Wining the Education Subsidy Prediction Challenge with Models Ensemble and Feature Engineering

Xuan Yang A0159066X
School of Computing, NUS
yancy100696@gmail.com

Zhendong Liu A0159369L
School of Computing, NUS
zhendong.nus@gmail.com

ABSTRACT

We have won the first prize with CN￥30,000 (or equivalently SG$6,000) bonus of a public machine learning challenge named National Big Data Innovation Competition Algorithm Competition(Education Topic)[1] hold by the Chengdu government in China in March 2017, which focus on predicting university students individual yearly subsidy amount by students daily behavior records. Taking the advantages of models ensemble and feature engineering, we stand out from over 2000 teams and exceed the second and the third team a big gap in the challenge. In this paper, we will talk about how did we win this challenge. The source code[2] is opened on Github under MIT license.

KEYWORDS

machine learning challenge, model ensemble,

feature engineering, subsidy prediction, accurate subsidy

1 INTRODUCTION

In China, the government puts a large amount of money every year on providing financial aid to the poverty students in universities. The challenge was set out for making use of machine learning technic to help the government better distribute the subsidy to real needed ones. This challenge provides us the real-world data of more than 10 thousand anonymous university students daily life behavior records, labeling with the history subsidy amount of each student (in the training data set). Our job is to design machine learning models, train them on the training data set and make predictions on the test data set. The output result could help either in finding out fake poverty students who are receiving the subsidy or finding out potential real poverty students who are still not subsided.

The daily records include students' behavior information such as the consume records of student card, student's GPA, student's book borrowing records, library enter records and dormitory enter records. Through these records, we can observe a student closely from his money cost, study status and personal habits. These will reflect a student's economic situation. For example, a student who is lacking money won't spend too much in his daily life.

The possible label of each example, in our case the yearly subsidy amount of each student, is given by a set of money amount $S = \{0, 1000, 1500, 2000\}$, which means this is a classification problem with 4 different categories. Among the values in the S, the 0 means that the student didn't get any subsidy, while the other values mean that the student has gained that amount of subsidy from the government.

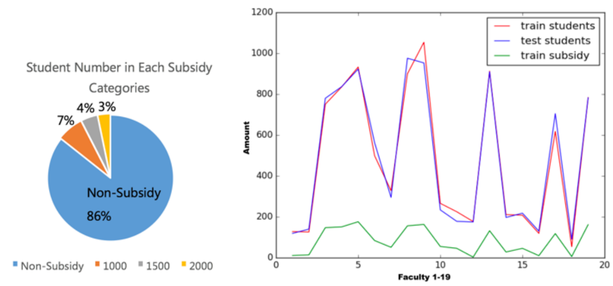

Figure 1: Number of Students in Each Subsidy Categories in Training Data (Left), Number of Total Students and Subsided Students in Each Faculty(Right)

As Figure 1 shows, the number of non-subsidy students is far bigger than the number of subsided students, which is also true within the university faculties. Therefore, we take this problem as a typical multi-categories data unbalanced classification problem. In the later section, we will talk about how to deal with the unbalanced data.

One thing we want to mention is the criteria of the challenge. To judge the model performance of each team, the organizer uses a Macro F1 value of the prediction result submitted by participants. The Macro F1 value is defined as follow:

$$F1_i = \frac{Precision * \ Recall * 2}{Precisio + Recall} \tag{1}$$

$$Macro\ F1 = \sum_{i=1}^{3} \frac{N_i}{N} * F1_i \tag{2}$$

where the $F1_i$ is calculated by the $Precision$ and $Recall$ of each category i in subsided categories $\{1000, 1500, 2000\}$, and the N is the number of total students while N_i is the is the number is the number of students in category i.

To be clear, we have drawn the architecture of our project, as shown in Figure 2. The first row in the picture presents how we dealing with the raw data to get the model input file. In this step, we dialed with redundant and missing values in the data and extract fea

[1]http://www.pkbigdata.com/static_page/cmpList.html
[2]https://github.com/lzddzh/DataMiningCompetitionFirstPrize

tures from each table, then union the feature files into one model input file and perform standardization. In the second row of the picture, we split the model input file into 5 cross-validation subsets. After the model parameters tuning, we ensemble different model's result to gain a better accuracy. At last, we perform the prediction on the test data set and get the result file containing the predicted label.

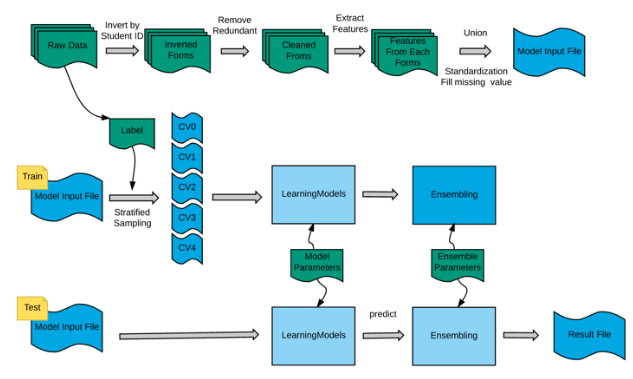

Figure 2: The Architecture of Our Project

The rest of this report are divided into 3 sections. In section 2, we will first describe the data set and the feature extraction. Here we will show how important the feature engineering is to a machine learning project. Then in section 3, the different models used in our project will be covered. An innovation ensemble method which raises the Macro F1 of our result by 3%. Lastly, in section 4 we will summarize the whole project and make an conclusion.

2 Data Set

2.1 Data Set Description

Taking the subsidy record in the year 2014, 2015 and the student daily behavior record in the academic year 2013-2014, 2014-2015, the data set contains 6 tables covering the book borrow records, student card consumes records, dormitory enter records, library records, student GPA records and subsidy records.

Table 1: Tables in the Data Set

Table Name	Collums Name	Training Data Size
Book Borrow Table	student ID, borrow date, book name, ISBN	239,947 rows, rows, 31MB
Student Card Consumes	student ID, consume type, location, purpose, timestamp, amount, balance	12,455,558 rows, 880MB
Dormitory Enter Records	student ID, timestamp, direction	2,115,064 rows, 66MB
Library Enter Records	student ID, door, timestamp	1,012,747 rows, 37MB
GPA ranking	student ID, Faculty, GPA ranking	9,130 rows, 111KB
Subsidy Category	Student ID, subsidy amount	10,885 rows, 97KB

We have presented all the training data set tables column and size in Table 1 to have a better overview of the data. The test data set owns the same type of information as the training data and is also similar to training data on the size, but it doesn't have the subsidy category. For more detail of the dataset, please refer to the appendix.

2.2 Data Clean

Because all the data are from the real word, so it is not so tidy. There are many redundant or missing values in the data, or in some cases, the data is not correct in logic. For example, we find that in the dormitory enter record, a student's entering record may not consistent with his leaving record, and the same situation happens in library entering records. In the book borrow table, there are some items don't have ISBN. The student card consumes record even appears negative value in the consume amount. Thus, the first task is to clean the data.

To remove the redundant items is relatively easy. As for missing values, we decided to fill them up according to their properties. For example, we may want to fill up a median value to a missing student GPA ranking, but for a student who has no book borrow record, we may just consider it as no borrowing history. We will fill up the values later after the feature extraction.

3 Feature Engineering

3.1 The Importance of Feature

3.1.1 Why important? We must aware that all of our model input is from the feature extraction. That is to say, the features decide the upper bound of you model performance. One Kaggle historical 1st personal ranking player, Xavier Conort said: "The algorithms we used are very standard for Kagglers. We spent most of our efforts in feature engineering. We were also very careful to discard features likely to expose us to the risk of over-fitting our model." In practice, it's also true that our model improved quickly every time when we improved our features, while the parameters tuning work can only change the model performance in a relatively small range. The most common situation is: we would found multiple 'best' parameters choices for our model, but the performance of these choices would reach to nearly one same point and then stopped growing. After improving our features, the performance of our model soon exceeded the previous score.

3.1.2 Maximum the input. Our lesson is that try to make use of the most of the data set as you can. For example, we ignored the location column in the student card consumes records at first. But it turned out to be one of the most useful information. There are many unexpected good features in our project. The conclusion is, in such a multi-aspect information input problem, the best choice is t hat don't give up any table or any column in the original data. Just assume all the columns of all the tables are useful. In practice, over feature extraction just leads to slow speed of the model but is not likely to reduce the model performance, but lack of feature extraction does.

3.1.3 Iteratively extract features. We didn't extract all the features at once, instead we used an iterative method. At the first iteration, we tentatively extracted 200 features from the data set, after which we were ranking at around the 120th. In the second iteration, we first observed which table and which column was providing good features and then extracted more features from these columns. This time we have 500 features and achieved the 14th at ranking. The last iteration, which increases the number to 1200 features, pushed our team to top 3 positions.

3.2 Dealing with The Consumes Records

Among all the tables in the data set. The student card consumes table is the largest, which contribute over 80% of all our features. Not only because it contains the most information, but also is the most important aspect to observe a student's economy situation. Here we will share our experiences on extracting high-quality features from this table. As for other tables, we are not going to cover too much on them since no thing special. You may find the all 1200 features names on our Github repository.

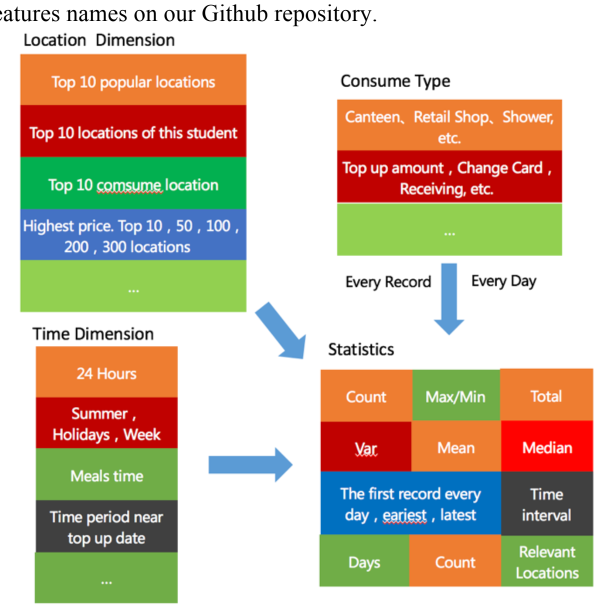

Figure 3: Different Dimensions Used in Feature Engineering

As shown in Fig 3, we designed many dimensions to extract more high-quality features from the student card consume table. Firstly, we take the different combination of the dimension's elements to make more reasonable features. Then for each feature value, we use 10 statistics to measure them more precisely. Using this method, we have successfully extracted more than 1000 features from the student consume table, many of which turned to be super useful as we will show in the next section.

Table 2: Top 20 Most Important Features (Given by RF)

Rank	Features
1	Whether Has Changed Card
2	Count Consume \$0-10 / Active Days
3	7:00AM-8:00AM Consume Rank
4	GPA Rank Multiply Consume Rank
5	6:00AM-8:00AM Consume Count/Active Days
6	The Percentage of 2000 Subsidy of His Faculty
7	The Percentage of 1000 Subsidy of His Faculty
8	\$0-2.5 Consume Records Count/Active Days
9	GPA Rank / Faculty Student Number
10	Student Card Top Up Amount / Active Days
11	The Var of Daily Consume
12	Consume Rank in Faculty
13	The Average Time of Breakfast
14	6:00AM-7:00AM Consume Amount
15	GPA Rank
16	7:00AM-8:00 Consume Max Amount
17	The Percentage of 1500 Subsidy of His Faculty
18	The Var of Daily Consume in Canteen
19	The 2rd Top Amount of His Location Count
20	11:00AM-12:00AM Consume Count

Among all the 1200 features, we display here the top 20 which are the most important features that affect the model result. To our surprise, the 'Whether Has Changed Card' feature become the No.1 important feature. It is hard to explain why, but when we check the original data, we find it is true in the data. So we can just say that it is a magic feature. The rest of the important features are from the combined features or the features reflecting students life schedule, which seems quite reasonable and interesting.

3.3 Feature Construction & Selection

The combination of existed features to produce new features proved to be a good method for squeezing out the information hidden in data. By combination, we mean use elementary arithmetic to combine two features and output one new feature. For example, in our project we divided the total consume amount by the number of active days, since different student have different active days in the data; we multiply the GPA ranking and the consume ranking of a student together, meaning the higher GPA one student get and less money he spent, the more likely he will get a subsidy. Other examples such as using the amount of in different consume categories divide the total consume amount so we have the percent of each category consume, holiday consume subtract non-holidays consume, … However, remember that you can't simply use this feature construction to combine all the existed features because the space is too large and the majority of construction output has no quality, which will slow down your model and even reduce the performance. Therefore the best way is to understand the domain knowledge of your prediction problem and construct the new features according to real experiences.

Another worth mention point is the feature selection, which is still an open question to the academic world. In our project, we have experienced deleting a few 'over-fitting' features and then having a better model result. But when tried harder to delete more featu res that we think are not relevant, our model performance fell. One must be extremely careful to delete the features, since each of them is rare and valuable.

4 Model

4.1 Preparing Model Input

After feature engineering, we also need some other preprocess: filling missing value, standardization and deal with the imbalance problem.

In practice, a lot of features contain missing values. We adopt different filling strategies for different features based on our understanding of the data. For example, for feature “total consuming amount”, if a student never has one consuming record, then we believe that system losses his or her information because it is impossible that one didn’t purchase anything in school within one year and most of the students have records in data. Consequently, we fill this feature using the average consuming amount. As for “consuming amount during 1:00 am-2:00 am”. We directly fill this feature as zero if the student doesn’t have record during this period. Obviously, not everyone stays up late to purchase at this period.

We have a variety of features with the different value range. On the one hand, consuming amount can be more than 10000$, On the other hand, the maximum value of consuming rank is 1. Although this will not affect the tree-based algorithms like random forest, gradient boosting decision tree and Adaboost, it can do harm to the efficiency and accuracy of scale variant models like SVM and Neural Network. In our project, we standardize each feature so that it looks like standard normally distributed data: Gaussian with zero mean and unit variance.

As we mentioned in the previous section, we are facing a data imbalance problem: most examples are non-subsidy students. We try three methods to solve this problem: oversampling, undersampling and setting sample weight when calculating loss function. After experiment, we found that undersampling will cause information loss and it is difficult to choose to maintain which examples. As for oversampling, actually it is equivalent to setting sample weight: Double an example in training set means you need to pay twice penalty when misclassification occurs. However, oversampling only support integer weight, you can’t duplicate one example 1.5 times. Therefore, we use the last mechanism: setting sample weight which allows us set weight in a flexible manner. Following is the loss function with sample weight. If algorithm misclassifies nth example, it needs to pay u_n times penalty. We give high weight to rare class examples in order to predict rare class examples as correctly as possible.

$$\frac{1}{N}\sum_{n=1}^{N} u_n * err(y_n, h(x_n)) \tag{3}$$

4.2 Offline Evaluation Mechanism

Because we can only submit online result at most twice per day, we need to build an effective offline evaluation mechanism to tune model parameters. We use 5-fold cross-validation in our project. Whenever we have a new idea we firstly verify it on cross-validation sets. We submit it to online evaluation system only if we obtain a good offline result.

Considering the data imbalance problem, we use stratified sampling to build cross-validation sets. Experiment shows that stratified sampling is useful to maintain the consistency between online score and offline score. If we don’t use the stratified sampling, proportion of different class examples in different cross-validation sets is different which causes the problem that offline score increase but online score decrease. Cross-validation sets with and without stratified sampling are showed in Figure 4.

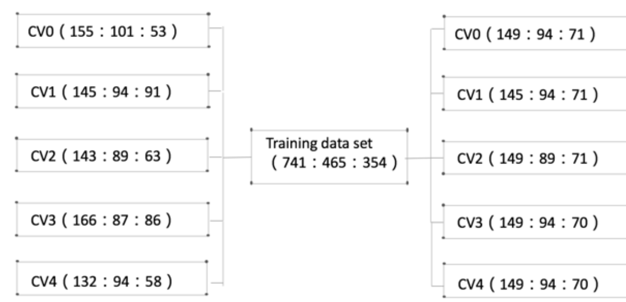

Figure 4: Left side is the cross-validation sets that generated without stratified sampling. The proportion of three class examples is different among 5 cross-validation sets. Right side is the cross-validation sets generated with stratified sampling.

4.3 Model Construction

We totally build eight different models in our project as shown in Table 3.

Table 3: Different models

algorithm	online score
Gradient Boosting Decision Tree (GDBT)	0.02889
Extreme Gradient Boosting (XGB)	0.02834
Random Forest (RF)	0.02732
Extremely Randomized Tree (ET)	0.03001
AdaBoost (Ada)	0.03006
Support Vector Machine (SVM)	0.02649
Shallow Neural Network (NN)	0.02704
K Nearest Neighbors (KNN)	0.02578

We use greedy coordinate descent and grid search to tune parameters. Greedy coordinate means we tune parameters in turn according to their influence to the algorithm.

We use GBDT as an example to describe how do we use these two methods to tune parameters. There are several important parameters in GBDT shown in Table 4. It is impossible to grid search all possible combinations of this seven parameters. After experiment, we found that sample_weight affects algorithm accuracy mostly. So we will first turn this parameter. Then we use grid search to tune max_depth and min_sample_leaf at same time because they work together to influence algorithm. We then turn min_sample_split , max_features in turn. Finally, we grid search learning_rate a nd n_estimators to obtain parameters. Actually we can switch the order of min_sample_split and min_sample_leaf. They have simil ar function that avoiding overfitting.

Table 4: Parameters in GBDT

Parameter	description
Sample_weight	Penalty weigh
Learning_rate	
N_estimators	Tree number
Max_depth	Maximum depth in each tree
Max_features	Number of candidate features in each split
Min_samples_leaf	Minimum number of samples needed to become a leaf node
Min_sample_split	Minimum number of samples needed to split a node

Another parameter tuning experience is that don't persist in ch oosing the best parameter in cross-validation sets. Figure 4 is a rea l example when we tune the parameter n_estimators in GBDT. W e can obtain best offline F1 value when n_estimators equals 450. However, F1 will decrease dramatically when n_estimators beyon d 450. Therefore, it is actually a boundary point. So we don't have a high confidence to say that this point is good for test data and m aybe this point is already in the decreasing phase for test data. In o ur project, we firstly choose a stable parameter interval with high performance which is [200,450] in this example. We will use the a verage value of the interval which is 325 as the best parameter.

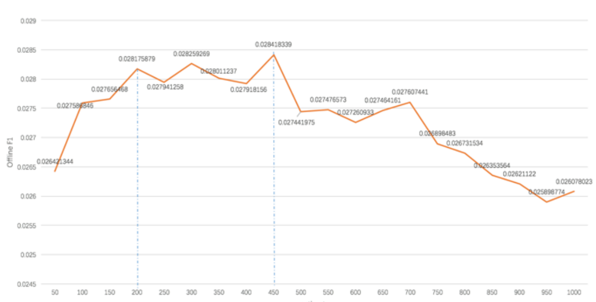

Figure 4: Real parameter tuning example in GBDT.

4.4 Model Ensemble

We discard the SVM, NN and KNN model in ensemble step be cause of the poor accuracy or efficiency. Therefore, we only retain GDBT, XGB, RF, ET and Ada models which are all tree-based e nsemble algorithm interestingly. We believe that only use five bas e learners for ensemble is far from enough. So we come up with s ome methods to extend one algorithm to many models.

Random seed can affect the model construction in a lot of tree-based models. For instance, in RF, random seed will affect the can didate feature set for each splitting and the examples after bootstra p sampling in each tree. Accordingly, choosing different random s eed can result in different models.

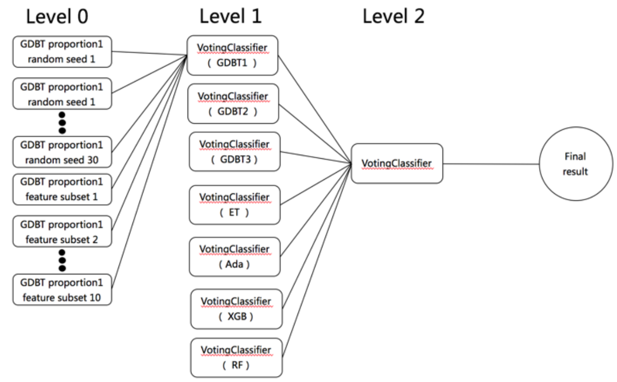

Figure 5: Model ensemble layers

We can also build different models by using different features. GBDT is able to output feature importance based on times one fea ture is used to split nodes. We give features id according to impor tance order. Then dividing them into five feature subsets by the w ay of feature_id %5 which guarantee features quality of each subs et. We can also randomly split the features into five subsets. This t en feature subset can be used to build ten different models.

We discovered that different prediction proportions on each lab el categories can achieve similar high Macro F1 result. For GBDT, all the following three prediction proportions can achieve good pe rformance. (2000subsidy : 1500:subsidy : 1000subsidy : 0) = (763 5:2124:634:550), (7808:2117:677:359), (8239:1929:484:309). We can adjust example weight for different classes to obtain the predi ction proportion we want.

Through above three extension methods, an algorithm can be e xtended to many different base learners. How do we fuse all these base learners are showed in Figure 5. We use three hierarchy stack ing method for ensemble. In level 0, we use 30 different seeds and 10 feature subsets to generate 40 base learners for a specific algor ithm. Classifiers in level 1 take the output from Level 0 as input a nd use majority vote algorithm to predict. The vote for each base l earner is equal because they are all constructed by the same algori thm and have similar accuracy. In level 2, we use a voting classifi er to fuse predictions from different kinds of algorithms to generat e the final result. In this layer, votes of different algorithms are ba sed on the corresponding accuracy. High accuracy model will obta in the high vote. Votes of different models are showed in figure 6. In the last day of competition, Ensemble helps us turn the table an d win the game finally.

algorithm	vote
GDBT1	2
GDBT2	0.5
GDBT3	0.5
XGB	1.5
Ada	1.5
RF	1
ET	1

Figure 6: Votes for different models

5 CONCLUSIONS

In summary, we introduced the whole process during competition: data preprocess, feature engineering, model construction and model ensemble. From our experience, feature engineering is the most important step. Features directly determine the final model accuracy. If features are in bad quality, model will become a garbage in garbage out system. Model ensemble seems like final sprint. However only if your base learners are variant, can you improve accuracy a lot through ensemble. Actually, we also try to use Adaboost or GBDT in level 1 or level 2 to fuse models. But these powerful models will cause overfitting problem because our training data is not very large. Accordingly, we choose voting classifier in level 1 and level2. Another interesting observation is that tree-based ensemble methods like GDBT, XGB, Ada, RF, ET outperform non-ensemble methods like SVM, NN, KNN in our project. Therefore, ensemble can help to reduce the bias and variance at the same time like what we have learned in class.

Appendix: Data Set Sample

```
Student ID, Borrow Date, Book Name, ISBN
9708,2014/2/25,"我的英语日记/ (韩)南银英著 (韩)卢炫廷插图","H315 502"
6956,2013/10/27,"解读联想思维: 联想教父柳传志","K825.38=76 547"
9076,2014/3/28,"公司法 gong si fa = = Corporation law / 范健, 王建文著 eng"
```

```
StuId, ConsumeType, Location, ConsumePurpose, ConsumeTime, Amount, Balance
1006,"POS消费","地点551","淋浴","2013/09/01 00:00:32","0.5","124.9"
1406,"POS消费","地点78","其他","2013/09/01 00:00:40","0.6","373.82"
13554,"POS消费","地点6","淋浴","2013/09/01 00:00:57","0.5","522.37"
```

```
Student ID, Subsidy Category›
10,0
22,1000
28,1000
64,1500
650,2000
```

```
Student ID, Faculty, GPA Rank
0,9,1
1,9,2
8,6,1565
9,6,1570
```

```
Student id, Door, Timestamp
3684,"5","2013/09/01 08:42:50"
7434,"5","2013/09/01 08:50:08"
8000,"进门2","2014/03/31 18:20:31"
5332,"小门","2014/04/03 20:11:06"
7397,"出门4","2014/09/04 16:50:51"
```

```
Student ID, Timestamp, Direction(0enter, 1exit)
13126,"2014/01/21 03:31:11","1"
9228,"2014/01/21 10:28:23","0"
```

REFERENCES

[1] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research 12.Oct (2011): 2825-2830.
[2] KAGGLE ENSEMBLING GUIDE（http://mlwave.com/kaggle-ensembling-guide/）
[3] Dietterich, Thomas G. "Ensemble methods in machine learning." International workshop on multiple classifier systems. Springer Berlin Heidelberg, 2000.
[4] de Abril, Ildefons Magrans, and Masashi Sugiyama. "Winning the kaggle algorithmic trading challenge with the composition of many models and feature engineering." IEICE TRANSACTIONS on Information and Systems 96.3 (2013): 742-745.
[5] Taieb, Souhaib Ben, and Rob J. Hyndman. "A gradient boosting approach to the Kaggle load forecasting competition." International Journal of Forecasting 30.2 (2014): 382-394.
[6] Puurula, Antti, Jesse Read, and Albert Bifet. "Kaggle LSHTC4 winning solution." arXiv preprint arXiv:1405.0546 (2014).