RESIDENTIAL ENERGY APPLIANCE CLASSIFICATION

Creator name: ZHIYIN WANG

Abstract

Energy efficiency is critical for transforming future energy systems and slowing the rapid growth of global energy consumption. To combat climate change and reduce the usage of fossil-fuel power plants, energy efficiency in our residential sectors become an essential part of area as energy consumption in residential sectors take more than 20% of total consumption. This paper studied the energy use of five different appliances over time in the residential sectors. The aim is to develop multiple algorithms to classify each appliance that can accurately predict the use of appliances in each time frame and features. To find classify models that have the best prediction power for each appliance, multiple classification algorithms are applied and analysed. Also, the study will go through different features in the dataset, analysed and investigated to identify important features that are influential in the modelling process. In the final decision of models, models are compared by error measurements such as F1 score, the model achieve a high F1 score are considered to have an overall high performance among all models.

Table of Contents

1. Introduction

This section describes the general information of the project, includes the background of the project, the main goal of the project and how the work is allocated in the study group.

1.1 Background

In the past few years, the concept of energy saving has been applied more frequently into various areas in people's life, such as the rise of Tesla and solar energy. However, climate change remains as one of the major issue human races have to face all together. In front of the global warming caused by human-induced emissions of greenhouse gas, energy efficiency become an essential part in slowing the growth of energy consumption. Residential sectors have taken more than 20% of the total energy consumption, which means the efficiency of energy used in our daily life has become a priority. To increase energy efficiency at our residential, the energy consumption in daily life needs to be investigated and modified to avoid over consumption and waste of energy.

1.2 Goal

In this project, data contains energy consumption of different appliances in residential are provided. The appliances include air conditioner, electric vehicle charger, oven, cloth washer and dryer. All the appliances are high energy consumption appliances in the residential sectors. The aim of this project is to develop multiple classify models for each individual appliances that have a high level of prediction power. Classification models with high accuracy can assist in the development of energy-saving technology for residential buildings.

Overall, five classification models are required to be developed and each model is corresponded to one appliance. The models should be machine learning methods that takes features and labels as input and produce accurate predicted labels as output. The five models should be able to detect the five appliances individually as accurate as possible.

1.3 Workload allocation

The workload can be separated in to following areas.

- Exploratory data analysis
- Feature analysis
- Model development for the five appliances
- Model analysis
- Model evaluation and comparison
- Report writing

2. Pre-processing and feature analysis

2.1 Data

Two datasets are given, train dataset with labels and test dataset without labels. Most of our model development will be focus on the train dataset. There is total 16 columns and 417720 rows in the train dataset. 16 columns include five target variables represent each of the appliance, one index variable and ten predictors. Then, the class of the variables are checked, all the target variables are in integer class. Predictors apart from “dayofweek” are in numeric class, “dayofweek” is in character class. Summary of the dataset is investigated to understand the distribution of each variable. The variable “dif” is the only variable that contains negative value, all other variables contain positive values and some of them includes zero.

2.2 Pre-processing

Though investigation on the data structure, it is observed that the target variables are in numeric class. To feed these target variables into our classification models, it is necessary to change the class into factor. Function as.factor() is used to change the type of class into factor. Apply log transformations on variables that have zero and negative values produce NaN value, it is necessary to add a minimum value to those variables and make all values become positive if transformation is required on the data. The function sum(is.na) is used to check for any NA values in the dataset, any NA value is removed for the convenience of modelling. Result shows there is no NA value in both train and test data.

2.3 Feature analysis

The feature in the dataset includes:

Load: the electricity consumption record from smart meter for the time (once per minute)
Dif : the difference between two sequential load data points.
Absdif: Absolute value of dif.
Max: the maximum load recorded in each timeframe
Var: Variance of load over a neighbourhood time window of 30 minutes around each load data
Entropy: The Shannon entropy that measures the “forecastability" of a time series data.
Nonlinear: The nonlinearity coefficient is used in Terasvirta's nonlinearity test.
Hurst: The hurst is used as a measure of the long-term memory of a time series.
Hourofday: The hour of the record, each hour is represented by a number between 0 to 23
Dayofweek: The day of the record, represent by Mon to Sun.

Features are plotted by histogram hist() function to further investigate their distribution. It is found that load, absdif, var, nonlinear have a right skewed distribution, and hurst has a left skewed distribution.

After applying transformations to the skewed variables to achieve a normal distribution, the effective transformations are log(load), sqrt(var) and log(nonlinear), these results produces a better shape close to normal distribution. The above transformations can be used in models need assumption of normal distribution to increase the accuracy.

The correlation between all variables are investigated using function cor(). “Load” and “ac” has a high correlation of 0.9 and “ac:max”, “ac:var” has a correlation around 0.5. Apart from these three pairs, all the other pairs have very low correlation, indicates it would be hard to build a good linear model based on given features for classification task. Despite of the low correlation between target variable and features, “load” appears to have the highest correlation on average with target variable. “Max”, on the other hand has the second highest correlation on average with target variable. The correlation analysis indicates “load” and “max” may have a significant impact on the appliance usage.

2.4 Additional data processing

In the process of model developing, the imbalance classification problem is identified through prop. table () function. It is observed that the labels target variables are imbalanced, where label 0 outnumbers label 1 by a large proportion. For example, in target variable “ev”, 99% of the label is 0. The imbalanced problem in the target variable strongly affects the performance of machine learning algorithms. The unequal distribution can cause our classifier to get biased towards the majority class label, in this case label 0. On the other hand, machine learning intends to work well with balanced class distribution, it is essential to solve this imbalance issue in the data to achieve a high accuracy in classifying label 1 correctly.

Sampling Methods are applied to resolve this issue. There are multiple types of sampling method available. The method applied in this study is under sampling. The under-sampling approach reduces the number of observations in the majority class to make the dataset become balanced. Some models are generated from the original dataset, and some are generated from the dataset processed with under sampling method. Under sampling is achieved by using ovun.sample () function from the ROSE package. The detail application and comparison of different sampling method are discussed later in the modelling process.

3. Models and model setups

3.1 Measurement of evaluation (F1 score)

The error measurement used in this project is F1 score. F1 score is a measurement of model’s accuracy on the classification of the dataset. The positive label in this project is set to label = 1. F1 score is the combination of precision and recall of a model. It reflects the ability of a model predicting the positive label. In this project where the labels are extremely imbalanced, F1 score is a better measurement than accuracy score. Accuracy score tends to become bias by the high proportion of negative predictions in labels. For example, 99% of the label is negative(0), if the model predict all labels to be 0, then the accuracy score will be 99% despite the fact model is not able to correctly make prediction on any positive label. Therefore, F1 score is a the suitable measurement in this project and it will be used as a measurement to compare between different models. Also, confusion matrix is produced using confusionMatrix() function in some scenarios to investigate the details of the predicted labels. Confusion matrix will not be used for comparison between models, it is used in model developing process to check the true positive and false positive values to have a better understanding of the proportions of labels predicted.

3.2 Logistic regression model

Logistic regression is a simple classification algorithm that models the probability of default. For example if default in this case is label 0, the output of logistic regression is the probability of the class of 0 with given predictors. The results are between 0 to 1, and usually determined with a 0.5 thresh hold. The model uses logistic function to model a binary classification problem. The process of logistic modelling is to estimate the values of parameters using maximum likelihood. The parameters are the β values in the logit function. Each term/features feed into the logistic model has a corresponding β value and p value. The p-value tests the null hypothesis of coefficient is equal to zero, indicates the significance of β value. A p-value is usually test against a 0.05 threshold, values less than 0.05 means a rejection of the null hypothesis, and the feature/predicator should not be included in the model.

The assumption of logistic regression model includes: the independence of errors, the linearity in the logit: independent variables need to be linearly related to the log odds, no multicollinearity between independent variables, and the absence of strongly influential outliers.

Setups:

In this project, logistic regression is implemented to the classify the five target variables. The model is first implemented with all features included. The summary of the model is checked for any p-values less than 0.05. Coefficients with a p-value less than the significant level will be removed in the model. After the model is checked, F1 score is computed for the model and saved for comparison. Transformed variables: log(load), sqrt(var) and interactions between variables that are considered may have an influence on the classification task is added to the model features intends to boost the model performance. The summary of the new model is then checked under the same process and repeat until no more features can be add/removed to increase the F1 score. The final model with the best F1 score is recorded to compare with other type of models.

3.3 Classification tree

Classification tree is one type of decision tree that used to predict a qualitative response. In the classification tree, training observations are used to train the tree and grow into recursive binary splits. When an observation is feed into a tree, the tree try to assign the observation to the most commonly occurring class with respect to the learned boundaries of the features. The classification error rate used in classification tree is the misclassification rate of the results, which is the amount of fraction of the observations that does not belong to the most occurred class the tree assigned to.

Random forest is implemented in this project, it is a special form of bagging trees. In bagging trees, multiple decision trees are built on bootstrapped training data. Random forest improves bagging trees by changing the number of predictors used in a single decision tree from all predictors to just part of the predictors. Usually, for classification task, square root of total number of predictors are used in a single split.

Boosting is very similar to bagging and random forest; the difference is boosting does not involve bootstrap sampling and the trees are grown sequentially. The trees in boosting are grown using information gain from a previous tree. Same as bagging trees and random forest, a parameter of number of trees needs to be determined. In additional, there is a shrinkage parameter λ in boosting that controls the learning rate of boosting trees.

The assumption in decision tree is simple, the dataset used in a tree is considered as the root of the tree.

Setups:

Both classification tree and random forest models are implemented to the modelling of all five target variables. Boosting was only applied to the classification on “ev”.

All features are used for all tree models as tree models does not need feature selection. The classification trees are fitted with function tree() first and to get an idea of the prediction power as a bench mark. Then cross validation is applied using function cv.tree() and prune the tree to avoid overfitting and remove redundant splits in trees. Results of tree pruning are visualized on plot and the tree size with the smallest cv error is used to produce prediction and calculate the F1 score. Summary of the tree model is checked for the number of actual predictors used in the model and the tree is also visualized though plot(tree) to visualize the importance of each variable in the tree. The predictors used on top nodes are most important.

Random forest is fitted using function randomforest(), with number of trees set to 25 for the entire dataset and 1000 for dataset after under sampling. Number of trees is set to 25 as the original dataset are too large, model training with number of trees over 25 requires too much training time. With data reduced by under sampling the number of trees can be increased. Mtry, the number of predictors are set to the square root of total number of parameters. With trained random forest model, the importance of predictors are visualized on plots using varImpPlot() function. Then F1 scores are computed for each of the models and recorded for comparison. train() function can be used to tune the ntree and mtry parameters in the modelling process but due to the limited computation power, the function seems to run very slow on the device. Thus, the parameters are manually tuned instead.

Boosting is only applied to “ev” using train () function with method = “gbm”. A list of lambdas between (0.001 to 10) is generated to find the best shrinkage. Also, cross validation is applied in gbm() function with cv.folds = 5 and the best shrinkage. The aim of the cross validation here is to find the best number of trees through comparing Bernoulli deviances. Then the boosting model is fitted with best number of trees and best lambda, F1 score is calculated and record for comparison.

3.4 Support vector machine (SVM)

Support vector machine is an classification approach used to capture non-linear boundaries between two classes. It is an extension from support vector classifier with an selection of kernel. The decision on kernel determines the computational approach in the modelling process. Kernels includes: “linear”, “polynomial”, “radial” etc. Using linear kernel is equivalent to the support vector classifier. Using polynomial kernel, leads to a more flexible linear boundary, where number of degree needs to be determined. There are two parameters in svm, cost and gamma in addition for non-linear kernels.

Setups:

Svm is applied to the classification of all five appliances on data after under sampling. The computation power on the device does not allow svm () to apply to the original dataset, due to the huge amount of data and memory requires. The svm () function is used to train the svm models, “kernel = linear, polynomial and radial” is tested separately to find out the optimal kernel for the classification tasks. Then, parameters are tuned manually, as tune () function seems to work extremely slow on my device for unknown reason. F1 scores are calculated for each parameter settings, the best performed gamma and cost is then used to compute final F1 scores of svm model for each appliance.

3.5 Neural network

A neural network is a net that consists multiple layers of neurons, each layer receiving inputs from previous layers, pass the outputs forward to the next layer. A simple neural network is implemented in this project with one hidden layer. The hidden layer intends to analyse the input features and extract the input values into k number of hidden units and produce predictions as output.

Setups:

H2o library is used to apply neural network classification on “ac”. The train and test data is read as h2o objects and a subset only contains “ac” label and features are extracted from the two objects. Then h2o.deeplearning() function is used to train the neural network which takes a 2:ncol(data) as X inputs (only features, without labels, label is the first column). The parameters are the number of hidden units k, the number of epochs. A range of k between 2:10 is tested, as there are 10 predictors in the dataset. Number of epochs is set to 50. The activation function used in this project is “Tanh” and the L2 penalization is applied through “l2 = 0.1” to avoid overfitting. F1 scores are then computed for models trained with each k value. The model with the highest F1 scores is the model with the best k number of hidden units. The result of the model with that k number of units is then recorded for comparison.

3.6 Discussion of model differences

Among all models, logistic regression is the easiest to implement, it requires the least amount of time to train and is the most interpretable algorithm. It can also be regularized to prevent overfitting. However, compare to decision trees and svm, logistic regression is constrained by its assumptions of linearity and multicollinearity. On the other hand, logistic regression tends to underperform on non-linear boundaries.

Decision trees transforms classification problem into tree representation, and it does not have any assumption on the given data. No feature selection is required in tree models as machine learning algorithm can select the predictors based on their importance automatically. However, the single tree grown in the decision tree method suffers from bias and variance and a change in value can leads to large differences in results. Bagging trees improve this by create serval subsets of data and grown multiple trees. Random forest further improves the issue where all the bagging trees used same predictors which leads to the high correlation between the predictions produced by the trees. Random forest optimized on this issue by only using some predictors in each split, so each predictors have a chance to contribute to the prediction, where in bagging trees strong predictors completely dominate the prediction. Therefore, in random forest, the trees are less correlated. One limitation of the tree models is the stability of the model. A small change in the data can result in a significant different in the structure of the trees. Thus, if the tree models are trained with a imbalanced data, when models are test against test data, the tree may be bias and not generalized enough to classify real world problem.

Support vector machine also have no assumption on the dataset, it works well for data with high dimensional spaces. In this project, the data has more than two dimensional spaces, svm is suitable to perform the task on this aspect. However, training svm is extremely expensive especially on a large dataset. Svm works well when the number of dimensions is more than the number of samples. In the project, svm is only applied to the data after under sampling. With reduction on the data size, svm can be trained faster with a relatively better performance compared to other models.

4. Experiment results

All models were initially developed on the original train data. In the table below, for target variable ac, all models applied received a high F1 score, where random forest has a perfect score of 1. However, when applying the same models to the other four variables, the models show an low performance according to F1 score, but Random forest still gets a F1 score that is close to 1. (The result of random forest on original data is not recorded in the table below, as they are useless) The unusual F1 score of random forest models leads to suspicion of the model accuracy on test data. Therefore, a output prediction produced by random forest was uploaded to Kaggle for testing. The score was 0.67 which is not as high as expected. After investigation on the dataset, it is to be found that the label distribution in “ev”, ”oven”, “wash” and “dryer” are highly imbalanced. Although random forest is capable of successfully classify the labels correctly in train data, as tree models are highly sensitive to small change in values, it underperforms on the test dataset. The method of under sampling is then implemented and random forest model and svm models are reproduced using the train dataset after under sampling process. The result F1 score from the new dataset are lower compare to the results from the original dataset but the model are more generalized as training with balanced data improves the performance of ML algorithms. For svm models, they are trained with the optimal cost and gamma. It is found that an increase in gamma increases the F1 score on train data, but also increase the risk of overfit which decrease the predict accuracy on test data. In considering of this issue, only comparing results from under sampling, random forest outperforms svm in all five tasks. Therefore, the best performing classifier among all models is the random forest model with under sampling method on train data. The Kaggle submission result also supports random forest with a score around 0.85. On the other hand, logistic regression and classification tree only performed well on "ac" classification, both of the models result in a 0 F1 score when modelling "wash" and "dryer" indicates these two models are unable to model a significant imbalance data as the algorithms are strongly influenced by the large amount of majority labels in training and prediction process.

Target variable	Train data	Model	F1 score
ac	Original data	Logistic regression(all features + log(load)+ sqrt(var))	0.9642
		Classification tree	0.9514
		Random forest(ntree = 25, mtry = 3)	1
		Neural network (k = 2)	0.9623
	Under sampling	Random forest(ntree = 200, mtry = 3)	0.9922
ev	Original data	Logistic regression(all features + log(load)+ sqrt(var))	0.0749
		Classification tree	0.5617
		Boosting	0.0042
	Under sampling	Random forest(ntree = 1000, mtry = 3)	0.3491
		Svm(kernel = radial, cost = 1000, gamma = 2)	0.2439
oven	Original data	Logistic regression (all features + log(load)+ sqrt(var))	0.3499
		Classification tree	0.5851
	Under sampling	Random forest(ntree = 1000, mtry = 3)	0.4925
		Svm(kernel = radial, cost = 1000, gamma = 2)	0.3700
wash	Original data	Logistic regression (all features + log(load)+ sqrt(var))	0
		Classification tree	0
	Under sampling	Random forest(ntree = 1000, mtry = 3)	0.2035
		Svm(kernel = radial, cost = 500, gamma = 2)	0.1904
dryer	Original data	Logistic regression (all features + log(load)+ sqrt(var))	0
		Classification tree	0
	Under sampling	Random forest(ntree = 1000, mtry = 3)	0.3683
		Svm(kernel = radial, cost = 1000, gamma = 2)	0.3278

The best F1 score in each classification task is highlighted in the table above. Although, some results of the model fitted on the original data have higher F1 score. The actual performance of the model on the test data remains poor. It is discovered that when dealing with imbalanced data, model developed from original train data tends to be bias towards the majority label. Hence, it emphasis the importance of using sampling method and balance the data as close to original data as possible.

5. Conclusion

Through comparison of multiple model performance, random forest model is determined to be the final model of this project. The model successfully reduced the issue of overfitting issue that occurs in decision trees. Also, the tree form of decision making enables it to handle non-linear boundaries between the clusters and outperform logistic regression models. Compare to another great ML algorithm svm, it is more capable of perform well in classification modelling with a large train dataset. When facing imbalanced problems in data, under sampling method is implemented and resolve the issue of the biased performance of classifiers towards majority class. Overall, random forest outperforms the rest of the model on both F1 scores and the ease of implementation.