Rider-Driver Supply and Demand Gap Forecast

Udacity Machine Learning Engineer Nanodegree Capstone Project Report

Levin Jian, August 2016,

Contents

1 Definition

1.1 Project Overview

The Di-Tech Challenge[1] is organized by DiDi Chuxing, China's largest ride-hailing company. It challenges contestants to use real data to generate predictive rider-driver supply and demand gap model, to direct drivers to where riders will need to be picked up.

Supply-demand forecasting helps to predict the gap of drivers and riders at a certain time period in a specific geographic area, and it is critical to enabling Didi to maximize utilization of drivers and ensure that riders can always get a car whenever and wherever they may need a ride.

When I learned of the challenge announcement, I just completed all the required courses in Udacity Machine Learning Engineer Nanodegree. It's exciting to choose this competition as my Capstone project, practice and consolidate what I've learned throughout the Nanodegree program by tackling a real world problem, and what's more, in a highly competitive ongoing contest!

This project report describes my solution for this competition.

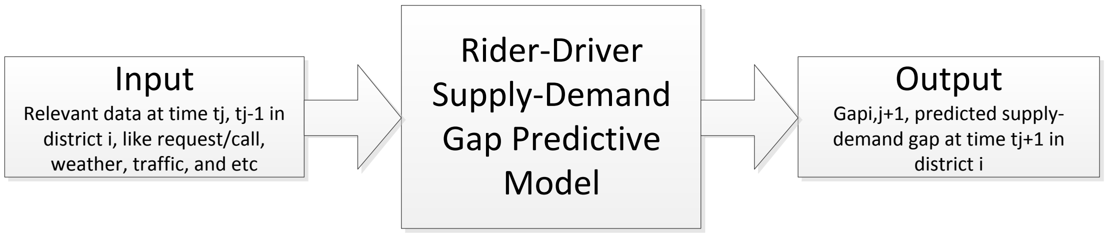
1.2 Problem Statement

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

Didi divides a city into n non-overlapping square districts $D=\{d_1,d_2,\cdots,d_n\}$ and divides one day uniformly into 144 time slots $t_1,t_2,\cdots,t_{144}$, each 10 minutes long.

At time slot t_j In district d_i, the number of passengers' requests is denoted as r_{ij}, and drivers' answers as a_{ij}, and the demand supply gap is $gap_{ij} : r_{ij} - a_{ij}$ Given relevant data concerning time slot t_j,t_{j-1} in district i, you need to predict $gap_{i,j+1}, \forall d_i \in D$.

This problem is, in essence, a supervised regression problem as shown below:


Input
Relevant data at time tj, tj-1 in district i, like request/call, weather, traffic, and etc
Rider-Driver Supply-Demand Gap Predictive Model
Output
Gapi,j+1, predicted supply-demand gap at time tj+1 in district i

Problem definition---a regression problem

1.3 Metrics

Given i districts and j time slots, for district di at time slot tj, suppose that the real supply-demand gap is gap_{ij} , and predicted supply-demand gap is s_{ij}, then:

MAPE = $\frac{1}{n}\sum_{i,j}(\frac{|gapij-sij|}{gapij})$, given $gap_{i,j}$ is not equal to zero

The lowest MAPE will be the best.
Sample calculation:
y_true = [3, -0.5, 2, 0,7]; y_pred = [2.5, -0.3, 2, 8,0]
MAPE(y_true, y_pred) = 0.391666666667

2 Analysis

2.1 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.

2.1.1 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	string	order ID	70fc7c2bd2caf386bb50f8fd5dfef0cf
driver_id	string	driver ID	56018323b921dd2c5444f98fb45509de

passenger_id	string	user ID	238de35f44bbe8a67bdea86a5b0f4719
start_district_hash	string	departure	d4ec2125aff74eded207d2d915ef682f
dest_district_hash	string	destination	929ec6c160e6f52c20a4217c7978f681
Price	double	Price	37.5
Time	string	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.

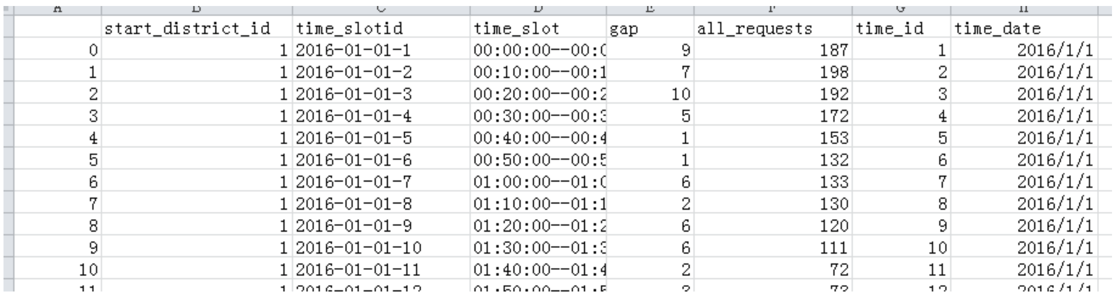
2.1.2 Aggregated Data

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.

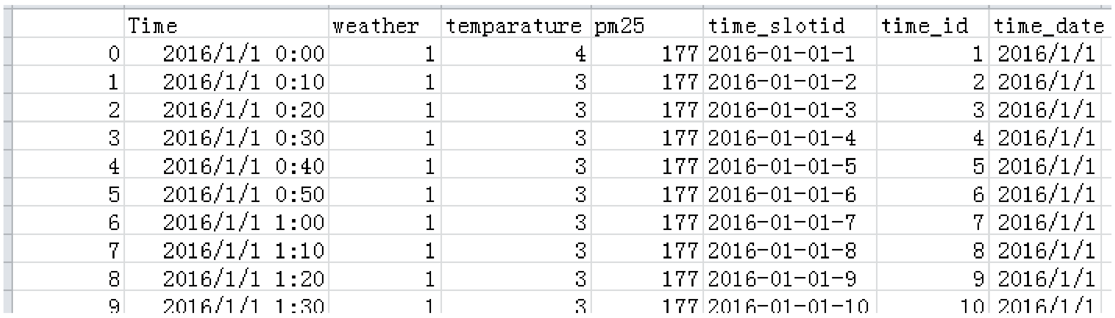


	start_district_id	time_slotid	time_slot	gap	all_requests	time_id	time_date
0	1	2016-01-01-1	00:00:00--00:0	9	187	1	2016/1/1
1	1	2016-01-01-2	00:10:00--00:1	7	198	2	2016/1/1
2	1	2016-01-01-3	00:20:00--00:2	10	192	3	2016/1/1
3	1	2016-01-01-4	00:30:00--00:3	5	172	4	2016/1/1
4	1	2016-01-01-5	00:40:00--00:4	1	153	5	2016/1/1
5	1	2016-01-01-6	00:50:00--00:5	1	132	6	2016/1/1
6	1	2016-01-01-7	01:00:00--01:0	6	133	7	2016/1/1
7	1	2016-01-01-8	01:10:00--01:1	2	130	8	2016/1/1
8	1	2016-01-01-9	01:20:00--01:2	6	120	9	2016/1/1
9	1	2016-01-01-10	01:30:00--01:3	6	111	10	2016/1/1
10	1	2016-01-01-11	01:40:00--01:4	2	72	11	2016/1/1
11	1	2016-01-01-12	01:50:00--01:5	3	73	12	2016/1/1

   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.

2. Weather table

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

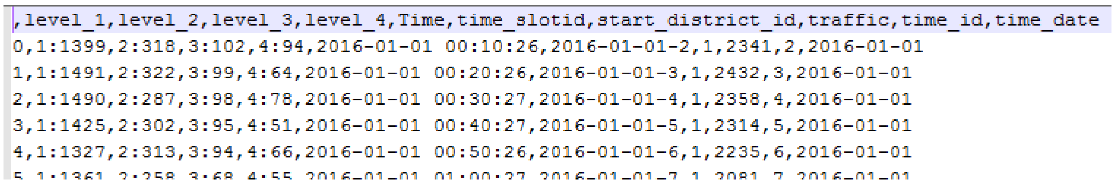


	Time	weather	temparature	pm25	time_slotid	time_id	time_date
0	2016/1/1 0:00	1	4	177	2016-01-01-1	1	2016/1/1
1	2016/1/1 0:10	1	3	177	2016-01-01-2	2	2016/1/1
2	2016/1/1 0:20	1	3	177	2016-01-01-3	3	2016/1/1
3	2016/1/1 0:30	1	3	177	2016-01-01-4	4	2016/1/1
4	2016/1/1 0:40	1	3	177	2016-01-01-5	5	2016/1/1
5	2016/1/1 0:50	1	3	177	2016-01-01-6	6	2016/1/1
6	2016/1/1 1:00	1	3	177	2016-01-01-7	7	2016/1/1
7	2016/1/1 1:10	1	3	177	2016-01-01-8	8	2016/1/1
8	2016/1/1 1:20	1	3	177	2016-01-01-9	9	2016/1/1
9	2016/1/1 1:30	1	3	177	2016-01-01-10	10	2016/1/1

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.

3. Traffic table

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



```
,level_1,level_2,level_3,level_4,Time,time_slotid,start_district_id,traffic,time_id,time_date
0,1:1399,2:318,3:102,4:94,2016-01-01 00:10:26,2016-01-01-2,1,2341,2,2016-01-01
1,1:1491,2:322,3:99,4:64,2016-01-01 00:20:26,2016-01-01-3,1,2432,3,2016-01-01
2,1:1490,2:287,3:98,4:78,2016-01-01 00:30:27,2016-01-01-4,1,2358,4,2016-01-01
3,1:1425,2:302,3:95,4:51,2016-01-01 00:40:27,2016-01-01-5,1,2314,5,2016-01-01
4,1:1327,2:313,3:94,4:66,2016-01-01 00:50:26,2016-01-01-6,1,2235,6,2016-01-01
5,1:1361,2:258,3:68,4:55,2016-01-01 01:00:27,2016-01-01-7,1,2081,7,2016-01-01
```

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.

4. POI information Table

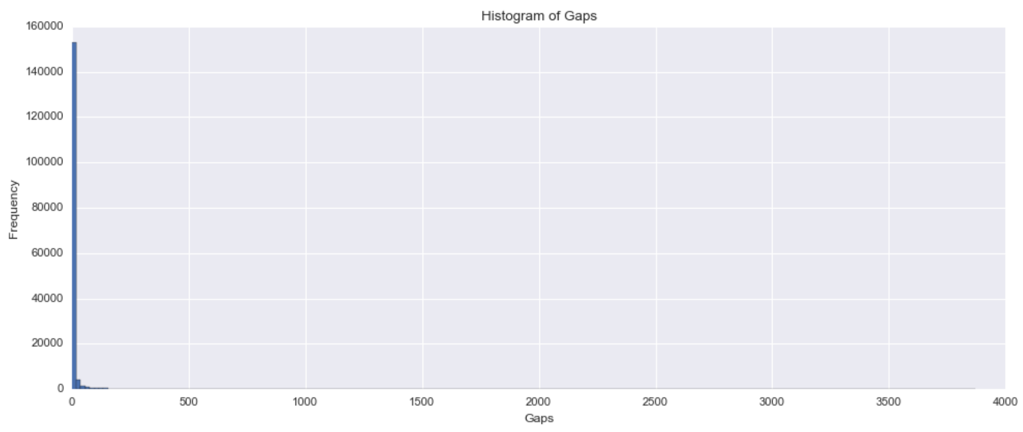
POI information table is the same as original table.

2.2 Data Statistics & Exploratory Visualization

2.2.1 Univariate

1. Gap

Gap is a continuous variable.

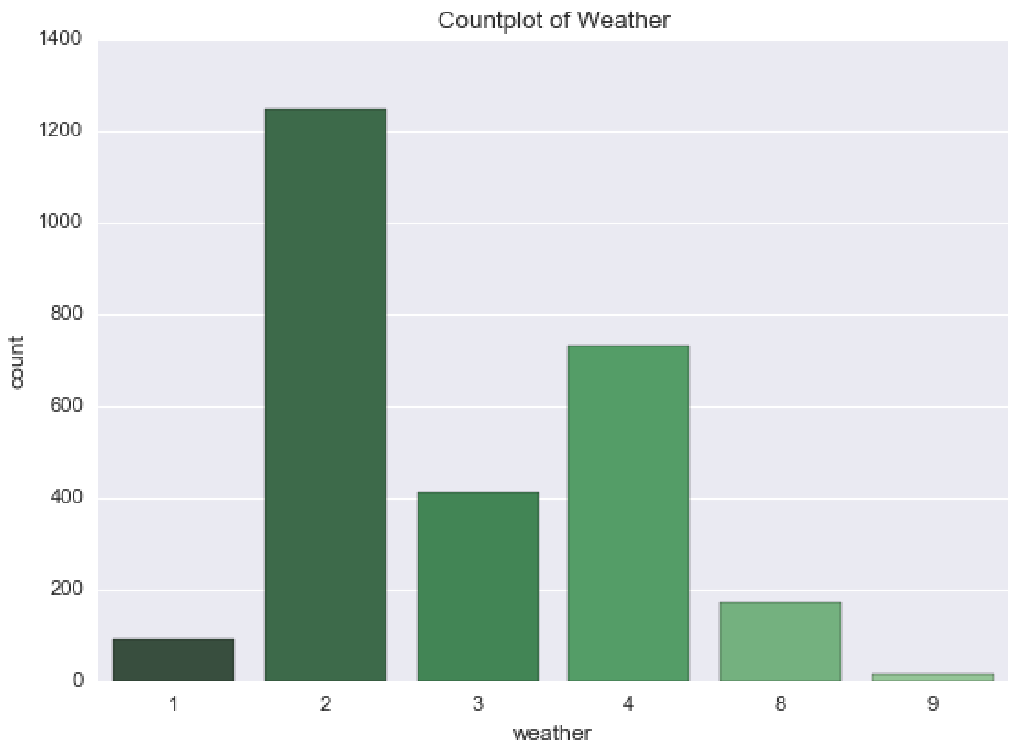


```
count    163491.000000
mean          9.275495
std          49.817422
min           0.000000
25%           0.000000
50%           1.000000
75%           4.000000
max        3872.000000
Name: gap, dtype: float64
```

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

2. Weather

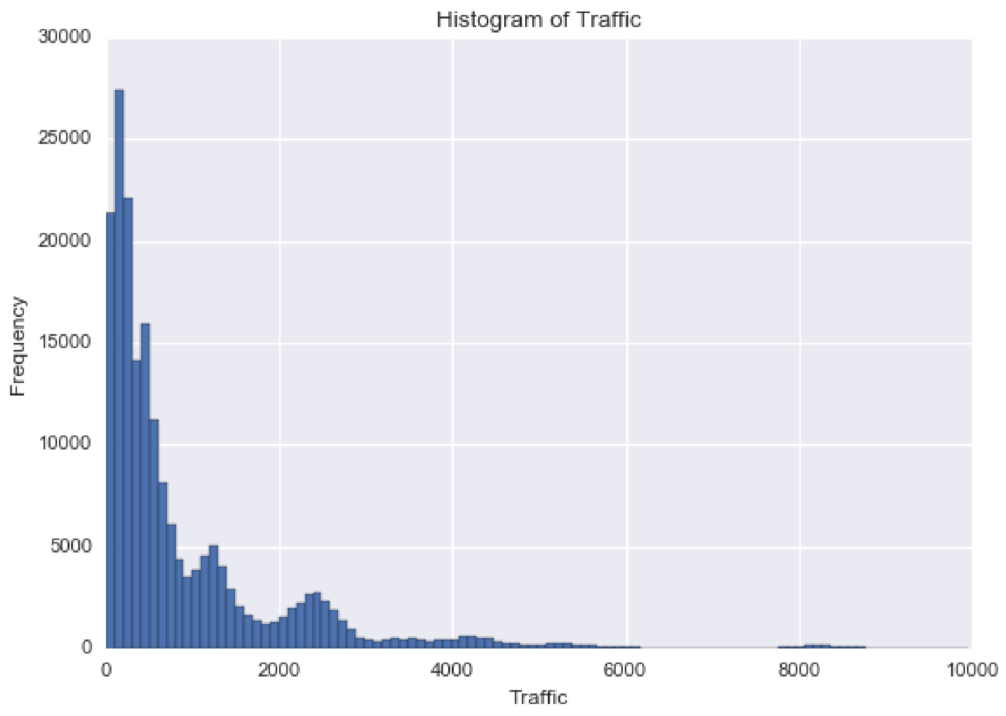
Weather is a categorical variable.



```
count    2670.000000
mean        3.098127
std         1.642939
min         1.000000
25%         2.000000
50%         2.000000
75%         4.000000
max         9.000000
```

3. Traffic

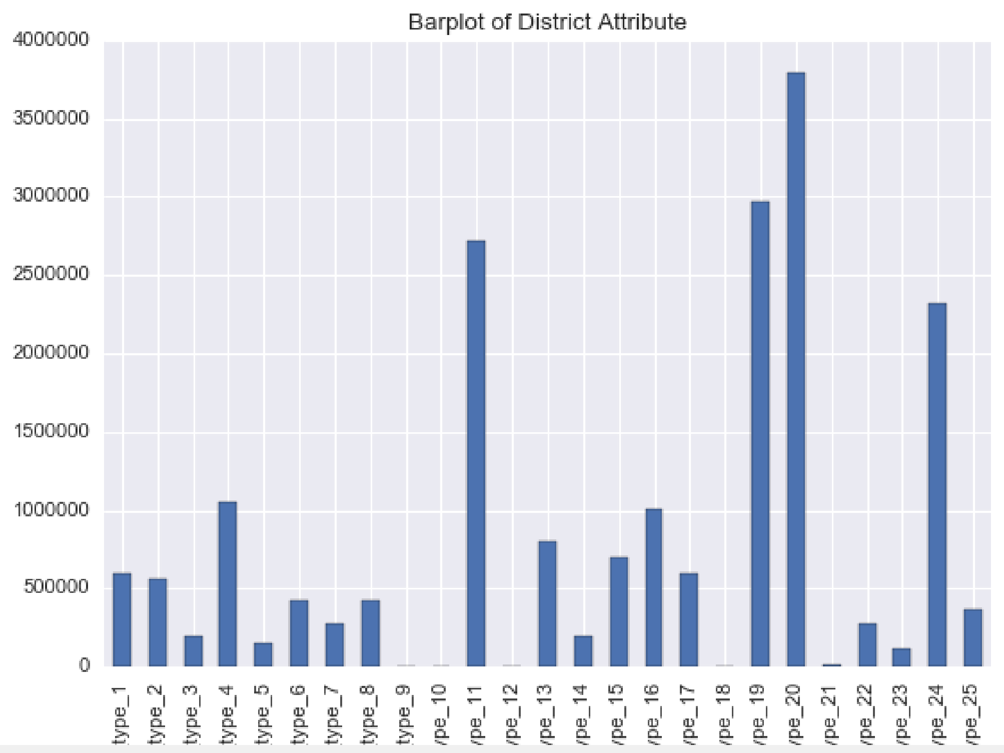
Traffic is continuous variable.



```
count    193553.000000
mean        977.248092
std        1294.344774
min           0.000000
25%         198.000000
50%         469.000000
75%        1241.000000
max        9956.000000
Name: traffic, dtype: float64
```

4. POI

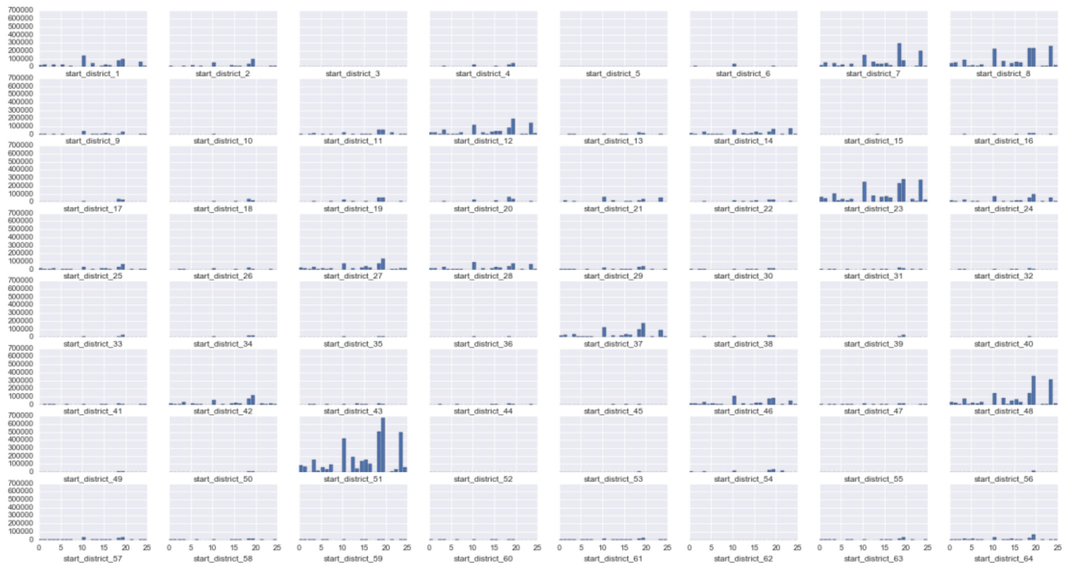
POI, District attributes are continuous variable.



```
count          25.000000
mean       782726.640000
std       1034453.786749
min            14.000000
25%        155044.000000
50%        421557.000000
75%        805100.000000
max       3789451.000000
dtype: float64
```

5. POI by District

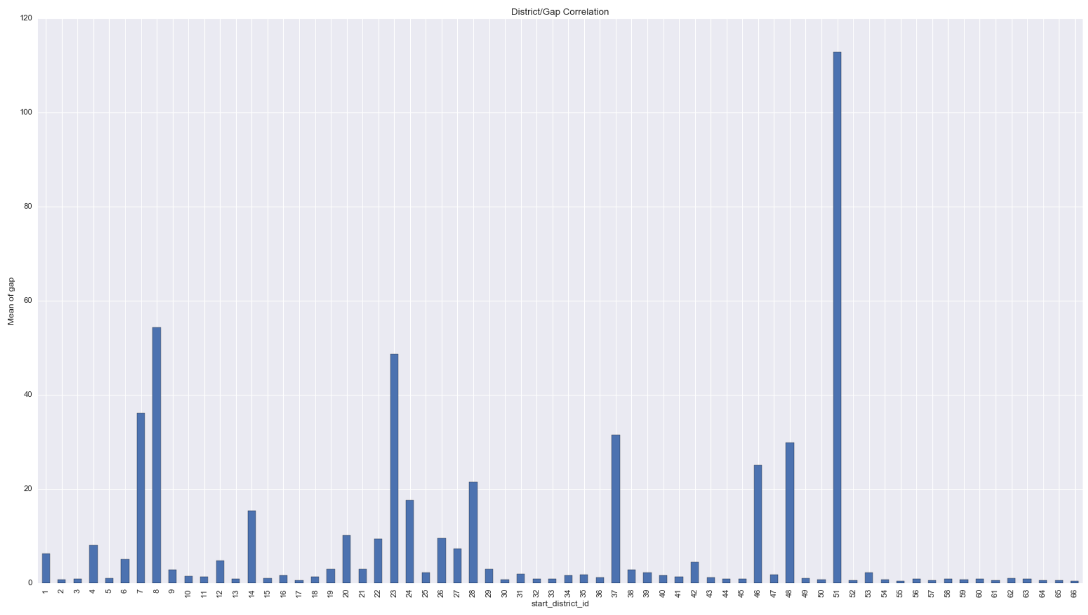
POI distribution for each district (66 districts in all)



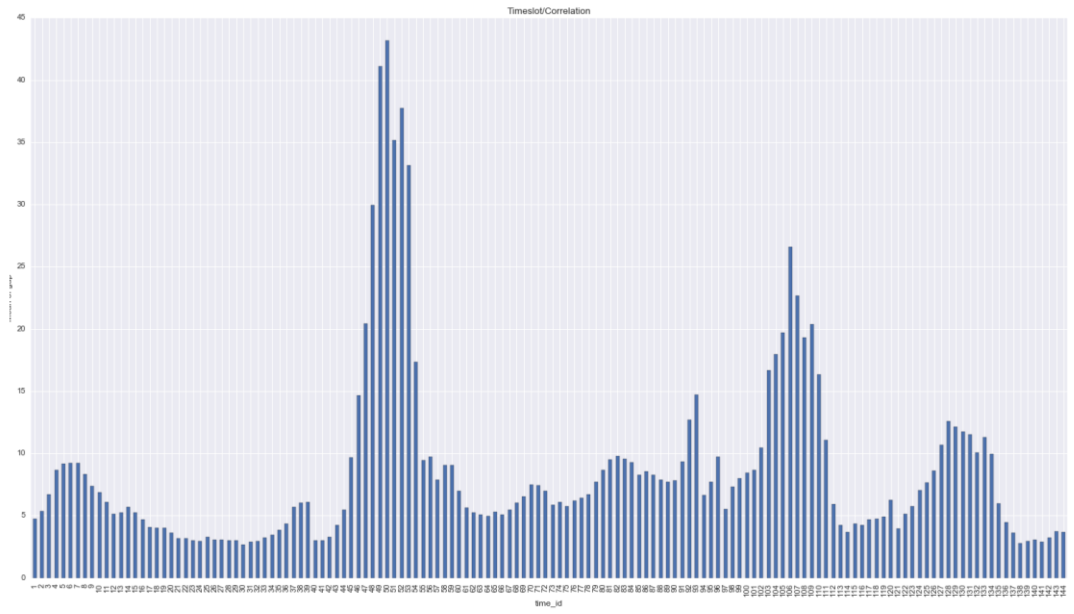
2.2.2 Bivariate

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.

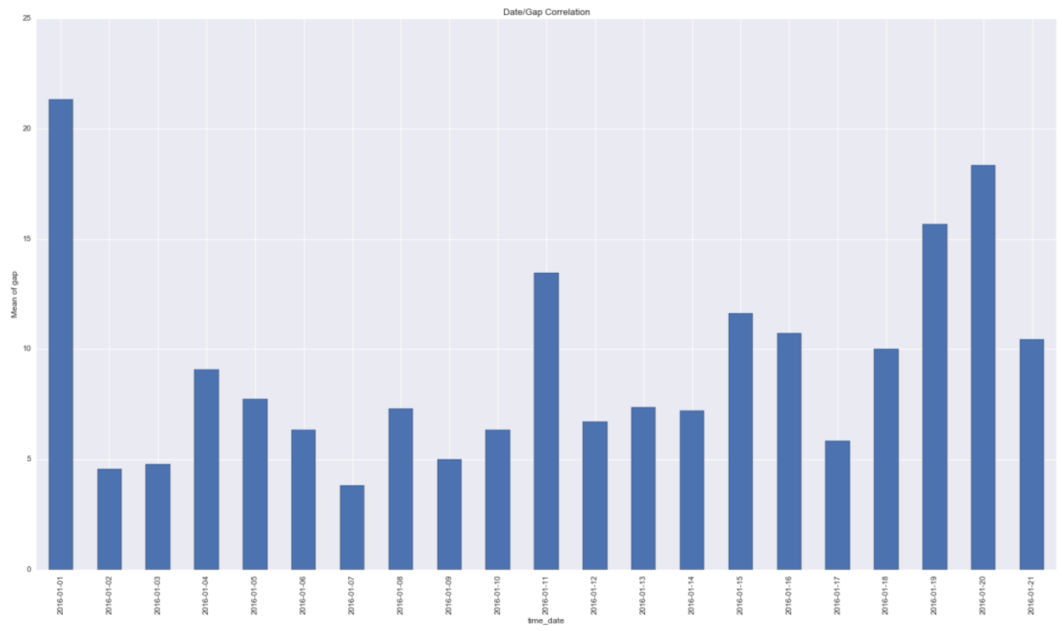
1. District/Gap



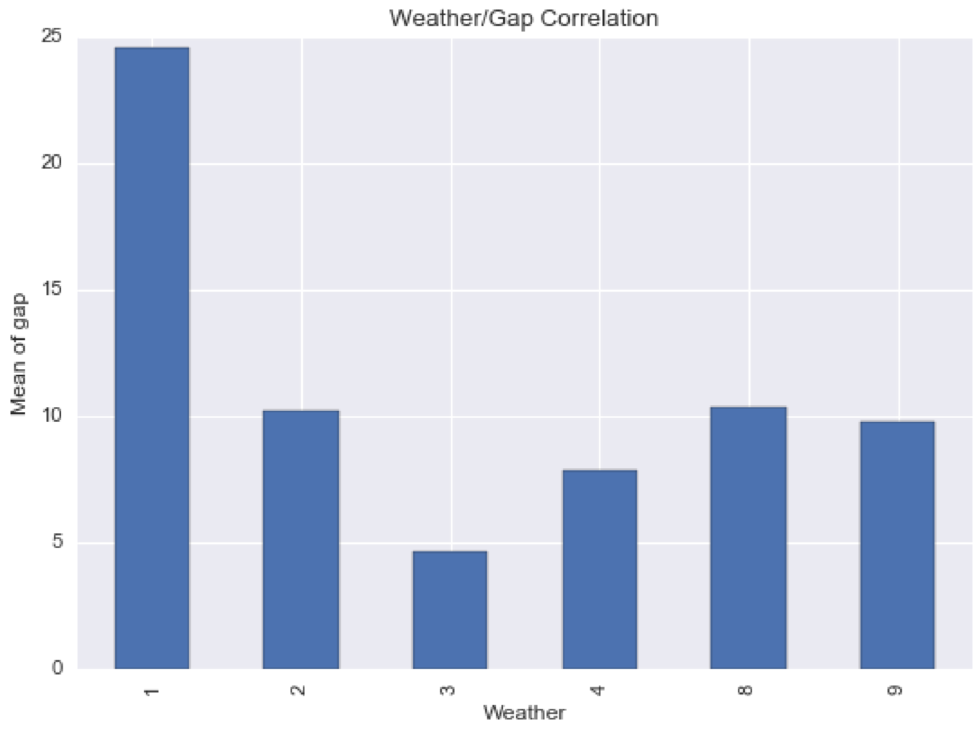
2. Timeslot/Gap



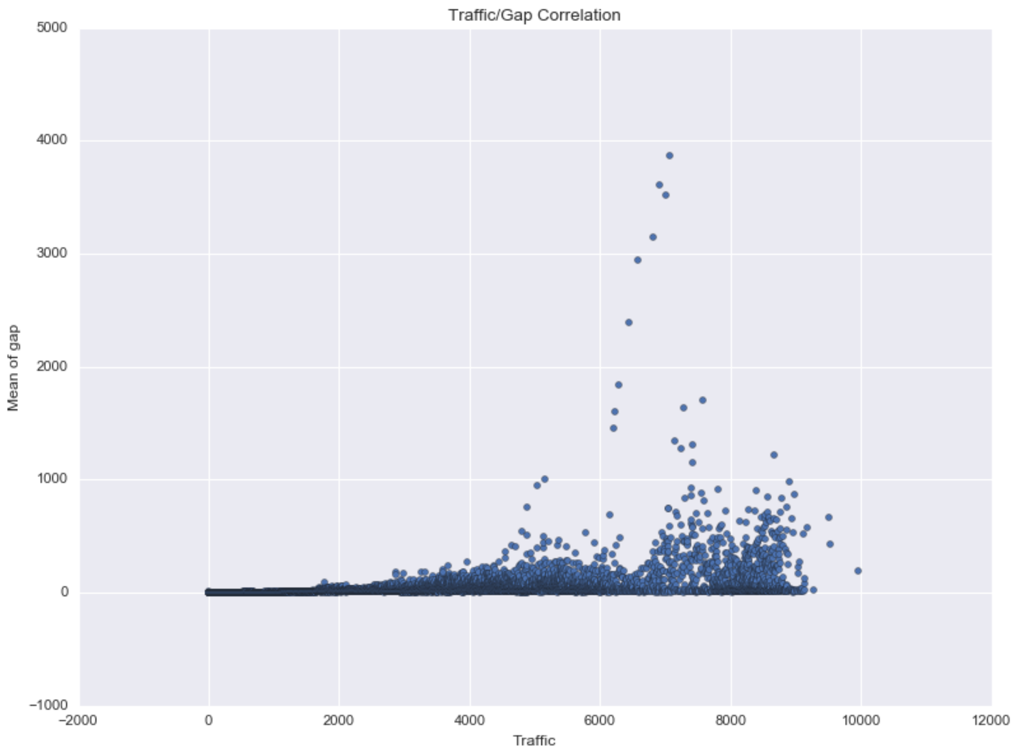
3. Date/Gap



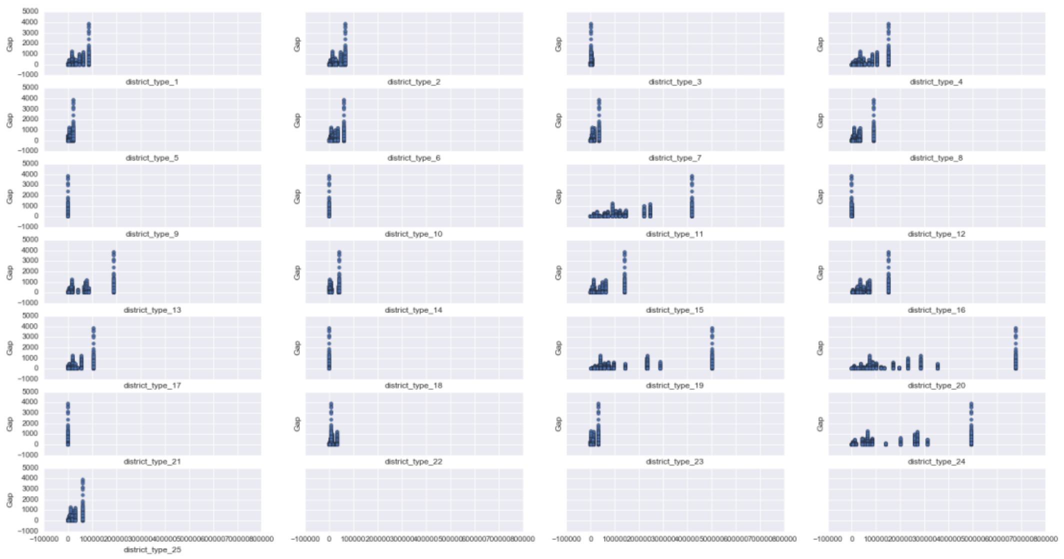
4. Weather/Gap



5. Traffic/Gap



6. POI/Gap



2.3 Algorithms and Techniques

The target variable to be predicted $\text{Gap}_{i,j+1}$ 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.

2. Random forest(Sklearn)

   A tree ensemble algorithm widely used in machine learning competitions. Very easy to use and very little parameters to tune.
3. 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.
4. 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.
2. 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.
3. 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.
4. Grid search

   Grid search is used to find optimal hyper parameter for models

2.4 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)

3 Methodology

3.1 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.

3.1.1 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: $Gap_{i,j-1}$
2) gap2: $Gap_{i,j-2}$
3) gap3: $Gap_{i,j-3}$
4) gap_diff1: $Gap_{i,j-2} - Gap_{i,j-1}$
5) gap_diff2: $Gap_{i,j-3} - Gap_{i,j-2}$
6) gap_mean: $Mean(Gap_{i,j-1}, Gap_{i,j-2}, Gap_{i,j-3})$
7) gap_std: $Std(Gap_{i,j-1}, Gap_{i,j-2}, Gap_{i,j-3})$

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_type_12,district_type_13,district_type_14,district_type_15,district_type_16,district_type_17,district_type_18,district_type_19,district_type_20,district_type_21,district_type_22,district_type_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: $Traffic_{i,j-1}$
2) traffic 2: $Traffic_{i,j-2}$
3) traffic 3: $Traffic_{i,j-3}$
4) traffic _diff1: $Traffic_{,j-2}$ - $Traffic_{,j-1}$
5) traffic _diff2: $Traffic_{i,j-3\,-}$ $Traffic_{i,j-2}$
6) traffic _mean: Mean($Traffic_{i,j-1,}$ $Traffic_{i,j-2,}$ $Traffic_{i,j-3}$)
7) traffic _std: Std($Traffic_{i,j-1,}$ $Traffic_{i,j-2,}$ $Traffic_{i,j-3}$)

Long term Gap

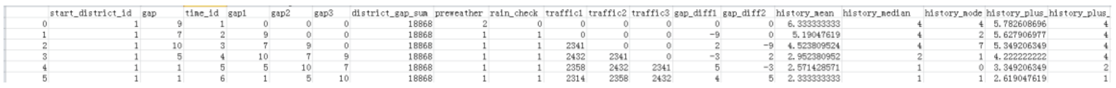
The historical gap statistics for all district/time combinations.

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:



	start_district_id	gap	time_id	gap1	gap2	gap3	district_gap_sum	preweather	rain_check	traffic1	traffic2	traffic3	gap_diff1	gap_diff2	history_mean	history_median	history_mode	history_plus_	history_plus_
0	1	9	1	0	0	0	18868	2	0	0	0	0	0	0	6.333333333	4	4	5.782608696	4
1	1	7	2	9	0	0	18868	1	1	0	0	0	-9	0	5.19047619	4	2	5.627906977	4
2	1	10	3	7	9	0	18868	1	1	2341	0	0	2	-9	4.523809524	4	7	5.349206349	4
3	1	5	4	10	7	9	18868	1	1	2432	2341	0	-3	2	2.952380952	2	1	4.222222222	4
4	1	1	5	5	10	7	18868	1	1	2358	2432	2341	5	-3	2.571428571	1	0	3.349206349	2
5	1	1	6	1	5	10	18868	1	1	2314	2358	2432	4	5	2.333333333	1	1	2.619047619	1

3.1.2 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.

3.1.3 Outlier/Missing value

For the gap column, its upper outer fence (Q3+3*IQR, see here 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.

3.2 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.

3.2.1 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)))

3.2.2 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.

3.2.3 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.

3.2.4 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.

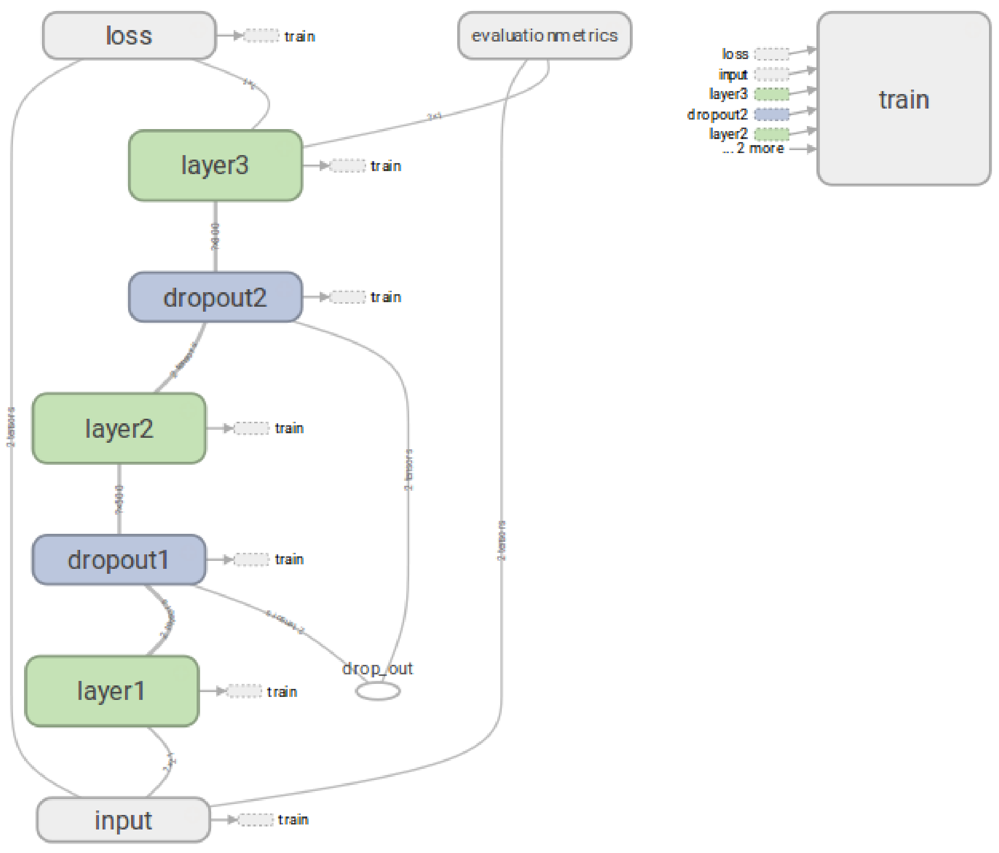
- Currently XGBoost library only accept KFold or StratifiedKFold instance in its cv API, so our customized folds (discussed in model evaluation and validation section) won't do. I modified the XGBoost library so that it can also accept array (in the form of [(train_index_1, valiation_index_1), (train_index_2, validation_index_2),..]). I plan to request the XGBoost library author to pull this code change in the near future.

3.2.5 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:

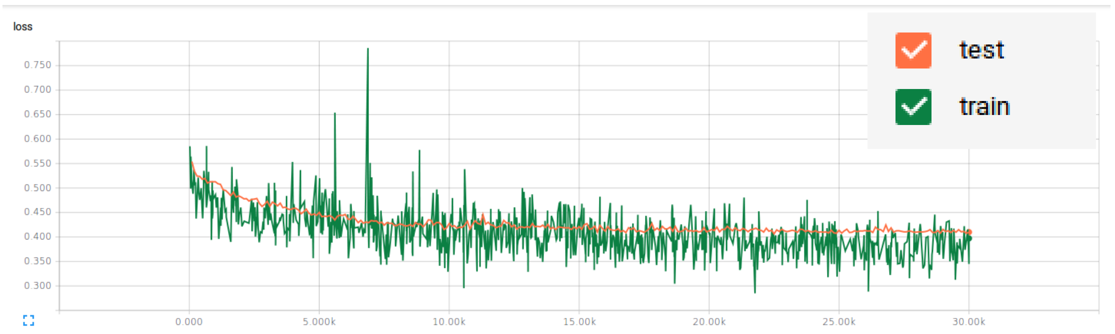


Neural network model architecture

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

3.2.6 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

3.2.7 Grid search

Grid search technique is used to search for optimal estimator parameters. Both exhaustive and randomized approaches are tried out in this project.
Using sklearn library, this technique is implemented in implement/tunealgorithm.py.

3.3 Refinement

3.3.1 KNN

1) Initial score: **0.540**
   With all the features and default KNN parameters.
2) 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

3) 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]

3.3.2 Random Forest

1) Initial score: **0.683**
   With all the features and default parameters
2) 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}

3.3.3 GBM

1) Initial score: **0.464**
   With all the features and default parameters
2) 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}

3.3.4 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**
2. 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**
3. Input features:

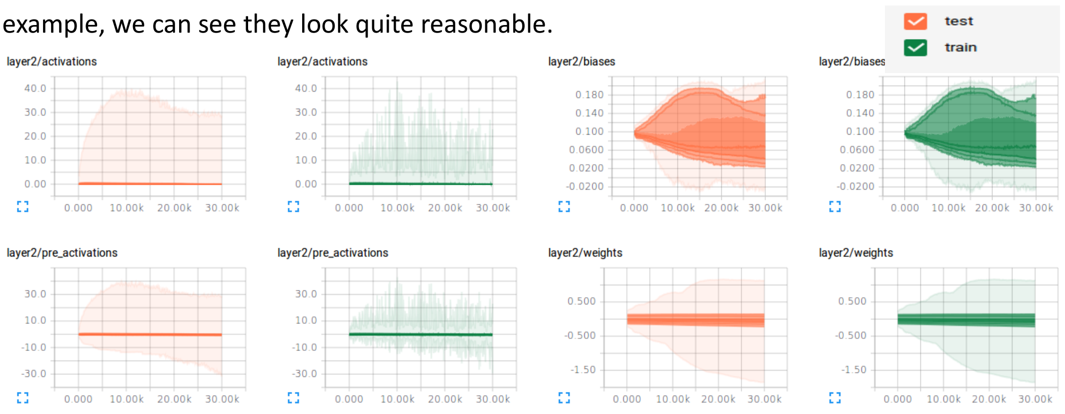
all features VS partial features:
all features: **0.430**
partial features(101,102,103,104,105,   201, 204,
203,301,401,402,501,502,503,601,602,603,604,605,606,8801,8802): **0.423**
Please see appendix section see feature id/name mapping.

4 Results

4.1 Model Evaluation and Validation

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

1. **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

2. **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

3. 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

4.2 Justification

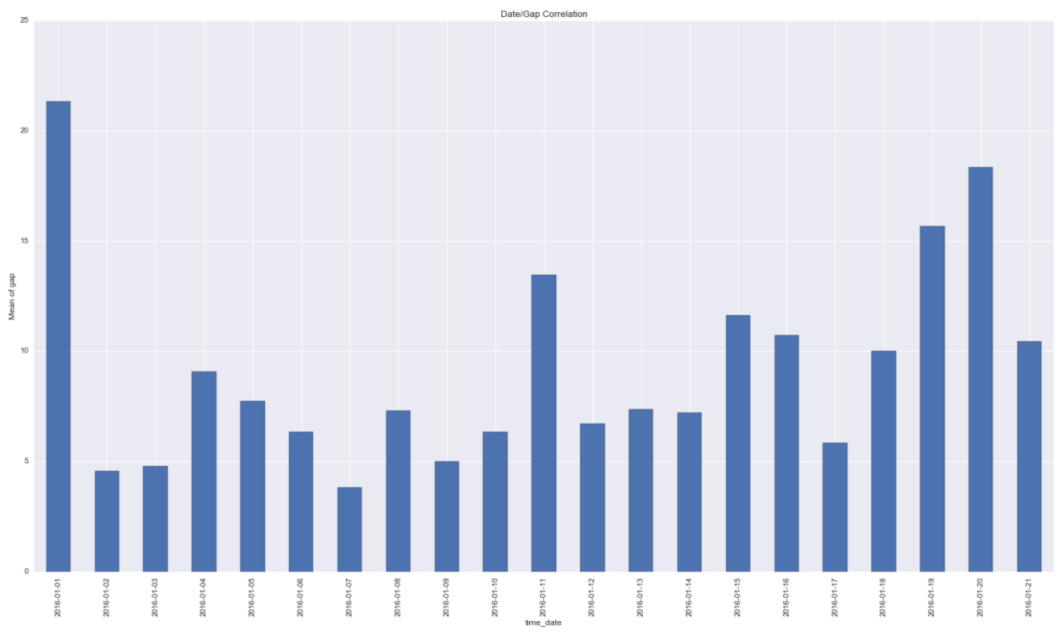
As mentioned in the benchmark section, the best score in leaderboard is about 0.39. Our final model's score is 0.42.

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.

5 Conclusions

5.1 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

5.2 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 design.
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 and, 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 model 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.

2. 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.

5.3 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).
2. Neural network architecture/fine tuning

   Experiment deeper neural network or Wide&Deep model(5)
3. 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.

6 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/
5. Wide&Deep neural network architecture

   https://www.tensorflow.org/versions/r0.10/tutorials/wide_and_deep/index.html

7 Appendix

7.1 Run the scripts

Required library to run the scripts

- Sklearn

- XGBOOST
- Tensorflow

Instructions for running the scripts

1. Download the source codes from here
2. Get data files(data_preprocessed.zip and data_raw.zip) from Dropbox shared link
3. Extract data_preprocessed.zip under root directory of the project. After extraction, all temporary preprocessed dump files will be under data_preprocessed folder
4. Extract data_raw.zip under root directory of the project. After extraction, all raw files will be under data_raw folder
5. In the console/terminal, set implement as current directory
6. Run scripts
   Run python didineuralmodel.py to train/validate neural network model
   Run python knnmodel.py to train/validate KNN model
   Run python randomforestmodel.py to train/validate random forest model
   Run python xgboostmodel.py to train/validation GBM model
   Run python forwardfeatureselection to try out greedy forward feature selection
   Run python tunemodels.py to try out grid search for models

7.2 Feature id/name mapping:

```
# gap features
featureDict[101] = ['gap1']
featureDict[102] = ['gap2']
featureDict[103] = ['gap3']
featureDict[104] = ['gap_diff1']
featureDict[105] = ['gap_diff2']
featureDict[106] = ['gap_mean']
featureDict[107] = ['gap_std']



#district features
featureDict[201] = ['start_district_id']
featureDict[202] = districtids
featureDict[203] = ['start_district_id_51',
'start_district_id_23','start_district_id_8','start_district_id_37']
featureDict[204] = ['district_gap_sum']
featureDict[205] = self.get_district_type_list()
featureDict[206] = ['poi_sum']

#time features
featureDict[301] = ['time_id']
```

```
featureDict[302] = timeids

#weatehr features
featureDict[401] = ['preweather']
featureDict[402] = ["rain_check"]

# Traffic features
featureDict[501] = ['traffic1']
featureDict[502] = ['traffic2']
featureDict[503] = ['traffic3']
featureDict[504] = ['traffic_diff1']
featureDict[505] = ['traffic_diff2']
featureDict[506] = ['traffic_mean']
featureDict[507] = ['traffic_std']


#historical features
featureDict[601] = ['history_mean']
featureDict[602] = ['history_median']
featureDict[603] = ['history_mode']
featureDict[604] = ['history_plus_mean']
featureDict[605] = ['history_plus_median']
featureDict[606] = ['history_plus_mode']

#cross features
featureDict[8801] = ['district_time']
featureDict[8802] = ['weather_time']
```