CS 6350.0U1 – BIG DATA MANAGEMENT AND ANALYTICS

KDD CUP 2017 - Highway Tollgates Traffic Flow Prediction

Travel Time & Traffic Volume Prediction

Project Report
Summer 2017
August 07 2017

Manohar Katam (mxk164930)

Shiva Podugu (sxp170130)

Sravani Lingam (sxl170330)

Venkata Kartheek Madhavarapu (vxm153830)

I. PROBLEM STATEMENT

The problem involves two challenges or tasks they are:

Task1: To **estimate the average travel time** from designated intersections to tollgates. In this task we have to estimate for every 20-minute time window the average travel time of vehicles for a specific route:

a. Routes from Intersection_id A to Tollgate_ids 2 & 3;

b. Routes from Intersection_id B to Tollgate_ids 1 & 3;

c. Routes from Intersection_id C to Tollgate_ids 1 & 3.

Task2: To **predict average tollgate traffic volume** for every 20-minute time interval window where tollgates 1 and 3 has entry and exit traffic and 2 has only entry traffic.

II. DATASET DESCRIPTION

Link to the dataset:
https://tianchi.aliyun.com/competition/information.htm?raceId=231597

The road network topology (as shown in Fig.1) is a directed graph formed by interconnected road links (as shown in Fig.2). A route to the tollgate in the network is represented by a sequence of links as shown in Fig.3. For every link, the vehicle traffic comes from one or more "incoming road links" and goes into one or more "outgoing road links" as shown in Fig.2.

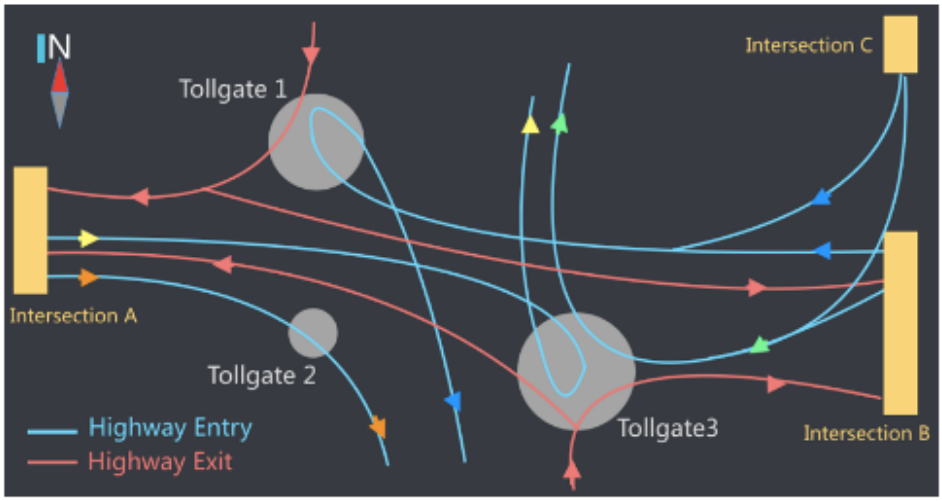


Fig.1 Road Network Topology

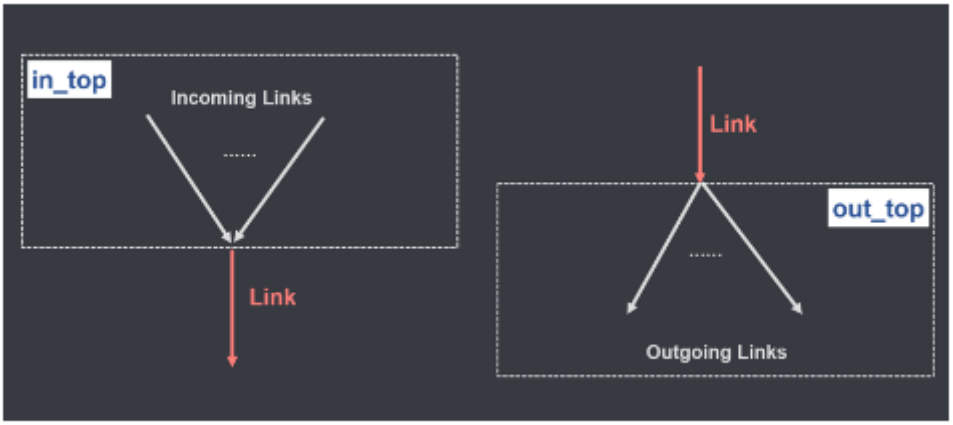


Fig.2 In_top and Out_top for a Road Link (number of incoming and ougoing links)

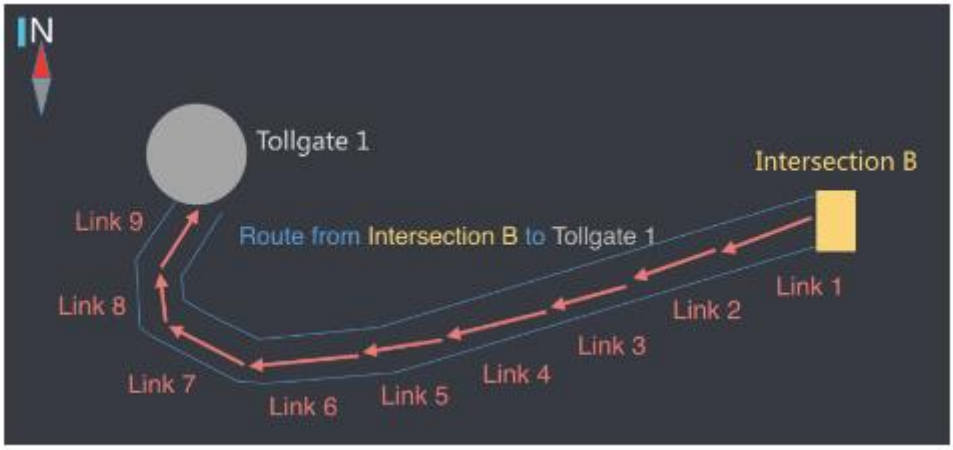


Fig.3 Link Sequence of a sample route

The dataset contains the following **attributes** distributed in five tables:

1. **Road link properties table:**

Link_id	link id
Length	length (meters) of road link
Width	width (meters) of road link
Lanes	number of lanes on the road link
In_top	incoming road links, separated by comma “,”
Out_top	outgoing road links, separated by comma “,”
Lane_width	lane width (meters)

2. **Vehicle Routes from Intersections to Tollgates**

Intersection_id	Intersection ID
Tollgate_id	Tollgate ID
Link_seq	A sequence of link IDs separated from intersection to tollgate separated by commas “,”

3. **Vehicle Trajectories Along Routes**

Intersection_id	Intersection ID
Tollgate_id	Tollgate ID
Vehicle_id	Vehicle ID
Starting_time	Time point when vehicle enters the route
Travel_seq	Trajectory in the form of a sequence of link traces separated by ";" and each trace consists of link id, enter time and travel time in seconds, separated by "#"
Travel_time	The total time (in seconds) the vehicle takes from intersection to the tollgate

4. **Traffic Volume through the Tollgates**

Time	The time when vehicle passes the tollgate
Tollgate_id	Tollgate ID
Direction	"0" for entry and "1" for exit
Vehicle_model	Capacity of the vehicle (range: 0-7) the bigger the higher capacity
Has_etc	Does vehicle has electronic toll collection tag? 0:No, 1:Yes
Vehicle_type	Vehicle type, 0:passenger type, 1:cargo vehicle

5. **Weather Data (every 3 hours) in the Target Area**

Date	Date
Hour	Hour
Pressure	Air pressure (in hPa)
Sea_pressure	Sea level pressure (in hPa)
Wind_direction	Wind direction (in Degree)
Wind_speed	Wind speed (in meters/sec)
Temperature	Temperature (in Degree Celsius)
Rel_humidity	Relative humidity
precipitation	Precipitation (in mm)

Number of Instances in the Dataset:

Task1: For travel time prediction, the training set contains data from July. 19th 2016 to Oct. 17th 2016. The number of instances in the trajectories data are: 109244.

Task2: For volume prediction, the training set contains data from Sep. 19th 2016 to Oct. 17th 2016. The number of instances in the volume data are: 543699.

Regression techniques used

We applied the following techniques on the data to complete the required analysis –

- Random Forest regression
- Linear regression
- Gradient Boosted tree regression

Programming Language we used

We used Python (Pyspark) language for the project with spark cluster on Databricks.

III. PRE-PROCESSING

Our pre-processing of data involved five steps:

1. Outlier pre-processing
2. Data Imputation: Detecting NA values or outliers and replacing them.
3. Removing the uncorrelated attributes: through correlation plots.
4. Adding new features (Feature engineering): Adding extra features as needed.
5. Data Augmentation (adding instances)

We will now present step by step how we have gone through our pre-processing:

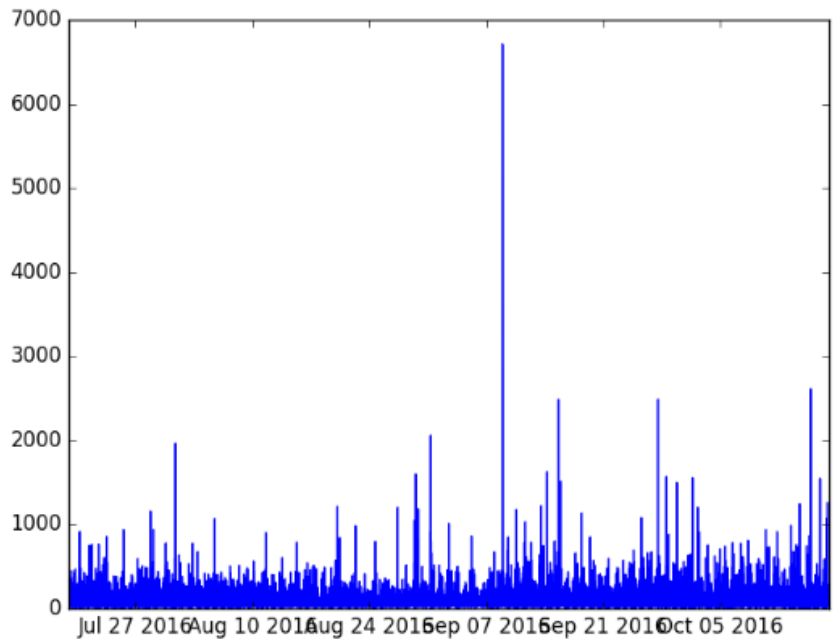
Step1: Restructuring of data according to the number of 20 minute time windows

Task1: We loaded the trajectory data, and the summary of travel_time is as below.

summary	travel_time
count	109244
mean	106.4474861777306
stddev	71.76168638789554
min	9.26
max	6711.11

We see that min travel_time (which is in minutes) is 9.26 where as max is 6711.11, there is scope for some outliers here, so we plotted travel_time and verified it.

Travel time of 6711 minutes is like 111 hours (4 and half days which is too long). So, we take mean which is 106 into account and fix some threshold and replace those outliers with best favorable value. **So, we replaced avg_travel_times which are more than 600 with average of its previous and next values.**



We grouped the data based on 20 minute time interval, intersection_id and tollagate_id. After grouping the data looks as below

```
1 trajectoryData_train.head()
```

Out[15]:

	starting_time	intersection_id	tollgate_id	averagetravltime
0	2016-07-19 00:00:00	B	3	70.85
1	2016-07-19 00:20:00	A	2	58.05
2	2016-07-19 00:20:00	B	1	79.76
3	2016-07-19 00:20:00	B	3	148.79
4	2016-07-19 00:40:00	B	1	137.98

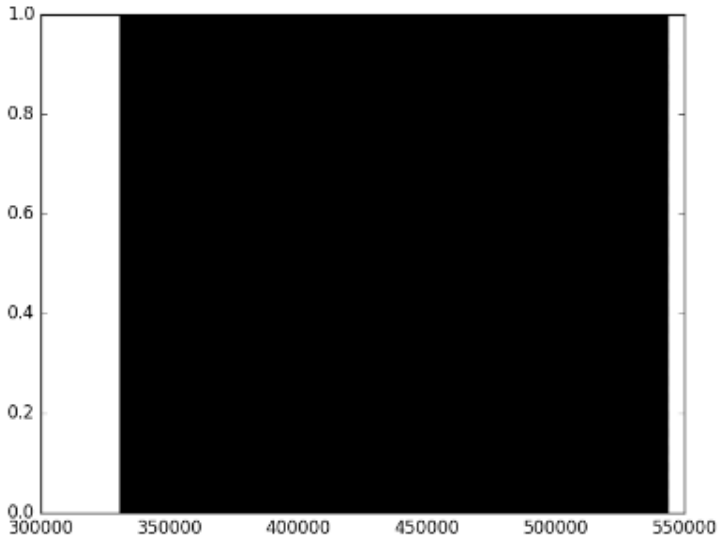
After analyzing the submission_sample_travelTime data (Provided by KDD2017), we found that the data for certain time intervals are missing in the actual data. So, We collected all the time intervals from submission sample data and stored it into an array and compared them with time interval of each intersection_id and tollagate_id in the actual data and appended the missing time intervals to the actual data with the average travel time of previous and next time intervals for that particular intersection and tollagates.

Task2:

Next, we explored the volume data, here is the summary of volume data:

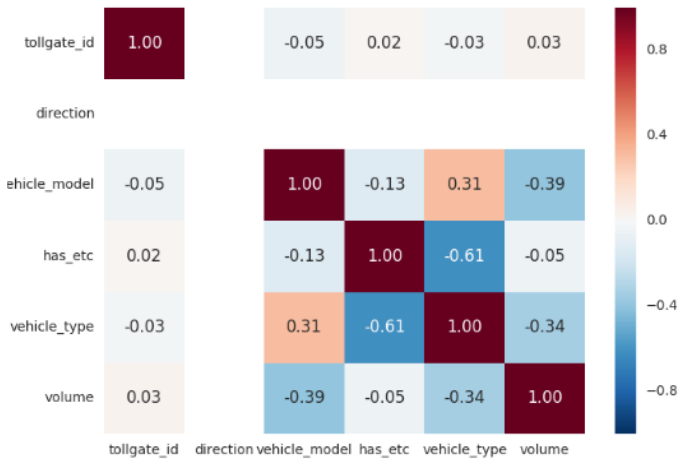
summary	tollgate_id	direction	vehicle_model	has_etc	vehicle_type
count	543699	543699	543699	543699	212710
mean	2.0861377342978376	0.39122749903899035	1.089689331781004	0.22142030792773207	0.22596963001269335
stddev	0.9256115347523565	0.48802559470553697	0.8383135050591561	0.41520316983435096	0.41822025131421003
min	1	0	0	0	0
max	3	1	7	1	1

We observed that vehicle type (0:passenger, 1:cargo) is null in half of the dataset, we verified this through a plot.



Since, vehicle_type is missing in half of the data, we consider this feature as not so important and we dropped this feature.

We grouped the volume data with 20 Minute time interval windows with pandas TimeGrouper. Then we plotted the effect of other attributes n class volume with correlation plot using the package “seaborn” and it’s method heatmap.



We see from above correlation plot that, nothing is really correlated with class volume. So, we removed the features vehicle_model, has_etc and vehicle_type features.

Data Augmentation: Next, important thing we did is data augmentation, that is we added new instances to the data as we felt, there is too less data to do our predictive analytics.

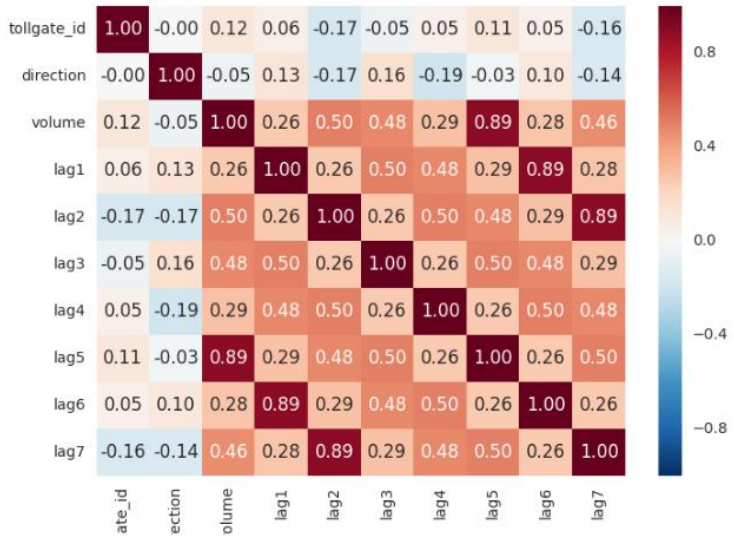
Data Augmentation involved adding the data with moving time windows from +5 and -5 and +3 and -3 i.e. for example consider below,

Time window	Window + 5	Window -5	Window + 3	Window -3
00:00 – 00:20	00:05 – 00:25	23:55 – 00:15	00:03 – 00:23	23:57 – 00:17

Adding Lag Features:

In time series regression problems, lags are very important because there is a dependency from one lag to other lag. In our problem, we have put 7 lags for volume because, the traffic flow once occurs, it will on steady rise and steady drop, to catch up with the trend we used lag features. After adding lag features to volumeData, volumeData now has 7 lag features is each is shifted by one row.

We plotted the correlation plot of lag features to see how lags are correlated with volume attribute. We see that there is a good amount correlation between them.

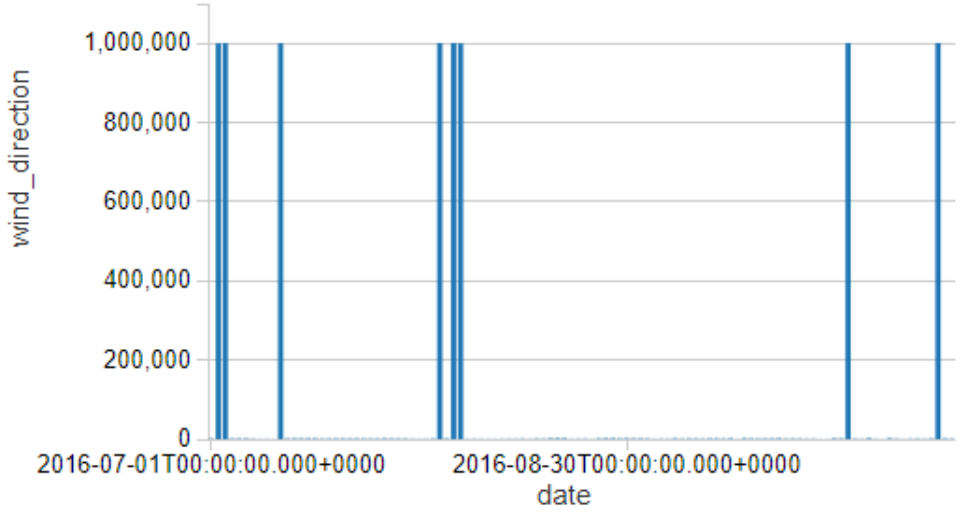


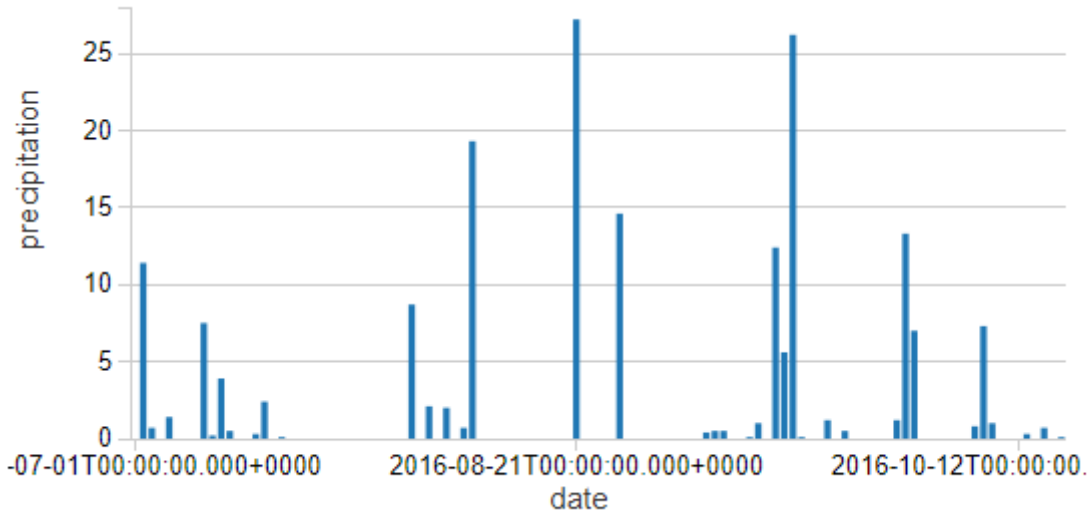
Step2: Adding the Weather Data to trajectory and volume data

Firstly, we loaded the weather data given in the dataset. Here is the summary of weather data:

summary	hour	pressure	sea_pressure	wind_direction	wind_speed	temperature	rel_humidity	precipitation
count	862	862	862	862	862	862	862	862
mean	10.5	1003.989675174014	1008.8669373549883	9449.430394431554	2.3025522041763313	27.400348027842274	73.87470997679814	0.21252900232018562
stddev	6.86720733525396	5.449205645852431	5.531462928628924	95832.73071923238	1.2157476132718992	4.869076841291541	15.900988240842684	1.3855084924132708
min	0	993.4	998.2	0.0	0.0	14.1	25.0	0.0
max	21	1018.4	1023.5	999017.0	7.5	39.4	98.0	27.2

From the weather table data summary we observed that wind_direction has outliers with value = 999017 and most of the precipitation values are zero with max value being 27.2, we verified this visually with graphs now.

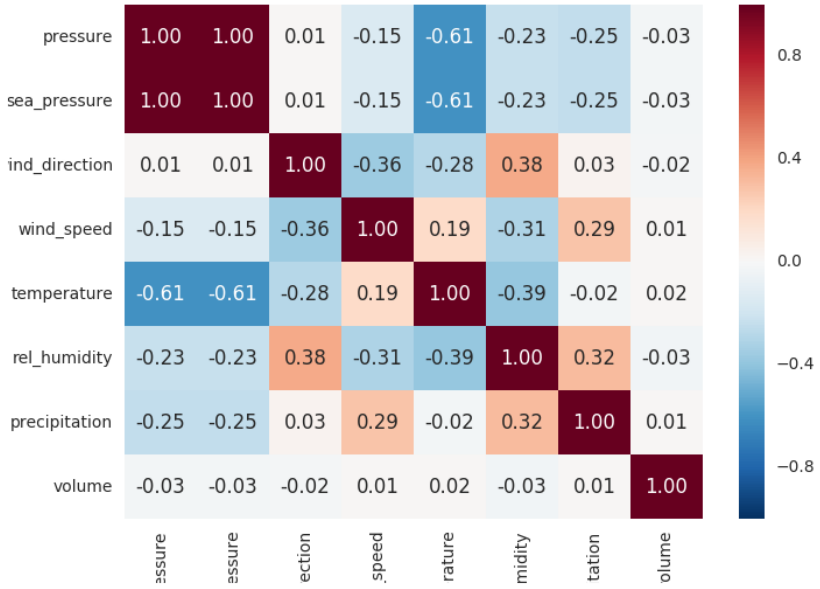




- **We replaced the outlier in wind_direction (999017 value) with the average of previous value and next value if next_vlaue is not equal to 999017 and if not we replace with previous value**.

- Although the precipitation attribute has lot of zeros, we did not remove it because precipitation (rain) is one of key features in deciding the traffic flow.

Next, we plotted the correlation of weather data with volume (Task2) and avg_travel_time (Task1).



We see from the above correlation plot that, no feature of climate is highly correlated with volume. We also see that pressure and sea_pressure are highly correlated with each other, so when we train the model we will remove sea_pressure attribute. Also, we will not remove precipitation and temperature attributes because they are somehow related with traffic flow in daily life.

We appended the weather features to volume and trajectory data based on starting_time.

Step3: Adding Date Time Features (like Holidays, weekdays, hour of the day)

Since this data is of china, we looked at holidays of china in the given date range of our data, the holidays are from September 15th to September 19th are Mid Autumn festival holidays and October 1st to October 7th are official National Holidays in china, so we have taken this holiday ranges into account.

(reference: http://www.officeholidays.com/countries/china/2016.php)

Based on the range of holidays we added the feature holiday (0: not a holiday, 1: holiday). We also added the feature what week day it is (Monday, Wednesday… Sunday). We also added the feature of what hour it is based on 24 hour clock (hour__0, hour__1, …., hour__23). We have also added which interval it is in minutes (0th interval, 20th interval, 40th interval).

At the end of this step, we have added this extra features to the trajectory and volume data.

Friday, Monday, Saturday, Sunday, Thursday, Tuesday, Wednesday, hour__0, hour__1, hour__2, hour__3, hour__4, hour__5, hour__6, hour__7, hour__8, hour__9, hour__10, hour__11, hour__12,

hour__13, hour__14, hour__15, hour__16, hour__17, hour__18, hour__19, hour__20, hour__21, hour__22, hour__23, `0`, `20`, `40`, holiday,

Step4: Extracting features from Links and Routes data tables for trajectory Data

In this step, we are analysing the link data and determining if the link is cross_in or cross_out and there by adding these columns to the existing link data.

Then we are aggregating the link data and route data based on the link id's present in the link sequence followed by the distance based on the intersection and link.

Adding various other data like number of links, count of links with respect to lanes and distance of each lane with respect to intersection and toll gate data.

At the end of this step, we have added the below features:

in_link_cross_conut	out_link_cross_count	length	link_count	1_length	2_length
3_length	4_length	1_count	2_count	3_count	4_count

IV. MODEL TRAINING AND VALIDATION

Performance Metrics Used: As specified in the competition we used one of the performance metrics as MAPE (Mean Absolute Percentage Error).

For Task1 MAPE:

$$MAPE = \frac{1}{R}\sum_{r=1}^{R}\left(\frac{1}{T}\sum_{t=1}^{T}\left|\frac{d_{rt} - p_{rt}}{d_{rt}}\right|\right)$$

where d_{rt} and p_{rt} are the actual and predicted average travel time for route r during time window t and R is the number of routes and T is number of predict time windows in the particular period.

For Task2 MAPE:

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

where C is the number of tollgate-direction pairs (tollgate 1-entry & exit, tollgate 2-entry, tollgate 3-entry & exit), T is the number of time windows in the particular period, and fct and pct are the actual and predicted traffic volume for a specific tollgate-direction pair c during time window t.

A part from the above metric MAPE, we also used MAE (Mean Absolute Error) and R^2 (Coefficient of Determination) for analysing the performance of regression techniques we used.

Mean Absolute Error (MAE)	$MAE = \sum_{i=0}^{N-1} \lvert \mathbf{y}_i - \hat{\mathbf{y}}_i \rvert$
Coefficient of Determination (R^2)	$R^2 = 1 - \frac{MSE}{\mathrm{VAR}(\mathbf{y}) \cdot (N-1)} = 1 - \frac{\sum_{i=0}^{N-1} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{\sum_{i=0}^{N-1} (\mathbf{y}_i - \bar{\mathbf{y}})^2}$

Pipeline Stages:

We have set the pipeline stages which includes a **StringIndexer with OneHotEncoder** which converts the catrgorical variable into binary sparse vectors. Next stage of pipeline includes a **VectorAssembler** which transforms all the features into feature vector.

After setting stages for pipeline, then we called fit() and transform() methods in which fit() computes feature statistics as needed and transform() actually transforms the features.

Once the Pipeline stage is over we have to model selection and hyper parameter tuning with ParamGridBuilder() and TrainValidationSplit() using different regression techniques as given below.

Model Selection and Hyper Parameter Tuning:

We used a ParamGridBuilder to construct a grid of parameters to search over and TrainValidationSplit (with trainRatio = 0.8) to try all combinations of values and determine best model using the evaluator (RegressionEvaluator).

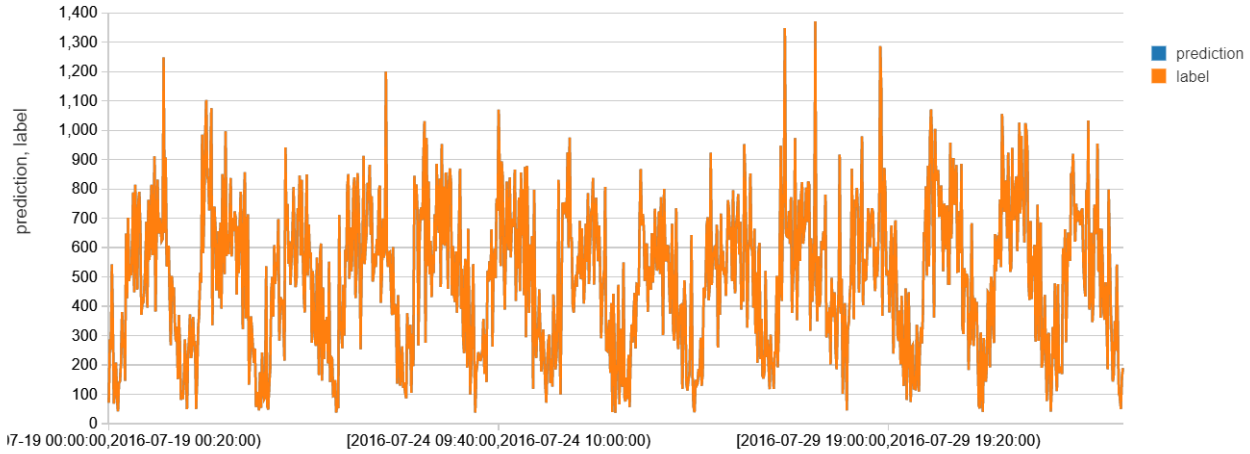
LINEAR REGRESSION:

For linear regression, we have set the grid parameters in this way:

Task1:

maxIter, [10, 20]

regParam, [0.9, 0.7, 0.5, 0.3, 0.1]
fitIntercept, [False, True]
elasticNetParam, [0.0, 0.5, 1.0]
Linear Regression for volume data (Task1) obtained a MAPE of **0.21014478581996687**
Other metrics:
MAE: 21.983,
RMSE: 31.877,
r2: 0.693.
The plot of actual v/s predicted is as follows for trajectory (task1) data.



Task 2:

maxIter, [10, 20]

regParam, [0.9, 0.7, 0.5, 0.3, 0.1]

fitIntercept, [False, True]

elasticNetParam, [0.0, 0.3, 0.5, 0.8]

Linear Regression for volume data (Task2) obtained a MAPE of **0.56683179152144236.**

Other metrics:

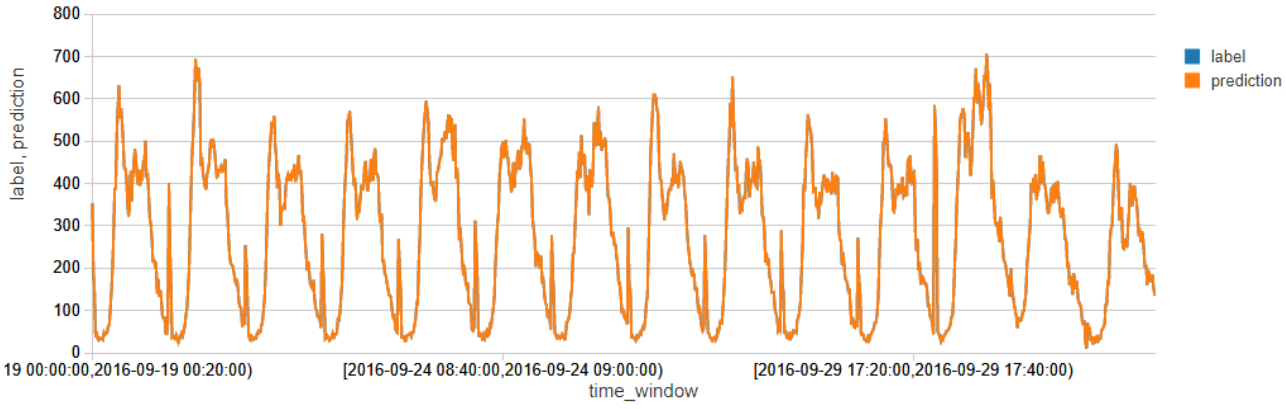
MAE: 9.868465,

RMSE: 15.233811,

r2: 0.892723.

R^2 (Coeeficient of determination) determines how close are predicted and actual values. The more close it is it will be 1 in ideal case. We have obtained a R^2 of 0.892 which is good.

The plot of actual v/s predicted is as follows for volume (task2) data.



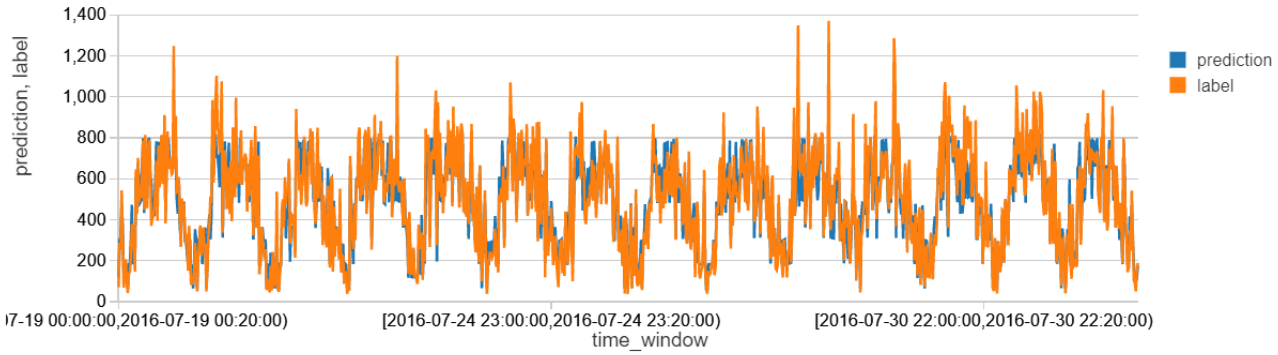
RANDOM FOREST REGRESSION:

Task1:

For Random Forest regression, we have set the grid parameters in this way:

maxDepth, [4, 8, 12, 20]
maxBins, [10, 20, 60]
numTrees, [100, 500, 1000]

MAPE: 0.2510082230774832
MAE: 26.2604
RMSE: 38.9364
r2: 0.543259
The plot of actual v/s predicted is as follows for trajectory (task1) data.



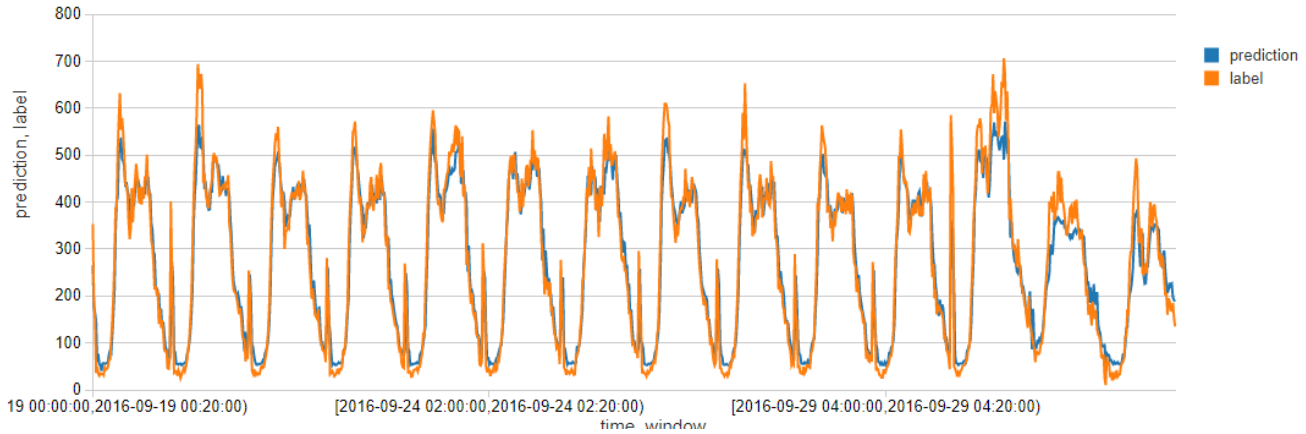
Task2:

For Random Forest regression, we have set the grid parameters in this way:

maxDepth, [4, 8, 12, 20]
maxBins, [10, 20, 60]
numTrees, [100, 500, 1000]

MAPE: 0.78647746101516436
MAE: 11.6561
RMSE: 17.6491
r2: 0.856009

The plot of actual v/s predicted is as follows for volume (task2) data.

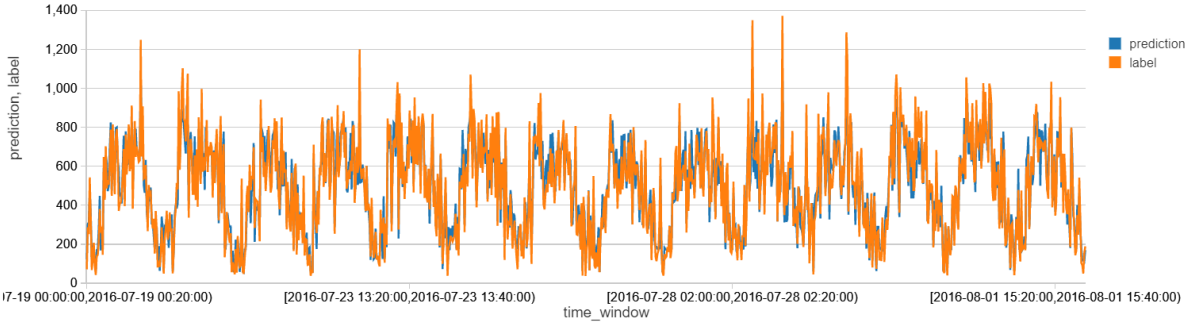


GRADIENT BOOSTED TREES REGRESSION:

For Gradient Boosted Trees regression, we have set the grid parameters in this way:

Task1:

maxDepth, [2,4,6,10, 20]
maxIter, [10, 20, 40]
MAPE: 0.23292589822325993
MAE: 24.7269
RMSE: 35.545
r2: 0.619359
The plot of actual v/s predicted is as follows for trajectory (task1) data.

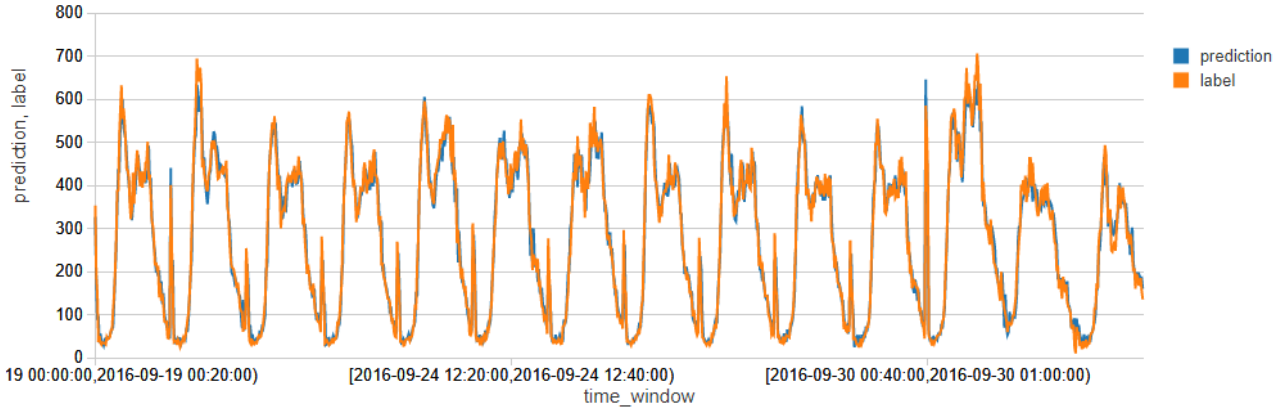


Task2:

maxDepth, [2,4,6,10, 20]
maxIter, [10, 20, 40]

MAPE: 0.55801966599733666
MAE: 9.85717
RMSE: 13.9279
r2: 0.910327

The plot of actual v/s predicted is as follows for volume (task2) data.



V. CONCLUSION

Task1 Results:

Regression Technique	**MAPE**	**MAE**	**RMSE**	**R^2**
Linear Regression	**0.21014478574298456**	**21.983251**	**31.877944**	**0.693847**
Random Forest Regression	**0.25100822307748322**	**26.2604**	**38.9364**	**0.543259**
Gradient Boosted Trees Regression	**0.23292589822325993**	**24.7269**	**35.545**	**0.619359**

Travel Time Prediction | Volume Prediction

Rank	Participant	Organization	MAPE
1	Convolution	Microsoft	0.1748
2	好想有个队友	Zhejiang University	0.1771

The winner of the competition has got an MAPE value of 0.1748 and we got MAPE value of 0.21, with this we say that our model is working well.

Task2 Results:

Regression Technique	MAPE	MAE	RMSE	R^2
Linear Regression	0.56683179152144236	9.868465	15.233811	0.892723
Random Forest Regression	0.78647746101516436	11.6561	17.6491	0.856009
Gradient Boosted Trees Regression	**0.55801966599733666**	**9.85717**	**13.9279**	**0.910327**

To our surprise we see that Linear Regression worked better than Random Forest regression, this may be due to the reason that the features have a smooth, nearly linear dependence on the covariates. In this type of situations linear regression will model the dependence better than random forests. And the other reason might be Random Forests's have a little trouble modelling linear combinations of a large number of features.

Linear regression and Gradient Boosted Tress regression gave almost equal results with minute change. As said above when data nearly linear dependence linear regression performs equally well with gradient boosted trees.

DATABRICKS PUBLIC LINKS:

Task1 Preprocessing:

https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/855389465014467/2984073192372450/4847935780219074/latest.html

Task1 Analysis with LR, RF and GBT using SparkMLlib:

https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/855389465014467/3665561683253945/4847935780219074/latest.html

Task2 Preprocessing:

https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/5008245478875404/727104593191945/440415051426816/latest.html

Task2 Analysis with LR, RF and GBT using SparkMLlib:

https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/5008245478875404/4245257127570951/440415051426816/latest.html

END OF PROJECT REPORT