A prediction and analyze on estate industry in big Toronto area

Tutor: Rami Yaerd
Members: Siyuan Huang, Zehua Zhou

Catalog

Instruction

purpose:

Toronto has been considered as the safest and lodgeable city in the world. More and more high quality immigration prefers to habit in Toronto. Also, this wave promotes the prosperous of Toronto's estate industry. This project aim to value the current estate industry in Toronto from several aspect (price, crime rate) by analyzing history data of the growth of price of different type of building (e.g., house, townhouse, condo).

Methodology:

1. LSTM:(Long Short Term Memory neural network) used for predicting the average price of selling building in the next 3 month
2. K-means: cluster the location of estate resources, compared the result of k-means and center point of heat map

Application:

This project has been implemented as a website https://timorchow.github.io/visualization/index-1.html with AWS support server EC2 instance. Users can see the advanced information by visit its subpage which contain data visualization part.

Programming language and dependency:

This project use python as data analyzing language and Javascript as data visualization's.

Major dependency: (python) Numpy, Scipy.cluster, Keras –GPU, MongoDB (Javascript) time, nvd3.

Data Collection:

The following table shows the data source and main data types (Table 1).

Source:	Data content
51ca.com	price, location, latitude, longitude, city, community, beds, dens, bathrooms, trading volumes, boundary of each city
cbc.ca/Toronto/features/crimemap	the number of criminal occurrences per 10,000 residents
map.google.ca	Schools, banks, restaurant nearby each house (or condo)

Table [1] Data Source

In order to collection information efficiently and easily, we write a script to crawl given website automatically.

51ca:

According to my analysis, the loading method is Ajax (Asynchronous javascript and xml), which means when we move the map and watch different area, the list on left will show the exact estates in this area (Figure 1).

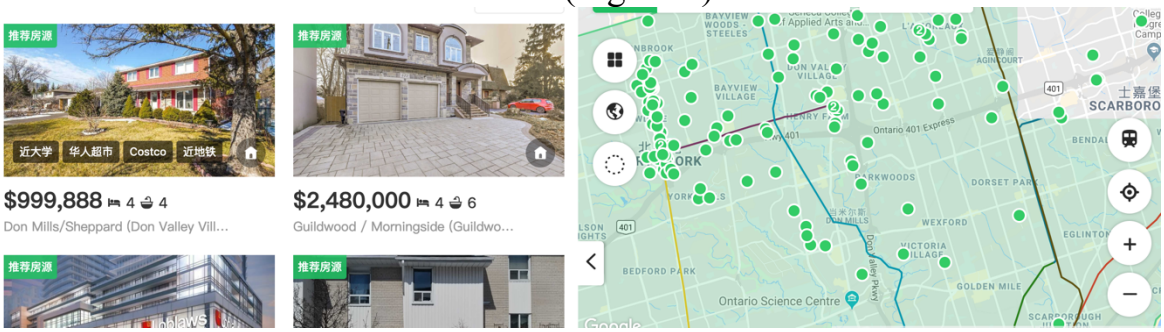


Figure [1] Estate agency website

By using google chrome DevTools, we check XHR (XML httpRequest) where we can get all DomString transferred by Ajax. Then we get a json string which by prasing contains 200 estate details (Figure 2). By this idea, we can move the map many times and get the estate details of the whole GTA area.

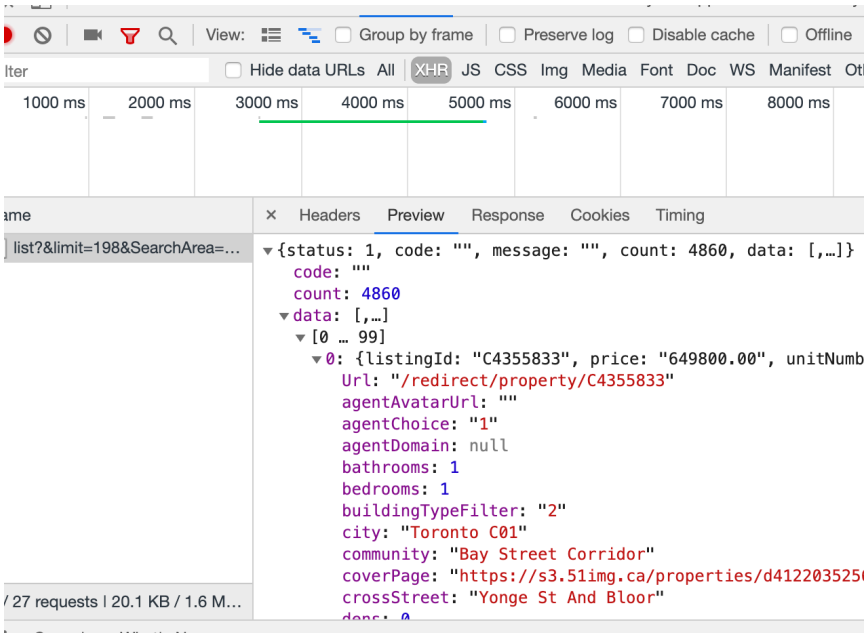


Figure [2] XHR in DevTools of Chrome

CBC crime:

CBC provides a CSV file which shows in an area the the number of criminal occurrences per 10,000 residents in 2018 (Table 2).

	A	B	C	D	E	F	G	H
1	Number	Name	2017	2018	Occurrences	Rank	Percent_Change	
2	1	West Humb	411	390	114.4	31	-5.10%	
3	2	Mount Olive	337	316	96.4	42	-6.20%	
4	3	Thistletown-	57	85	84	52	49.10%	
5	4	Rexdale-Kip	76	59	56.3	94	-22.40%	
6	5	Elms-Old Re	68	77	80.7	60	13.20%	
7	6	Kingsview V	163	134	61.6	87	-17.80%	
8	7	Willowridge	97	131	61.4	88	35.10%	
9	8	Humber Hei	40	41	38.8	112	2.50%	
10	9	Edenbridge-	42	57	38.3	114	35.70%	
11	10	Princess-Ro	31	48	43	106	54.80%	
12	11	Eringate-Ce	68	58	30.8	125	-14.70%	
13	12	Markland W	37	42	40.3	111	13.50%	
14	13	Etobicoke W	71	73	66.8	79	2.80%	
15	14	Islington-Ci	274	268	70.4	74	-2.20%	
16	15	Kingsway Sc	36	37	40.4	110	2.80%	
17	16	Stonegate-(	122	131	53.1	97	7.40%	
18	17	Mimico	205	201	75.6	68	-2%	
19	18	New Toront	157	137	125.7	20	-12.70%	

Table [2]

In order to analysis this area, we have get the latitude and longitude of each area. Consequently, Google GeoCoding API is chosen to convert address (or intersection) to geographic coordinate (like latitude and longitude). A python script is written to call google api and do this job (Figure 3).

```python
def geocoding_save_to_mongodb():
    '''
    transfer a given place to coordinate
    :return:
    '''
    my_assault = my_db['assault']
    address_list = my_assault.find()
    address_list = [address for address in address_list]

    pre_url = "https://maps.googleapis.com/maps/api/geocode/json?address
    GEOCODE_API = "AIzaSyD3bduGNl2aykdb_elYJxtVjB_CTFJhHE0"

    # travel every place, send a request, get result
    for address in address_list:
        address_name = address['Name']
        url = pre_url.format(address_name, GEOCODE_API)
        req = requests.get(url)
        data = json.loads(req.content)
        # content = '''{ "results" : [ { "address_components" : [ { "lon
        latitude = data['results'][0]['geometry']['location']['lat']
        longtitude = data['results'][0]['geometry']['location']['lng']
        data = {
            'lat': latitude,
            'lng': longtitude,
        }
        # save to mongodb
        my_assault.update({'_id': address['_id']},
                          {'$set': data})
        print(address['Number']+": "+address['Name']+" Success")
```

Figure [3] PythonScript Using Google API

Data Cleaning:

Use a series of rules to filter the repeated and useless information.
This is original data:

```
[{
    "listingId":"N4327501",                          # id of estate
    "price":"2298000.00",                            # price
    "unitNumber": null,                              # unit number (if it has)
    "streetAddress":"14 Cynthia Cres",               # address of street
    "crossStreet":"Yonge & King Rd",                 # cross of estate
    "community":"Oak Ridges",                        # community
    "city":"Richmond Hill",                          # city
    "type":1,
    "bedrooms":3,                                    # number of bedrooms
    "dens":2,                                        # number of dens
    "bathrooms":5,                                   # number of bathrooms
    "coverPage":"https://s3…df61c6.jpg",             # cover link
    "lastUpdate":1550266453,                         # last update timestamp
    "slug":"richmond-hill-real-estate/oak-r…a-crescent-n4327501",
    "latitude":"43.957096500",                       # latitude
    "longitude":"-79.459259300",                     # longitude
    "Url":"/redirect/property/N4327501",
    "favoriteId":"0",
    "display":"1",
    "buildingTypeFilter":"1",
    "openHouse":"0",
    "agentAvatarUrl":"",
```

```
        "tags": [
        ],
        "agentDomain": null,
        "agentChoice":"0"
    },
    ....
]
```


There are many useless estate details including dens, bathrooms, url, slug, agentDomain and others, which will be filtered before saving into database via adjustment statement. Pymongo was used to save these data into mongodb. We also use the same idea to clean assault and surrounding stores information.

Data Storage:

Nowadays, there is a trend that more and more people choose nosql as database due its efficiency, flexibility and low cost. MongoDB use json document to storage data, which is well-defined, human-friendly, and commonly-understand.

We use Amazon AWS as database server (figure 4):

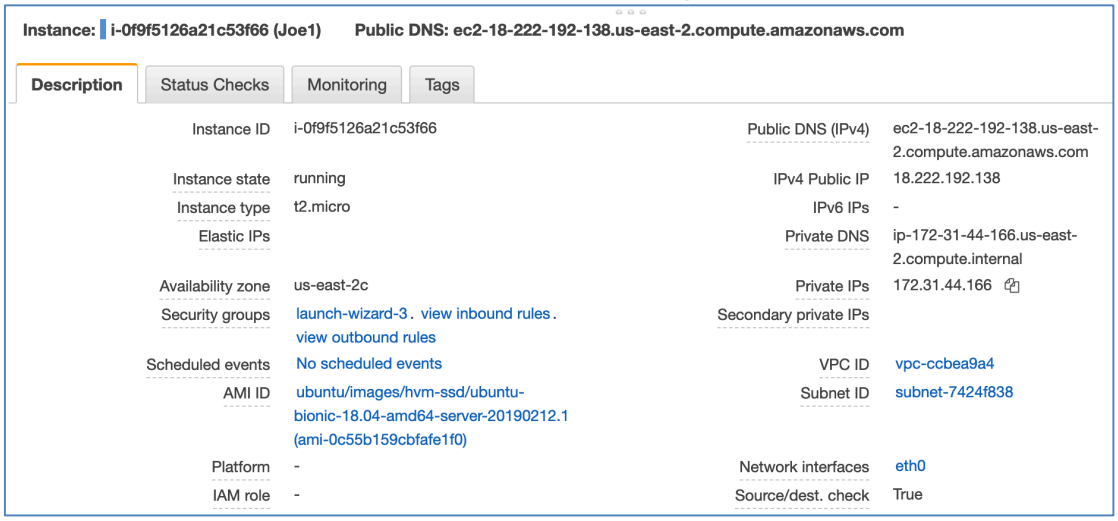


Figure [4] AWS server details

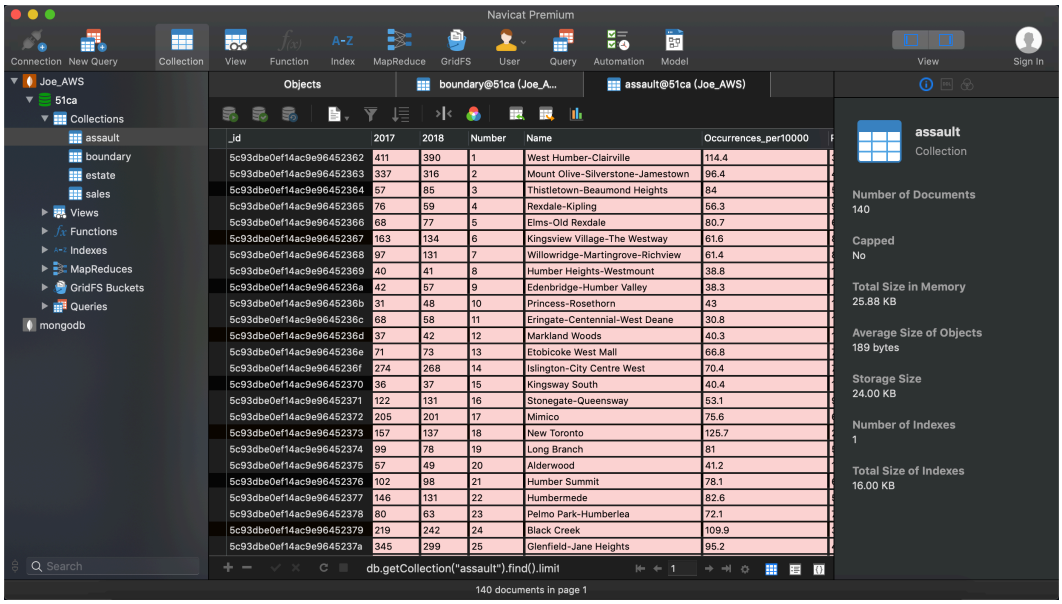


Figure [5] A collection in MongoDB under navicat environment

Data Reading:

For python, there is a library named pymongo, which is a tool to control mongodb by python. The method of searching is collection.find() ():

```
AWS = '18.222.192.138'
my_client = pymongo.MongoClient("mongodb://{}:27017/".format(AWS))
my_db = myclient['51ca']
my_estate = mydb["estate"]
result = my_estate.find({'city_name': 'toronto'})
```

Figure [6] get data from mongodb

This api uses find() with given pattern to search. This code is searching all the estates in Toronto city. In searching process, there is a problem that there are too many estate records in database, it is time-inefficient to get all of those once. Consequently, each time 200 records are inquired.

Data Prediction:

LSTM is an advanced deep learning model derived from RNN (recurrent neural network). LSTM stands for Long short-term memory. They are a special kind of Neural Network called Recurrent Neural Networks. Neural Networks is a machine learning technique where you stack up layers

containing nodes. Input data (features) go into the nodes of the input layers and the information is combined in a weighted manner and passed onto the next layer and so on, until it comes out of the output layer. Now the expected output (target) is compared with the model's output and the weights are updated accordingly. Based on markov process, these sequence model inherit the information from history. As well as, because they are learning algorithm, large amount of parameters help them perform better in regression problem.

The following figure is recurrent structure(figure [7]):

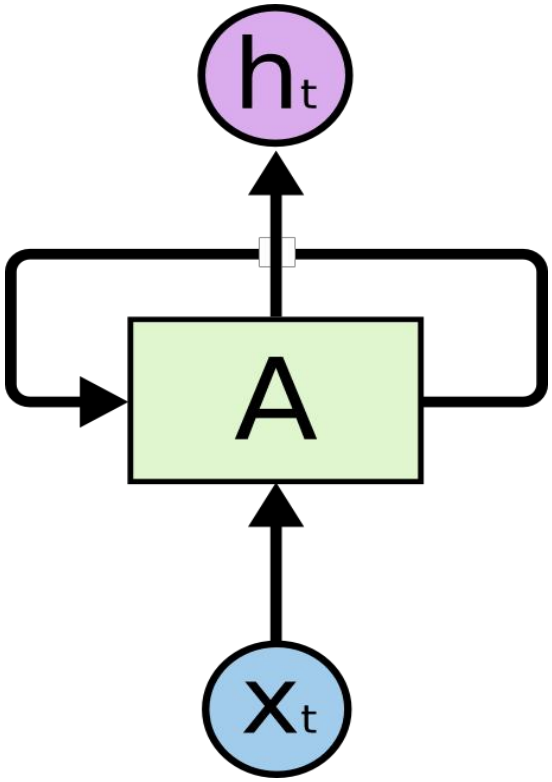
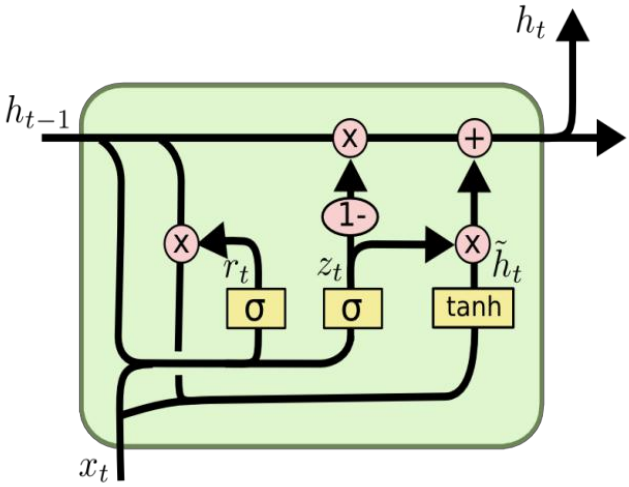


figure [7]

mathematical expression of RNN: $h^{(t)} = \text{sigma}(z^{(t)}) * \text{sigma}(W^{(t)} + Wh^{(t-1)}) + b)$

sigma = tanh(x)

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data. However, in reality, estate industry is more like a markov process which means the data in the future has relation with the closest memory. Since, the structure need to change in order to adapt the memory mechanism. Consequently, we rethink about the model we choose. It should have a gate to judge the whether the meaning value of history should be forgotten or not. Then comes to LSTM (long short term memory structure) showed in Figure []



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

figure [8] LSTM structure and formula

Follow the instruction above, due to the specification of data in estate industry, LSTM is chosen as the predict model for future development.

LSTM in Keras:

Keras is a high-level neural networks API, written in Python and capable of running on top of Tensorflow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. LSTM is packaged in the subclass of Sequential () in Keras. Figure [9] shows the way to construct LSTM in code

```
model = Sequential()
model.add(LSTM(256,input_shape=(3,1)))
model.add(Dense(1))
model.compile(optimizer='adam',loss='mse')
#Reshape data for (Sample,Timestep,Features)
X_train = X_train.reshape((X_train.shape[0],X_train.shape[1],1))
X_test = X_test.reshape((X_test.shape[0],X_test.shape[1],1))
#Fit model with history to check for overfitting
print("x_train_shape",X_train.shape)
history = model.fit(X_train,y_train,epochs=1000,validation_data=(X_test,y_test),shuffle=Fal
)#1
)#2
Xt = model.predict(X_test)
```

Figure [9] core code to build up a LSTM model in Keras

The training data is the average price of sold estate resources from 2010-2017.Specifically, the parameter “input_shape” represent the batch you feed to the network. And it should be adapted to the shape of input data for example the input_shape in this program is (3,1), and the shape of data’s input should be like (10,3,1). 10 can be changed for the requirement.

Result of prediction:

the orange poly line shows the test prediction of the price for the next 3 month at each point. In https://timorchow.github.io/visualization/index- 5.html. The data we used is the average price of estate in Toronto from 2014-2015. Figure [10] shows the figure in page.

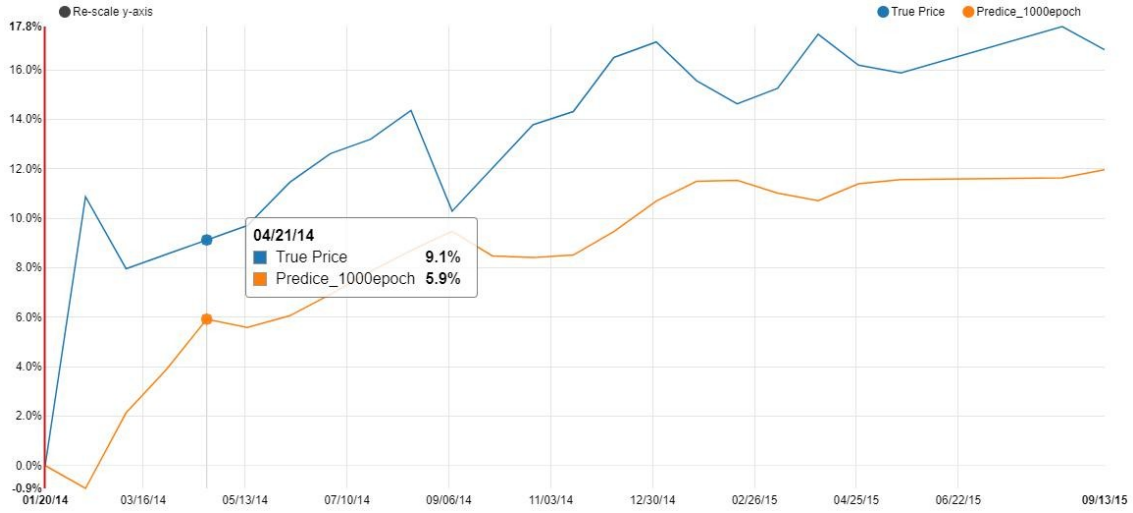


Figure [10] test result

In keras, weights and bias are initializing as normal distribution but generate it randomly. We cannot precisely get the same prediction for every time.

And there always a problem of deep learning. Whether more time we trained that means better result we will obtain. The answer is not sure in this project. Mathematically, more complex the model is, the less possible we may find the optimized problem. Because in reality, the model will be tested by new data for every time. And for each time if we didn’t do a back propagation the model may not be able to show the compatibility with new data. For this project, we have 80% training data and 20% test data, and

compared different epoch time of training Figure [11][12][13][14]shows the comparison

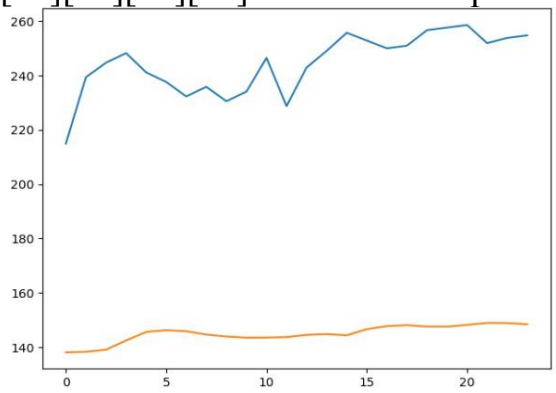


figure [11] trained after 3 epoch

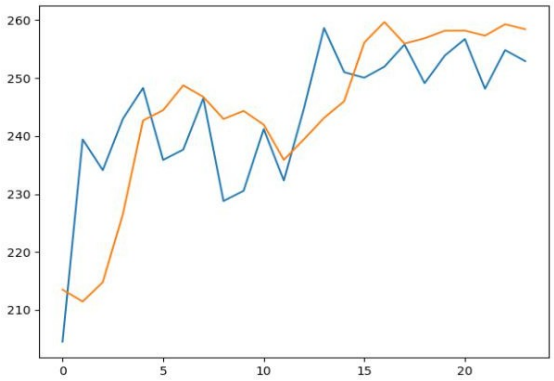


figure [12] trained after 100 epoch

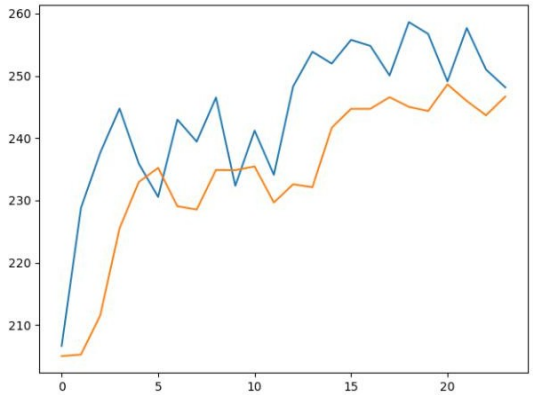


figure [13] trained after 3 epoch

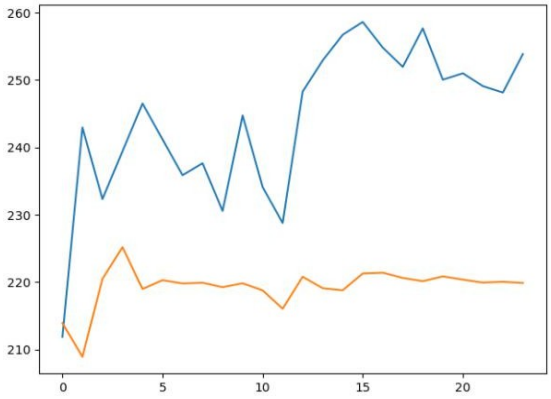


figure [14] trained after 100 epoch

From the same tested data model with 1000 epoch times training perform better than others. Though the detail still cannot be described precisely, the trend is nearly the same with the raw. And we can find the problem we mention before. Model trained after 10000 epochs shows the similar distribution as model trained only 3 epochs. And the testing data loss its variability after 12th test. So, the epoch time should be adjusted by developer according to the project and the reality. And 1000 epoch can be regard as an evidence for purchase decision Figure [15] shows the result for the last data.

```
32/92 [=========>....................] - ETA: 0s - loss: 4.6483e-04
92/92 [==============================] - 0s 1ms/step - loss: 0.0034 - val_loss: 0.0042
Epoch 999/1000

32/92 [=========>....................] - ETA: 0s - loss: 4.6482e-04
92/92 [==============================] - 0s 794us/step - loss: 0.0034 - val_loss: 0.0042
Epoch 1000/1000

32/92 [=========>....................] - ETA: 0s - loss: 4.6481e-04
92/92 [==============================] - 0s 1ms/step - loss: 0.0034 - val_loss: 0.0042
predicted:[[248.62218]], actual:[[255.7717191]]
```

Figure [15] training result during runtime

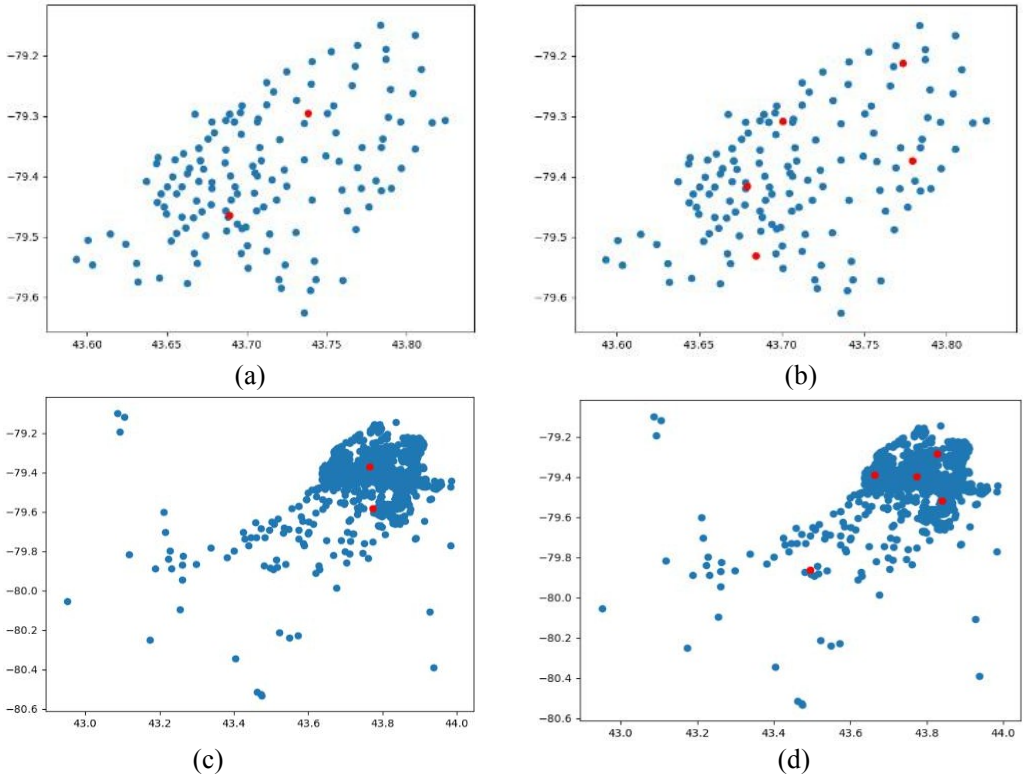
K-means

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups).

The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:

1. The centroids of the K clusters, which can be used to label new data
2. Labels for the training data (each data point is assigned to a single cluster)

This project utilizes K-means algorithm to find the densest place of estate trading. And compared the result in the heat map. If they don't meet together we can research the location provide by k-means. It may indirectly or directly give us the hint of the factor may influence the price. Figure (16) a, b shows when k=2 and 5 a center in the crime map. C and d shows the case in price map



Figure[16] experiment result of k-mean

from 4 above graph, we can find that the number of k in k-means is used to define the number of angle the cluster has.

K-means and heat map:

For the analyse purpose, we set the experimental result into google map, and the mean point doesn't meet the hottest place in the heat map. And further more we find that two mean point have big luxury mall and big sports center separately.

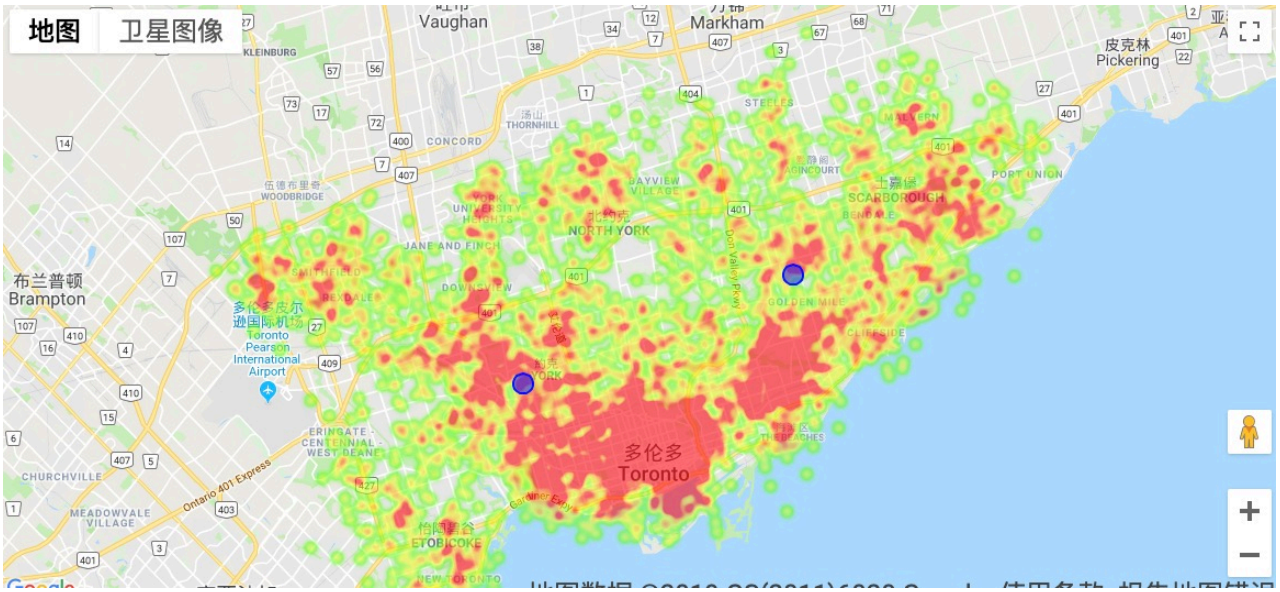


Figure [17] assault point with mean point

HeatMap:

A heat map is a graphic representation of data in which values are represented by colors. They allow you to track and record what visitors are doing on your website based on their mouse movements. In other words, heat maps track mouse movements, such as clicking on links and buttons, hovering or scrolling. Based on the action they track and quantify, heat maps can be classified into click maps, scroll maps and hover maps.

Price and assault HeatMap:

Gmpolt is a python library to draw on google map based on google map api. It is similar to common graphics lib like matplotlib but easy to use in this case. The idea is to get all the coordinate of estates, then draw it on map. The will be red where the price is high. But heatmap is showing density of point, so how can we show the price information? There is a way to achieve it. We make random points around one estate, the number of point depends on price, which means more surrounding points represent higher price. Price index is used to show how many points we make, which is price divided by 500000. For example, if a house selling at $200000, so we make 4 surrounding points. If an estate selling price is lower than $400000, the price index is one. I chose a Gaussian distribution in Numpy to make surrounding points (figure 7).

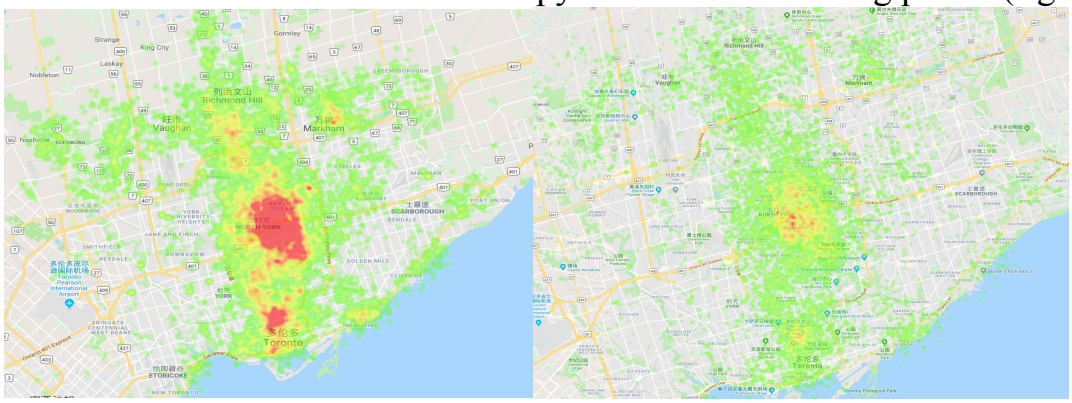


Figure [18] a Two heatmaps of price

The same idea was applied in crime heatmap (figure 8). We have a series of coordinate with numbers of assault happened surrounding there. The number is normalized by divided by 5. Then use Gaussian distribution to make surrounding point and density to show the assault events.

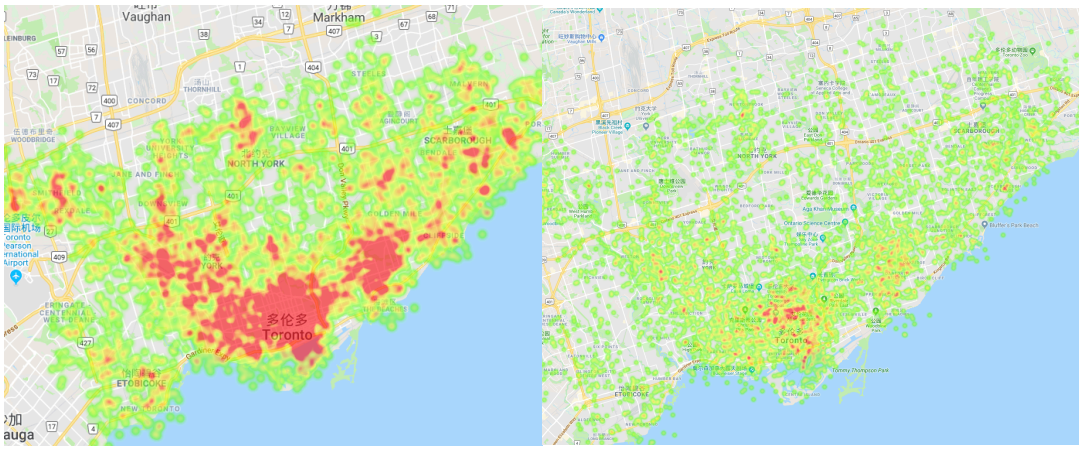
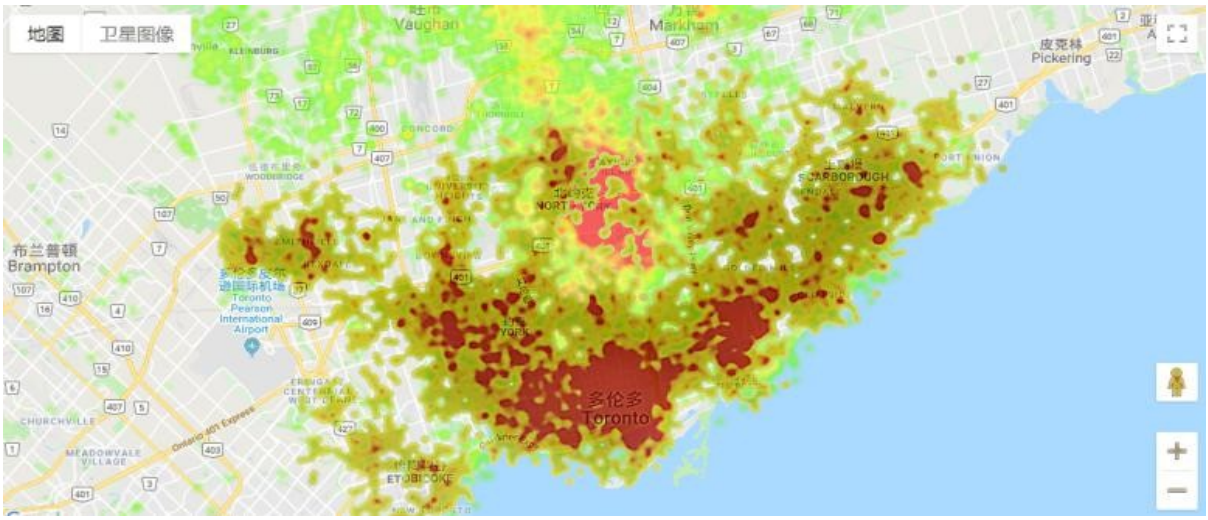


Figure [18] bTwo heatmaps of crime

Combining above information we merge figure [17] and figure [18] shows in Figure[19], we can approximately give it a conclusion : the place with high criminate rate may not be prosperous in estate industry.



Figure[19] merge criminate point and house selling porint

Bounding map:

In gmplot, there is a function named gmplot.plot(). We give 2 lists (latitude and longitude), color and width, then draw a polygon on google map (figure 9). One thing should be noticed, we cannot give all board points one time, because it will get messy. In order to avoid it, the boundary of each city should be added into map one by one.

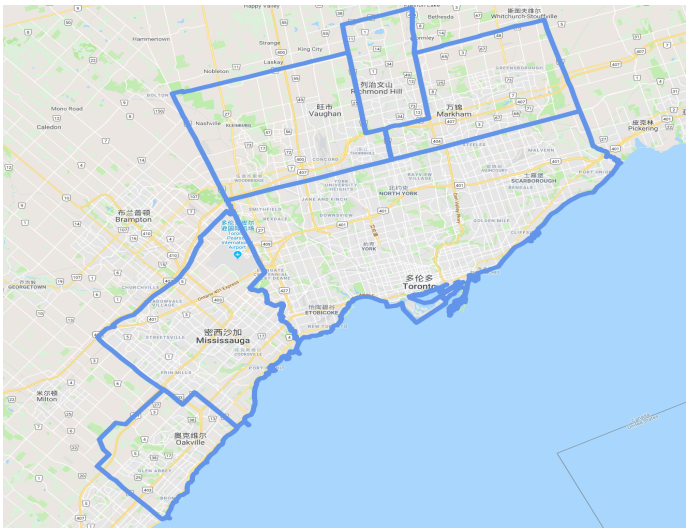


Figure [9] Bounding

Data Visualization:

This project consider web as the carrier of data visualization application, so javascript and its libs commonly will be used. Here we use the nvd3.js as the main dependency of the program. This lib provides plenty of click event and dynamic effect. Also, it is an attempt to build re-usable charts and chart components for d3.js without taking away the power that d3.js gives you. This is a very young collection of components, with the goal of keeping these components very customizable, staying away from your standard cookie cutter solutions.

Timestamp:

Timestamp is a very important concept in nvd3.js. Normally, Timestamp contains 13 bits or 10 bits int number to represent the time format in m/d/y. And in nvd3 we need 13 bits. So, we should transfer the format of date at first.

We have already show the line chart previously in figure [21]. Compared to show the value directly, we show the percentage of variety between current data and the historical data. So, reader can focus on the changed it had. As well as, it is good for predict the trend. In nvd3.js, we add the event that once we stay our cursor on the graph. it can dynamically show the change for each point. And we can re-scale the y axle, to see the graph evenly which shows in figure [21]

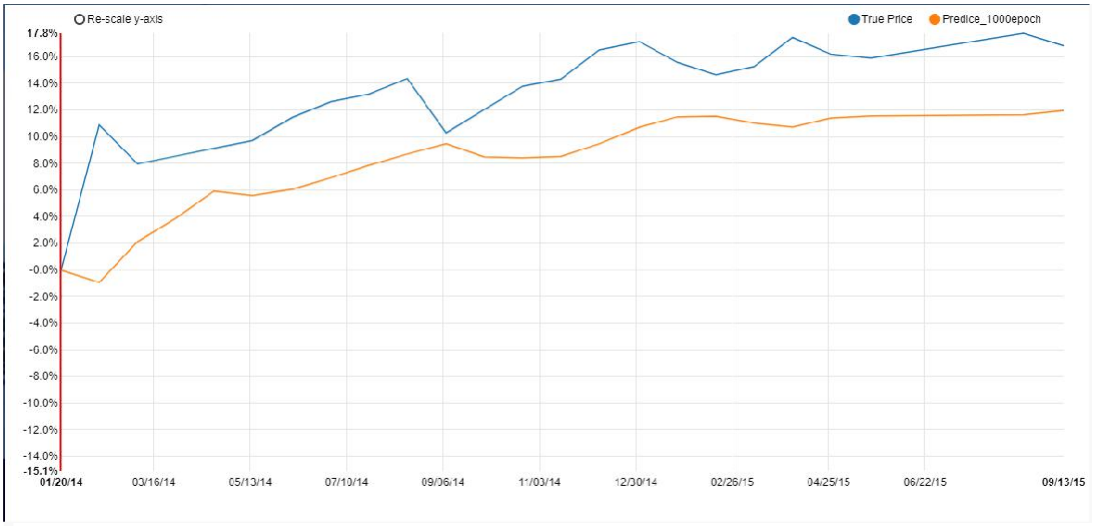


Figure [21]

Pie chart:

In order to show the static data, this project used pie chart to display. Data is the area which sold estates belong to, and the type of them

Under nvd3.js we visualize them by nv.piechart(), and also add the duration for HiddenEach(), this function allow our graph to dynamically display the number and percentage to satisfy the users' requirement. For example, in Figure [22] and Figure [23], after hiding the 'Vaughan', the percentage of Markham change from 9% to 10%. This method is friendly for the dynamical comparison.

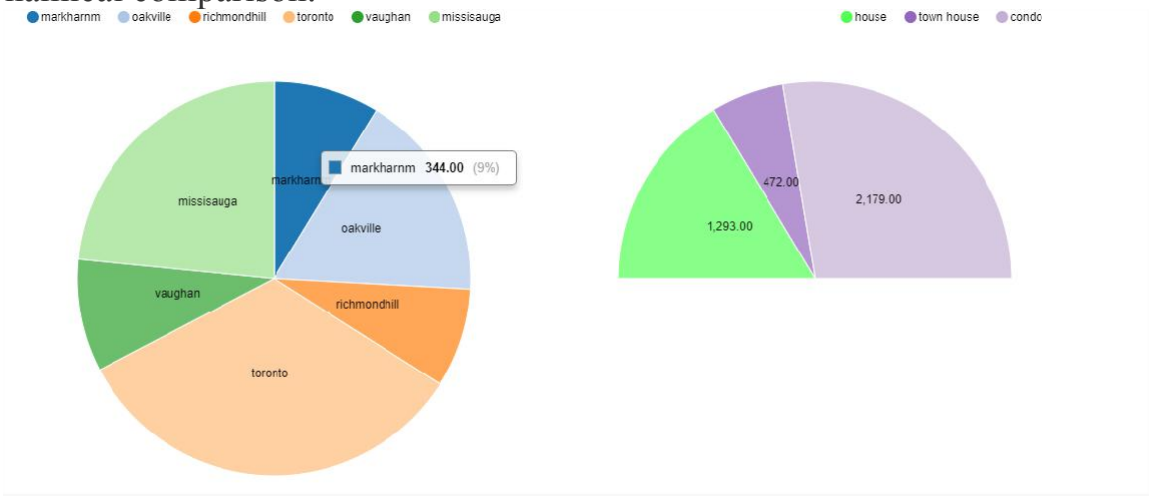


Figure [22] pie chart with Vaughan

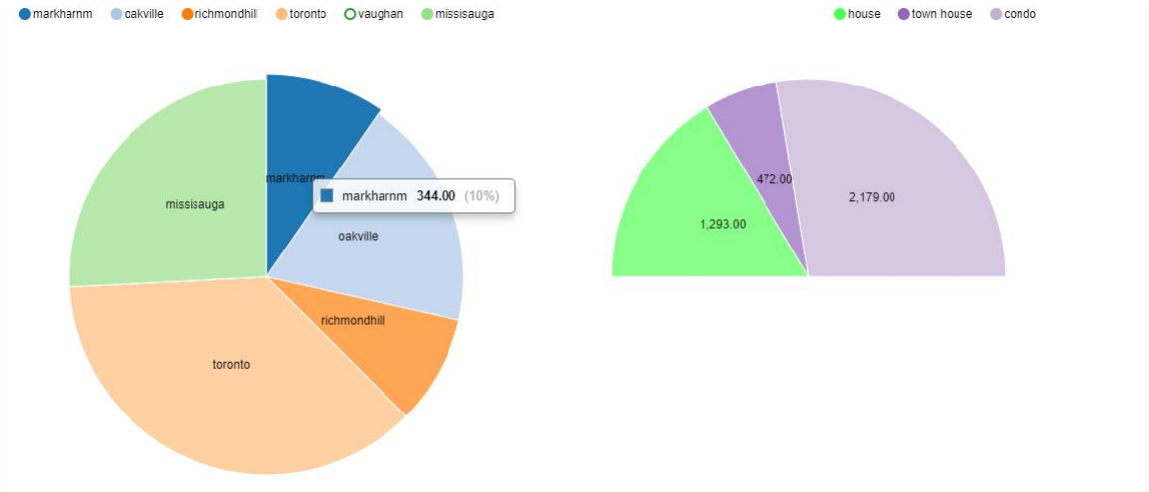


Figure [23] pie chart without Vaughan

Stacked chart

This graph shows the variance of demand of 3 different type estate (house, town house), From 2010 to 2017.

Stacked chart in nvd3.js give us another mode to compare the value in each year, showed in figure [24] and figure [25]. From the fist mode, we calculate the total number and the price of each type can be seen clearly. In the second mode, it emphasis the percentage of each number, and we normalize the sum for each point. This mode can be seen as another type of pie cha

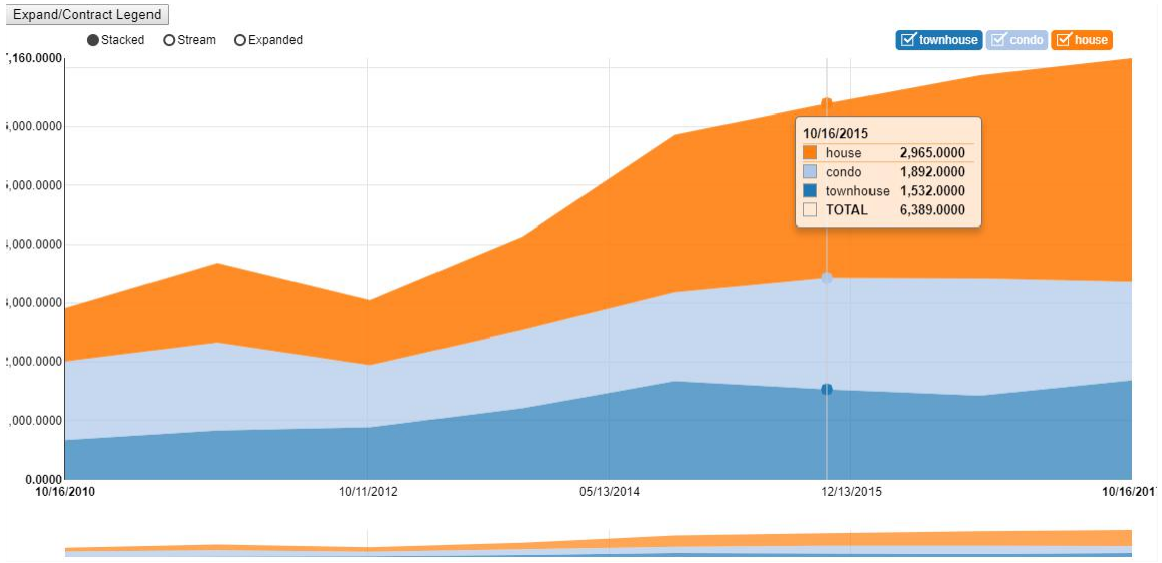


Figure [24]

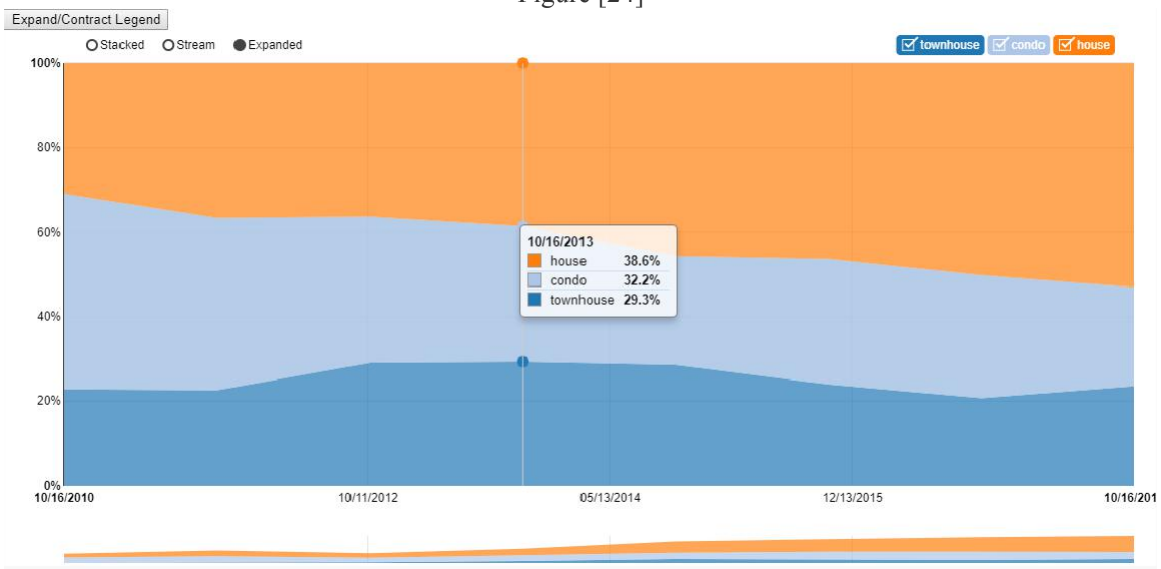


Figure [25]

conclusion

The core of data visualization and data analyze is data. In this project, we pay plenty of the time to collect data. Make sure that the reality and the dimension of the data can satisfy our requirement.

About the data analyze part, basically we try 2 different advanced machine learning method to give a comment to the current estate industry from time aspect and location aspect. Unfortunately, the LSTM can't give us a precise description of the future price, otherwise we can make an easier decision when we are buying house. And the k-means shows a very good result in the location description, and in our further research we will limit the range from center of k-mean, give this algorithm a interval to describe the relationship of the element implemented.

About data visualization, we utilize the dynamic effect in nvd3.j, also in Google's heat map. Click event and cursor event are package pretty well in nvd3.js which allow us to design a beautiful page and fantastic effective for dynamically display.

References

[1] Lloyd, Stuart P. "Least squares quantization in PCM." Information Theory, IEEE Transactions on 28.2 (1982): 129-137

[2] Bottou, L. and Bengio, Y. Convergence properties of the K-means algorithm. Adv. Neural Info Processing Systems, 1995, 7, 585–592.

[3] Du, Q. and Wong, T-W. Numerical studies of Macqueen's K-means algorithm for computing the centroidal Voronoi tessellations. Int. J. Computers Math. Applics, 2002, 44, 511–523.]

[4] Hochreiter, S. and Schmidhuber, J. Long short-term memory. Technical Report FKI-207- 95, Fakult at f ur Informatik, Technische Universit at Munchen.1995

[5] P. Simon, The Visual Organization: Data Visualization, Big Data and the Quest for Better Decisions, Harvard Business Review, June 13

[6] Instruction of NVD3.Js. http://www.nvd3.org.html, http://https://github.com/novus/nvd3.html