**Driver Supply to Passengers Challenge**

## Problem Statement

A passenger calls a ride (request) by entering the place of origin and destination and clicking “Request Pickup”. A driver answers the request (answer) by taking the order.

Didi divides a city into n non-overlapping square districts D={d1,d2,⋯,dn } and divides one day uniformly into 144 time slots t1,t2,⋯,t144, each 10 minutes long.

At time slot tj In district di, the number of passengers' requests is denoted as rij, and drivers' answers as aij, and the demand supply gap is gapij : rij - aij Given relevant data concerning time slot

tj,tj-1 in district i, you need to predict gapi,j+1, ∀di∈D.

This problem is, in essence, a supervised regression problem.

#### Problem definition---a regression problem

## Metrics

Given i districts and j time slots, for district di at time slot tj, suppose that the real supply-demand gap is gapij , and predicted supply-demand gap is sij, then:

MAPE.

The lowest MAPE will be the best.

# Analysis

## Data Exploration

This section describes all available original input data provided by competition organizer, and how these original data are aggravated for the purpose of more convenient preprocessing later on.

## Original Data

This whole section about original data is extracted from competition website(2).

The training set contains three consecutive weeks of data for City M in 2016, and you need to forecast the supply-demand gap for a certain period in the fourth and fifth weeks of City M. The test set contains the data of half an hour before the predicted time slot. The specific time slots where you need to predict the supply-demand gap are shown in the explanation document in the test set.

The **Order Info Table**, **Weather** Info Table and **POI** Info Table are available in the database, while the District Definition Table and Traffic Jam Info Table are derived from other tables in the database. All sensitive data has been anonymised.

### Order Info Table

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Meaning | Example |
| order\_id | strin  g | order ID | 70fc7c2bd2caf386bb50f8fd5dfef0c  f |
| driver\_id | strin  g | driver ID | 56018323b921dd2c5444f98fb45509d  e |

|  |  |  |  |
| --- | --- | --- | --- |
| passenger\_id | strin  g | user ID | 238de35f44bbe8a67bdea86a5b0f471  9 |
| start\_district\_has  h | strin  g | departure | d4ec2125aff74eded207d2d915ef682  f |
| dest\_district\_hash | strin  g | destinatio  n | 929ec6c160e6f52c20a4217c7978f68  1 |
| Price | doubl  e | Price | 37.5 |
| Time | strin g | Timestamp of the  order | 2016-01-15 00:35:11 |

The Order Info Table shows the basic information of an order, including the passenger and the driver (if driver\_id =NULL, it means the order was not answered by any driver), place of origin, destination, price and time. The fields order\_id, driver\_id, passenger\_id, start\_hash, and dest\_hash are made not sensitive.

### District Info Table

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Meaning | Example |
| district\_hash | string | District hash | 90c5a34f06ac86aee0fd70e2adce7d8a |
| district\_id | string | District ID | 1 |

The District Info Table shows the information about the districts to be evaluated in the contest. You need to do the prediction given the districts from the District Definition Table. In the submission of the results, you need to map the district hash value to district mapped ID.

### POI Information Table

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Meaning | Example |
| district\_hash | string | District hash | 74c1c25f4b283fa74a5514307b0d0278 |
| poi\_class | string | POI class and its  number | 1#1:41 2#1:22 2#2:32 |

The POI Info Table shows the attributes of a district, such as the number of different facilities. For example, 2#1:22 means in this district, there are 22 facilities of the facility class 2#1. 2#1 means the first level class is 2 and the second level is 1, such as entertainment#theater, shopping#home appliance, sports#others. Each class and its number is separated by \t.

### Traffic Jam Info Table

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Meaning | Example |
| district\_hash | string | Hash value of the  district | 1ecbb52d73c522f184a6fc53128b1ea1 |
| tj\_level | string | Number of road sections at  different | 1:231 2:33 3:13 4:10 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | congestion  levels |  |
| tj\_time | string | Timestamp | 2016-01-15 00:35:11 |

The Traffic Jam Info Table shows the overall traffic status on the road in a district, including the number of roads at different traffic jam levels in different time periods and different districts. Higher values mean heavier traffic.

### Weather Info Table

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Meaning | Example |
| Time | string | Timestamp | 2016-01-15 00:35:11 |
| Weather | int | Weather | 7 |
| temperature | double | Temperature | -9 |
| PM2.5 | double | pm25 | 66 |

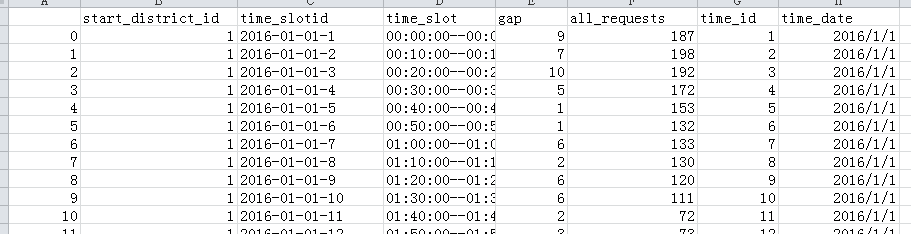
The Weather Info Table shows the weather info every 10 minutes each city. The weather field gives the weather conditions such as sunny, rainy, and snowy etc; all sensitive information has been removed. The unit of temperature is Celsius degree, and PM2.5 is the level of air pollutions.

## Aggregated Data

The original data is organized day by day; we will need to aggregate them into one big table for more convenient preprocessing later on. Also we will derive gap information at specific time in specific district, as gap is the target we will need to predict.

As a result, we extracted and aggregated below four big tables.

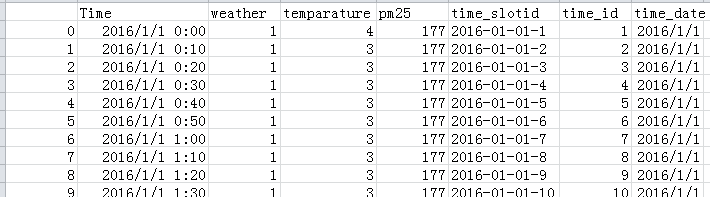
1. Gap table

The gap table contains the supply demand gap for each district/timeslot pair.

The complete tables are under data\_raw folder, and there are 3 such tables(training\_data\_gap.csv, test\_set\_1\_gap.csv, test\_set\_2\_gap.csv), for train dataset, testset1, and testset2 separately.

1. Weather table

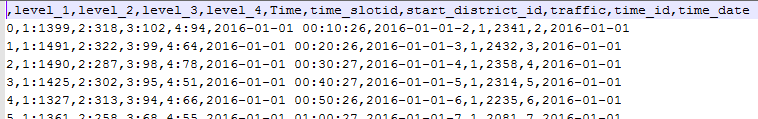
The weather table contains weather information for the dates in the train/test dataset.



The complete tables are under data\_raw folder, and there are 3 such tables(training\_data\_weather.csv, test\_set\_1\_weather.csv, test\_set\_2\_weather.csv), for train dataset, testset1, and testset2 separately.

1. Traffic table

The traffic table contains traffic information for the dates in the train/test dataset.



The complete tables are under data\_raw folder, and there are 3 such tables(training\_data\_traffic.csv, test\_set\_1\_traffic.csv, test\_set\_2\_traffic.csv), for train dataset, testset1, and testset2 separately.

1. POI information Table

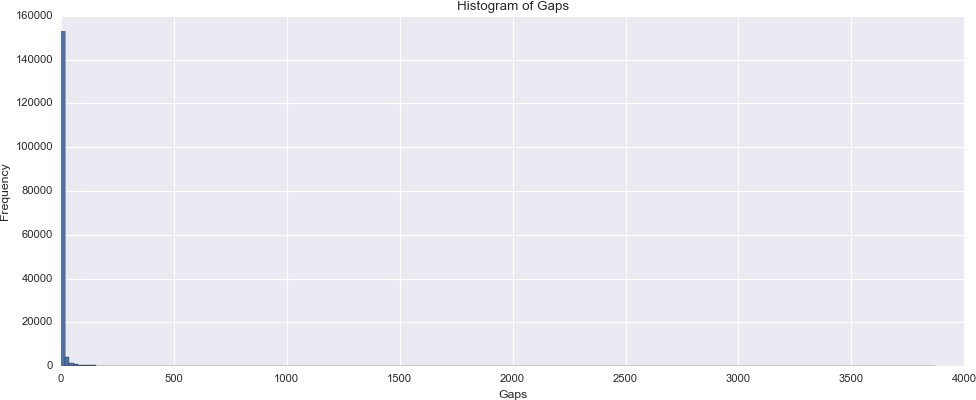
POI information table is the same as original table.

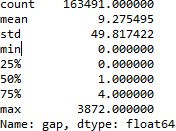
## Data Statistics & Exploratory Visualization

## Univariate

1. Gap

Gap is a continuous variable.

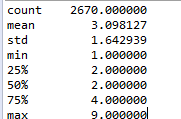
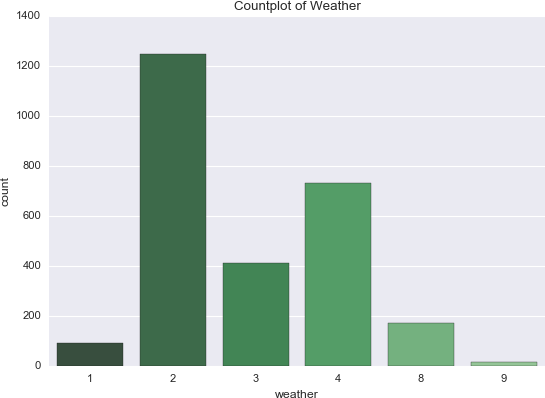




Among all rows, 60811 rows (accounting for 37% of total rows) has value 0. These rows will be excluded from the final training dataset.

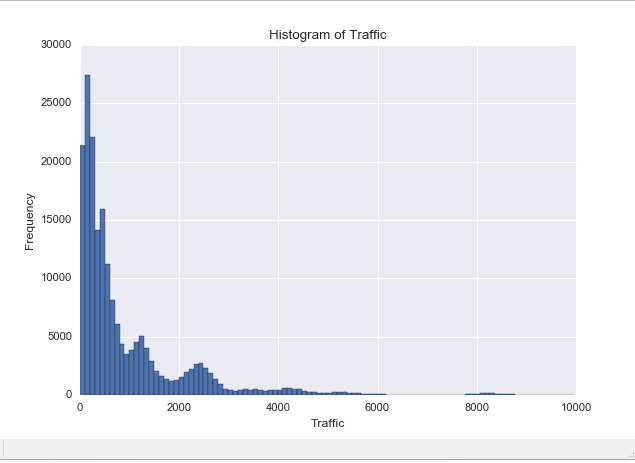
1. Weather

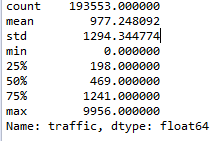
Weather is a categorical variable.



1. Traffic

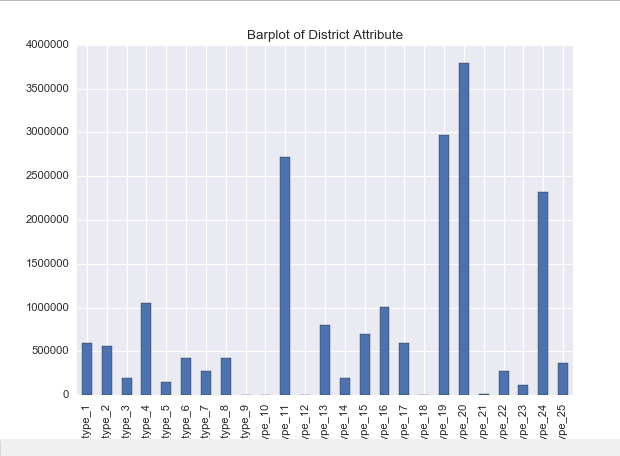
Traffic is continuous variable.

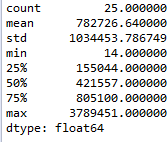




1. POI

POI, District attributes are continuous variable.





1. POI by District

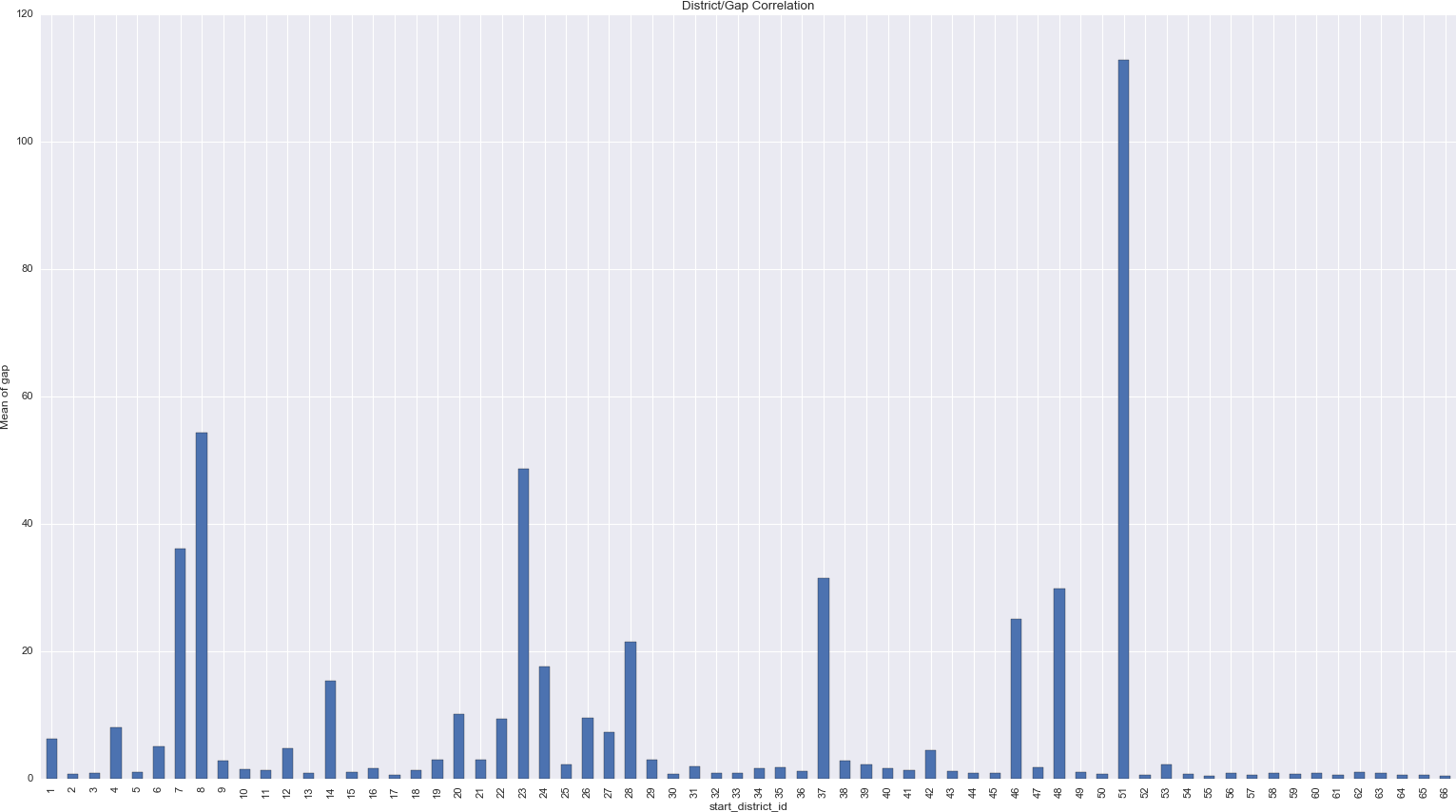
POI distribution for each district (66 districts in all)



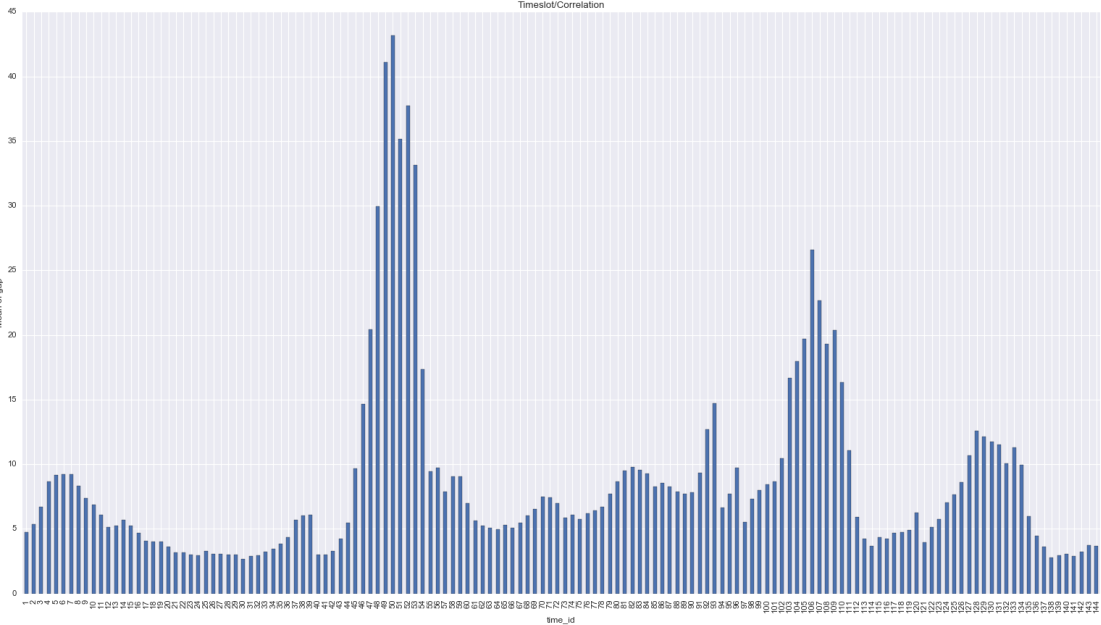
## Bivariate

This section visualize how gap correlate with various features. From below charts, we can see gap has much correction with district, timeslot, and has little correlation with traffic and weather.

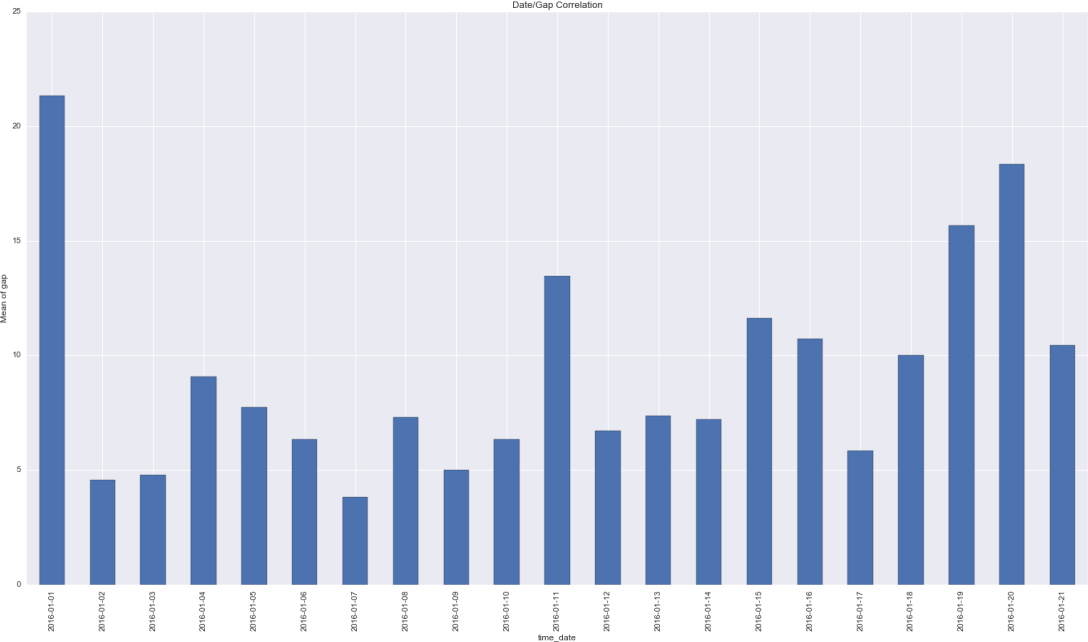
1. District/Gap



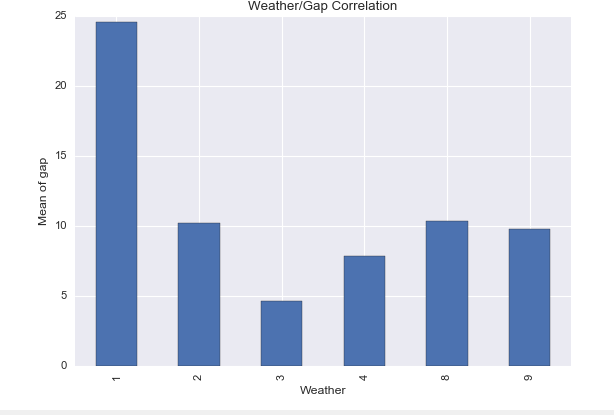
1. Timeslot/Gap



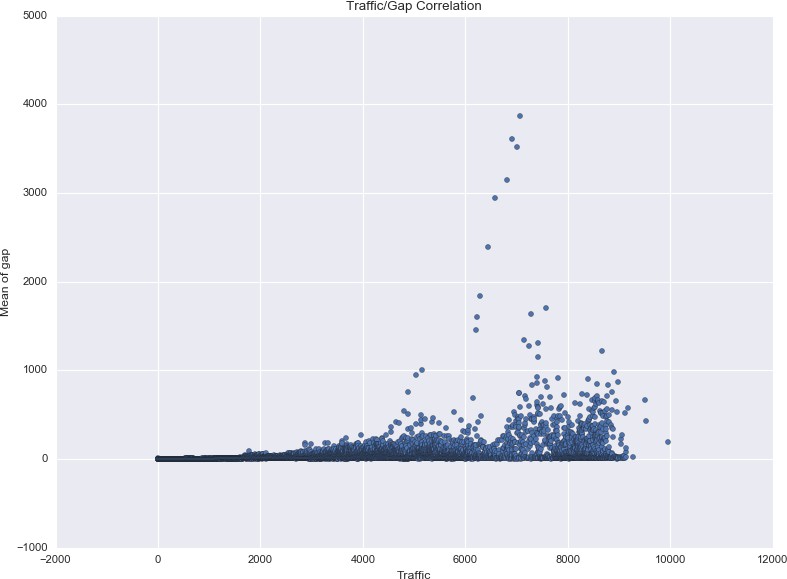
1. Date/Gap



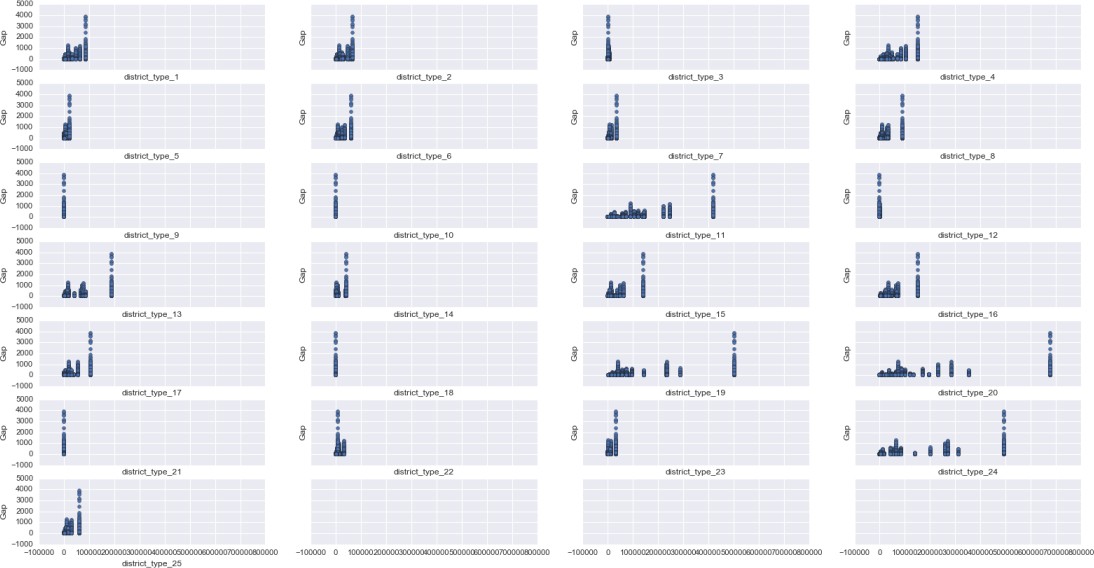
1. Weather/Gap



1. Traffic/Gap



1. POI/Gap



## Algorithms and Techniques

The target variable to be predicted Gapi,j+1 is a continuous variable, so we can attempt to solve this problem by regression algorithm. In this project, below algorithms are experimented:

1. KNN(Sklearn)

An instance based learning algorithm, it can be easily deployed and to see if what score a

simple algorithm can achieve. Need to beware that its prediction accuracy can quickly degrade when number of features grows due to curse of dimensionality.

1. Random forest(Sklearn)

A tree ensemble algorithm widely used in machine learning competitions. Very easy to use and very little parameters to tune.

1. Gradient Boosting machine (XGBoost)

A very popular algorithm used by the winning solutions in various Kaggle competitions. It has quite some parameters to be fine tuned.

1. Neural network (Tensorflow)

The champion learning algorithm in computer vision, speed recognition, and etc. It’s worth trying it out in this problem and see if how it might perform.

Below major techniques are used in this project:

1. Cross validation

As the problem is posed as a competition, the label for test dataset is not known to contestants. Having a solid cross validation strategy will allow us to quickly experiment new ideas without being limited by official test dataset prediction submission, which occurs only once a day

Also the cross validation will reduce the chance of overfitting public leaderboard in the process of fine tuning models.

1. Feature engineering

Original Input data is not directly usable and has to be transformed into a tabular format. Some new features will also be devised to improve model prediction capability.

1. Feature selection

For some algorithms, greedy forward feature selection is used to select input features out of all the engineered features; while for other algorithms, all engineered features are used as input features.

1. Grid search

Grid search is used to find optimal hyper parameter for models

## Benchmark

During the first round of competition, it is known that the best MAPE score in leaderboard is

0.246. Besides, the Didi MAPE metrics used by competition organizer is not a standard MAPE, in that it excludes those samples whose labels are zero, but at the same time counting them in the denominator, fortunately we can roughly map standard MAPE score to Didi MAPE as below.

DidiMAPE = standard MAPE / 0.636364

0.636364 is the percentage of non-zero gap samples.

As a result, the state of the art benchmark for this project is 0.387 (0.246/0.636364 )

# Methodology

## Data Preprocessing

After aggregating original data, we obtained four big tables as discussed above. Now we will utilize these four tables and generate one big table (with features and labels), this big table will then be fed into standard machine learning algorithm later on.

For each row in the big table, it can be uniquely identified ‘start\_district\_id’ and ‘time\_id’, and also contains other columns described below.

All the data preprocessing is implemented in preprocess/preparedata.py file.

## Feature Engineering

In all, below features are developed and experimented. All of them are listed in the Appendix section.

#### Recent Gap

Gap information in the half past hour, for a district i timeslot j, specifically:

1. gap1: Gapi,j-1
2. gap2: Gapi,j-2
3. gap3: Gapi,j-3
4. gap\_diff1: Gapi,j-2 - Gapi,j-1
5. gap\_diff2: Gapi,j-3 - Gapi,j-2
6. gap\_mean: Mean(Gapi,j-1, Gapi,j-2, Gapi,j-3)
7. gap\_std: Std(Gapi,j-1, Gapi,j-2, Gapi,j-3)

#### District related

1. start\_district\_id: it ranges from 1 to 66
2. 'start\_district\_id\_51', 'start\_district\_id\_23','start\_district\_id\_8','start\_district\_id\_37': the districts that has the biggest sum of gap
3. district\_gap\_sum, the sum of the gap for the district
4. district\_type\_1,district\_type\_2,district\_type\_3,district\_type\_4,district\_type\_5,district\_type\_ 6,district\_type\_7,district\_type\_8,district\_type\_9,district\_type\_10,district\_type\_11,district\_ty pe\_12,district\_type\_13,district\_type\_14,district\_type\_15,district\_type\_16,district\_type\_17,d istrict\_type\_18,district\_type\_19,district\_type\_20,district\_type\_21,district\_type\_22,district\_t ype\_23,district\_type\_24,district\_type\_25: poi information for the district
5. poi\_sum: sum of poi for the district

#### Time related

1) time\_id: it ranges from 1 to 144

#### Weather related

1. preweather, the weather in the past half an hour
2. rain\_check, whether it was raining in the past half an hour

#### Traffic related

1. traffic1: Traffici,j-1
2. traffic 2: Traffici,j-2
3. traffic 3: Traffici,j-3
4. traffic \_diff1: Traffic,j-2 - Traffic,j-1
5. traffic \_diff2: Traffici,j-3 - Traffici,j-2
6. traffic \_mean: Mean(Traffici,j-1, Traffici,j-2, Traffici,j-3)
7. traffic \_std: Std(Traffici,j-1, Traffici,j-2, Traffici,j-3)

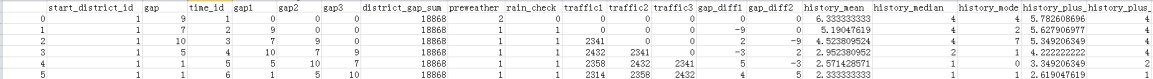
#### Long term Gap

The historical gap statistics for all district/time combinations.

history\_mean, history\_median, history\_mode, history\_plus\_mean, history\_plus\_median, history\_plus\_mode

#### Cross features

1. district\_time, combination of district and time
2. weather\_time, combination of weather and time Below is a subset of the preprocessed data:



## Feature transformation

#### One hot encoding

One hot encoding are applied to categorical variables like start\_district\_id, time\_id and cross features.

One hot encoding transform each categorical feature with m possible values into m binary features, with only one active. Theoretically, such a transformation can make it possible/easier for the model to learn the non-linear correlation between the label and categorical feature.

#### Scaling

KNN and Neural Network require that the input features to be normalized in order to do effective learning. A simple MinMaxScaler is applied to scale all features to range (0,1).

Xscaling = (X - Xmin) / (Xmax - Xmin)

Range scaling is used, as opposed to standardization (StandardScaler), because it can preserve zero entries for those columns generated by one hot encoding transformation above.

## Outlier/Missing value

For the gap column, its upper outer fence (Q3+3\*IQR, see [here](http://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm) for details) is 24.0. There are quite some outliers whose gap exceeds 24.0. However, it’s observed that all the data entries themselves are accurate and similar outliers also occur in test dataset. As a result, these outliers are retained.

As the evaluation metrics is MAPE, those rows that has zero gap will not be included in final score

calculation, in another word, we do not need to train our model to predict gap zero, and thus we can remove all rows with gap column being zero.

There is no missing value in the dataset so we don’t have to do anything in this regard.

## Implementation

All the algorithm implementation scripts are under implement folder of github project, please reference appendix section of this report for how to run the scripts.

## Metrics

MAPE metrics is implemented in evaluation/sklearnmape.py file. Basically it’s just one line np.mean(np.abs((y\_true - y\_pred) / y\_true.astype(np.float32)))

## KNN

Using sklearn library, KNN is implemented by implement/knnmodel.py.

The input features is scaled to range (0,1) before being fed into the model. Forward feature selection and grid search is used to fine tune the model. Cross validation is also used to evaluate the model.

To simplify model implementation, many common functionality like model training, validation, testing, saving are done in a base class classed BaseModel in utility/sklearnasemodel.py. KNN model reuse these codes by inheriting this base class.

## Random Forest

Using sklearn library, RandomForest is implemented by implement/randomforestmodel.py.

Grid search is used to fine tune the model, and cross validation is also used to evaluate the model.

Just like KNN model, we inherit BaseModel to simplify implementation.

## GBM

Using XGB library, GBM is implemented by implement/ xgboostmodel.py.

Grid search is used to fine tune the model, and cross validation is also used to evaluate the model.

The code implementation part is quite simple and straightforward. Two particular notes:

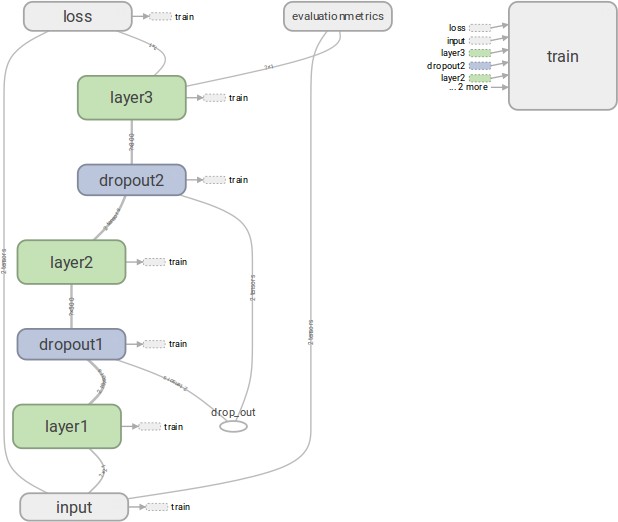
* Original Python API, as opposed to sklearn API, is used as sklearn API do not support early stopping during parameter grid search.
* Currently XGBoost library only accept KFold or StratifiedKFold instance in its cv API, so our customized folds (discussed in model evaluation and validation section) won’t do. I modified the XGBoost library so that it can also accept array (in the form of [(train\_index\_1, valiation\_index\_1), (train\_index\_2, validation\_index\_2),..]). I plan to request the XGBoost library author to pull this code change in the near future.

## Neural Network

Using tensorflow library, neural network model is implemented by implement/didineuralmodel.py.

The neural network model is fined tuned via feature selection, model architecture tuning, loss function experiment, learning rate adjustment and etc. Cross validation is used to evaluate the model.

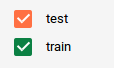
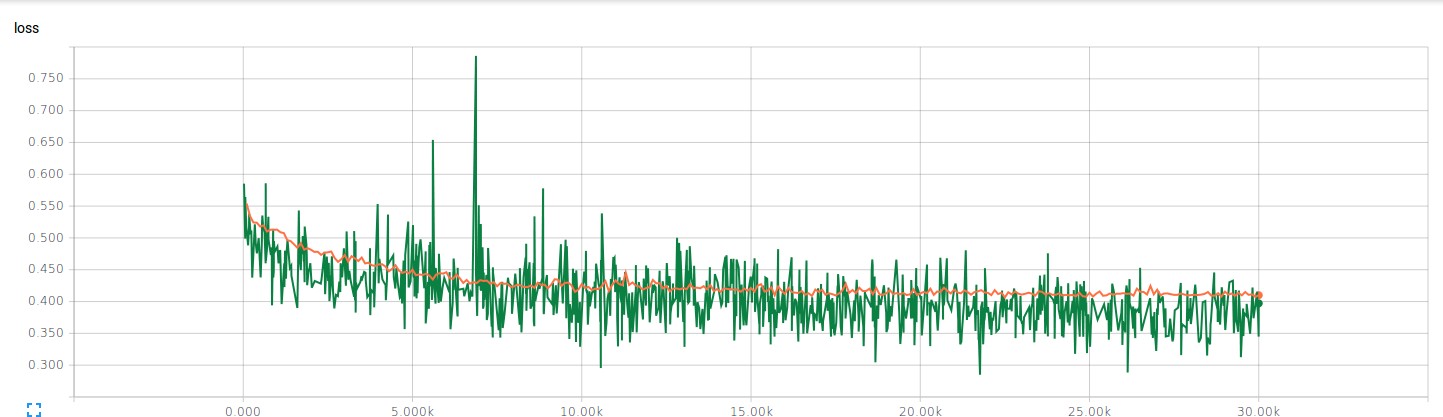
The final model architecture is as below:



#### Neural network model architecture

The number of nodes for each layer is as below: Input: 21, layer1: 500, layer2: 300, layer3:1

The final model’s learning chart is as below:



#### Neural network learning chart

The final model’s score is 0.423, and the training takes about 30 minutes to complete

## Greedy Forward feature selection

Greedy forward feature selection adds one feature dimension at a time to a set of already selected features (initially the set is empty), and checks how good the feature combination is performing as model input, then the best feature (in terms of model accuracy) is then added to the set of selected features, and then then next iteration begins.

This techniques is implemented by implement/ forwardfeatureselection.py

## Grid search

Grid search technique is used to search for optimal estimator parameters. Both exhaustive and randomized approaches are tried out in this project. Grid search is done on cross validation folds to avoid overfitting.

Using sklearn library, this technique is implemented in implement/tunealgorithm.py.

## Refinement

## KNN

1. Initial score: **0.540**

With all the features and default KNN parameters.

1. Forward feature selection: **0.475**

With forward feature selection and deault KNN paramters [history\_mode'], 0.5376

['history\_mode', 'gap1'], 0.5093

['history\_mode', 'gap1', 'history\_median'], 0.4832

['history\_mode', 'gap1', 'history\_median', 'gap3'], 0.4895 ['history\_mode', 'gap1', 'history\_median', 'gap3', 'start\_district\_id\_51', 'start\_district\_id\_23', 'start\_district\_id\_8', 'start\_district\_id\_37'], 0.4750

['history\_mode', 'gap1', 'history\_median', 'gap3', 'start\_district\_id\_51', 'start\_district\_id\_23', 'start\_district\_id\_8', 'start\_district\_id\_37', 'history\_plus\_mode'], 0.47942

1. Grid search for hyper parameters: **0.447**

Search optimal n\_neighbors parameter, ranging from 30 to 35

Best Score/parameter: means, 0.44659450725, std, 0.00389443248812, details,[ 0.45069229

0.44135876 0.44773247]

## Random Forest

1. Initial score: **0.683**

With all the features and default parameters

1. Grid search for hyper parameters: **0.671**

With all the features, grid search {'n\_estimators': [100], 'max\_features':['auto', 0.3,0.8,0.5]} Best score/parameters: mean: 0.67052, std: 0.01996, params: {'max\_features': 0.8,

'n\_estimators': 100}

## GBM

1. Initial score: **0.464**

With all the features and default parameters

1. Grid search for hyper paramters: **0.453**

Search [0.01], 'max\_depth':range(5,8), 'colsample\_bytree': [0.6,0.8,1.0]}

Best score/parameters: mean: 0.45301966666666665, std: 0.0013799657805741092,

params: {'colsample\_bytree': 0.8, 'silent': 1, 'lambda ': 1, 'min\_child\_weight': 1, 'subsample':

0.8, 'eta': 0.01, 'objective': 'reg:linear', 'max\_depth': 7}

## Neural Network

Experiments are done on quite several key aspects of the neural network model, among them

1. Number of hidden layer

With layer2 VS without layer2 Without layer2: **0.445**

With layer2: **0.423**

1. Loss function MAPE VS MSE

Use either MAPE or MSE as the loss function. It turns out that using MAPE as the loss function is much more superior than using MSE. This is probably because MAPE as an evaluation metric is not very stable(3), and does not correlate very well with MSE. Since our final evaluation metric is MAPE, optimize over MAPE is more direct and efficient.

MSE as loss function: **0.824**

MAPE as loss function: **0.423**

1. Input features:

all features VS partial features:

all features: **0.430**

partial features(101,102,103,104,105, 201, 204,

203,301,401,402,501,502,503,601,602,603,604,605,606,8801,8802): **0.423**

Please see appendix section see feature id/name mapping.

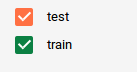
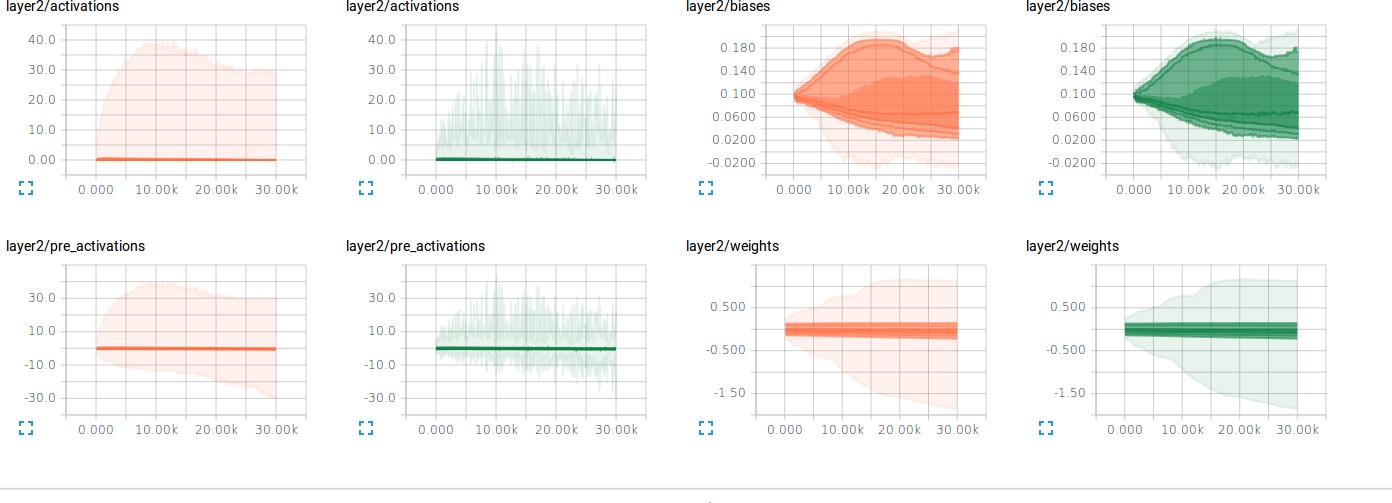
# Results

## Model Evaluation and Validation

Neural network model is chosen as the final model because it outperforms KNN, RandomForest, GMB.

#### Final module’s parameters

Final module’s parameters mainly consist of weights and bias in its layers. Pick layer2 as an



example, we can see they look quite reasonable.

#### Neural network model layer2 weights/bias/activations

1. **Cross validation strategy:**

The competition organizer did not publish its test dataset even after the competition ends, so cross validation is the best approach we can utilize to ensure/measure the performance and robustness of the model.

The “Gap” prediction is, in essence, a time series prediction; we are required to use information in the past to predict the “Gap” in the future. A common k-fold cross validation split is not appropriate here since it practically allows the model to use future information to predict the gap, As a result, a time based forward chaining cross validation strategy is used. Specifically, three folds are used:

1) training 2016-1-1 ------ 2016-1-16, validation 2016-1-17 2016-1-21

2) training 2016-1-1 ------ 2016-1-17, validation 2016-1-18-----2016-1 21

3) training 2016-1-1 ------ 2016-1-18, validation 2016-1-19 2016-1—21

Moreover, to best mimic the train/test split. Not all timeslots are used in test training/validation, specifically,

For validation, use the same timeslots listed in the test dataset. [46, 58, 70, 82, 94, 106, 118, 130,

142]

For training, timeslots in test dataset plus two adjacent timeslots [ 44 45 46 47 48 56

57 58 59 60 68 69 70 71 72 80 81 82

83 84 92 93 94 95 96 104 105 106 107 108 116 117 118 119 120 128

129 130 131 132 140 141 142 143 144]

Cross validation strategy is implemented by preprocess/ splittrainvalidation.py

#### Final result robustness

The score of final model on the three CV folds are mean: 0.423, std: 0.006 scores: [ 0.415 0.425 0.43 ]

The range of the scores is 0.015 and acceptable, and we would say the module’s accuracy is roughly 0.42

## Justification

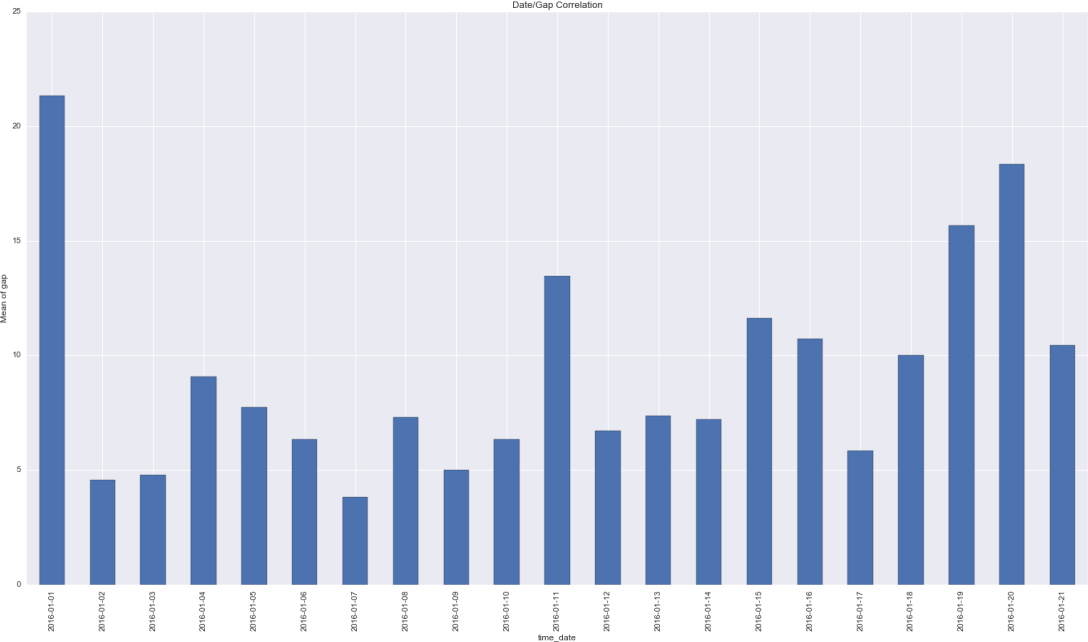
As mentioned in the benchmark section, the best score in leaderboard is about 0.39. Our final model’s score is 0.42.

A score of 0.42 is not really state of the art for this competition, but still it is quite respectable. This score gap of 0.03 might be further narrowed by experimenting some improvement ideas listed in improvement section of this document.

# Conclusions

## Free-Form Visualization

Date/Gap correlation chart is an interesting and telling visualization. Its x axis represents the date, ranging from January 1st to January 21st. Its y axis represents the mean of gap for each day. From the chart, we can see that weekday/weekend does not appear to have much impact on the gap. As a result, no such date related features are developed.



#### Date/Gap correlation

## Reflection

The development life cycle of this machine learning project can be summarized using below steps:

1. Define the problem as a supervised regression learning problem
2. Define MAPE as the metrics to evaluate the models
3. Explore the data type, basic statistics and distribution of features and labels, perform univariate and bivariate visualizations to gain insight into the data, and guide feature engineering and cross validation strategy.
4. Identify KNN, Random Forest, GBM Neural network as potential models/algorithms, Find out state of art benchmark that these models aims to reach/beat.
5. Perform feature engineering, feature transformation, outlier and missing value handling.
6. Implement models by leveraging Sklearn, XGBoost, and Tensorflow learning libraries.
7. Fine tune the models via iterative feature selection/engineering, model selection, hyper parameter tuning. Cross validation is used to ensure that the models generalize well into unseen data.

Some interesting aspects of the project:

1. MAPE instability

MAPE is known to be unstable, and choose it as the evaluation metric is probably not a very good idea. I guess that’s why during the second round of the completion, the organizer changed the evaluation metric to MAE.

While comparing the models, I often observe an interesting point: MSE and MAPE do not correlate with one other very well. Sometimes the model with higher MSE score has lower MAPE score. I think this is the very reason why we are seeing a significant score improvement after changing loss function from MSE to MAPE in neural network model, directly optimizing

over MAPE is more efficient since our final evaluation metric is MAPE.

1. Power of neural network

Before doing this project, I’ve been under the impression that neural network is well fit for sequence data like image, speech, while XGB excels at structured data.

Our problem is a typical structured data, and I am much surprised to see neural network outperforms XGB here. Anyway, a pleasant surprise and a great deal to explore in the arena of neural network.

## Improvement

Potential ways to further push the score:

1. Model ensemble

Model ensemble is commonly used in almost all Kaggle completions, so it would be a promising trick to boost the score (4).

1. Neural network architecture/fine tuning

Experiment deeper neural network or Wide&Deep model(5)

1. Feature engineering

Feature engineering is often perceived as more important than model selection/fine tuning for many problems, so this is also a promising direction to explore.

# Reference

1. The Di-Tech Challenge Website<http://research.xiaojukeji.com/competition/main.action?competitionId=DiTech2016>
2. Input Data Format<http://research.xiaojukeji.com/competition/detail.action?competitionId=DiTech2016>
3. MAPE instability<https://en.wikipedia.org/wiki/Mean_absolute_percentage_error>
4. Model ensemble

<http://mlwave.com/kaggle-ensembling-guide/>

1. Wide&Deep neural network architecture<https://www.tensorflow.org/versions/r0.10/tutorials/wide_and_deep/index.html>