MSBD5002 Final Project

FAN JiaHao
20616960
jfanas@connect.ust.hk

WANG Yuxiang
20635564
ywangkh@connect.ust.hk

YE Guangxian
20644369
gyeaa@connect.ust.hk

Abstract

Highway is the way specially designed for cars to drive at a high speed. Nowadays, it plays an increasingly important role in modern transportation. As a modern infrastructure, the highway has brought great help to the physical distribution, investment attraction, industrial structure adjustment and resource development along the route. As of December 28, 2018, the total mileage of highways in China has reached 140,000 kilometers, ranking first in the world. However, with the development of technology and times, the automobile industry is developing rapidly as well. Then the number of cars has gradually increased, and congestion often appears on highways, making highway no longer 'high-speed'. In order to avoid traffic jams, we hope to estimate the volume at each highway tollgate, so as to divert traffic during periods when congestion may occur and reduce traffic pressure.

1 Introduction

1.1 Project and Data set Description

In the task 2, given related volume data gathered from Sep. 19th to Oct. 17th, we plans to predict exit travel volume every 20 minutes later at each tollgates. 5 tollgate-direction pairs in total.
There are three tables for us to handle: Traffic Volume at Tollgates(Table1), weather (table7) and Traffic Volume through the Tollgates(table6). Table1 has four attribute,with 10064 records of data to handle. table 7 has six attributes,with 863 records of data to handle.Table 6 has eight attributes, with 543700 records of data to handle. After the feature engineering, there are totally 10064 records of training data, and 421 records of testing data, with 28 and 27 features each.

1.2 Feature engineering Description

There are three steps in the feature engineering process:

- Firstly, we need to handle the missing value and do the out-liners detection. Fill all the null value with 0 and remove some out-liners. The performance of whether removing specific out-liners will be estimated in the cross validation process.
- Secondly, we convert some data columns into numerical format and merge three tables (weather, main traffic volume, additional traffic volume) together into one big table. And below is the brief introduction of how we handle the data attribute. For the time windows, we separate it into minutes, month, days, hours, seconds and two more. The year is ignored as it is the same as all the data. What should be mentioned is the time interval to be detected, one is the hour that car passed the tollgate (3 hours), another one is the minutes that car passed. Those attributes are essential for us to merge the three tables. We make a pickle dictionary of the table six. The key is combination of tollgateid, direction, timestamp (year, month, hour) and start time. And we fill the data in the 20min average volume table. Similar

for the weather table, only the hour attribute should be modified as it is measured within three hours.

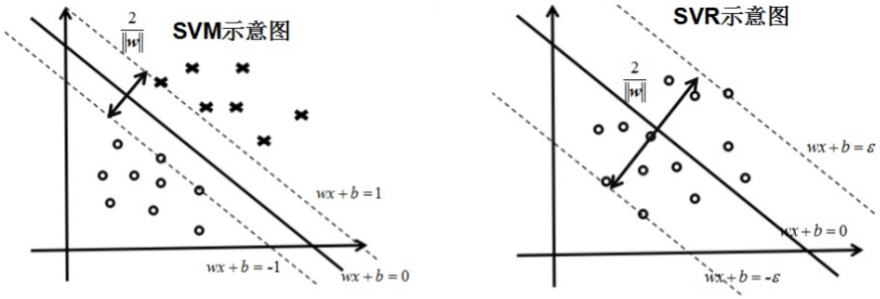
- Thirdly, we try to add some additional features based on the scenario. For the vehicle model attribute in table 6, we calculate the probability of that the car appeared in that that timestamp belongs to one specific car type. Totally eight types of car. So eight attributes are derived. Since the task aims to predict the volume two hours late, we add the actual volume two hour before as additional feature. Besides, we do the normalization of the data, making them within 0 to 1.

2 Models

After the data preprocessing, we choose different models to fit the data and do the prediction. To test the accuracy of our models, we divide the entire dataset into training set and testing set, with a ratio of 4:1. In general, we used three different models: random forest, support vector regression (SVR) and XGBoost.

2.1 Support Vector Regression

Support vector regression (SVR) is an important application branch of support vector machine (SVM). The difference between SVR and SVM is that there is ultimately only one class of sample points in SVR, and the optimal hyperplane it seeks is not the "most open" of two or more classes of sample points as in SVM, but rather the one that minimizes the total deviation of all sample points from the hyperplane. That is to say, SVR is to minimize the "distance" to the farthest point of the sample from the hyperplane.



$$\begin{cases} \min \frac{1}{2}\|w\|^2 \\ s.t.\ y_i(wx_i+b) \geq 1, \quad \forall i \end{cases}$$

$$\begin{cases} \min \frac{1}{2}\|w\|^2 \\ s.t.\ |y_i - (wx_i+b)| \leq \varepsilon, \quad \forall i \end{cases}$$

Figure 1: SVM and SVR

Based on sklearn, we can use svr to fit our training set. To get the better result, we use grid search to tune parameters. We choose two parameters: C and gamma.

- C: penalty factor. If C is too small, it will lead to underfitting. Otherwise, it will lead to overfitting.
- gamma: If gamma becomes larger, the number of support vectors, which can affect the speed of training and prediction, will decrease. Otherwise, it will increase.

After tuning parameters, we finally decide the parameters as C=200,kernel = 'rbf', gamma = 0.01.

2.2 Random Forest

Random forest belongs to the bagging algorithm in ensemble learning, which is a common and efficient model in terms of regression and classification tasks. The forest means it consists of many decision trees. It's random because it selects samples randomly with the bootstrapping method. That is to say, if the dataset D has m samples, we randomly select one sample from D at a time, and then put it back to D. Repeat the procedure for n times, then we can get one training set with n samples. When it comes to the regression tasks, each decision tree in the model is a regression tree. Due to the bagging method, the model repeats the bootstrap for k times and then get k training sets, which are independent with each other. Based on each training set, it can train a basic regressor. No matter what feature the regressor chooses, the goal is to minimize the mean-square error. To get the final prediction, the model will merge these regressors and calculate the average value of all regressors as the final result.

Based on sklearn, we can use random forest to fit our training set. To get the better result, we use grid search to tune parameters. We choose four parameters: n_estimators, min_samples_leaf, max_depth and min_samples_split.

- n_estimators: the number of weak learners. If it is too small, it will lead to undetfitting. If it is too big, the calculation will be complex.
- max_depth: a value which restrict the depth of subtree. When the data or features are large, it need to be controlled.
- min_samples_leaf: a value which restrict the minimal number of samples in leaf node. When the number of samples in one leaf node is smaller than the value, this node will be cut. When the data volume is large, it needs to be controlled.
- min_samples split: this value limits the conditions under which the subtree can continue to divide, and if a node has fewer samples than min_samples_split, it will not continue to attempt to select the optimal feature for division.

After tuning parameters, we finally decide the parameters as n_estimators=50, max_depth=9, min_samples_leaf=1, min_samples_split=5, max_features = '$sqrt'$, random_state=0.

2.3 XGBoost

Extreme gradient boosting (XGBoost) is a highly efficient algorithm which is based on gradient boosting decision tree (GBDT). It uses the boosting method, a typical method in ensemble learning. Boosting refers to construct multiple weak classifiers to make predictions on the dataset and then integrating the results with some specific strategy as the final prediction result. During the process, those weak classifiers are dependent to each other, which is different from bagging. The principle of GBDT is that adding up all the results of weak classifiers, and then the next classifier fits the gradient/residual of the error function of the predicted value.
Based on the GBDT, XGBoost makes much improvement. The biggest difference is that XGBoost applies the regularization factor into object function. This can avoid the overfitting and increase the accuracy of the model efficiently. The object function is:

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant$$

To understand the formula, we consider the process of XGBoost from the beginning (Figure 2). In each iteration, add a new CART tree to fit the residual of the former tree.

From the figure we can see that the prediction result at time t equals to the result of t-1 plus $f_t(x_i)$. Usually, we consider the loss function as mean-square error function, which is a quadratic function. However, lots of loss function are not quadratic, so we use Taylor expansion to do the transformation. So we can get(Figure 3):
After installing the xgboost package, we can use XGBoost to fit our training set. To get the better result, we use grid search to tune parameters. We choose five parameters: n_estimators, min_child_weight, max_depth, gamma and subsample.

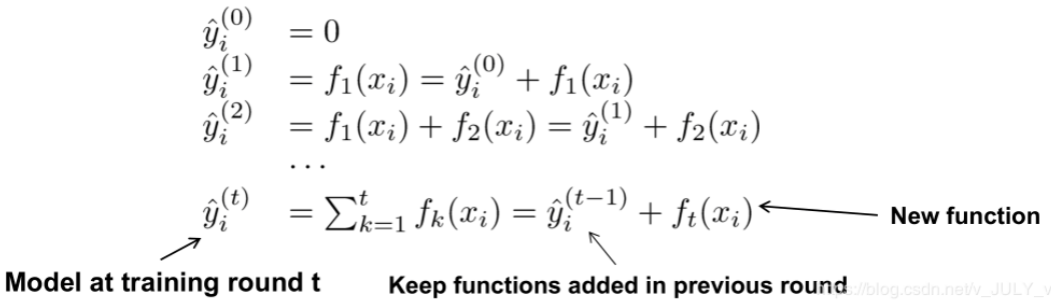


Figure 2: Function in XGBoost

$$Obj^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y_i}^{(t-1)}) + g_i f_t(x_i) + 1/2 h_i {f_t}^2(x_i)] + \Omega(f_t) + constant$$

$$g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right), h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$$

$\Omega(f_t)$ contains L_1 regularization and L_2 regularization.

Figure 3: Final object function

- n_estimators: the number of estimators.
- min_child_weight: determine the minimum leaf node's sum sample weights. It can be used to avoid overfitting. However, when it becomes too large, it will lead to underfitting.
- max_depth: the max depth of tree. When it becomes large, model can learn more in details and become more accurate.
- gamma: it specifies the minimum loss function drop required for node splitting. The larger the value, the more conservative the algorithm.
- subsample: it controls the proportion of random sampling for each tree. By reducing the value of this parameter, the algorithm becomes more conservative and avoids overfitting. However, if this value is set too small, it may result in underfitting.

After tuning parameters, we finally decide the parameters as n_estimators=150, max_depth=9, min_child_weight=3, subsample=1, gamma=0.4, learning_rate=0.1.

2.4 Stacking

Stacking is another useful ensemble learning method. In bagging or boosting, we choose voting or average ways to do the merge of basic classifications. In stacking, the merge strategy is based on another machine learning algorithm, instead of a simple merge method. In this task, we use xgboost as the basic model, followed by a linear model.

3 RMSE and Visualization

After tuning parameters, we use cross validation to calculate the final root mean square error (rmse) of each model. The result can be seen in the table below.

Then we do the visualization to evaluate the prediction directly. We apply the models into testing dataset which we split ahead of time. We draw scatter plot in which x_axis represents the real value and y_axis represents the prediction value. When the trend of these points is close to the function of y=x, the model is accurate.

	RMSE
SVR	25.1058297558632
Random Forest	21.6037440072698
XGBoost	13.6593652130443
sclf	13.6592718648050

Figure 4: RMSE results

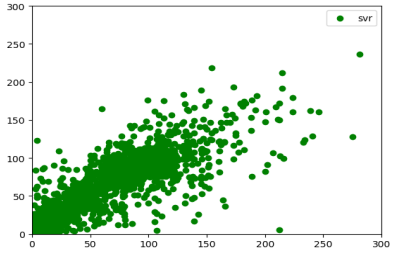


Figure 5: SVR

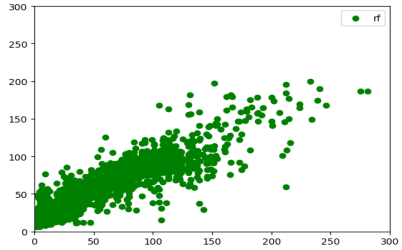


Figure 6: Random Forest

From the figure 9 we can see that the plot of xgboost and stacking regressor are almost the same. Their rmse results are quite close to each other. Therefore, we choose these two models to do further tasks.

4 Evaluation

To evaluate how good our model is, we collected the data from Oct. 18th to Oct. 24th as our test dataset, especially for the specific rush hours. Actually, our model is useful to use the data in one time slot to predict the volume of the same tollgate pair after two hours. As we do not have the real data for the time windows that need to be predicted, in this part, we would only use the given time window as the predicted data, implement MAPE method on model evaluation to have an overview (assume the volume in two hours does not change too much), and also use some data visualizations to show our prediction results comparing with actual data to make sense.

4.1 MAPE Evaluation

Firstly, let us introduce what the MAPE is. Let C be the number of tollgate-direction pairs which are 1-entry, 1-exit, 2-entry, 3-entry and 3-exit. Then T is the number of time slots during the time period, and f_{ct} and p_{ct} is the actual and predicted traffic volume for each tollgate pair c during time window t respectively. The MAPE formula for traffic volume prediction is defined as:

$$MAPE = \frac{1}{C}\sum_{c=1}^{C}(\frac{1}{T}\sum_{t=1}^{T}\left|\frac{f_{ct}-p_{ct}}{f_{ct}}\right|)$$

As we did not know the actual volume in the test dataset, let use just use the training set as a reference. Here we only choose the data of the first tollgate with direction 1(i.e. 1-exit) to see the performance on MAPE. As the cross-validation result had already shown that our model is not overfitting, this way is secure enough. Using the prediction results and the actual results of volumes, we could calculate the MAPE for our prediction on different models. We could easily find that Stacking Regressor is the best one with the lowest MAPE value, this conclusion is the same with which we talked in RMSE calculation. The result is 0.335373 which is pretty good, as it means that for this gate, the error will not exceed 35 percent of the true value.

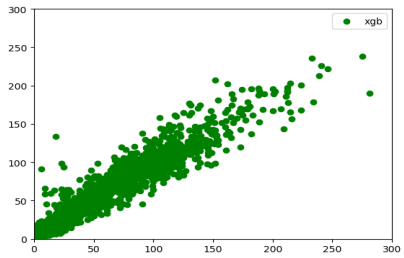


Figure 7: XGBoost

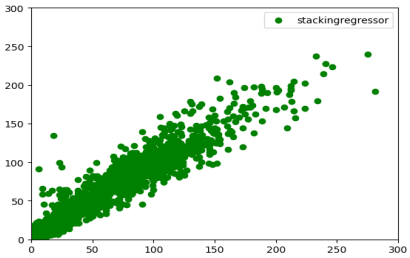


Figure 8: StackingRegressor

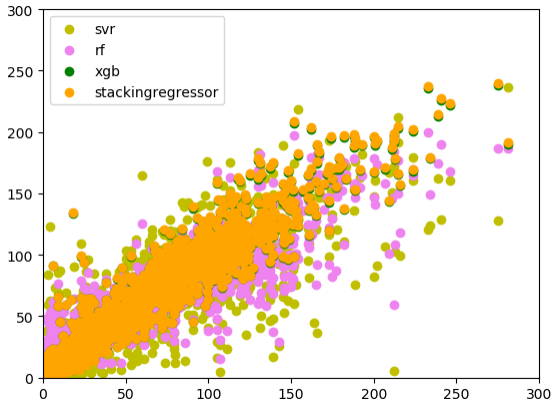


Figure 9: Comparison visualization on different models

4.2 Visualizations

In order to make it easier to see the performance of our models, we also choose 1-exit to made visualizations. The line chart was used only for the best model we had – Stacking Regressor, to give a overview as Figure 11.

From the chart, it is obvious that there is a certain trend over these lines. Every day, there would be an obviously increasing at beginning and a decreasing at the end. This tendency is also visible in our prediction. We could find that although there were some errors between, the overall trend is very consistent, which means that our model has truly learned our dataset.

From above, we could say that our model reached a pretty good performance.

Conclusion

In fact, in today's society, people are paying more and more attention to traffic problems. Traffic jams are a phenomenon that occurs in almost every developed city. And how to relieve traffic pressure is also a headache for government departments. But if we can predict the traffic jam before the traffic jam occurs, and take some measures to deal with it, such as traffic diversion, vehicle type limit and speed limit, etc., then we can solve this problem well, and control people's emotional changes and public opinion Trend as well.

From this project, we successfully implemented many basic models, like the linear regressor, the SVR model, the Random Forest model and so on. From those we chose Xgboost and the linear model and then make stacking of those two to get our best model. This project allowed use to have a richer experience in data processing, for example data pre-processing, features mining, parameters tuning and so on, it helped us to have a deeper understanding of the subject of big data technology. Maybe our results are not perfect, hopefully through further study of data mining in the future, we could improve the accuracy on our predictions.

Model	MAPE
SVR	3.012247
Random Forest	1.159324
XGBoost	0.350802
Stacking	0.335373

Figure 10: MAPE results on different models

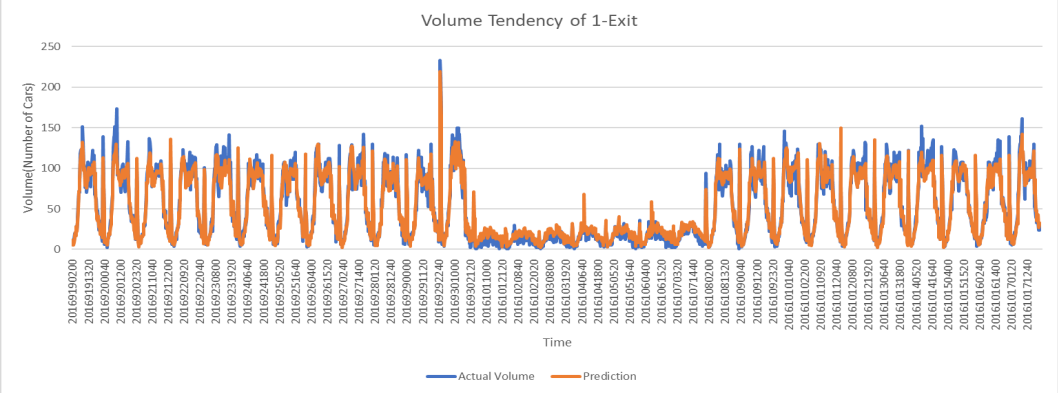


Figure 11: Visualization of prediction and actual volume on 1-exit