**Evaluate how the weather and hours affects traffic of New York by better model**

The status of taxi trip duration and number of request will greatly depend on the time slot and the area of one city. In peak hour, the number of request will go up and meanwhile, the trip duration will increase as well in some area. Additionally, the weather condition in different month will appear with different temperature, precipitation or snow-fall. That may cause the different consequence on the traffic flow.

However, how do we evaluate specifically by kinds of approaches: model, animation?

**These are the steps we are going to do:**

1. Collect data with datetime, and position information(lat,log).
2. Process the data.
3. Feature extraction.
4. Generate and yield related data for animation: speed, distance, time cyclic.
5. Clustering by mini K means
6. Split the training data to train and validate.
7. Use the TPOT to train classifier and handle the cross validation.
8. Animation: Basic data processing
9. Plot geograph and build gif.
10. Plot correlation by any other language or platform: java and web by Flare.
11. Explore the correlation by sentiment analysis

**Collect data with datetime, and position information(lat,log):**

Raw data collected from diverse departments, such as National Weather Service and traffic department. 3 data source will be collected and merged by different attributes according to different requirement.

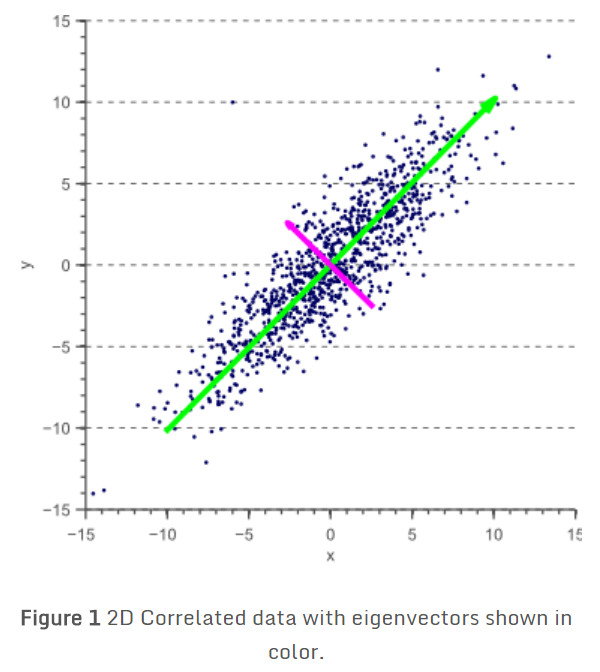
Firstly, we need the data is based on individual trip attributes: pickup\_datetime, dropoff\_datetime, pickup\_lat-long etc. Next, the data extracted by OSRM contains fastest routes information for each data point: Starting\_Street, End\_Street, total\_distance, total\_TravelTime etc, could provide information about duration information. Finally, the weather data contains the weather data: Min\_Temperature, Max\_Temperature, Average\_Temperature, Precipitation, Snow\_fall, snow\_depth, is important to uncover the potential correlation with traffic.

**Process the data:**

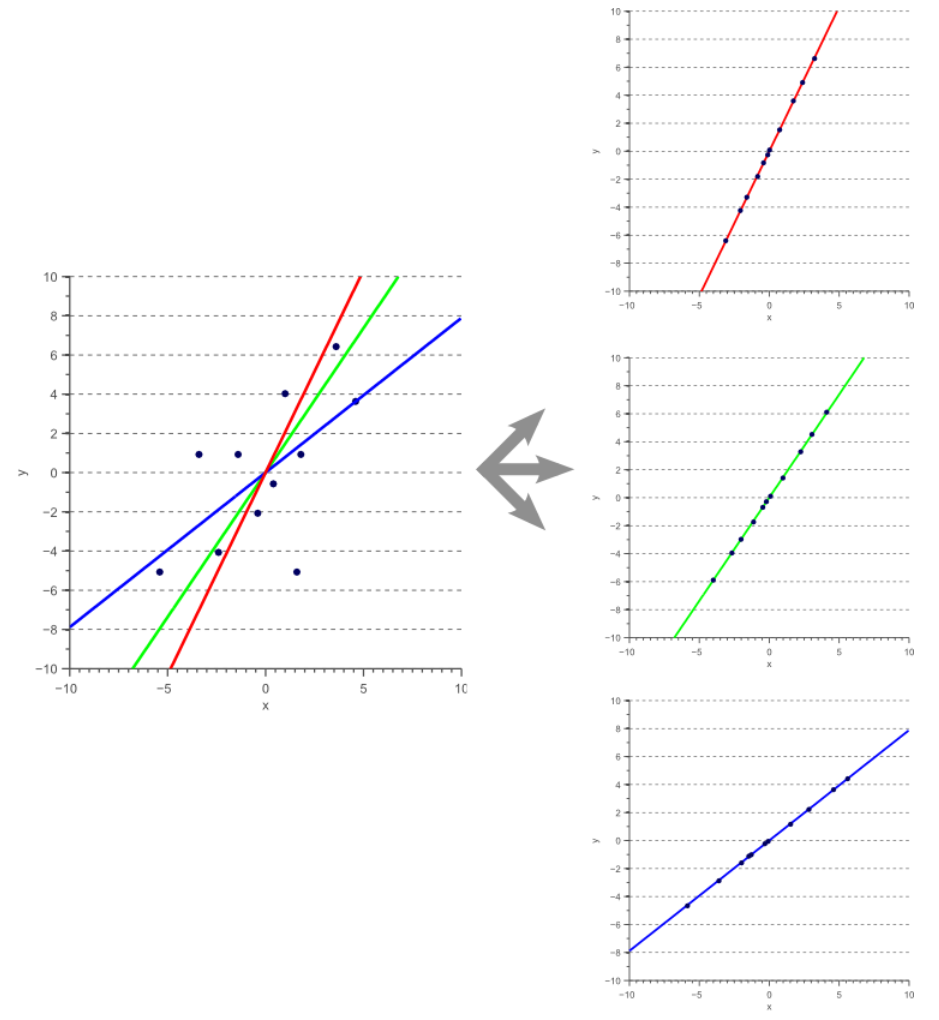
The raw data will mixed with empty, improper data. Besides, the format of date in the 3 data files are quite different, thus, unify them by functions and merge the related dimensions needed. It is not difficult.

**Feature extraction:**

The dimensions of these 3 dataset more than 50.  It showed that classifiers tend to overfit the training data in high dimensional spaces. The question then rises which features should be preferred and which ones should be removed from a high dimensional feature vector. In this section, If all features in this feature vector were statistically independent, one could simply eliminate the least discriminative features from this vector. However, in practice, many features depend on each other or on an underlying unknown variable. A single feature could therefore represent a combination of multiple types of information by a single value. Removing such a feature would remove more information than needed. Before eliminating features, we would like to transform the complete feature space such that the underlying uncorrelated components are obtained.



We found that these so called ‘principal components’ are obtained by the eigendecomposition of the covariance matrix of our data. The dimensionality is then reduced by projecting the data onto the largest eigenvectors.



**Generate and yield related data for animation: speed, distance, aggregation.**

For further analysis, the existing dimensions could not satisfy the analysis, and then, yielding more dimensions are necessary. Average speed will be calculate by haversine distance and duration, and dimensions of aggregation will focus on temporal, geospatial, pickup/dropoff aggregation.

**Clustering by** [**MiniBatchKMeans**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans)

For retrieving the aggregation of pickup/dropoff, we are considering Mini-K-means. Why we do not use K means? K-means is one of the most used clustering algorithms. With the increasing size of the datasets being analyzed, this algorithm is losing its attractive because its constraint of needing the whole dataset in main memory. For this reason several methods have been proposed to reduce the temporal and spatial cost of the algorithm.

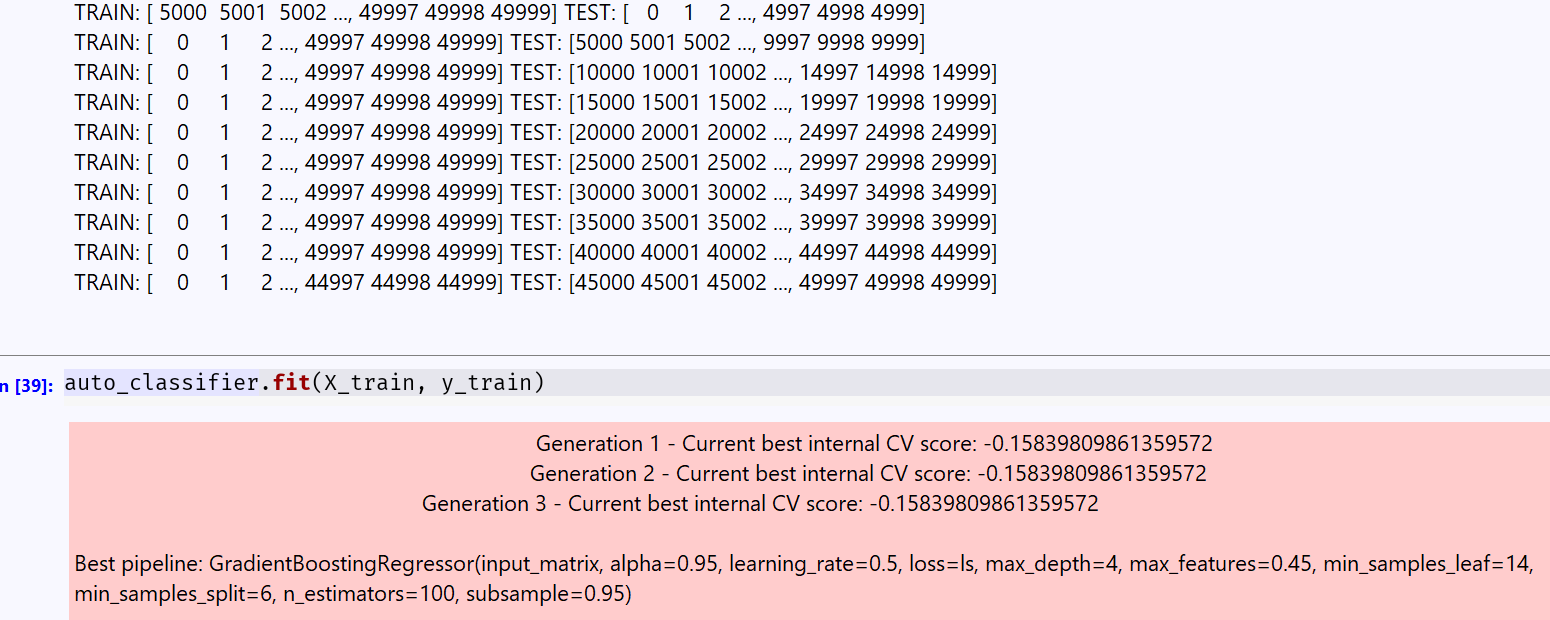
The [MiniBatchKMeans](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html" \l "sklearn.cluster.MiniBatchKMeans" \o "sklearn.cluster.MiniBatchKMeans) is a variant of the [KMeans](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans) algorithm which uses mini-batches to reduce the computation time, while still attempting to optimize the same objective function.

Mini-batches are subsets of the input data, randomly sampled in each training iteration. These mini-batches drastically reduce the amount of computation.

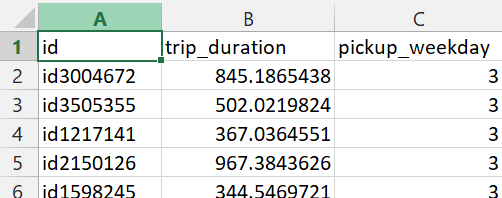
**Split the training data for TPOT to train, and handle the cross validation.**

How to choose the best pipeline to fit best? Different models are better suited for different tasks. we apply a “tuned” random forest to the problem, but we’re going to find that the random forest isn’t well-suited for signal processing tasks. What if we tried a different model, for example a logistic regression? Not sure about it. Always try out many different machine learning models for every machine learning task that you work on. Trying out—and tuning—different machine learning models is another tedious yet vitally important step of machine learning pipeline design.

TPOT will automate the most tedious part of machine learning by intelligently exploring thousands of possible pipelines to find the best one for your 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. In this case, we apply the TPOT after splitting the training data.



Predict the data based on best pipeline:



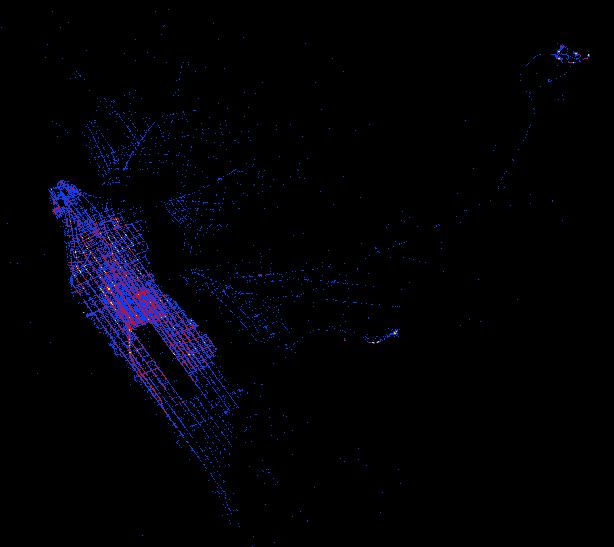
**Animation: Basic data processing**

### Let us do some preprocessing on the duration data: visualize the trip duration given using log-scale distplot in sns to get some finding: From the chart, the trip-durations are like Gaussian and few trips have very large duration. Most of the trips are e^4 = 1 minute to e^8 ~ 60 minutes: probably are taken inside Manhattan or in New York only.

### 

**Plot geograph and build gif.**

### For plotting the heatmap, we have taken an empty image and make it a color it black so that we can see colors where the lat-longs are falling. To visualize we need to consider each point of this image as a point represented by lat-long, assign a different color for different count range. The whole Manhattan is blue colored and with few green points as well, that shows that in Manhattan most of the trips are getting originated.



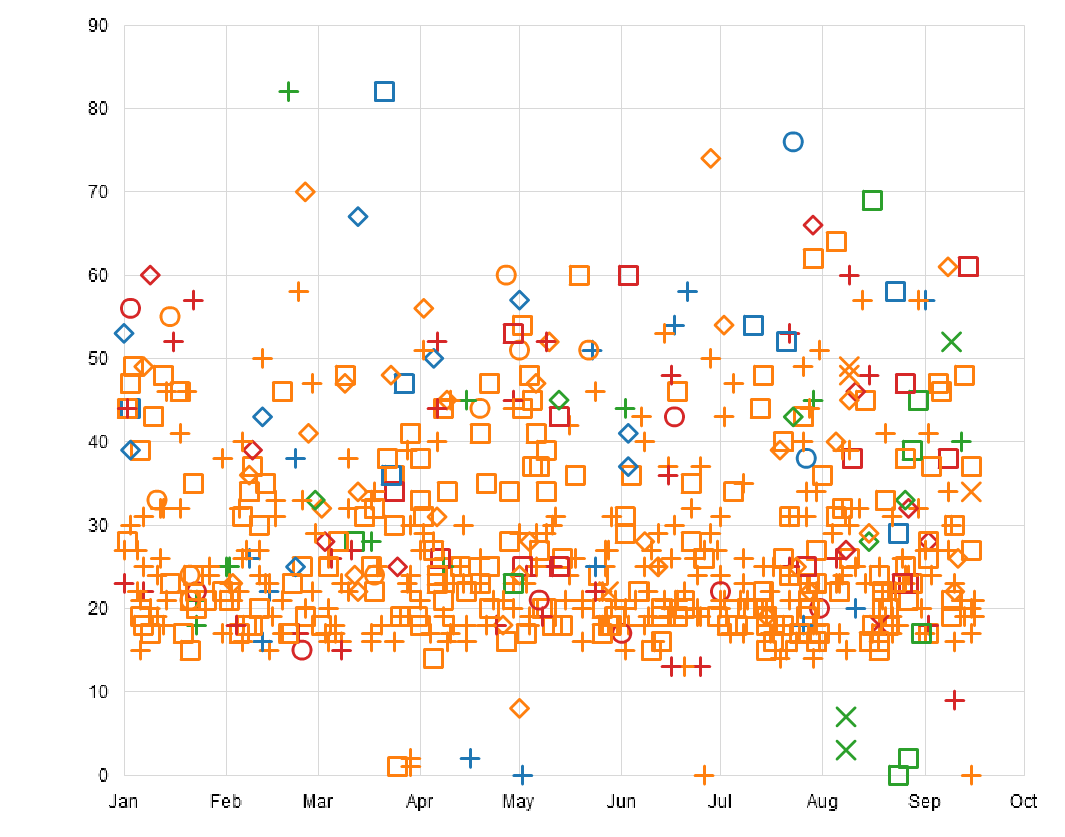
For further animation, let's just plot the heatmap and make an animation of pickup and see how with pickup hours the traffic changes: In some hour vary a large number of pickup are done that means traffic will be there and trip\_duration will be a little bit longer. Thus, function for color change in animation, the function to generate return a pic of plottings, and the function to create a gif of heatmaps would be defined.

**Plot correlation by any other language or platform: java and web by Flare.**

We get used to plot nested in notebook by seaborn or other tools in python, but could it be extended to web or flash? Yes. And could it be interactive as well? Yes. We could use Flare.

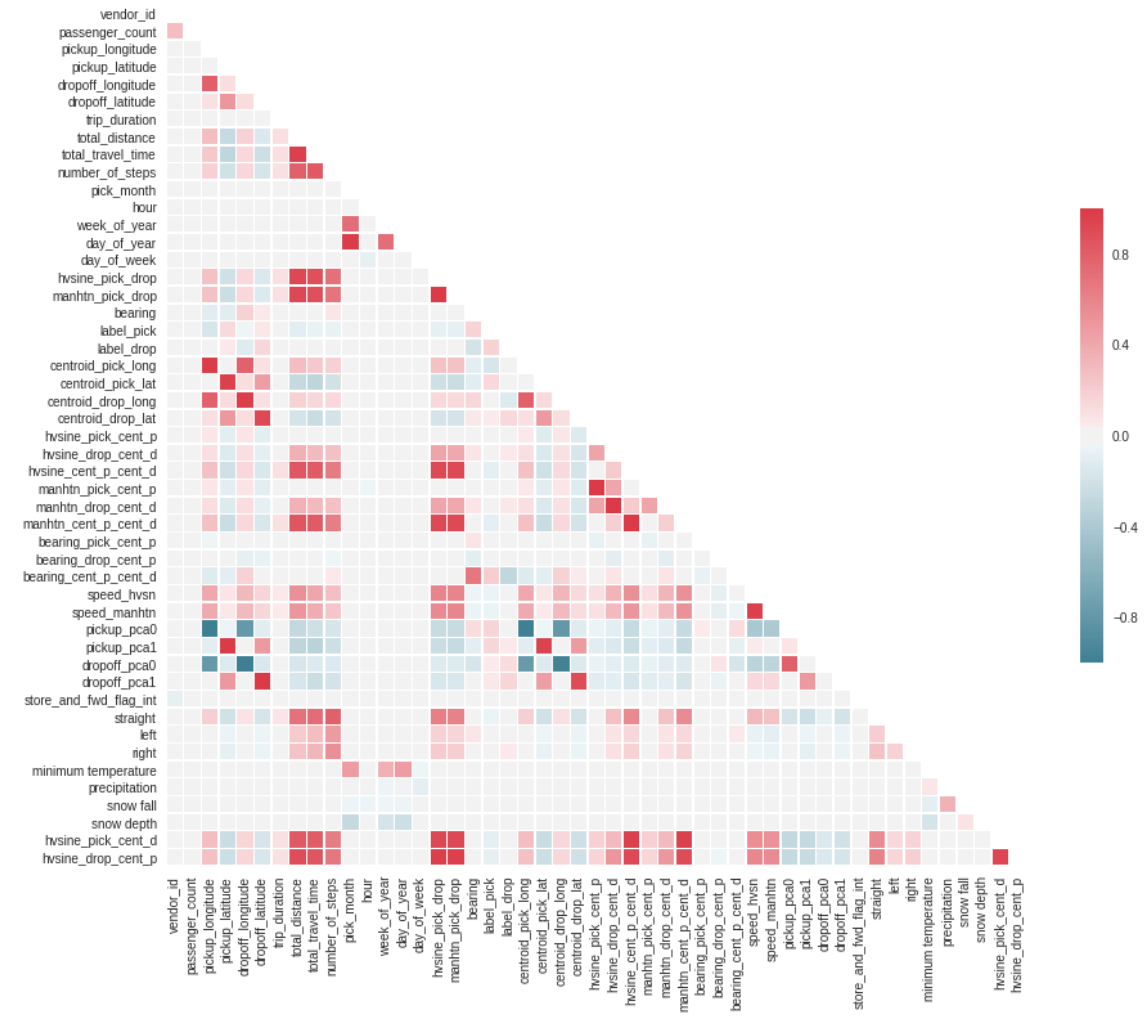
Flare is an ActionScript library for creating visualizations that run in the Adobe Flash Player. Even better, flare features a modular design that lets developers create customized visualization techniques without having to reinvent the wheel.

However, it will be implemented in java with a specific IDE Flex and Flex SDK.



Clearly, it is more likely the accidents will occur when the temperature is below 40F.

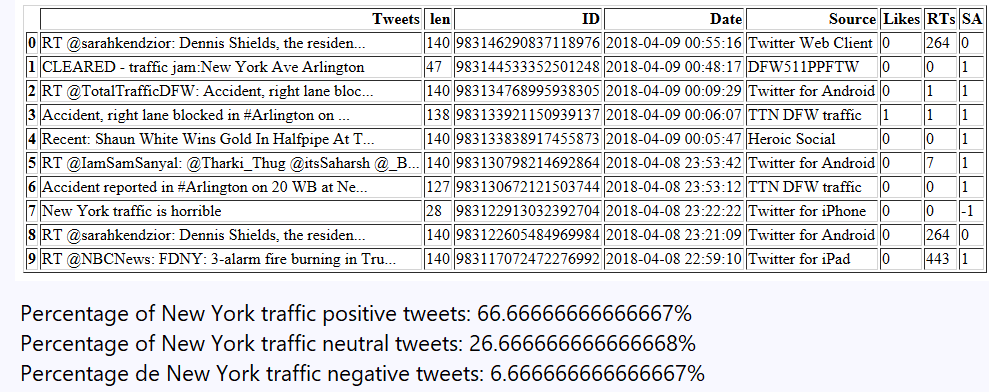
Additionally, for increasing number of dimensions we would like to explore, we will check correlation using a heatmap and check how the features are correlated by the heatmap & function of python.

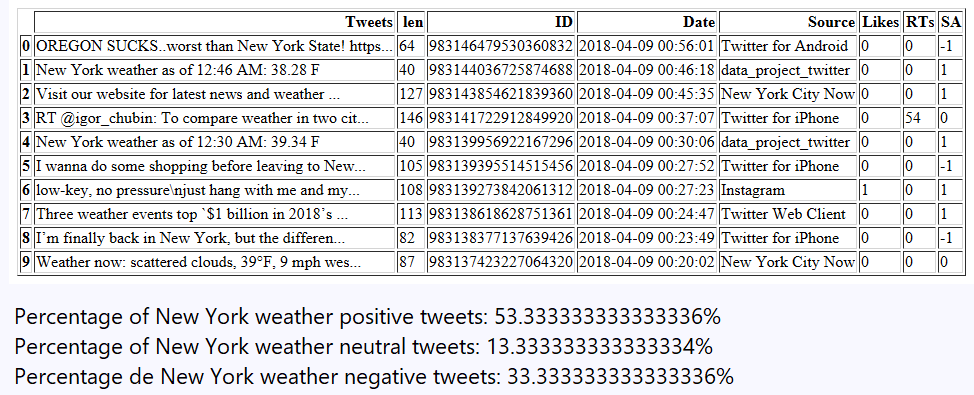


**Explore the correlation by sentiment analysis**

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.

For this case, the potential correlation between weather and traffic will be individually explored by twitter. If the traffic with positive opinion reflects the better weather or worse weather? Vise versa. Based on the reflection and weather forecasting, we could further suggests if the drivers could detour to avoid traffic digestion or not.





Findings: It seems a little linear correlated between weather & traffic opinion, especially the positive opinions.