CSYE 7245-Big Data Systems and Intelligence Analysis

Prediction on Traffic Situation based on Synthetic Analysis

Ziwei Fan 001855517

fan.ziw@husky.neu.edu

Abstract

The time spent on road primarily depends on multiple perspectives: Regions, weather, accidents etc. For taxi or Uber driver, **how to avoid traffic congestion and achieve benefit maximization?** In details:

(1) How could the drivers keep track of the traffic situation by visualization?
(2) How could drivers acquire the distribution information about requests number?

For these questions, the focus of paper is to:

(1) Visualize the traffic and requests number by geospatial information.
(2) Explore how weather data impacts the traffic with data & explore if it could be validated by sentiment analysis.
(3) Predict the total ride duration of taxi trips in New York based on TPOT.

Weather data, geospatial data, and traffic data, have been utilized to be visualized according to the different requirements. For prediction in this case, the best pipeline generated by TPOT is DecisionTreeRegressor. Additionally, the correlation between traffic and weather is different from the common sense. It is just a weak negative correlation -0.385 by sentiment analysis and similar with the result of Pearson Correlation.

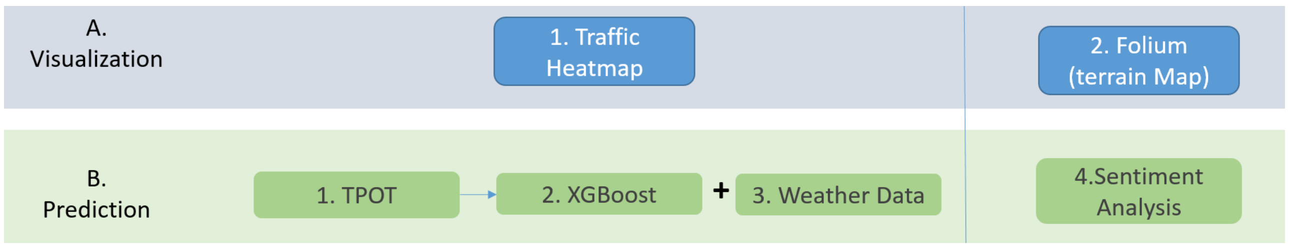
Introduction

Longer trip duration and traffic congestion are terrible issues for traffic system and taxi drivers. For leveraging the traffic resources and avoid traffic jam, the synthetic traffic monitoring and prediction are necessary.

For this goal, diverse sorts of data in the system should be considered:

(1) In terms of weather, it is shown that tremendous range of weather/climate condition may (not must) affect the traffic, according to the 'Road Weather Management program'. [1]
(2) For drivers, the heatmap of hour changing will directly reflect the hot areas of traffic. Additionally, the prediction on duration trip will be more accurate to get to know what the hot area will be and show the hot areas predicted for the drivers.
(3) Besides, they could keep track which regions have the most number of requests and they could receive the traffic information from operation center, so that they could balance and make a decision to achieve benefit maximization.

In the implementation, the basic architecture is as below:



Data source

1. **New York City Taxi Trip Duration:**
   The data was originally published by the NYC Taxi and Limousine Commission (TLC)[6].It contains pickup time, geo-coordinates, number of passengers, and several other variables.
2. **Taxi with OSRM:**
   OSRM(Open source routing machine)is a Modern C++ routing engine for shortest paths in road networks. Get in Touch. Features. Flexible import of OpenStreetMap data.[7] This dataset is used to extract information about the fastest routes for each data point.
   It contains data about Starting & End Street, total distance, number of steps, distance of per step etc.
3. **New York Weather Data:**
   Weather data collected from the National Weather Service. It contains the first six months of 2016, for a weather station in central park. It contains for each day the minimum temperature, maximum temperature, average temperature, precipitation, new snow fall, and current snow depth.

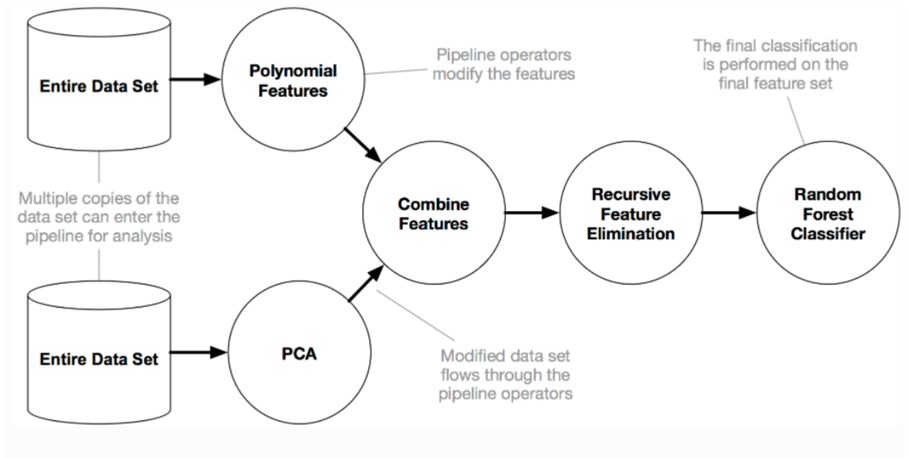
Method

Prior to the training, some work for pro-pressing has been completed. Firstly, the raw data could be collected from different departments and be pre-processed, such as weather data from National Weather Service and 2 data sheet from traffic department. Moreover, the text information will be gathered from media, and filter the keyword and analysis based on streaming. The approaches will be explained in more details:

(1) TPOT:

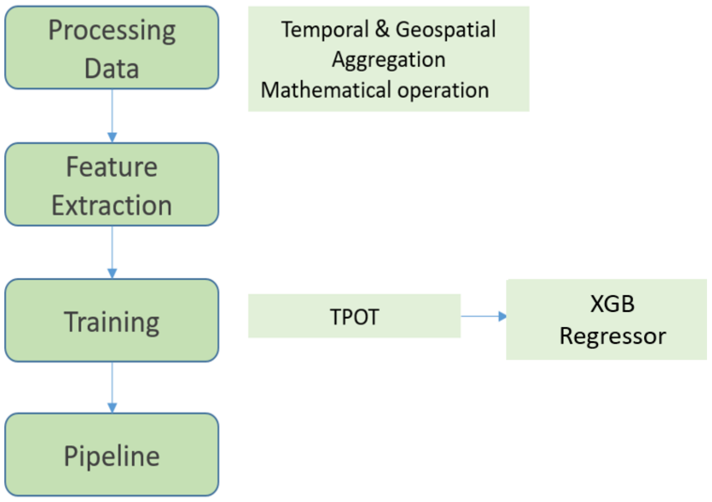
TPOT package is used to handle the cross validation and hyper-parameters and predict the trip duration.[3][4]

It is a Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming. It will automate the most tedious part of machine learning by intelligently exploring thousands of possible pipelines to find the best one for data. Once TPOT is finished searching (or you get tired of waiting), it provides you with the Python code for the best pipeline it found so you can tinker with the pipeline from there.


Entire Data Set
Polynomial Features
Pipeline operators modify the features
The final classification is performed on the final feature set
Multiple copies of the data set can enter the pipeline for analysis
Combine Features
Recursive Feature Elimination
Random Forest Classifier
Entire Data Set
PCA
Modified data set flows through the pipeline operators

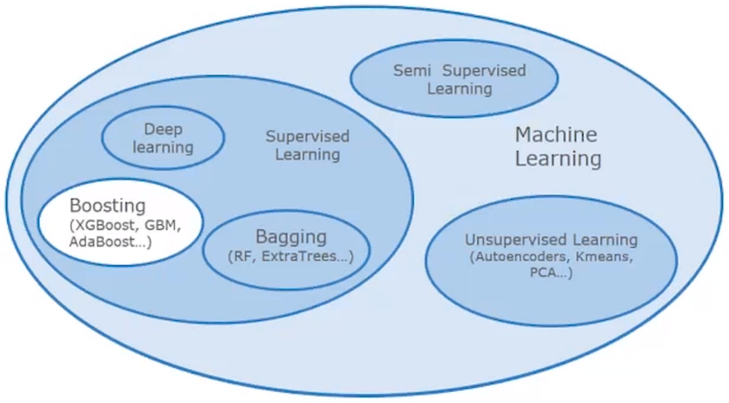
An example TPOT pipeline

In this case, I will apply TPOT by this progress:

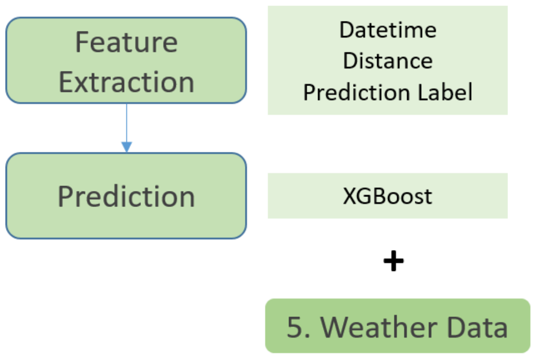


(2) XBGoost:

An XGBoost (Extreme Gradient Boosting) is an ensemble model that aggregates several trees to provide a more generalizable Machine Learning model. In this program, it might be generated as one of the results of TPOT. However, in this case, another model-GradientBoostingRegressor has been generated after TPOT processing. Thus, XGBoost could be applied for further validation or mixed with weather data for exploring correlation.


Semi Supervised Learning
Deep learning
Supervised Learning
Machine Learning
Boosting (XGBoost, GBM, AdaBoost...)
Bagging (RF, ExtraTrees...)
Unsupervised Learning (Autoencoders, Kmeans, PCA...)

In order to further predict with weather data, and validate the result generated by TPOT.

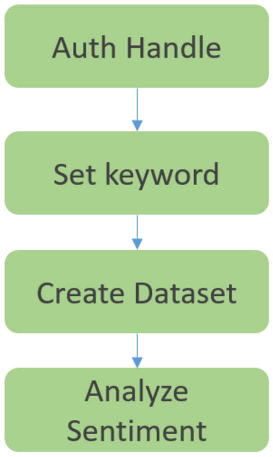


(3) Sentiment analysis:

Sentiment Analysis is the process of ‘computationally’ determining whether a piece of writing is positive, negative or neutral. It’s also known as opinion mining, deriving the opinion or attitude of a speaker. [2] With the different keyword, it is likely that we could explore the attitude correlation between traffic and weather.

As one of the approaches to validate the correlation explored previously, sentiment analysis has been applied based on twitters. In this validation, the attitudes towards weather and traffic have been checked routinely and compared with the correlation results calculated previously: If the results are similar.

It will focus on the twitter dataframe and explore if there is any correlation between weather and traffic. This is an approach to validate the Pearson correlation calculated by algorithm.

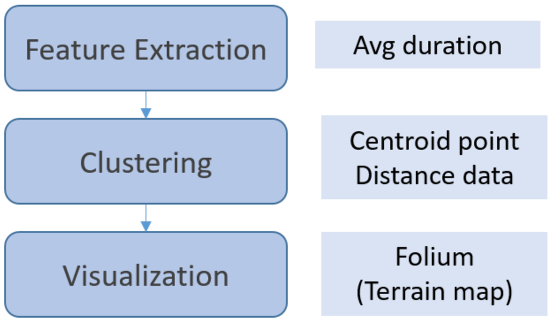


(4) Visualization by heatmap & Folium:

For real terrain visualization, Folium is map-based interactive package and it is open source as well. It's based on leaflet and enables both the binding of data to a map for choropleth visualizations, as well as passing Vincent/Vega visualizations as markers on the map.[5]

In the driving, the visualized and colorful map with request information could navigate the driver to the region with more request.

I will apply it into scenario of the request number: Folium is applied for generating terrain map in order to reflect the number of request more directly.

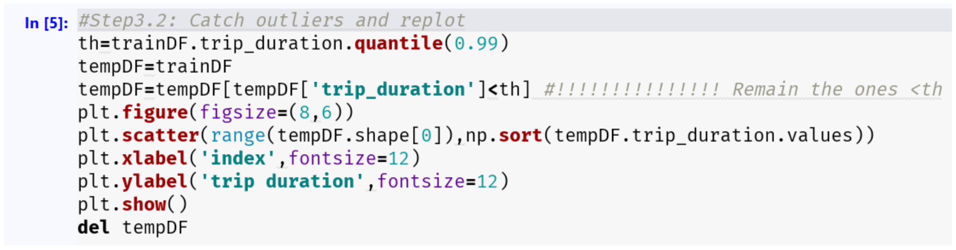


Result and validation

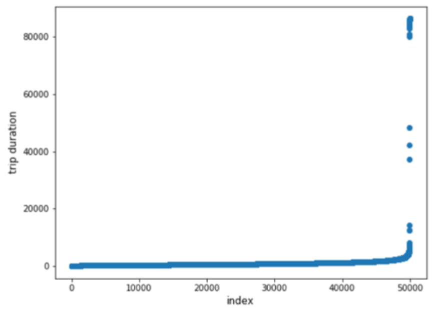
Based on the methods mentioned briefly, the visualization could be presented directly according to the data frame after feature extraction and model training. Additionally, a model has been generated by TPOT. The prediction on trip duration has been conducted with RMSE. In details:

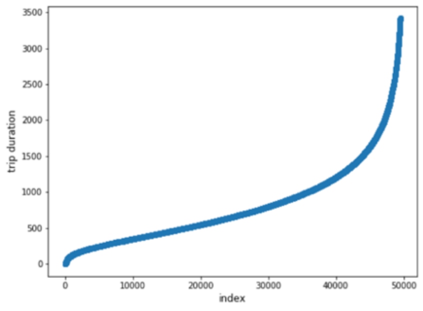
1. **Pro-processing and check the data:**

   The new plot has been smoothened after removing the null data, outliers. In this case, we just remain the ones within the range of threshold (0.99).



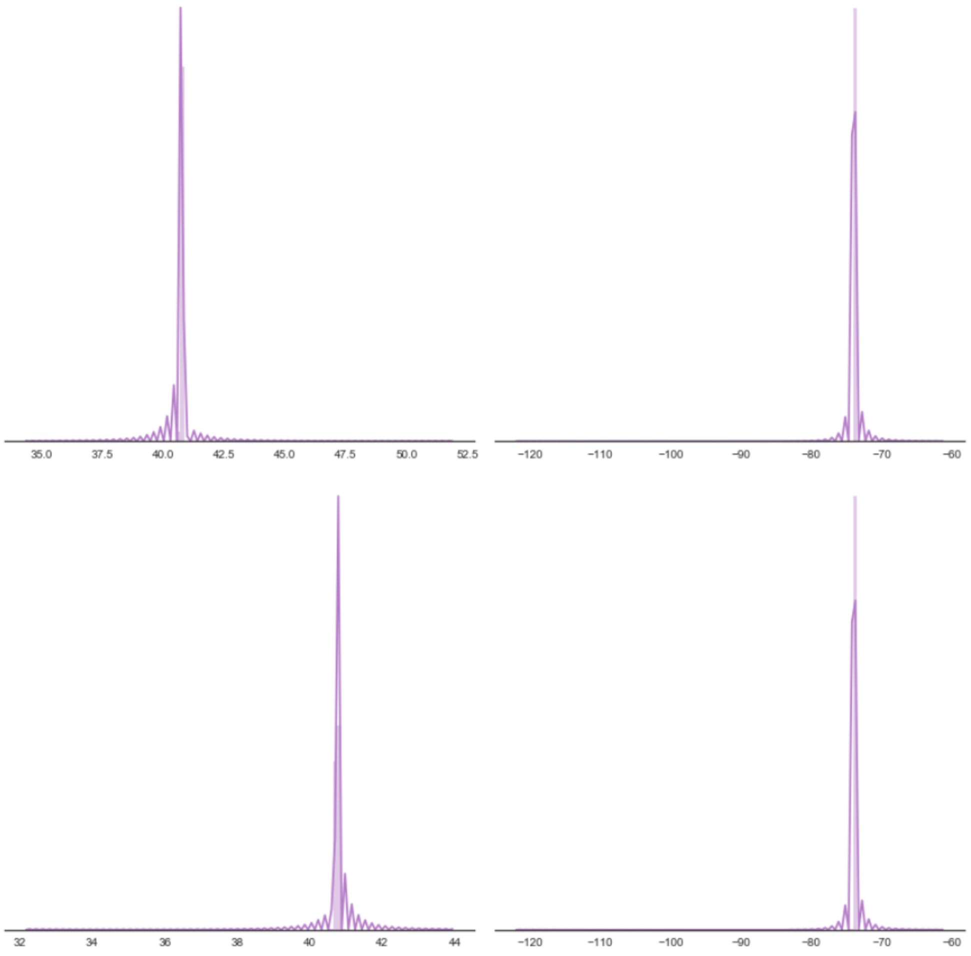
```
In [5]: #Step3.2: Catch outliers and replot
        th=trainDF.trip_duration.quantile(0.99)
        tempDF=trainDF
        tempDF=tempDF[tempDF['trip_duration']<th] #!!!!!!!!!!!!!!!! Remain the ones <th
        plt.figure(figsize=(8,6))
        plt.scatter(range(tempDF.shape[0]),np.sort(tempDF.trip_duration.values))
        plt.xlabel('index',fontsize=12)
        plt.ylabel('trip duration',fontsize=12)
        plt.show()
        del tempDF
```





After remove outlier

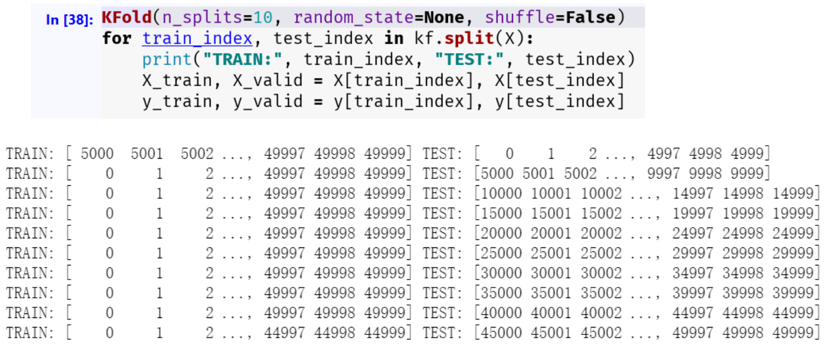
Then, let us check the lat-long distributions are then used them to have a heat map kind of view of given lat-longs in next steps. It is obvious that we get the findings that most duration centered around these ranges: Latitude should be between 40.6 to 40.9 and Longitude should be between -74.05 to -73.70. Then we will focus on these range more closely.



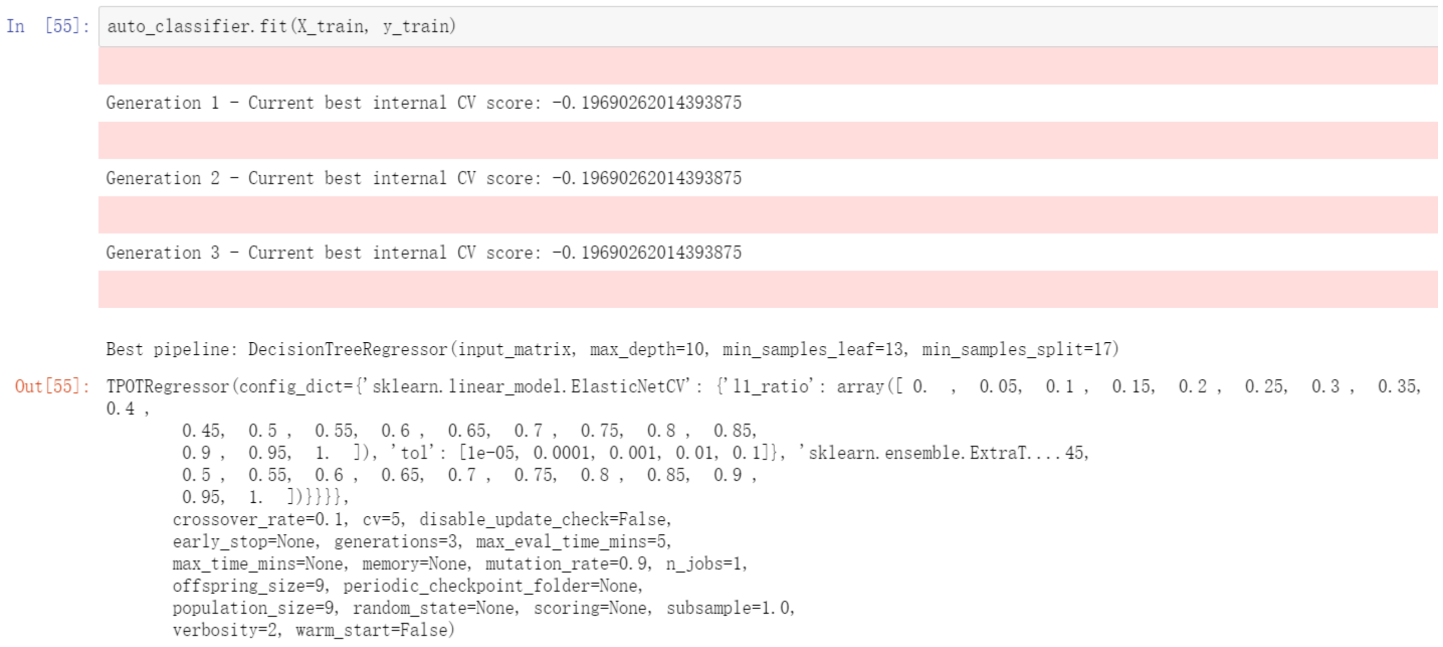
2. Pipeline:

After importing TPOTRegressor and KFold, the data has been trained by TPOTRegressor with n_splits=10.

The best result of TPOT is Decision Tree Regressor with proper parameters.



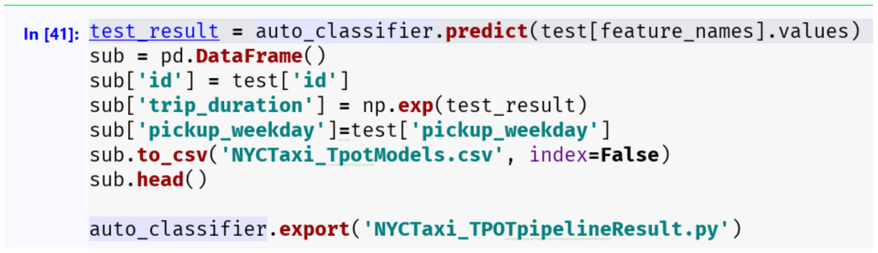
```
In [38]: KFold(n_splits=10, random_state=None, shuffle=False)
         for train_index, test_index in kf.split(X):
             print("TRAIN:", train_index, "TEST:", test_index)
             X_train, X_valid = X[train_index], X[test_index]
             y_train, y_valid = y[train_index], y[test_index]

TRAIN: [ 5000  5001  5002 ..., 49997 49998 49999] TEST: [    0     1     2 ...,  4997  4998  4999]
TRAIN: [    0     1     2 ..., 49997 49998 49999] TEST: [5000 5001 5002 ..., 9997 9998 9999]
TRAIN: [    0     1     2 ..., 49997 49998 49999] TEST: [10000 10001 10002 ..., 14997 14998 14999]
TRAIN: [    0     1     2 ..., 49997 49998 49999] TEST: [15000 15001 15002 ..., 19997 19998 19999]
TRAIN: [    0     1     2 ..., 49997 49998 49999] TEST: [20000 20001 20002 ..., 24997 24998 24999]
TRAIN: [    0     1     2 ..., 49997 49998 49999] TEST: [25000 25001 25002 ..., 29997 29998 29999]
TRAIN: [    0     1     2 ..., 49997 49998 49999] TEST: [30000 30001 30002 ..., 34997 34998 34999]
TRAIN: [    0     1     2 ..., 49997 49998 49999] TEST: [35000 35001 35002 ..., 39997 39998 39999]
TRAIN: [    0     1     2 ..., 49997 49998 49999] TEST: [40000 40001 40002 ..., 44997 44998 44999]
TRAIN: [    0     1     2 ..., 44997 44998 44999] TEST: [45000 45001 45002 ..., 49997 49998 49999]
```



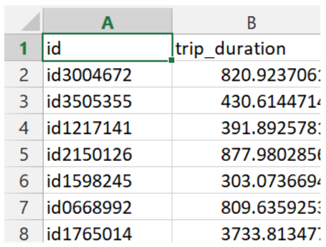
```
In  [55]: auto_classifier.fit(X_train, y_train)

Generation 1 - Current best internal CV score: -0.19690262014393875

Generation 2 - Current best internal CV score: -0.19690262014393875

Generation 3 - Current best internal CV score: -0.19690262014393875

Best pipeline: DecisionTreeRegressor(input_matrix, max_depth=10, min_samples_leaf=13, min_samples_split=17)

Out[55]: TPOTRegressor(config_dict={'sklearn.linear_model.ElasticNetCV': {'l1_ratio': array([ 0.  ,  0.05,  0.1 ,  0.15,  0.2 ,  0.25,  0.3 ,  0.35,
         0.4 ,
                0.45,  0.5 ,  0.55,  0.6 ,  0.65,  0.7 ,  0.75,  0.8 ,  0.85,
                0.9 ,  0.95,  1.  ]), 'tol': [1e-05, 0.0001, 0.001, 0.01, 0.1]}, 'sklearn.ensemble.ExtraT....45,
                0.5 ,  0.55,  0.6 ,  0.65,  0.7 ,  0.75,  0.8 ,  0.85,  0.9 ,
                0.95,  1.  ])}}}},
               crossover_rate=0.1, cv=5, disable_update_check=False,
               early_stop=None, generations=3, max_eval_time_mins=5,
               max_time_mins=None, memory=None, mutation_rate=0.9, n_jobs=1,
               offspring_size=9, periodic_checkpoint_folder=None,
               population_size=9, random_state=None, scoring=None, subsample=1.0,
               verbosity=2, warm_start=False)
```

3. Predict by the 'best' pipeline:

After generating the best pipeline, now we could predict the trip duration with it:



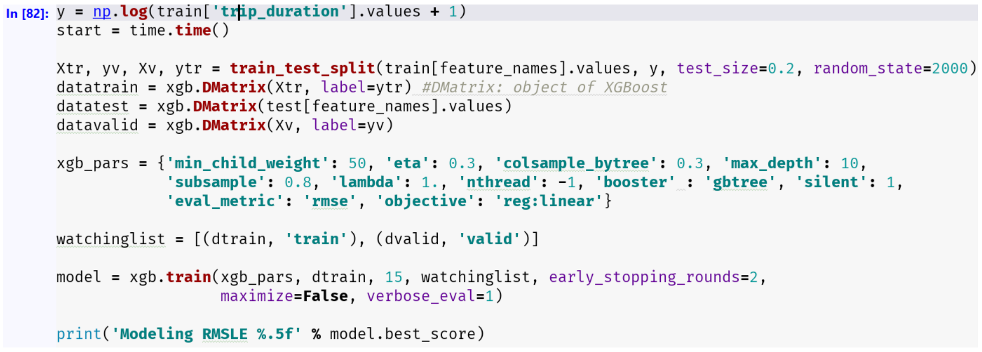
```
In [41]: test_result = auto_classifier.predict(test[feature_names].values)
         sub = pd.DataFrame()
         sub['id'] = test['id']
         sub['trip_duration'] = np.exp(test_result)
         sub['pickup_weekday']=test['pickup_weekday']
         sub.to_csv('NYCTaxi_TpotModels.csv', index=False)
         sub.head()

         auto_classifier.export('NYCTaxi_TPOTpipelineResult.py')
```



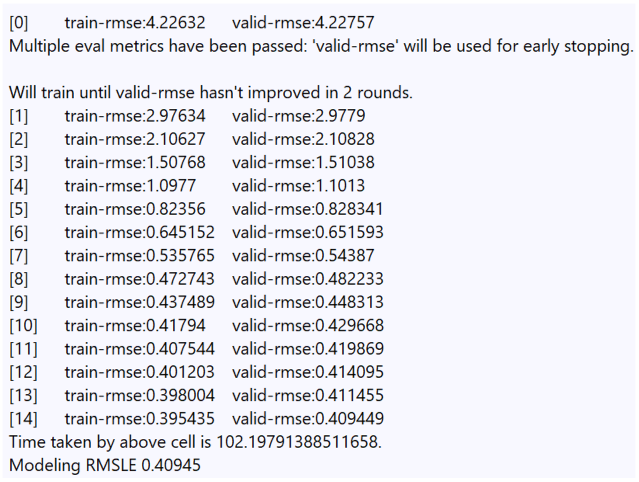
	A	B
1	id	trip_duration
2	id3004672	820.923706
3	id3505355	430.614471
4	id1217141	391.892578
5	id2150126	877.980285
6	id1598245	303.073669
7	id0668992	809.635925
8	id1765014	3733.81347

4. Train and validate by XGBoosting:

It will train until valid RMSE has not improved in 2 rounds. Finally, the modeling RMSLE is 0.40945. (Root Mean Squared Error (RMSE) and Root Mean Squared Logarithmic Error (RMSLE) both are the techniques to find out the difference between the values predicted by machine learning model and the actual values.)



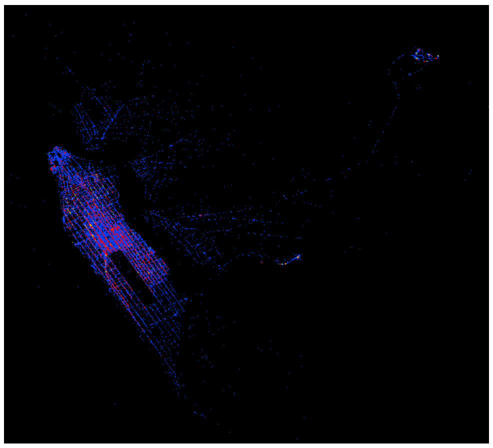
```
In [82]: y = np.log(train['trip_duration'].values + 1)
         start = time.time()

         Xtr, yv, Xv, ytr = train_test_split(train[feature_names].values, y, test_size=0.2, random_state=2000)
         datatrain = xgb.DMatrix(Xtr, label=ytr) #DMatrix: object of XGBoost
         datatest = xgb.DMatrix(test[feature_names].values)
         datavalid = xgb.DMatrix(Xv, label=yv)

         xgb_pars = {'min_child_weight': 50, 'eta': 0.3, 'colsample_bytree': 0.3, 'max_depth': 10,
                     'subsample': 0.8, 'lambda': 1., 'nthread': -1, 'booster' : 'gbtree', 'silent': 1,
                     'eval_metric': 'rmse', 'objective': 'reg:linear'}

         watchinglist = [(dtrain, 'train'), (dvalid, 'valid')]

         model = xgb.train(xgb_pars, dtrain, 15, watchinglist, early_stopping_rounds=2,
                           maximize=False, verbose_eval=1)

         print('Modeling RMSLE %.5f' % model.best_score)
```



```
[0]     train-rmse:4.22632      valid-rmse:4.22757
Multiple eval metrics have been passed: 'valid-rmse' will be used for early stopping.

Will train until valid-rmse hasn't improved in 2 rounds.
[1]     train-rmse:2.97634      valid-rmse:2.9779
[2]     train-rmse:2.10627      valid-rmse:2.10828
[3]     train-rmse:1.50768      valid-rmse:1.51038
[4]     train-rmse:1.0977       valid-rmse:1.1013
[5]     train-rmse:0.82356      valid-rmse:0.828341
[6]     train-rmse:0.645152     valid-rmse:0.651593
[7]     train-rmse:0.535765     valid-rmse:0.54387
[8]     train-rmse:0.472743     valid-rmse:0.482233
[9]     train-rmse:0.437489     valid-rmse:0.448313
[10]    train-rmse:0.41794      valid-rmse:0.429668
[11]    train-rmse:0.407544     valid-rmse:0.419869
[12]    train-rmse:0.401203     valid-rmse:0.414095
[13]    train-rmse:0.398004     valid-rmse:0.411455
[14]    train-rmse:0.395435     valid-rmse:0.409449
Time taken by above cell is 102.19791388511658.
Modeling RMSLE 0.40945
```

5. Visualization by Traffic Heatmap:

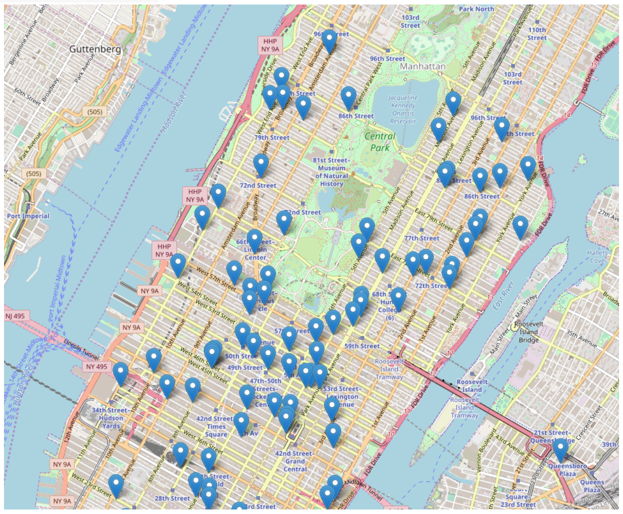
The heatmap will present the request according to different hour. The picture has taken a summary of lat-long and their count, and assign a different color for different count range.



6. Visualization by Terrain map Folium:

Before generating map and adding the pick up and drop coordinates, firstly, the function has been defined to calculate summary of given list of clusters.

```
In [69]: osm = show_fmaps(train_data, path=1)
         osm
```



7. Validation by Sentiment Analysis:

Let us check if the sentiment of twitters has some consistency with the result generated by Pearson Correlation.

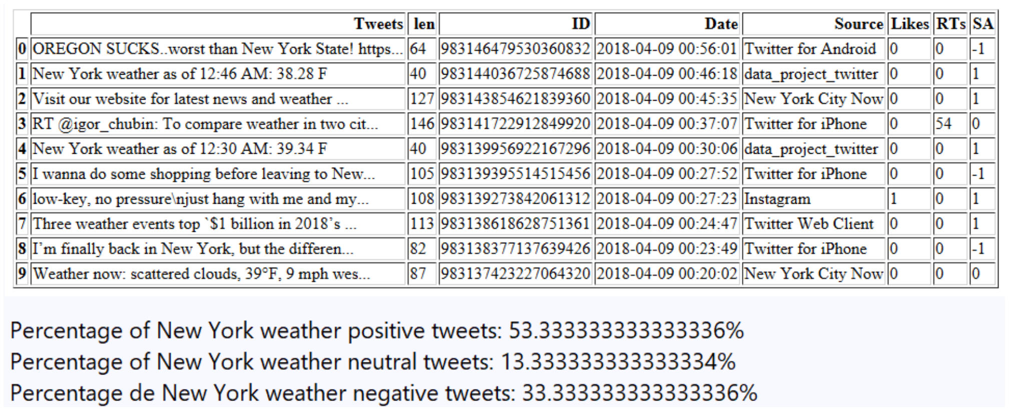
After handling authorization, defining the keyword, adding tweets data into dataframe, sentiment analysis could be applied by importing TextBlob.

```
In [98]: # We create an extractor object:
         api = twitter_setup()

         keyword="New York traffic"
         tweets=searchKeyword(keyword)
         dftweets=createDataFrame(tweets)

         print("Number of {} tweets extracted: {}.\n".format(keyword, len(tweets)))

         # We display the first 10 elements of the dataframe:
         display(dftweets.head(10))

         dftweets=AddDataFeature(dftweets, tweets)

         print("After adding features:\n")
         display(dftweets.head(10))

         dftweets=addSentimentFeature(dftweets)
         print("After adding Sentiment Features:\n")
         display(dftweets.head(10))

         pos_tweets, neu_tweets, neg_tweets=classifyTweets(dftweets)
```

	Tweets	len	ID	Date	Source	Likes	RTs	SA
0	RT @sarahkendzior: Dennis Shields, the residen...	140	983146290837118976	2018-04-09 00:55:16	Twitter Web Client	0	264	0
1	CLEARED - traffic jam:New York Ave Arlington	47	983144533352501248	2018-04-09 00:48:17	DFW511PPFTW	0	0	1
2	RT @TotalTrafficDFW: Accident, right lane bloc...	140	983134768995938305	2018-04-09 00:09:29	Twitter for Android	0	1	1
3	Accident, right lane blocked in #Arlington on ...	138	983133921150939137	2018-04-09 00:06:07	TTN DFW traffic	1	1	1
4	Recent: Shaun White Wins Gold In Halfpipe At T...	140	983133838917455873	2018-04-09 00:05:47	Heroic Social	0	0	1
5	RT @IamSamSanyal: @Tharki_Thug @itsSaharsh @_B...	140	983130798214692864	2018-04-08 23:53:42	Twitter for Android	0	7	1
6	Accident reported in #Arlington on 20 WB at Ne...	127	983130672121503744	2018-04-08 23:53:12	TTN DFW traffic	0	0	1
7	New York traffic is horrible	28	983122913032392704	2018-04-08 23:22:22	Twitter for iPhone	0	0	-1
8	RT @sarahkendzior: Dennis Shields, the residen...	140	983122605484969984	2018-04-08 23:21:09	Twitter for Android	0	264	0
9	RT @NBCNews: FDNY: 3-alarm fire burning in Tru...	140	983117072472276992	2018-04-08 22:59:10	Twitter for iPad	0	443	1

```
Percentage of New York traffic positive tweets: 66.66666666666667%
Percentage of New York traffic neutral tweets: 26.666666666666668%
Percentage de New York traffic negative tweets: 6.666666666666667%
```

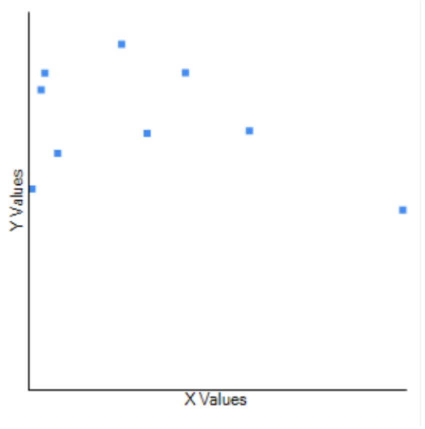
From this result, we find out the percentage of traffic positive tweets is 66.6% and percentage of negative tweets is 6.6%;



	Tweets	len	ID	Date	Source	Likes	RTs	SA
0	OREGON SUCKS..worst than New York State! https...	64	983146479530360832	2018-04-09 00:56:01	Twitter for Android	0	0	-1
1	New York weather as of 12:46 AM: 38.28 F	40	983144036725874688	2018-04-09 00:46:18	data_project_twitter	0	0	1
2	Visit our website for latest news and weather ...	127	983143854621839360	2018-04-09 00:45:35	New York City Now	0	0	1
3	RT @igor_chubin: To compare weather in two cit...	146	983141722912849920	2018-04-09 00:37:07	Twitter for iPhone	0	54	0
4	New York weather as of 12:30 AM: 39.34 F	40	983139956922167296	2018-04-09 00:30:06	data_project_twitter	0	0	1
5	I wanna do some shopping before leaving to New...	105	983139395514515456	2018-04-09 00:27:52	Twitter for iPhone	0	0	-1
6	low-key, no pressure\njust hang with me and my...	108	983139273842061312	2018-04-09 00:27:23	Instagram	1	0	1
7	Three weather events top `$1 billion in 2018’s ...	113	983138618628751361	2018-04-09 00:24:47	Twitter Web Client	0	0	1
8	I’m finally back in New York, but the differen...	82	983138377137639426	2018-04-09 00:23:49	Twitter for iPhone	0	0	-1
9	Weather now: scattered clouds, 39°F, 9 mph wes...	87	983137423227064320	2018-04-09 00:20:02	New York City Now	0	0	0

```
Percentage of New York weather positive tweets: 53.333333333333336%
Percentage of New York weather neutral tweets: 13.333333333333334%
Percentage de New York weather negative tweets: 33.333333333333336%
```

After 10 iterations of sentiment analysis, the correlation could be calculated as the chart. Pearson Correlation Coefficient is -0.385. That means there is a weak negative correlation between weather and traffic: It is likely that if the sentiment of weather is positive, the traffic will be a little worse. Vice versa. However, the correlation is not strong as imagine.



Conclusion

1. The trip duration could be predicted according to history records. The best model is DecisionTreeRegressor and current best internal CV score is -0.19.
2. The weather roughly affected the traffic, but not too much as common sense: It is used to be known that the traffic will getting worse with a worse weather. However, the traffic may be affected by diverse aspects of weather: perhaps the measures have been taken in advance, the numbers of vehicles has decreased due to severe weather. Thus, the weather could affect the traffic with -0.385 of Pearson correlation.
3. The geospatial heatmap could be plotted in visualized map based on features extracted and merged.

Future work

4. Further sentiment experience: In the sentiment analysis, just the keyword with broader and neutral meaning have been filtered and searched, like ‘weather’, ‘traffic’, ‘temperature’ etc. How about negative and positive keyword and will it affect the result? Like ‘accidents’, ’crash’, ’gale’, ‘sunshine’ etc.
5. Validation by more models: The model generated by TPOT may quite different. One time it was XGBoostRegressor. It changed to GradientBoostingRegressor and sometimes it become to DecisionTree. What is the reason?

References

[1] How Do Weather Events Impact Roads? https://ops.fhwa.dot.gov/weather/q1_roadimpact.htm
[2] Sentiment Analysis: https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/

[3] TPOT: http://epistasislab.github.io/tpot/
[4] TPOT: A Python tool for automating data science http://www.randalolson.com/2016/05/08/tpot-a-python-tool-for-automating-data-science/
[5] Folium Documentation: https://folium.readthedocs.io/en/latest/
[6] NYC TLC Trip data:
http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
[7] Open source routing machine: http://project-osrm.org/
[8] sklearn.tree.DecisionTreeRegressor Tutorial: http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html