, some appliances such as `ac`, `ev` and `oven` obtained high F1 scores while others such as `wash` and `dryer` don't. The lowest score was when predicting the `dryer` usage using the SVM model. On the other hand, the highest F1 score was when predicting `ev` using the Decision Tree Model.

Finally, the following table shows the highest F1 score for each appliance and the best model chosen for the prediction task.

	Appliance	Best_Model	F1_Score
0	ac	Decision Tree	0.997275
1	ev	Decision Tree	0.999751
2	oven	Decision Tree	0.998868
3	wash	Decision Tree	0.996456
4	dryer	Decision Tree	0.997550

Table 2: Final Model and F1-Score

Decision Tree Model obtains the highest F1 score for all the appliances. For comparison purposes, after scoring each model, the predicted outputs were submitted and loaded to Kaggle competition in order to see whether the predictions were computed correctly, and the format of submission were correct.

6 Conclusion

We have built 5 models which include **Logistic Regression, Random Forest, XGBoost, SVM and Decision Tree.** Each of them was built for predicting target variables such as **`ac`, `ev`, `oven`, `wash` and `dryer`** which correspond whether those appliances are turned on or off. For measuring the performance of each model, a F1 score were used for comparing which model is the best for each appliance usage prediction.

Originally, the dataset given was processed as a time series when the order of rows had significance. We tried to use Tsfresh package for rolling time series and feature extraction but it was not used in the final implementation as it was optional, so we decided to include the section just in case a deep analysis wants to be carried out.

The model chosen for making predictions in all appliances was `Decision Tree` because these models gave the highest value of F1 score for all the appliances. For getting better results, this model's hyperparameters might be tuned for improving its F1 score. The tuning process could be included for each model too.

The Decision tree classifier works well for continuous and categorical variables which suit our dataset. Decision Tree solves the problem of machine learning by transforming the data into a tree representation. Each internal node of the tree representation denotes an attribute, and each leaf node denotes a class label. They provide a clear indication of which fields are most important for prediction or classification. It requires less computation and can handle large datasets.

From this project we learnt how to conduct time series analysis, data processing, EDA, and various feature engineering methods for a machine learning model, apart from this we learnt to differentiate between various classification models such as traditional and ensemble methods and implement hyperparameter tuning for the models. We also learnt the problem of imbalanced classification and how techniques like SMOTE and downsampling can be used for securing a better F1 score.

If the test dataset were ordered, we could have applied the Tsfresh package for feature extraction and have applied PCA to carry out a better analysis. Furthermore, we wished to implement a time series forecasting framework and feature extraction along with Recurrent Neural networks and Long Short-Term Memory neural networks as a future work for better forecasting and comparing the f1 score with the traditional machine learning and deep learning models.

References

- https://stackoverflow.com/questions/65682019/attributeerror-str-object-has-no-attribute-decode-in-fitting-logistic-regre
- https://machinelearningmastery.com/feature-selection-machine-learning-python/
- https://tsfresh.readthedocs.io/en/latest/text/feature_extraction_settings.html
- https://tsfresh.readthedocs.io/en/latest/text/forecasting.html
- https://tsfresh.readthedocs.io/en/latest/text/feature_filtering.html
- https://tsfresh.readthedocs.io/en/latest/text/quick_start.html
- https://medium.datadriveninvestor.com/time-series-classification-using-feature-extraction-16209570a22e
- https://towardsdatascience.com/ultimate-pandas-guide-time-series-window-functions-a5362b782f3e
- https://towardsdatascience.com/time-series-analysis-resampling-shifting-and-rolling-f5664ddef77e
- https://www.kaggle.com/malekzadeh/smart-home-data-processing-weather-vs-energy
- https://www.kaggle.com/msand1984/appliance-energy-prediction
- https://github.com/pipette/Electricity-load-disaggregation/blob/master/python_notebooks/Sandbox%20EDA%26HMM.ipynb
- https://www.dataquest.io/blog/tutorial-time-series-analysis-with-pandas/
- https://www.earthdatascience.org/courses/use-data-open-source-python/use-time-series-data-in-python/date-time-types-in-pandas-python/resample-time-series-data-pandas-python/
- https://regenerativetoday.com/a-complete-guide-to-time-series-analysis-in-pandas/
- https://www.edureka.co/community/46099/advantages-of-using-decision-tree#:~:text=Decision%20trees%20generate%20understandable%20rules,important%20for%20prediction%20or%20classification.
- https://www.activestate.com/blog/comparing-decision-tree-algorithms-random-forest-vs-xgboost/#:~:text=One%20of%20the%20advantages%20of,classification%20and%20regression%20predictive%20models.
- https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a