FIT5149 S1 2022 Assessment 2 Report Group No. 68

Residential Energy Appliance Classification

AUTHOR1 : Muxuan Hu
AUTHOR 2: Sang Ye
AUTHOR 3: Yifan Wang

April 27, 2022

Abstract

In recent years, as public attention to the environment increases due to the excessive emissions of fossil energy from cars, some experts are gradually thinking about the alternative resources to fossil-fuel. As a result, the electric vehicles may be the preference, because electricity is an environmentally friendly and renewable resource. However, the challenges of this through will challenge the city's electricity supply system. In this case, to do the survey related to the charging habits of the customers could cope with this kind of challenge to some extent.

In this project, we will firstly do the EDA engineer to do the feature extraction and selection. After that, the focus of us is to compare different classifiers based on this project and finally select the Light Gradient Boosting Machine (LGBM) to detect whether the current user is charging the electric vehicles. Meanwhile, the performance of the LGBM is amazing, which may be a powerful tool for the future prediction tasks.

Content

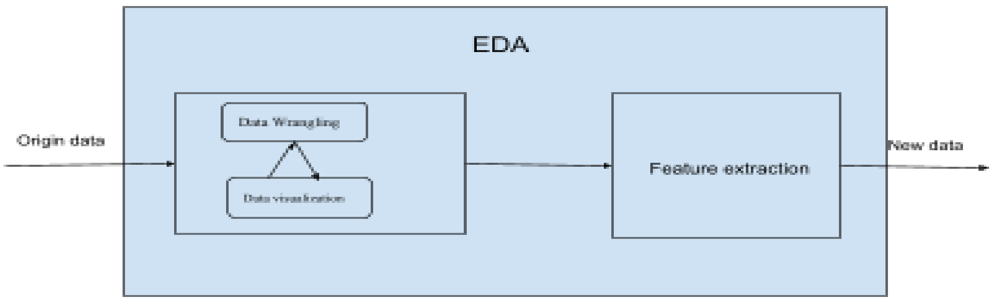
1 Introduction

The burning of fossil fuels releases a large amount of carbon dioxide, which will lead to global warming and thus affect environmental changes. Electric vehicles (EVs) are widely loved as a green energy source. With the increasing demands for electricity such as: new energy vehicles, furniture, these will cause unexpected peak load and voltage problems in the energy distribution network. To better prepare for EV charging in the system, load monitoring provides detailed electricity consumption information of individual appliances or to detect appliance use.

In this project, Our group uses three different models to predict labels(whether EV is being charged or not) by using the data from the household total energy consumption including: dataid, localminute, total_load. Muxuan Hu will mainly contribute to the EDA part, Sang Ye will focus on the feature selection based on the EDA and Yifan Wang will choose three different and suitable models to finish this classification task. Finally, We will give our conclusion about model performance and load detecting.

2 Pre-processing and features used

In this part of EDA, we mainly use pictures and text for data cleaning and data visualisation. During this process, we will analyse the relationship of each attribute, which is a preparation for the subsequent feature selection and cleaning.



2.1 Data Wrangling

Before we do feature engineering, we need to clean the data, including operations such as deleting duplicate values and detecting data anomalies. In the training dataset, a small percentage of total_load values are negative. Since the amount of data is very large and the continuity of time cannot be broken, we choose to directly become positive here.

In addtition, We also changed the localminute property to the standard time format in pandas, which provides traversal for our subsequent operations.

2.2 Data Visualisation

2.2.1 Time series decomposition

One of the traditional methods to analyse the time series is the time series decomposition, which can reduce our time series data into its following three components ("Different Types of Time Series Decomposition", 2022) :

1. trend : The trend of a time series refers to the general direction in which the time series moves. As we can see the figure 2.2.1, It is obvious that the peak electricity consumption occurs from June to September.
2. Seasonal: the seasonal component refers to data that rises and falls at consistent frequencies. We can find out that ,from June to September, The slope of the curve obviously changes rapidly, and the peak is higher than the others. In addition, It is very interesting to see that there are some peaks in May
3. Raised: some of the data can not be explained by using trend and Seasonal. As can be seen from the Raised chart, the reason why there will be a similar trend in May compared to August and September is because there are some outliers.

Overall, The peak electricity consumption period for users tends to occur from July to September. There are also outliers at some point in May.

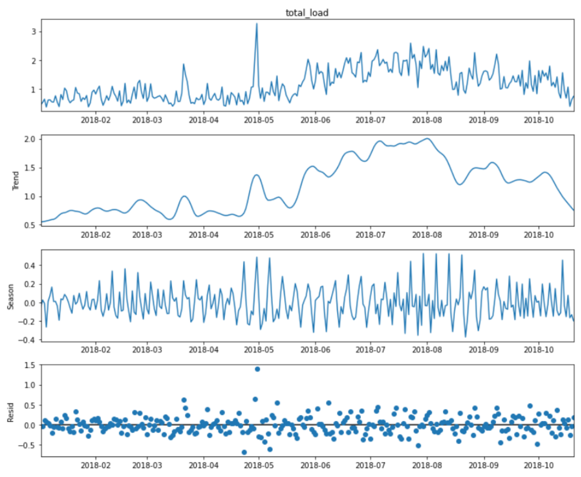


Figure 2.2.1 the decomposition of the time series

2.2.2 the headmap about month and week

Through the above analysis, we know that month has a great relationship with electricity consumption, so we also draw the relationship between week and month (Figure 2.2.2). From this graph, it is clear that there is no strong relationship between week and electricity consumption, whose colours are the same every week. So, We don't need the property of the week.

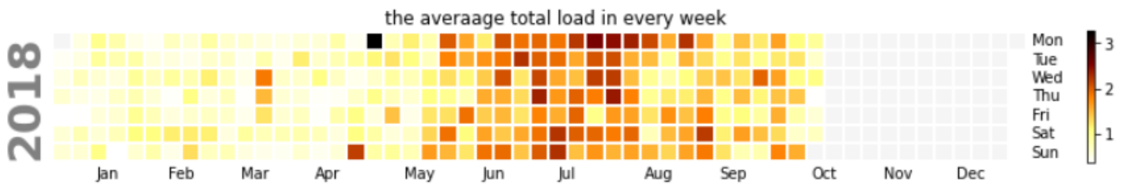


Figure 2.2.2 the headmap about month and week

2.2.3 Hourly Radar Chart

After we analyse the month, day and week, we have to put our attention into the hour. We created a radar chart for about an hour, seeing figure 2.2.3. It is very straightforward that most people tend to use electricity in the afternoon from 14:00 to 19:00 and get the same answer that June to September are the main electricity consumption month:

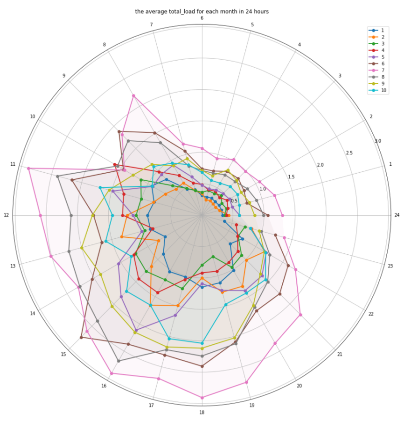


Figure 2.2.3 Hourly Radar Chart

2.3 Feature extraction

Because the raw data only contains the 'localminute' and 'total_load' features, we would like to dive into the feature extraction according to the existing featuers. According to the distribution of the label shown in Figure 2.3.1, the uncharging time (target = 1) is much longer than the charging time. In this case, we would like to extract some new information according to the existing features.

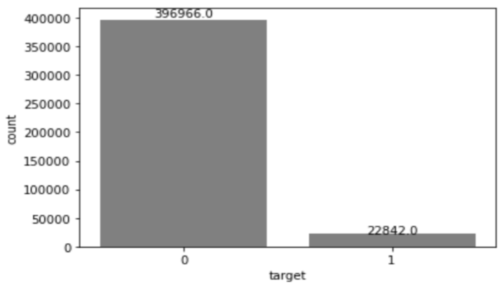
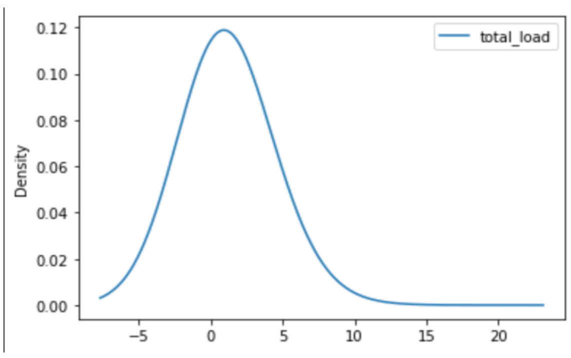


Figure 2.3.1 distribution of target label

Firstly, we will dive into the analysis of total_load column. As are shown by figure 2.3.2 and 2.3.3, most values in total_load column are centred between 0 and 2. Meanwhile, from the perspective of statistics, the values that exceed Q3+1.5IQR could be regarded as outliers. However, from the figure 2.3.3, there are a large number of outliers. So, we would like to extract this part of values and explore the relationship between them and the target label as is displayed in figure 2.3.4.



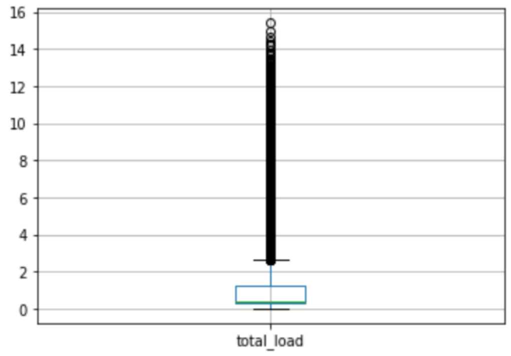
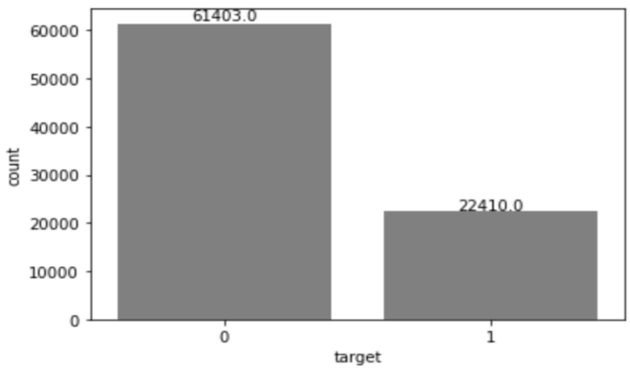


Figure2.3.2 the distribution of total_load values *Figure 2.3.3* boxplot of total_load

By combining the figure 2.3.1 and 2.3.4, it could be seen that most of the charging time occurred when the total_load is high. In this case, we could assume that there is a strong relationship between total_load and target label. Subsequently, we calculated the average charging time and the corresponding average total_load as is shown in figure 2.3.5.



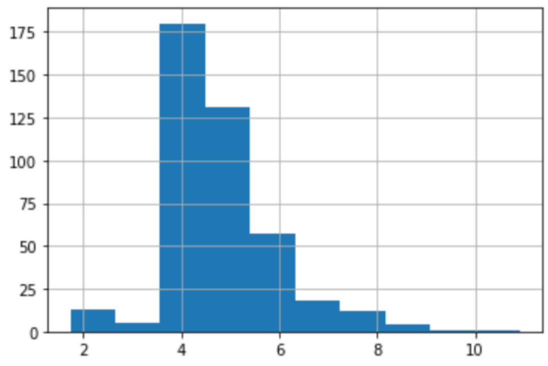


Figure 2.3.4 The distribution of target label based on the total_load 'outliers

Figure 2.3.5 distribution of average total_load when charging

In summary, we insert a new boolean feature named 'high_load' determined by whether total_load exceeds 4.15.

After that, we would focus on the 'localminute' column, the information inside it could be divided into four parts: year, month, day, and time. However, year data can be ignored since all data are of the same year. Similarly, the data is a kind of streaming data and the basic time unit is minutes. As a result, it could be assumed the minute and second data may not be useful for this project. So, to extract the month, day, and hour data may be our preference. The rationality of extracting these features will be illustrated in Figure 2.3.2

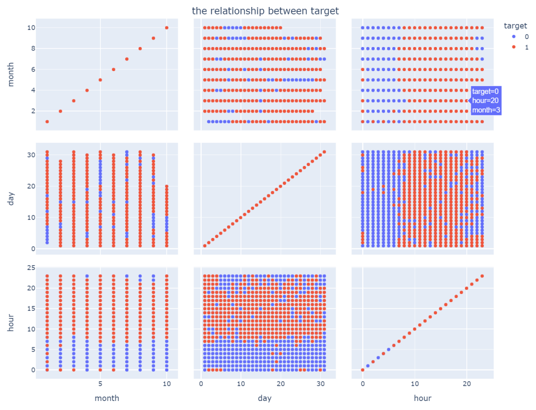


Figure 2.3.2 Correlation between label and some time features

From figure 2.3.2, it could be observed that there are some regular distributions of the target label. For example, when the hour is between 8am and 23pm, the label is one most times, and the similar situation also appears when the month is between June and August. As is aforementioned, the total load between 23 and 8 o'clock (usually be regarded as midnight) is much lower than the other times. Also, the lower the total_load is, the less likely the target label is one. As a result, we could make the assumption that when the hour is between 23 pm and 8am, it is likely that the target label is zero. In summary, we insert a new boolean feature named 'mid_night' determined by whether localminute is between 23 pm and 8am.

3 Models

3.1 KNN

The full name of KNN is K Nearest Neighbours, which means K nearest neighbours. Its working principle is to use the training data to divide the feature vector space and use the division result as the final algorithm model. There is a sample data set, also known as a training sample set, and each data in the sample set has a label, that is, we know the correspondence between each data in the sample set and the category to which it belongs.

After inputting unlabeled data, compare each feature of this unlabeled data with the feature corresponding to the data in the sample set, and then extract the classification label of the data with the most similar features (nearest neighbour) in the sample.

3.2 XGBoost

XGBoost algorithm is a tree-based boosting algorithm. Unlike GBDT, XGBoost only uses the information of the first-order derivative, while XGBoost does a second-order Taylor expansion of the loss function, and adds a regular term to the objective function to weigh the complexity of the objective function and the model to prevent overfitting. .

The core algorithm idea of XGBoost is not difficult, it is basically:

1. Continuously add trees, and continuously perform feature splitting to grow a tree. Each time you add a tree, you learn a new function f(x) to fit the residual of the last prediction.
2. When we get k trees after training, we need to predict the score of a sample. In fact, according to the characteristics of this sample, each tree will fall to a corresponding leaf node, and each leaf node will correspond to a score.
3. Finally, you only need to add up the scores corresponding to each tree to get the predicted value of the sample.

In addition, the design concept of XGBoost has the following advantages: Fast, Portable write less code and Fault tolerant.

3.3 LightGBM

Light Gradient Boosting Machine is a distributed gradient boosting framework based on a decision tree algorithm. To meet the needs of the industry to shorten the model calculation time, the design ideas of LightGBM are mainly two points:

Reduce the use of memory for data to ensure that a single machine can use as much data as possible without sacrificing speed.
Reduce the cost of communication, improve the efficiency of multiple machines in parallel, and achieve linear acceleration in computing.

3.4 Discussion of model difference(s)

KNN is a typical non-parametric method. It is a very simple algorithm, but it has high computational complexity and high space complexity, and the classification effect is not ideal when the samples are unbalanced.

XGBoost is a method based on pre-sorting. It will calculate the feature value of each unique. The advantage is that it can find a specific feature value as a segmentation point. The disadvantage is that the calculation and storage costs of this method are both Very big. And the particularly accurate segmentation points found in this way may have over-fitting, and the method of growing each tree is to grow by layers, that is, level-wise learning. Layer-by-layer growth will bring a lot of time benefits. Each layer can operate on all data, but there will be some unnecessary operations, such as some nodes not needing to split calculations.

LightGBM adopts a histogram-based decision tree algorithm, which converts traversal samples into traversal histograms. It not only reduces the memory usage, but can use the histogram to make a difference to reduce the computational complexity

The LightGBM algorithm uses the Gradient One-Side Algorithm (GOSS) to filter out samples with small gradients in the training process, reducing a lot of computation. GOSS first sorts all the absolute values of the features to be split in descending order, selects some data with the largest absolute value, and then randomly selects some data from the remaining smaller gradient data.

Then multiply this data by a constant, so that the algorithm can pay more attention to the under-trained samples without worrying about changing the distribution of the original data set.

LightGBM adopts the growth strategy of leaf-wise algorithm to build the tree, which reduces a lot of unnecessary calculations. XGBoost adopts a level-wise growth strategy, which traverses the data once and can split the leaves of the same layer at the same time, although this is convenient for multi-threaded optimization, controls the model complexity and is not easy to overfit. However, this strategy treats the leaves of the same layer indiscriminately and is very inefficient. The leaf-wise algorithm adopted by LightGBM only splits the leaf node with the largest current split gain each time. This method can easily grow deep trees and cause overfitting, so a limit is generally imposed on the maximum depth of the tree.

It may grow a deep decision tree, resulting in overfitting. LightGBM is a bias-based algorithm, so it is sensitive to outliers. In finding the optimal solution, the idea that the optimal solution is a synthesis of all features is not considered.

4 Experiment setups

For the KNN model, we need to find only a suitable value of K.

In the LightGBM model, since the target to be predicted is a binary classification, first set the objective to binary and the evaluation matrix to binary_logloss. However, in the original data set, the ratio of the number of positive and negative labels is too large, so set scale_pos_weight to 17, that is, the ratio of positive and negative labels in the original data set. The learning rate is set to a small 0.01 and the regularisation parameter α defaults to 1.

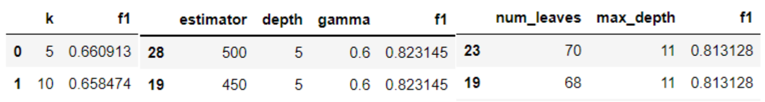
Num_iterations specifies the number of augmentation iterations (the tree to build), 500 is enough by looping. max_depth sets the tree depth. In addition, because LightGBM uses the leaf-wise algorithm, num_leaves is used when adjusting the complexity of the tree. These two parameters are looped to find their better combination.

The XGBoost model is similar to LightGBM parameters, with an objective function set to binary:logistic, learning rate of 0.1 and min_child_weight, the minimum weight sum of all observations of a subset, is 1. In addition, the number of loops n_estimators and the maximum depth of the tree max_depth and the loss function reduce the threshold gamma to find the best combination by looping.

A cross-validation function was used during all model training. Since it is time series data, the original order cannot be disrupted. Divide the complete dataset into small datasets of the same length by writing a function and take the first 80% of each small dataset as the training set and the remaining 20% as the test set. Each training result on a small dataset uses the f1_score function in the sklearn.metrics package to calculate the f1 value and finally average it.

5 Experimental results

When the above three models are put into Blocked Cross-Validation, we can see the parameters of each model with the highest F1 score(Figure 5.1). It is obvious to see that the LGBM model is the best monetizer, around 82%. So, we finally chose LGBM as our final model.



	k	f1
0	5	0.660913
1	10	0.658474

	estimator	depth	gamma	f1
28	500	5	0.6	0.823145
19	450	5	0.6	0.823145

	num_leaves	max_depth	f1
23	70	11	0.813128
19	68	11	0.813128

Figure 5.1 the F1 score for K-means, LGBM and XGBOOST

6 Conclusion

To sum up, LGBM is the best model in this project. Through this classification task, we learned that the PCA is not particularly friendly to discrete variables. When data is required for processing, traditional cross-validation is not adapted , which needs to consider time-series relationships. In the fugure, In dealing with timing problems, we should learn neural networks such as RNN, LTSM, etc. These models have strong memory capabilities. In addition, we also need to deepen our understanding of traditional models to help us better perform parameter tuning and model selection.

7 References

[1]A Visual Guide to Time Series Decomposition Analysis. (2022). Retrieved 27 May 2022, from https://betterprogramming.pub/a-visual-guide-to-time-series-decomposition-analysis-a1472bb9c930

[2]Cross Validation in Time Series. (2022). Retrieved 27 May 2022, from https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4

[3]Different Types of Time Series Decomposition. (2022). Retrieved 27 May 2022, from https://towardsdatascience.com/different-types-of-time-series-decomposition-396c09f92693

[4] Miller, G.A.: WordNet: a lexical database for English. Communications of the ACM 38(11), 39–41 (1995)

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionally. In: Ad- vances in Neural Information Processing Systems 26, pp. 3111–3119 (2013)

[6] Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Meth- ods in Natural Language Processing, pp. 1532–1543 (2014)

[7]Zach, V. (2022). How to Show Values on Seaborn Barplot (With Examples) - Statology. Retrieved 27 May 2022, from https://www.statology.org/seaborn-barplot-show-val