Technical Report

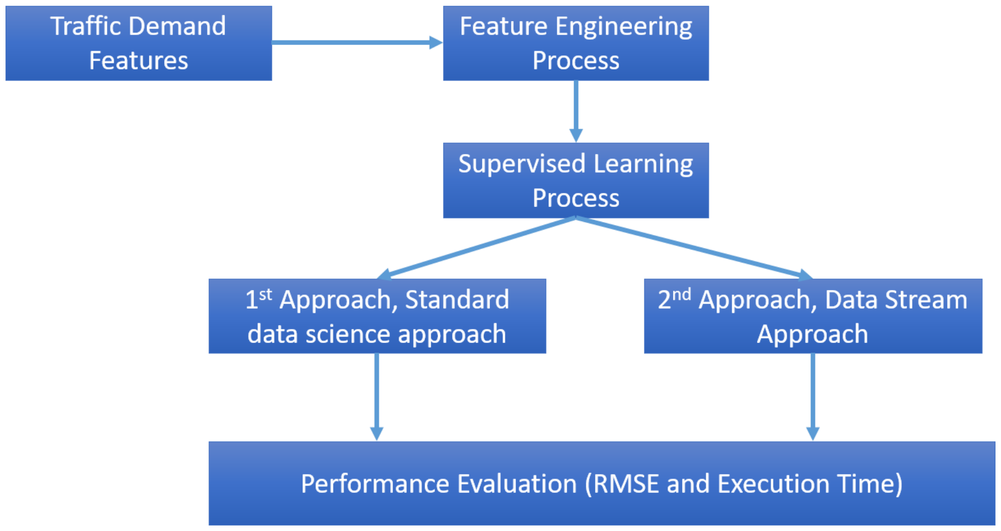

Traffic Demand Features
Feature Engineering Process
Supervised Learning Process
1st Approach, Standard data science approach
2nd Approach, Data Stream Approach
Performance Evaluation (RMSE and Execution Time)

Fig.1. Research Diagram

Result Highlight

I use two approaches in processing predictions from traffic demand shown in Fig.1. The first approach is the standard data science approach, namely the standard data science process approach, which performs a feature selection, and then the feature is included in the algorithm learning machine. The second approach is to use a data stream model approach.

This approach is used because seeing the potential for this dataset can be very large because it is a travel demand data that is rapidly generated by many users. Simple machine learning approach has the potential to have weaknesses when the data becomes very large because the training process will consume CPU memory so in this problem, I try to approach the data stream so that the learning process can be done in a stream and iterative manner. It is expected that with this approach, the memory of the CPU does not experience exhaust resources when the data becomes very large.

Table.1. RMSE evaluation uses Standard machine learning and data streams

Standard Machine Learning			Data Stream	
Linear Regression	Decision Tree	Random Forest	FIMT-DD	FIMT-DD-ARDEV
0.15	0.042	0.034	0.10	0.06

Based on Table.1, I use a standard machine learning and data stream model to see RMSE performance from traffic demand data. For the data stream model algorithm, I tried to propose an improvement of the algorithm with the name FIMT-DD-ARDEV with changes to the bound the use of standard deviations for splitting trees. To use a data stream, the FIMT-DD-ARDEV model has better RMSE performance compared to FIMT-DD (standard data stream algorithm). With RMSE values of 0.10 (FIMT-DD) and 0.06 (FIMT-DD-ARDEV). In standard machine learning, random forest has the best RMSE, which is 0.034.

In Table.1. Indeed, it can be seen that the results of the machine learning standard with random forest have smaller RMSE values compared to the data stream model. However, keep in mind that the data will gradually increase due to generated data from user behavior, especially if the area of coverage increases. It will be a big problem when the standard machine learning algorithm requires resource memory to fit all data in memory. Resource memory will run out when the data that has to be trained increases continuously. Therefore on this challenge, I tried to use two approaches. The standard machine learning approach and the data stream approach, where the data stream approach will do iterative training based on the incoming data stream.

Here are some of the results of the analysis that I have done for traffic data. The data stream algorithm that I use in this traffic demand challenge is also based on the following reference publication:

- Ari Wibisono, Devvi Sarwinda, ***Average Restrain Divider of Evaluation Value (ARDEV) in data stream algorithm for big data prediction***, Knowledge-Based Systems, Volume 176, 2019, Pages 29-39, ISSN 0950-7051,
- Ari Wibisono, Wisnu Jatmiko, Hanief Arief Wisesa, Benny Hardjono, Petrus Mursanto, ***Traffic big data prediction and visualization using Fast Incremental Model Trees-Drift Detection (FIMT-DD)***, Knowledge-Based Systems, Volume 93, 2016, Pages 33-46, ISSN 0950-7051,
- Ari Wibisono, H. A. Wisesa, W. Jatmiko, P. Mursanto, and D. Sarwinda, "**Perceptron rule improvement on FIMT-DD for large traffic data stream**," 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, 2016, pp. 5161-5167.
- E. Ikonomovska, J. Gama, and S. Džeroski, ***Learning Model Trees from Evolving Data Streams***", Data Mining and Knowledge Discovery Journal, pages 1–41, 2010. Springer, Nederlands.

Data Analysis and Feature Selection

Based on the problem described on the website https://www.aiforsea.com/traffic-management, the purpose of this challenge is to be able to accurately predict travel demand based on historical booking data in several areas. The basic features provided are geohash6, day, timestamp, and demand.

In solving this problem, I tried two approaches:

1. The first approach is the data science approach standard, which starts from the cleansing dataset, features selection, and takes several datasets for training and tests into machine learning algorithms.

2. The second approach is to approach the data stream; this approach is slightly different from the standard approach. In this approach, the dataset is an iterative training stream for the model so that large enough data can be slowly trained into the model.

Feature Engineering Process

File source code : 1-feature-combine.ipynb

Feature Engineering Process :

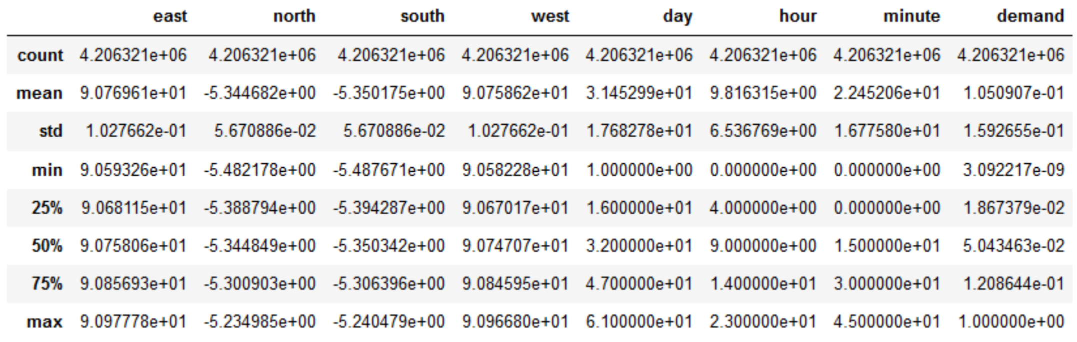
1. Convert geohash6 to east, north, south, west
   a. Utilize https://github.com/hkwi/python-geohash
2. Split timestamp feature to hour and minute
3. Combine east, north, south, west, day, hour, minute, demand into one tuple and use std out from bash script
4. Execute nohup python feature-combine.py &>combine.csv&

1st approach (Standard Machine Learning)

File source code : 2-traffic-demand-analysis.ipynb

Standard machine learning approach :

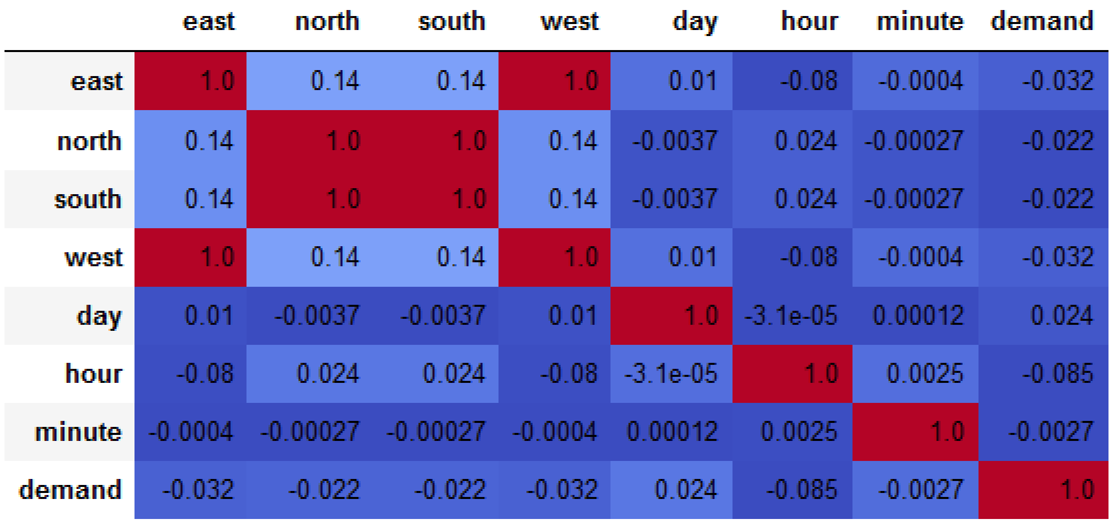
1. View dataset statistic



	east	north	south	west	day	hour	minute	demand
count	4.206321e+06	4.206321e+06	4.206321e+06	4.206321e+06	4.206321e+06	4.206321e+06	4.206321e+06	4.206321e+06
mean	9.076961e+01	-5.344682e+00	-5.350175e+00	9.075862e+01	3.145299e+01	9.816315e+00	2.245206e+01	1.050907e-01
std	1.027662e-01	5.670886e-02	5.670886e-02	1.027662e-01	1.768278e+01	6.536769e+00	1.677580e+01	1.592655e-01
min	9.059326e+01	-5.482178e+00	-5.487671e+00	9.058228e+01	1.000000e+00	0.000000e+00	0.000000e+00	3.092217e-09
25%	9.068115e+01	-5.388794e+00	-5.394287e+00	9.067017e+01	1.600000e+01	4.000000e+00	0.000000e+00	1.867379e-02
50%	9.075806e+01	-5.344849e+00	-5.350342e+00	9.074707e+01	3.200000e+01	9.000000e+00	1.500000e+01	5.043463e-02
75%	9.085693e+01	-5.300903e+00	-5.306396e+00	9.084595e+01	4.700000e+01	1.400000e+01	3.000000e+01	1.208644e-01
max	9.097778e+01	-5.234985e+00	-5.240479e+00	9.096680e+01	6.100000e+01	2.300000e+01	4.500000e+01	1.000000e+00

Fig.2. Dataset statistic

2. Features correlation



	east	north	south	west	day	hour	minute	demand
east	1.0	0.14	0.14	1.0	0.01	-0.08	-0.0004	-0.032
north	0.14	1.0	1.0	0.14	-0.0037	0.024	-0.00027	-0.022
south	0.14	1.0	1.0	0.14	-0.0037	0.024	-0.00027	-0.022
west	1.0	0.14	0.14	1.0	0.01	-0.08	-0.0004	-0.032
day	0.01	-0.0037	-0.0037	0.01	1.0	-3.1e-05	0.00012	0.024
hour	-0.08	0.024	0.024	-0.08	-3.1e-05	1.0	0.0025	-0.085
minute	-0.0004	-0.00027	-0.00027	-0.0004	0.00012	0.0025	1.0	-0.0027
demand	-0.032	-0.022	-0.022	-0.032	0.024	-0.085	-0.0027	1.0

Fig.3. Correlation Matrix

Based on the feature correlation, there is no significant correlation between one feature with other features, but that does not mean that the feature cannot predict traffic demand. This feature engineering is only an illustration of the connection between features.

3. Split train and test and K-Fold validation Initialization
4. Display boxplot for RMSE result

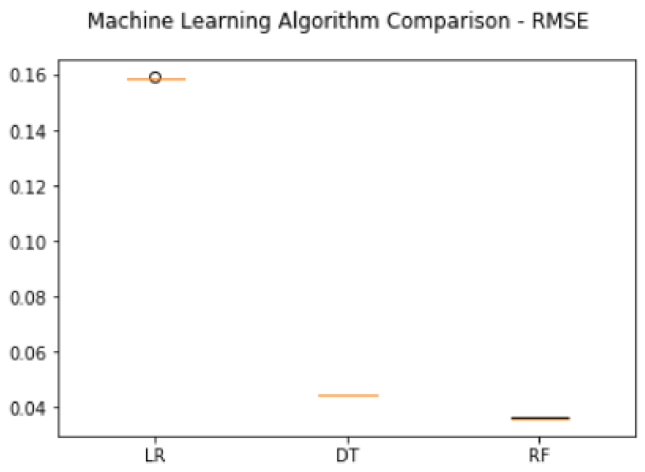


Fig.4. Boxplot of RMSE Value of Machine Learning Algorithm

5. Display Result (MAE, RMSE, Fitting time)

Table. 2. Machine Learning Error and Training Time Evaluation

Linear Regression	Decision Tree	Random Forest
MAE: 0.0961 RMSE: 0.157 Model Fitting Time: 0.813 s	MAE: 0.026 RMSE: 0.042 Model Fitting Time: 21.693 s	MAE: 0.021 RMSE: 0.034 Model Fitting Time: 167.236 s

Based on the result of MAE, RMSE, and fitting time, the results are quite good and reasonable running time is obtained by the decision tree. The best results are owned by random forest, but for the running time it is 8x longer than the decision tree.

6. Save a model to file (included in ipynb)
7. Load and test the model (included in ipynb)

2nd approach (Data Stream)

File source code: moa-ardev.jar, moa-std.jar

Data stream model approach :

1. View dataset statistic
   (The explanation is as the same as the 1st approach)
2. Features correlation
   (The explanation is as the same as the 1st approach)
3. Convert the file into arff format, the framework that I use in this data stream process is the MOA framework (Massive Online Analysis) https://moa.cms.waikato.ac.nz/
4. Prediction sequential evaluation
   Iterative learning and Error evaluation are done every of 10,000 instances

Please use this script only for evaluation based on the the data stream, the model output from this script cannot be load, you can use point 6 Script for save the model and load the model

```
// Script to run EvaluatePrequentialRegression FIMT-DD
java -cp moa-fimt-dd-ardev.jar moa.DoTask "EvaluatePrequentialRegression -l
trees.FIMTDD -s (ArffFileStream -f (combine-traffic-demand.arff)) -f 10000 -q
10000 -d (ardev-traffic-demand.csv) -o (ardev-traffic-demand.pred) -O (ardev-
traffic-demand.moa)"

// Script to run EvaluatePrequentialRegression FIMT-DD-ARDEV
java -cp moa-fimt-dd-std.jar moa.DoTask "EvaluatePrequentialRegression -l
trees.FIMTDD -s (ArffFileStream -f (combine-traffic-demand.arff)) -f 10000 -q
10000 -d (std-traffic-demand.csv) -o (std-traffic-demand.pred) -O (std-traffic-
demand.moa)"

-s : stream file
-f and -q : evaluation every 10000 instances
-d : output of evaluation result (MAE,RMSE, evaluation time, tree size)
-O : task result file (this file cannot be loaded, if you want to build model,
please use 2. script to save and load the model
```

5. Display Result (MAE, RMSE, Fitting time)

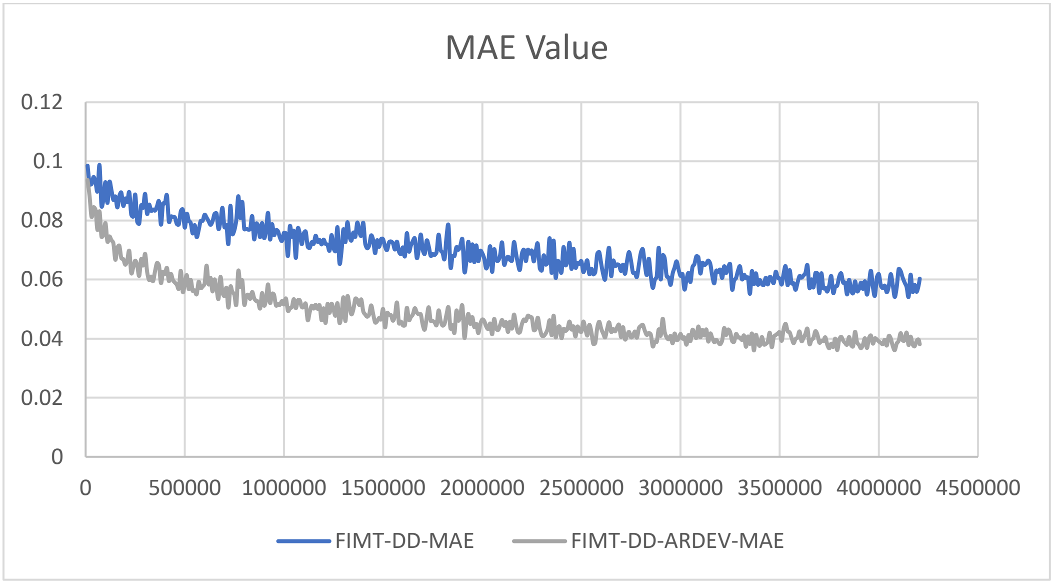


Fig.5. MAE Value Comparison of FIMT-DD and FIMT-DD-ARDEV

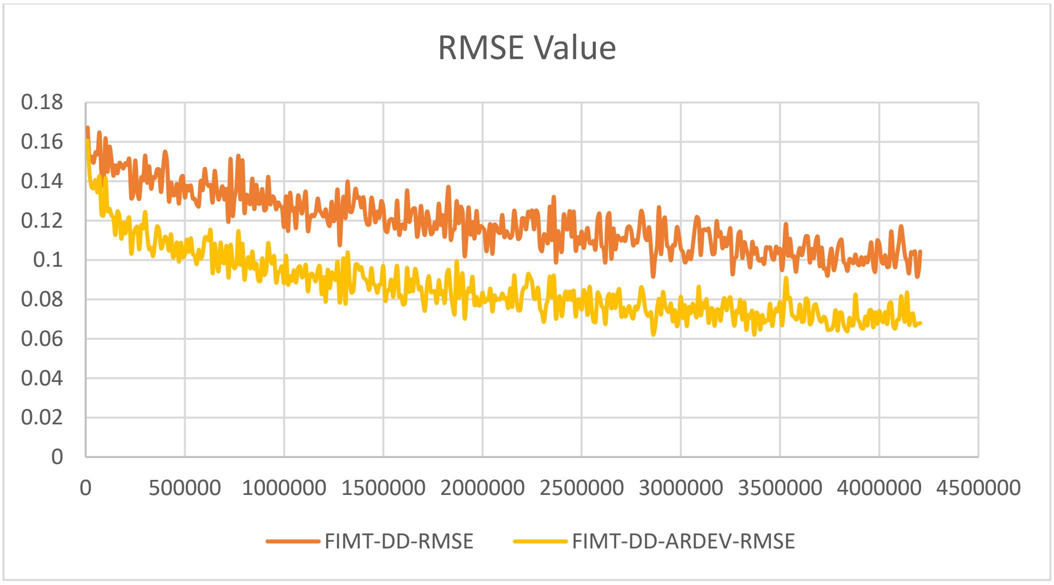


Fig.6. RMSE Value Comparison of FIMT-DD and FIMT-DD-ARDEV

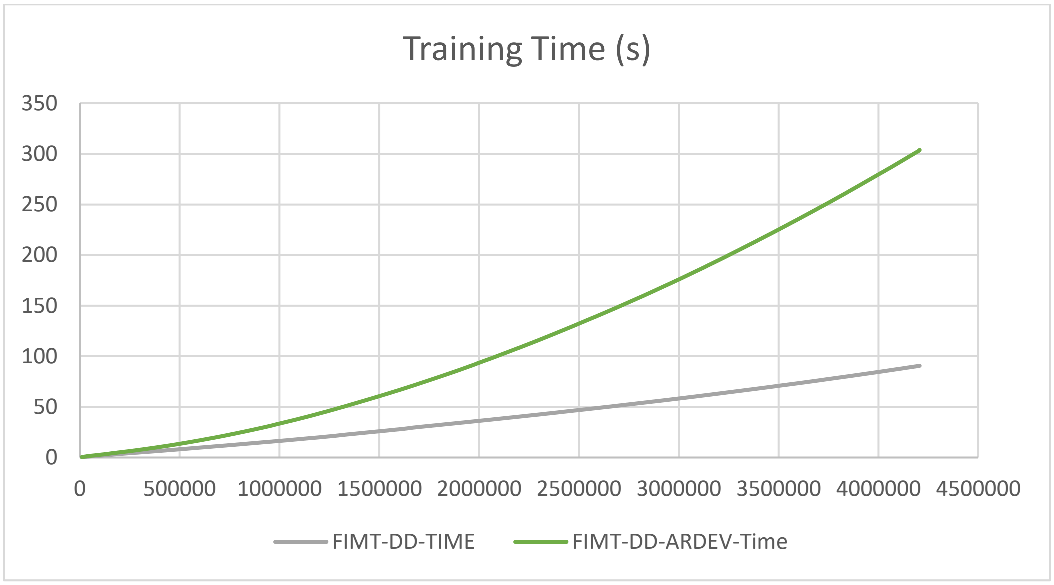


Fig.7. Training Time Value Comparison of FIMT-DD and FIMT-DD-ARDEV

To produce the results of this evaluation (Fig. 5, Fig. 6, Fig. 7), you can get the code in step 4. Prediction Sequential Evaluation. Based on the result MAE, RMSE, and fitting time. The results of the FIMT-DD-ARDEV accuracy show good results compared to the FIMT-DD standard. It needs to be recognized that running time ARDEV is longer than the FIMT-DD standard, but ARDEV FIMT-DD is much faster to reduce error value compared to FIMT-DD. For example RMSE with a value of 0.1 obtained by FIMT-DD-ARDEV on an evaluation of an instance of 500,000 (13 seconds) while FIMT-DD is obtained in an instance evaluation of 4,200,000 (90 seconds)

6. Save and load the model to file

   Please use this script to save model and load the model

```
// Script to run LearnModel FIMT-DD
java -cp moa-std.jar moa.DoTask "LearnModelRegression -l trees.FIMTDD -s (ArffFileStream
-f (combine-traffic-demand.arff)) -O (std-traffic-demand.moa) -m 100000000"
// Script to test Model Evaluation FIMT-DD
java -cp moa-std.jar moa.DoTask "EvaluateModelRegression -m file:std-traffic-demand.moa -
s (ArffFileStream -f (combine-traffic-demand.arff)) -i 100000"

// Script to run LearnModel FIMT-DD-ARDEV
java -cp moa-ardev.jar moa.DoTask "LearnModelRegression -l trees.FIMTDD -s
(ArffFileStream -f (combine-traffic-demand.arff)) -O (ardev-traffic-demand.moa) -m
100000000"
// Script to test LearnModel FIMT-DD-ARDEV
java -cp moa-ardev.jar moa.DoTask "EvaluateModelRegression -m file:ardev-traffic-
demand.moa -s (ArffFileStream -f (combine-traffic-demand.arff)) -i 100000"

Parameters in LearnModelRegression
-m : max instance
-s : stream file to learn by the model ( you can change this file as data training for
the model)
-O : save the model

Parameters in EvaluateModelRegression
-m : load model file
-s : stream file to learn by the model ( you can change this file to test the model)
-i : amount of instances to be tested
```