

# 基于上海地铁刷卡记录的 数据研究

10174503110 印张悦

华东师范大学 上海浦东 201208

# 基于上海地铁刷卡记录的数据研究

**摘要：**近年来，上海地铁飞速发展，乘坐地铁的人次越来越多，本文通过地铁闸机的刷卡信息进行分析和统计并制作一个让人们出行更加便捷、安全的软件原型，采用局部加权线性回归的方法，通过卡号和站名得到当天各个站点的客流信息，通过进站和出站时间分析不同时间段出发站点到目的站点花费的时间，从而估算出一天内不同时间段的时间花费。

**关键词：**地铁刷卡信息，局部加权线性回归，客流量，时间花费，python

**分类号：**C37

**Abstract:** In recent years, Shanghai Metro has developed rapidly and more and more people take the subway. This paper analyzes and counts the card information of the subway gates and makes a software prototype that makes people travel more convenient and safe. The method of local weighted linear regression is adopted. The passenger flow information of each station on the day is obtained by the card number and the station name, and the time spent by the departure site to the destination site in different time periods is analyzed by the inbound and outbound time, thereby estimating the time cost of different time periods in a day.

\* 作者简介： 印张悦，本科在读，yinzhangyue@126.com

**Keywords:** Subway card information, local weighted linear regression,  
traffic, time spent, python

**Classification:** C37

\* **作者简介:** 印张悦, 本科在读, yinzhangyue@126.com

目录

1 引言 ..... 4

1.1 研究背景..... 4

1.2 研究方法 ..... 4

2 地铁咨询软件基本框架搭建 ..... 5

2.1 最短路径算法与其改进 ..... 5

2.2 站点间距离的计算及时间预测 ..... 6

2.3 费用预测 ..... 7

2.4 搭乘可行性告知 ..... 7

2.5 结果展示 ..... 8

3 建模方法 ..... 9

3.1 基本思想 ..... 9

3.2 数学原理 ..... 11

4 数据集与数据探索 ..... 12

4.1 数据集介绍 ..... 12

4.2 数据探索及预处理 ..... 12

4.2.1 数据探索 ..... 12

4.2.2 数据预处理 ..... 13

4.3 数据分析 ..... 14

4.3.1 统计地铁站数量并获取地铁站名称 ..... 14

4.3.2 统计当天客流量信息 ..... 16

4.4 数据可视化 ..... 17

4.4.1 普通折线图可视化 ..... 17

4.4.2 wordcloud 词云可视化 ..... 17

4.4.3 Excel 3D 可视化 ..... 19

4.5 地铁咨询软件的优化和局部加强线性回归数据清洗 ..... 19

4.5.1 地铁咨询软件的优化 ..... 20

4.5.2 局部加强线性回归数据清洗 ..... 20

5 局部加权线性回归 ..... 23

5.1 局部加权线性预测客流量 ..... 23

\* 作者简介： 印张悦，本科在读， yinzhangyue@126.com

5.2 局部加权线性预测不同时间段的时间花费 .....	26
5.3 结合局部加权线性预测对软件的改进 .....	29
6 总结 .....	30
7 附录 .....	31
探索: python 网络爬虫.....	31
参考文献 .....	34

# 1 引言

## 1.1 研究背景

近年来,上海地铁飞速发展,乘坐地铁的人次越来越多,每天的闸机刷卡数据规模宏大,如何利用好这些刷卡记录来帮助乘客选择合适的出行路线和出行时间,准确的计算出所需的费用和时间以方便乘客的出行。不同于市面上简单的地图软件,本文通过数据的分析推断出一天当中不同时间段搭乘同一条线路所需的时间变化,根据用户所处的时间段告知用户所需的时间。

## 1.2 研究方法

基于机器学习的局部加权线性回归算法预测客流量和不同时间段相同路线的时间花费,基于最短路径算法以及根据现实情况的改进版本分析乘客的出行路线,并推荐给乘客。基于机器学习的方法和数据科学的思想,通过 python 实现数据的分析、模型的搭建和软件原型的设计。

## 2 地铁咨询软件基本框架搭建

本文优先介绍软件的基本框架搭建，这一部分不涉及与数据分析结果模型的关联，是目前绝大多数地图软件推荐地铁最佳路径的基本想法。本文从最短路径算法与其改进，站点间距离的计算和搭乘可行性告知三方面论述。

### 2.1 最短路径算法与其改进

最短路径算法，本文采用 Dijkstra 算法：

①基本思想：设置一个集合  $S$  存放已经找到最短路径的顶点， $S$  的初始状态只包含源点  $v$ ，对  $v_i \in V-S$ ，假设从源点  $v$  到  $v_i$  的有向边为最短路径。以后每求得一条最短路径  $v, \dots, v_k$ ，就将  $v_k$  加入集合  $S$  中，并将路径  $v, \dots, v_k, v_i$  与原来的假设相比较，取路径长度较小者为最短路径。重复上述过程，直到集合  $V$  中全部顶点加入到集合  $S$  中。

②设计数据结构：

- 1、图的存储结构：带权的邻接矩阵存储结构。
- 2、数组  $dist[n]$ ：每个分量  $dist[i]$  表示当前所找到的从始点  $v$  到终点  $v_i$  的最短路径的长度。初态为：若从  $v$  到  $v_i$  有弧，则  $dist[i]$  为弧上权值；否则置  $dist[i]$  为  $\infty$ 。
- 3、数组  $path[n]$ ： $path[i]$  是一个字符串，表示当前所找到的从始点  $v$  到终点  $v_i$  的最短路径。初态为：若从  $v$  到  $v_i$  有弧，则  $path[i]$  为  $vvi$ ；否则置  $path[i]$  空串。
- 4、数组  $s[n]$ ：存放源点和已经生成的终点，其初态为只有一个源点  $v$ 。

```

def path_search(start, goal):
    """Find the shortest path from start state to a state
    with min change times such that is_goal(state) is true."""
    if start == goal:
        return [start]
    explored = set()
    queue = [[start, ('', 0)]]
    while queue:
        path = queue.pop(0)
        s = path[-2]
        # print(path, s)
        linenum, changetimes = path[-1]
        if s == goal:
            path.pop(-1)
            path.append(action)
            return path
        for state, action in sh_subway[s].items():
            if state not in explored:
                linechange = changetimes
                explored.add(state)
                # print(explored)
                if linenum != action:
                    linechange += 1
                # print(path)
                path2 = path[:-1] + [action, state, (action, linechange)]
                # print(path[:-1])
                queue.append(path2)
                queue.sort(key=lambda path: path[-1][-1])
    return []

```

根据这一思想，我们可以实现经过站点最少乘坐方式，但介于换乘的时间代价和人力代价，实际情况人们往往更愿意选择换乘次数最少的方法，因此对 Dijkstra 算法进行改进，增加变量 linechange 记录生成线路所搭乘地铁线路的数目，选取其中路径最短换乘最少的线路，推荐给用户。（代码实现见上图）

## 2.2 站点间距离的计算及时间预测

在高德地图开放 API 上爬取地铁站坐标，通过地铁站的经纬度计算最短球面距离，计算公式为：

$$d(x1, y1, x2, y2) = r * \arccos(\sin(x1) * \sin(x2) + \cos(x1) * \cos(x2) * \cos(y1 - y2)),$$

其中，x1, y1 是纬度\经度的弧度单位，r 为地球半径。介于地铁站的路线并非球面最短距离，为减少误差，求取经过每站所需的距离（利用刚刚得出的最短路径），而不是起点到终点的距离，但这些误差仍不可避免，积少成多可能产生

\* 作者简介： 印张悦，本科在读，yinzhangyue@126.com

极大的误差，因此根据数据来进行分析的优势就显现了出来（距离计算代码见下图）。根据求得的距离和经过的站数推断所需的时间，但这样时间是固定的，像笔者这样经常乘坐地铁的人一定知道搭乘所需的时间在一天中是变化的，如何预测不同时间段内从某一站到某一站的时间花费我们会在 4.5 小节中重点讨论。

```
def cal_dis(latitude1, longitude1, latitude2, longitude2):
    latitude1 = (math.pi/180.0)*latitude1
    latitude2 = (math.pi/180.0)*latitude2
    longitude1 = (math.pi/180.0)*longitude1
    longitude2 = (math.pi/180.0)*longitude2
    #因此AB两点的球面距离为:[arccos[sina*sinx+cosb*cosx*cos(b-y)]]*R...(a,b,x,y)
    #地球半径
    # print(latitude1, longitude1, latitude2, longitude2)
    R = 6378.1
    temp = math.sin(latitude1) * math.sin(latitude2) + math.cos(latitude1) * math.cos(latitude2) * math.cos(
        longitude2 - longitude1)
    if float(repr(temp)) > 1.0:
        temp = 1.0
    d = math.acos(temp) * R
    return d
```

## 2.3 费用预测

通过所得路径的长度，计算所需的费用，计算方式如下。从官方申明也可以看到，上海地铁也是采用最短路径算法计算票价的，但可能由于实际路线和球面最短距离的偏差产生不准确。这个时候也需要从数据集中获取信息。

### 票价定价规定

按照市物价主管部门批复的轨道交通网络票价体系，即：轨道交通实行按里程计价的多级票价，0~6公里3元，6公里之后每10公里增加1元；票价计算采用最短路径法，即：当两个站点之间有超过1条换乘路径时，选取里程最短的一条路径作为两站间票价计算依据。

## 2.4 搭乘可行性告知

为避免用户在不恰当的时间搭乘地铁路线，爬取地铁首末班车时间，利用 python 的 datetime 模块得到当前时间，判断是否在首班车之后，再根据所需的时间推断是否在末班车之前，这样的提示对于对上海地铁比较陌生的人士或者游客是非常友好的，代码实现见下图。

\* 作者简介： 印张悦，本科在读，yinzhangyue@126.com



```
def oper_time(s1, s2, time_1):
    f3 = open("operation_hours.txt", "r")
    charge1, charge2 = False, False
    time1 = datetime.strptime("23:59", "%H:%M")
    time2 = datetime.strptime("0:00", "%H:%M")
    for line1 in f3:
        line1 = line1.split()
        # print(line1)
        if s1 == line1[0]:
            time1 = datetime.strptime(line1[5], "%H:%M")
            charge1 = True
        if s2 == line1[0]:
            time2 = datetime.strptime(line1[6], "%H:%M")
            charge2 = True
        if charge1 == True and charge2 == True:
            break
    # print(time1, time2)
    now_time = datetime.now().strftime("%H:%M")
    now_time = datetime.strptime(now_time, "%H:%M")
    other_time = datetime.strptime("0:10", "%H:%M")
    cal_time = datetime.strptime("0:00", "%H:%M")
    other_time = other_time - cal_time
    time_ = time_ - cal_time
    if (now_time - other_time > time1) and (now_time + other_time + time_ < time2):
        print("在地铁运营时间内，欢迎搭乘")
    else:
        print("建议搭乘其他交通工具")
    f3.close()
```

## 2.5 结果展示

```
***推荐路线***
东靖路 line6
巨峰路 line12
东陆路 line12
复兴岛 line12
爱国路 line12
隆昌路 line12
宁国路 line12
江浦公园 line12
大连路 line4
临平路 line4
海伦路 line4
宝山路 line4
上海火车站 line4
中潭路 line4
镇坪路 line4
曹杨路 line4
金沙江路 line4
共17站
无相关历史记录，正在通过路径查找计算
预计金额4元
预计用时0小时55分钟
在地铁运营时间内，欢迎搭乘
预计到达时间: 19:23
```

## 3 建模方法

在数据科学研究中，线性回归是经常用到的重要手段，用来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法，运用十分广泛。其表达式为  $y = w'x + e$ ， $e$  为误差服从均值为 0 的正态分布。

线性回归：

1: 函数模型 (Model) :

$$h_w(x^i) = w_0 + w_1x_1 + wx_2 + \dots + w_nx_n^{+j}$$

$$h_w(x^j) = w^T x_i = W^T X^{+j}$$

$$X = \begin{bmatrix} 1 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad W = \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_n \end{bmatrix}^{+j}$$

假设有训练数据

$$D = \{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$$

那么为了方便我们写成矩阵的形式

$$X = \begin{bmatrix} 1, x_1^1, x_2^1, \dots, x_n^1 \\ 1, x_1^2, x_2^2, \dots, x_n^2 \\ \dots \\ 1, x_1^n, x_2^n, \dots, x_n^n \end{bmatrix} \quad XW = h_w(x^i)$$

2: 损失函数 (cost) :

现在我们需要根据给定的X求解W的值，这里采用最小二乘法。

### 3.1 基本思想

经过尝试，本文所探讨的数据集采用线性回归预测出来的效果不佳，因此我们采用局部加权的线性回归模型。

局部加权线性回归 (Local Weights Linear Regression) 也是一种线性回归，不同的是，普通线性回归是全局线性回归，使用全部的样本计算回归系数。而局部加权线性回归，通过引入权值 (核函数)，在预测的时候，只使用与测试点相近的部分样本来计算回归系数。

\* 作者简介： 印张悦，本科在读， yinzhangyue@126.com

损失函数：

$$J(w) = \frac{1}{n} \sum_{i=1}^n \alpha_i (y_i - w * x_i)^2 = \frac{1}{n} \alpha \|Y - X * w\|^2$$

同理推导：

$$\begin{aligned} J(w) &= \frac{1}{n} \alpha \|Y - X * w\|^2 = \frac{1}{n} (Y - Xw)^T \alpha (Y - Xw) \\ &= \frac{1}{n} (Y^T - w^T X^T) \alpha (Y - Xw) \\ &= \frac{1}{n} (Y^T \alpha Y - w^T X^T \alpha Y - Y^T \alpha Xw + w^T X^T \alpha Xw) \end{aligned}$$

其中， $\alpha$ 为是权重的对角矩阵。对 $w$ 求导得到：

$$\begin{aligned} \frac{dJ(w)}{dw} &= \frac{1}{dw} \left( \frac{1}{n} (Y^T \alpha Y - w^T X^T \alpha Y - Y^T \alpha Xw + w^T X^T \alpha Xw) \right) \\ &= \frac{1}{n} (0 - X^T \alpha Y - X^T \alpha Y + 2X^T \alpha Xw) \\ &= \frac{1}{n} (-2X^T \alpha Y + 2X^T \alpha Xw) = 0 \end{aligned}$$

所以，得到：

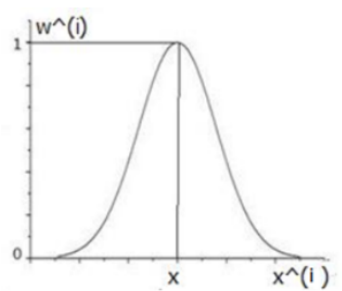
$$w = (X^T \alpha X)^{-1} X^T \alpha Y$$

加权的帽子矩阵为：

$$\hat{H} = X(X^T \alpha X)^{-1} X^T \alpha$$

## 3.2 数学原理

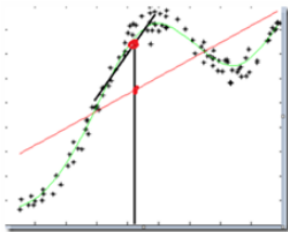
上式中参数  $x$  为新预测的样本特征数据，它是一个向量，参数  $\tau$  控制了权值变化的速率， $w^{(i)}$  和  $x$  的图像如下



可以看到

- (1) 如果  $|x^{(i)} - x| \approx 0$ ，则  $w^{(i)} \approx 1$ 。
- (2) 如果  $|x^{(i)} - x| \approx +\infty$ ，则  $w^{(i)} \approx 0$ 。

也即，离  $x$  很近的样本，权值接近于1，而对于离  $x$  很远的样本，此时权值接近于0，这样就是在  $x$  局部构成线性回归，它依赖的也只是  $x$  周边的点



图中红色直线使用线性回归做的结果，黑色直线使用LWR做的结果，可以看到局部加权回归的效果较好。

在我们原始的线性回归中，对于输入变量  $x$ ，我们要预测，通常要做：

1. Fit  $\theta$  to minimize  $\sum_i (y^{(i)} - \theta^T x^{(i)})^2$ .
2. Output  $\theta^T x$ .

而对于局部加权线性回归来说，我们要做：

1. Fit  $\theta$  to minimize  $\sum_i w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$ .
2. Output  $\theta^T x$ .

$w^{(i)}$  为权值，从上面我们可以看出，如果  $w^{(i)}$  很大，我们将很难去使得  $(y^{(i)} - \theta^T x^{(i)})^2$  小，所以如果  $w^{(i)}$  很小，则它所产生的影响也就很小。

通常我们选择  $w^{(i)}$  的形式如下所示：

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$

上式中参数  $x$  为新预测的样本特征数据，它是一个向量，参数  $\tau$  控制了权值变化的速率， $w^{(i)}$  和  $x$  的图像如下

## 4 数据集与数据探索

### 4.1 数据集介绍

本次分析的主要数据来源于 SODA 在 2015 年开放的部分数据，其余为搭建软件所需要的数据是本人爬取的，包括地铁站名、地铁站经纬度和地铁站的首末班车时间，爬取的过程在附录中介绍。原始数据集包括 6 个不同的属性，卡号、日期、时间、刷卡地点、交通工具、金额和是否优惠，共计 10 余万条数据，都是 2015 年 4 月 1 日地铁闸机记录下来的刷卡数据。

卡号	日期	时间	刷卡地点	交通工具	金额	是否优惠
602141128	2015/4/1	7:51:08	1号线莘庄	地铁	0	非优惠
602141128	2015/4/1	9:07:57	11号线昌吉东路	地铁	6	优惠
2201252167	2015/4/1	19:20:33	7号线场中路	地铁	4	非优惠
2201252167	2015/4/1	8:55:44	1号线陕西南路	地铁	4	非优惠
2201252167	2015/4/1	18:43:14	1号线陕西南路	地铁	0	非优惠
2201252167	2015/4/1	8:19:00	7号线上大路	地铁	0	非优惠
2001530605	2015/4/1	9:28:17	10号线国权路	地铁	5	非优惠
2001530605	2015/4/1	18:29:43	10号线国权路	地铁	0	非优惠
2001530605	2015/4/1	19:15:59	2号线金科路	地铁	5	非优惠
2001530605	2015/4/1	8:41:14	2号线广兰路	地铁	0	非优惠
101438819	2015/4/1	19:02:48	8号线大世界	地铁	0	非优惠
101438819	2015/4/1	16:44:23	7号线行知路	地铁	0	非优惠
101438819	2015/4/1	15:53:13	7号线行知路	地铁	4	非优惠
101438819	2015/4/1	19:59:09	3号线宝杨路	地铁	5	非优惠
101438819	2015/4/1	15:15:35	1号线黄陂南路	地铁	0	非优惠
101438819	2015/4/1	17:17:11	1号线黄陂南路	地铁	4	非优惠
101438819	2015/4/1	14:34:52	1号线黄陂南路	地铁	3	优惠
740892727	2015/4/1	16:46:43	7号线静安寺	地铁	0	非优惠
740892727	2015/4/1	13:47:35	7号线静安寺	地铁	3	非优惠
740892727	2015/4/1	17:02:06	4号线东安路	地铁	3	非优惠
740892727	2015/4/1	16:08:05	7号线静安寺	地铁	0	优惠

### 4.2 数据探索及预处理

导入 python 进行处理，由于原始数据集的编码问题，导入一直乱码，后来采用转换成 utf8 编码 txt 格式的方法成功导入。

#### 4.2.1 数据探索

通过对数据集的分析发现所有记录都是同一天发生的，且交通工具均为地铁，对于重复数据没有任何意义，因此全部删除，是否优惠根据笔者的了解是取决于此卡当月的消费额，而我们只有一天的数据，因此对我们的分析没有帮

\* 作者简介： 印张悦，本科在读，yinzhangyue@126.com

助，也应该删除。卡号数据通过时间和地点的联系我们可以总结出同一个人一天的行程，通过和费用的联系可以总结出某一站到某一站的花费的金额，通过时间则可以清洗出不同时间段内某一站到某一站的时间花费，进行局部加权线性回归预测不同时间段内的从某一站到某一站所需的时间。站名出现的次数我们可以总结出各个站点的客流量，与时间相联系可以用局部加权线性回归预测客流量随时间的变化。

## 4.2.2 数据预处理

导入 python 后将 string 断开成 list，并清楚重复数据（日期和交通工具）以及与此次分析无关的数据（是否优惠），最后得到的数据集如下图所示：

卡号	时间	刷卡地点	金额
602141128	7:51:08	1号线莘庄	0
602141128	9:07:57	11号线昌吉东路	6
2201252167	19:20:33	7号线场中路	4
2201252167	8:55:44	1号线陕西南路	4
2201252167	18:43:14	1号线陕西南路	0
2201252167	8:19:00	7号线上大路	0
2001530605	9:28:17	10号线国权路	5
2001530605	18:29:43	10号线国权路	0
2001530605	19:15:59	2号线金科路	5
2001530605	8:41:14	2号线广兰路	0
101438819	19:02:48	8号线大世界	0
101438819	16:44:23	7号线行知路	0
101438819	15:53:13	7号线行知路	4
101438819	19:59:09	3号线宝杨路	5
101438819	15:15:35	1号线黄陂南路	0
101438819	17:17:11	1号线黄陂南路	4
101438819	14:34:52	1号线黄陂南路	3
740892727	16:46:43	7号线静安寺	0

```
def Pretreatment(a):  
    for i in range(len(a)):  
        a[i] = [_ for _ in a[i].split()]  
  
def step1(a):    #delete subway, favourable and date  
    length = len(a)  
    b = a[0][1]  
    for i in range(length):  
        a[i].pop(6)  
        a[i].pop(4)  
        a[i].pop(1)  
    return b
```

预处理代码如上图所示。

## 4.3 数据分析

### 4.3.1 统计地铁站数量并获取地铁站名称

由于是 2015 年的数据，首先统计当时有多少个地铁站并获得当时的地铁站名称（从 1 号线到 16 号线按顺序输出）。输出结果如下图所示：

当时共有313个车站  
共康路地铁站  
彭浦新村地铁站  
上海南站地铁站  
常熟路地铁站  
莘庄地铁站  
中山北路地铁站  
陕西南路地铁站  
新闸路地铁站  
衡山路地铁站  
上海火车站地铁站  
人民广场地铁站  
富锦路地铁站  
宝安公路地铁站  
汶水路地铁站  
呼兰路地铁站

采用字典使站点名与地铁线路名称对应，再按 key 值排序后换回给主函数 Analysis。实现代码如下图所示：



```

def step2(a):  #get the name of subway
    length = len(a)
    dic = {}
    for i in range(length):
        # print(a[i][2])
        line, location = map(str, a[i][2].split("号线"))
        if int(line) in dic:
            dic[int(line)].append(location)
        else:
            dic[int(line)] = []
    # print(dic.keys())
    new_dic = {}
    for i in sorted(dic.keys()):
        dic[i] = list(set(dic[i]))
        new_dic[i] = dic[i]
        # print("{}号线:".format(i), end='')
        # print(dic[i])
    sum = 0
    for i in new_dic:
        # print(i, end=":")
        # print(new_dic[i])
        sum += len(new_dic[i])
    # print(sum)
    return new_dic, sum

```

### 4.3.2 统计当天客流量信息

根据地铁站名出现的次数统计各个站点当天的客流量。统计结果如下图所示：

地铁站名	客流量
人民广场	24911
上海火车站	22167
徐家汇	18140
静安寺	17779
陆家嘴	16578
中山公园	15948
南京东路	15241
陕西南路	14984
莘庄	14943
南京西路	11730
莲花路	11088
上海南站	11040
世纪大道	10530
漕河泾开发区	10513
宜山路	10423
淞虹路	10284
浦电路	9691
曹杨路	9648
娄山关路	9519

\* 作者简介： 印张悦，本科在读，yinzhangyue@126.com

用库函数 count 时由于是二重循环，处理大量数据时效率极低，我们发现这其实  $O(n)$  的复杂度即可实现，实现代码如下图所示：

```
def step3(a): #the number of people in each subway
    length = len(a)
    dic = {}
    data = []
    for i in range(length):
        line, location = map(str, a[i][2].split("号线"))
        data.append(location)
    station = set(data)
    # for i in station: #data.count(i) O(n^2)复杂度
    #     dic[i] = data.count(i)
    for i in station:
        dic[i] = 0
    for i in data: #O(n)复杂度
        dic[i] += 1
    # print(dic)
    return dic
```

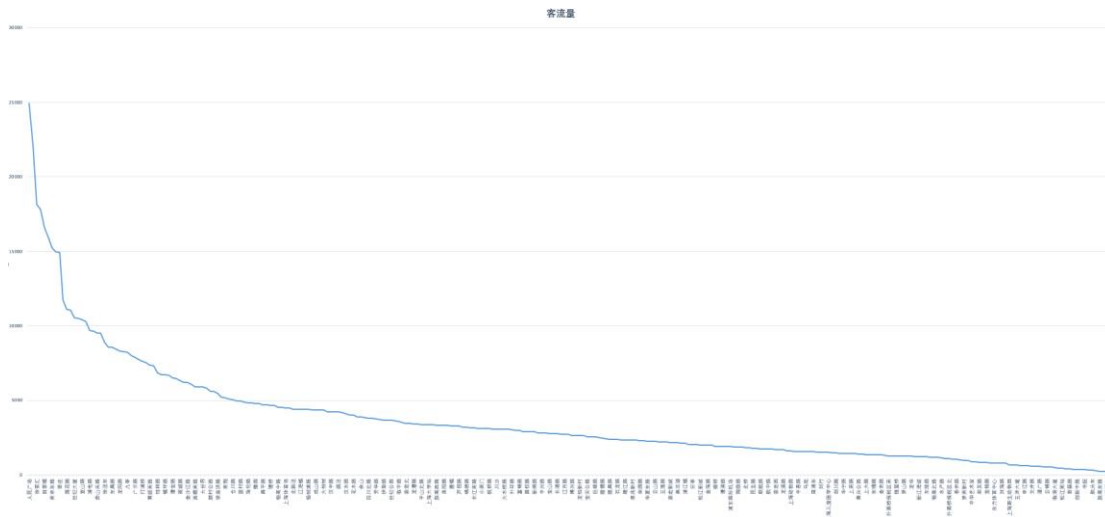
通过python自带的count实现，二重循环， $O(n^2)$ 复杂度

其实遍历一遍， $O(n)$ 即可实现

## 4.4 数据可视化

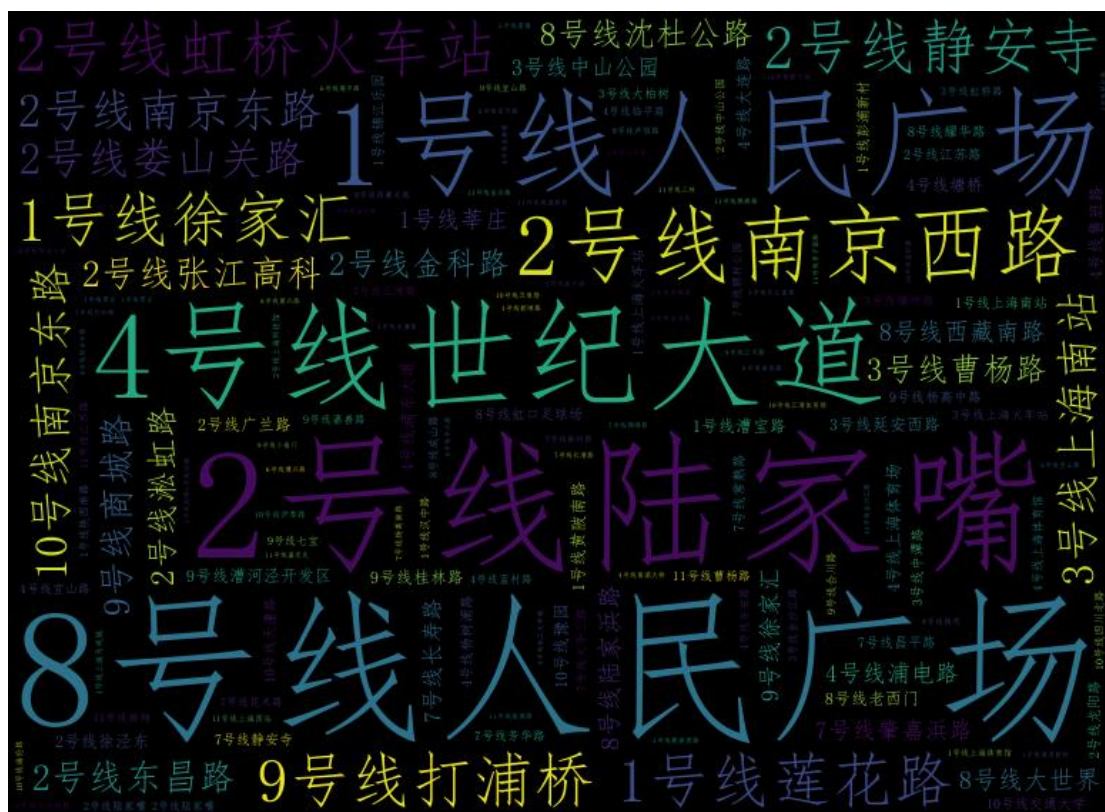
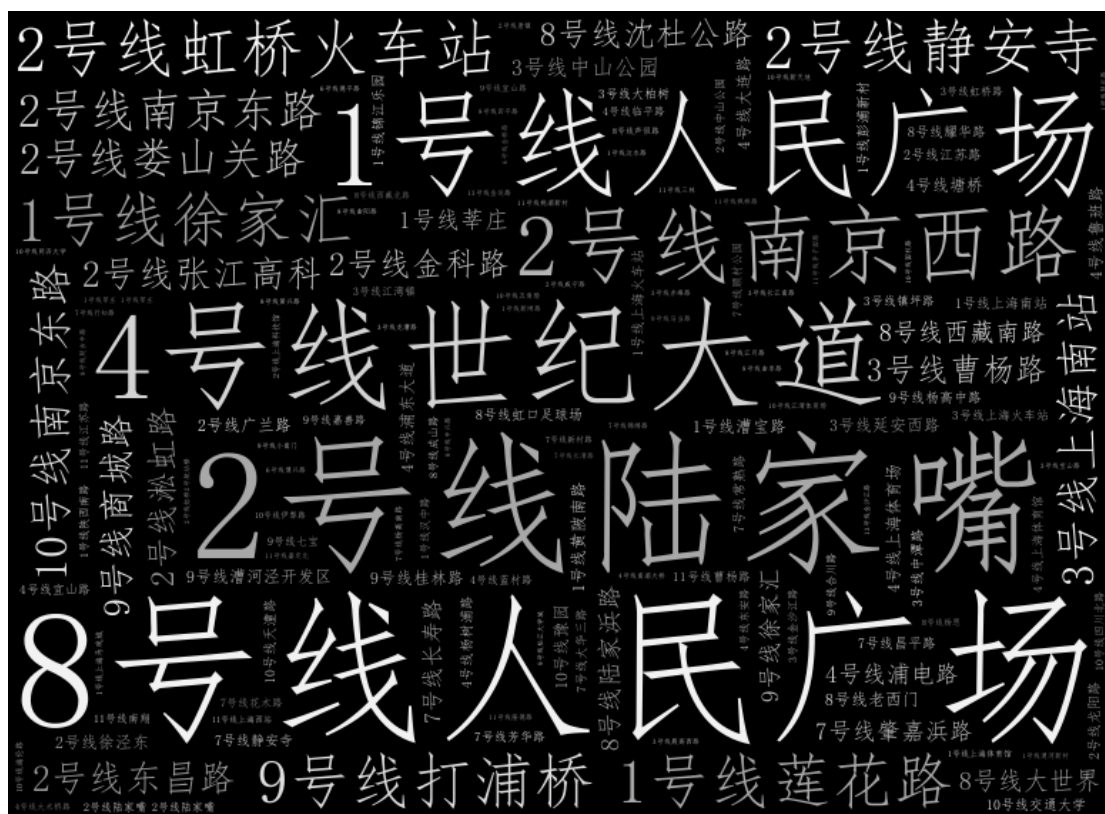
### 4.4.1 普通折线图可视化

折线图是一种很简洁的表现方式，但样本过多时显得十分冗杂。



### 4.4.2 wordcloud 词云可视化

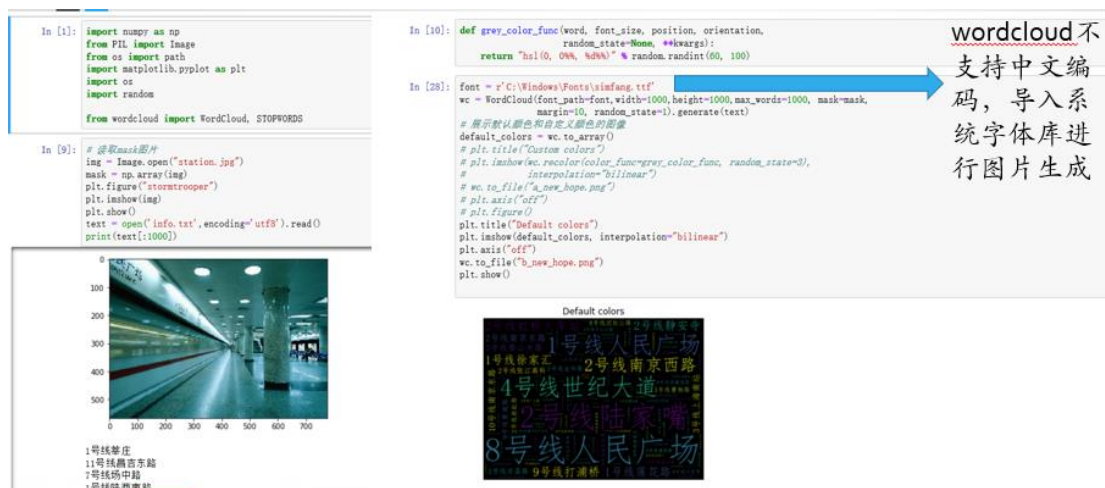
利用 Python 的 wordcloud 库生成可视化图片，字的大小越大说明出现的次数越多，非常简约直观。生成效果如下图所示：



实现代码见下图：

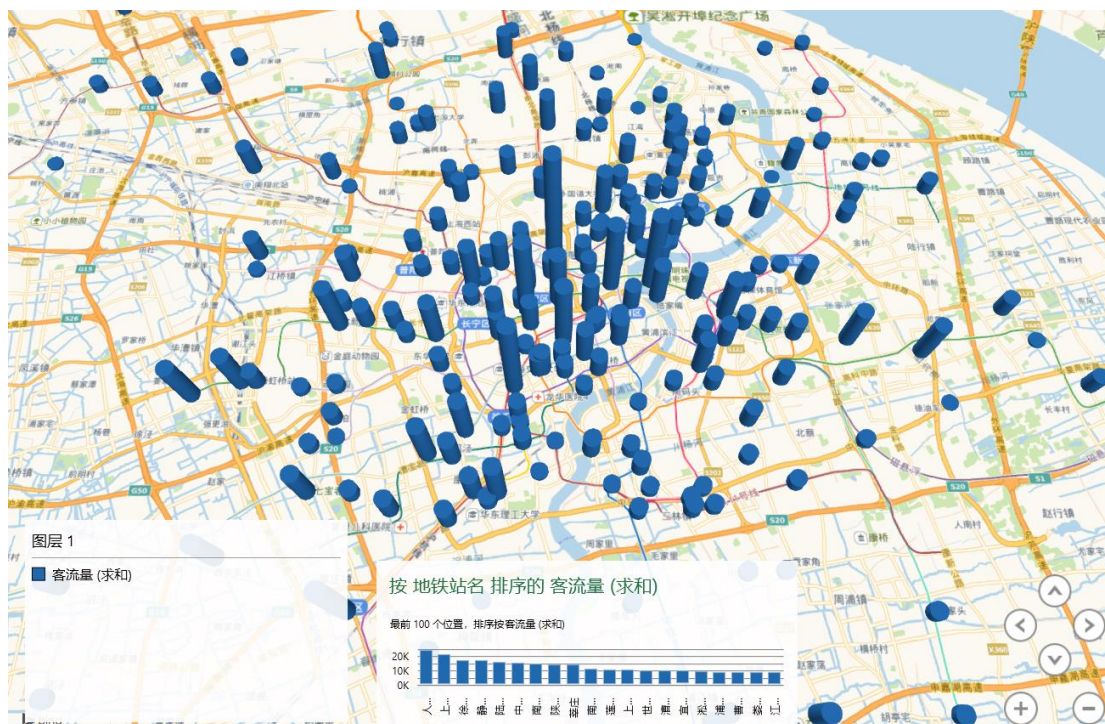
\* 作者简介: 印张悦, 本科在读, yinzhangyue@126.com





### 4.4.3 Excel 3D 可视化

Excel 有一个非常强大的功能就是建立三维地图，操作非常简单，生成的图表可以像电子地图一样移动放大缩小，生成效果如下：



### 4.5 地铁咨询软件的优化和局部加强线性回归数据清洗

在 2.2 小节中我们提到采用距离和站数推断时间的方法固定死板，而我们在现实生活中也很难讲清楚什么时候坐地铁最快，因为早晚高峰客流量大但是班次多，其他时间段客流量少班次也少，这个问题很难通过简单的因素分析回答。我们在 4.2.1 小节数据探索中提到通过卡号和时间相联系并进行局部加权线性回归可

\* 作者简介： 印张悦，本科在读，yinzhangyue@126.com

以预测不同时间段内的从某一站到某一站所需的时间。采用这样的分析手段这个问题就迎刃而解了。

### 4.5.1 地铁咨询软件的优化

根据上文的分析采用对数据集中卡号、站点和时间的分析利用线性回归可以准确预测出不同时间段内的从某一站到某一站所需的时间，根据用户所处的时间段告知用户所需的时间，灵活准确，展现出数据分析的 power。再通过清洗出的数据中费用这一栏得到从某站到某站的费用花费，基于实际，准确无误。

### 4.5.2 局部加强线性回归数据清洗

清洗需要解决的问题：

- 首先要清洗的数据量非常大，仅仅是一天当中的数据就有 100 多万条，而且复杂度是  $O(n^2)$ ，一般的计算机很难承受。
- 其次清洗出有效数据的困难较大，存在很多缺失数据，原因是刷卡没有被闸机记录。举个例子，如果一天当中往返就是有 4 次记录，如果其中一次未被记录则对应另一次也作废；如果有 2 次未被记录，而且恰好是一次出站和另一次的进站，那么时间会明显增大，这类数据也要被丢弃。
- 最后是要排除人为的不合格的数据，地铁里面有些快递小哥就负责在地铁里转运物资，这类记录的异常是人为引起的，不是我们想要的数，对于这类异常值要被清洗。

清洗过程中我们遇到了大量的异常数据，见下图：

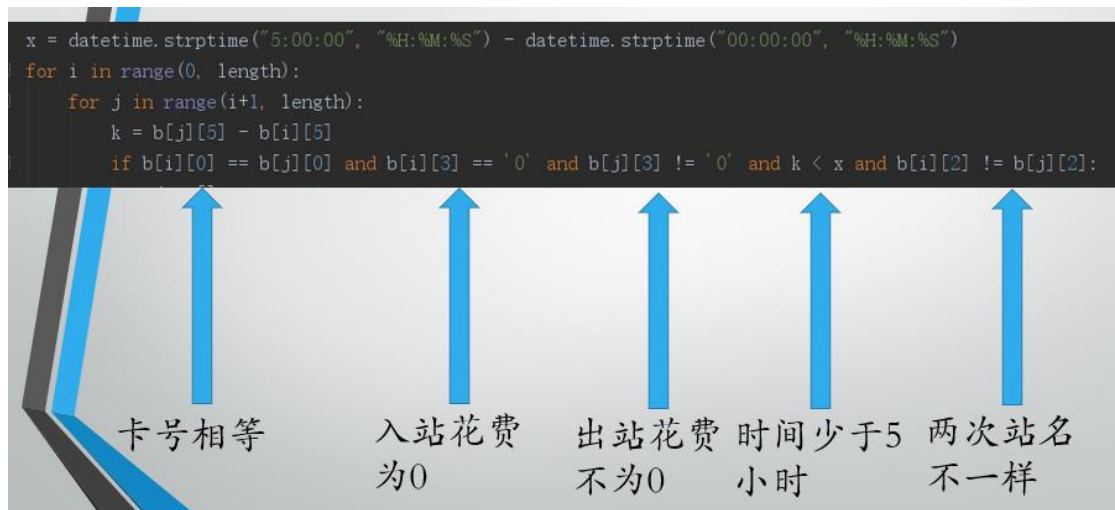
2003591684	6号线云山路	6号线云山路	8:04:50	17:57:46	9:52:56//异常数据
2702935490	4号线东安路	2号线虹桥2号航站楼	8:04:50	8:39:57	0:35:07
2202704911	3号线江杨北路	10号线龙柏新村	8:04:50	9:30:37	1:25:47
202824526	7号线杨高南路	10号线四川北路	8:04:50	8:44:29	0:39:39
2201628716	1号线通河新村	9号线打浦桥	8:04:50	8:54:13	0:49:23
3003336600	8号线市光路	2号线娄山关路	8:04:50	9:01:56	0:57:06
3000242452	1号线彭浦新村	4号线浦电路	8:04:50	8:54:48	0:49:58
2303716877	9号线松江大学城	9号线漕河泾开发区	8:04:50	8:41:28	0:36:38
2903171195	3号线长江南路	2号线南京西路	8:04:50	8:47:59	0:43:09
3100404009	6号线金桥路	6号线外高桥保税区南	8:04:50	8:29:59	0:25:09
2902392393	2号线唐镇	8号线沈杜公路	8:04:50	9:51:00	1:46:10
2001546965	10号线五角场	1号线汉中路	8:04:50	8:41:58	0:37:08
2502191937	2号线娄山关路	2号线徐泾东	8:04:51	8:32:20	0:27:29
3100984486	11号线桃浦新村	1号线徐家汇	8:04:51	8:40:45	0:35:54
300859670	7号线岚皋路	9号线桂林路	8:04:51	8:39:45	0:34:54
2002243009	8号线市光路	1号线新闻路	8:04:51	8:46:48	0:41:57
2003732138	1号线通河新村	1号线通河新村	8:04:51	19:21:39	11:16:48//异常数据
2803543362	7号线锦绣路	7号线芳华路	8:04:51	18:18:35	10:13:44
301842915	7号线美兰湖	7号线静安寺	8:04:51	8:54:41	0:49:50
2900458627	3号线大柏树	3号线宜山路	8:04:51	8:45:56	0:41:05
2004170750	2号线娄山关路	3号线石龙路	8:04:51	8:40:58	0:36:07
2601486529	1号线人民广场	11号线武威路	8:04:51	8:40:07	0:35:16
3002111286	8号线曲阜路	2号线金科路	8:04:51	8:45:37	0:40:46
3102088478	2号线威宁路	2号线威宁路	8:04:51	17:39:03	9:34:12
2401550387	7号线长寿路	11号线昌吉东路	8:04:52	8:57:44	0:52:52
2202482286	6号线洲海路	9号线漕河泾开发区	8:04:52	9:11:17	1:06:25
2801384784	11号线御桥	10号线伊犁路	8:04:52	8:50:07	0:45:15
2303801250	3号线殷高西路	3号线上海火车站	8:04:52	8:31:37	0:26:45
2803143926	6号线东靖路	6号线东靖路	8:04:52	20:04:20	11:59:28//异常数据
2104053777	3号线东宝兴路	3号线延安西路	8:04:52	8:34:02	0:29:10

第一次清洗产生的大量异常数据

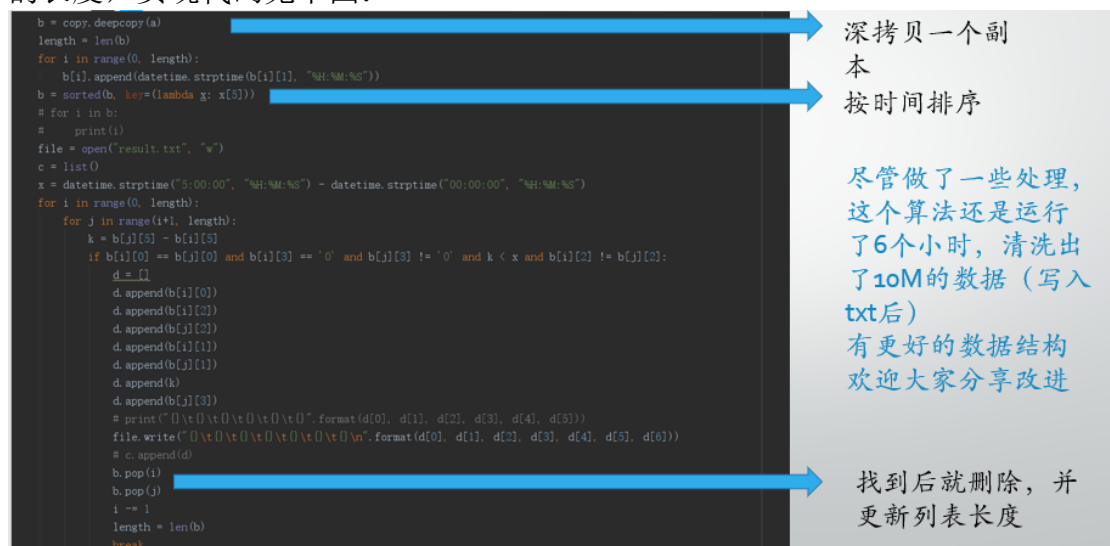
于是我们对判断条件做了不断地改进，对所用时间和两次的站名都做出了限制，以便得到有效数据：

清洗主要代码见下图：

\* 作者简介： 印张悦，本科在读，yinzhangyue@126.com



为了提高算法效率，在循环的同时对列表中的数据不断删减并更新数据集的长度，实现代码见下图：



最后得到了较好的清洗结果，部分结果如下图所示：



卡号	进站地点	出战地点	进站时间	出站时间	花费时间	花费金额
3300421029	10号线国权路	10号线海伦路	7:37:07	7:46:26	0:09:19	3
2802090516	10号线国权路	10号线海伦路	8:11:50	8:24:49	0:12:59	3
2104345058	10号线国权路	10号线海伦路	8:20:44	8:31:02	0:10:18	3
2303623507	10号线国权路	10号线海伦路	9:06:16	9:16:59	0:10:43	3
740485847	10号线国权路	10号线海伦路	9:41:40	9:51:18	0:09:38	3
2200754501	10号线国权路	10号线海伦路	11:08:27	11:22:01	0:13:34	3
3102471686	10号线国权路	10号线虹桥1号航站楼	7:37:49	8:27:01	0:49:12	5
500522122	10号线国权路	10号线虹桥路	6:17:12	6:49:24	0:32:12	4
300089616	10号线国权路	10号线虹桥路	7:11:49	7:43:46	0:31:57	4
3102611952	10号线国权路	10号线虹桥路	8:59:16	9:33:47	0:34:31	4
701516124	10号线国权路	10号线虹桥路	9:42:41	10:14:43	0:32:02	4
2702784630	10号线国权路	10号线虹桥路	10:26:12	11:00:09	0:33:57	4
2103836678	10号线国权路	10号线江湾体育场	7:41:52	7:46:17	0:04:25	3
600340681	10号线国权路	10号线交通大学	6:41:07	7:10:57	0:29:50	4
3003764958	10号线国权路	10号线交通大学	7:05:01	7:37:41	0:32:40	4
2803308175	10号线国权路	10号线交通大学	7:09:30	7:41:35	0:32:05	4
2500891073	10号线国权路	10号线交通大学	9:42:26	10:12:25	0:29:59	4
2601006546	10号线国权路	10号线龙柏新村	7:40:22	8:31:08	0:50:46	5
2302327522	10号线国权路	10号线龙柏新村	9:42:49	10:31:15	0:48:26	4
3800023624	10号线国权路	10号线龙溪路	7:57:06	8:37:19	0:40:13	5
2603294362	10号线国权路	10号线南京东路	7:09:28	7:27:59	0:18:31	4
2301771514	10号线国权路	10号线南京东路	7:43:56	8:05:22	0:21:26	4
2801452761	10号线国权路	10号线南京东路	7:51:43	8:09:19	0:17:36	4
2902469392	10号线国权路	10号线南京东路	7:54:10	8:14:43	0:20:33	4
3600006726	10号线国权路	10号线南京东路	7:58:02	8:20:03	0:22:01	4
2903787147	10号线国权路	10号线南京东路	8:06:49	8:26:37	0:19:48	4
2903342533	10号线国权路	10号线南京东路	8:08:54	8:29:35	0:20:41	4
2103712436	10号线国权路	10号线南京东路	8:09:00	8:28:39	0:19:39	4
2203392064	10号线国权路	10号线南京东路	8:09:37	8:26:30	0:16:53	4
2803427312	10号线国权路	10号线南京东路	8:18:45	8:38:01	0:19:16	4
2301160246	10号线国权路	10号线南京东路	8:35:55	8:55:28	0:19:33	4
2001234038	10号线国权路	10号线南京东路	8:38:42	8:59:47	0:21:05	4
3002940490	10号线国权路	10号线南京东路	8:58:07	9:18:22	0:20:15	4

\* 作者简介： 印张悦，本科在读，yinzhangyue@126.com

## 5 局部加权线性回归

本文在建模部分已经提到了算法的主要思想这里我们主要讨论局部加权线性回归在客流量和不同时间段的时间花费上的实现。

### 5.1 局部加权线性预测客流量

客流量在不同时间段的数据经过简单清洗即可得到，这里以数据量最大的人民广场站为例，以下是数据清洗代码：

```
if __name__ == '__main__':
    f = open("liner_data.txt", "r")
    a = list()
    for line in f:
        b = line[:-4]
        a.append(float(b))

    max = 0
    min = 20
    length = len(a)
    # for i in range(0, length):
    #     if a[i] > max:
    #         max = a[i]
    #     if a[i] < min:
    #         min = a[i]
    # print(max, min)
    b = list(0 for x in range(0, 18))
    for i in range(0, length):
        b[int(a[i])-5] += 1
    f2 = open("test_.txt", "w")
    length = len(b)
    for i in range(0, length):
        f2.write("{}\t{}\n".format(i+5, b[i]))
    f2.close()
```

局部加权线性回归算法中需要用到 numpy 进行矩阵运算和 matplotlib 函数进行绘图。我们将客流量划分成 5 点到 22 点 18 个时间段，以较粗的颗粒度拟合曲线，并使用训练样本自身进行训练效果的测试。

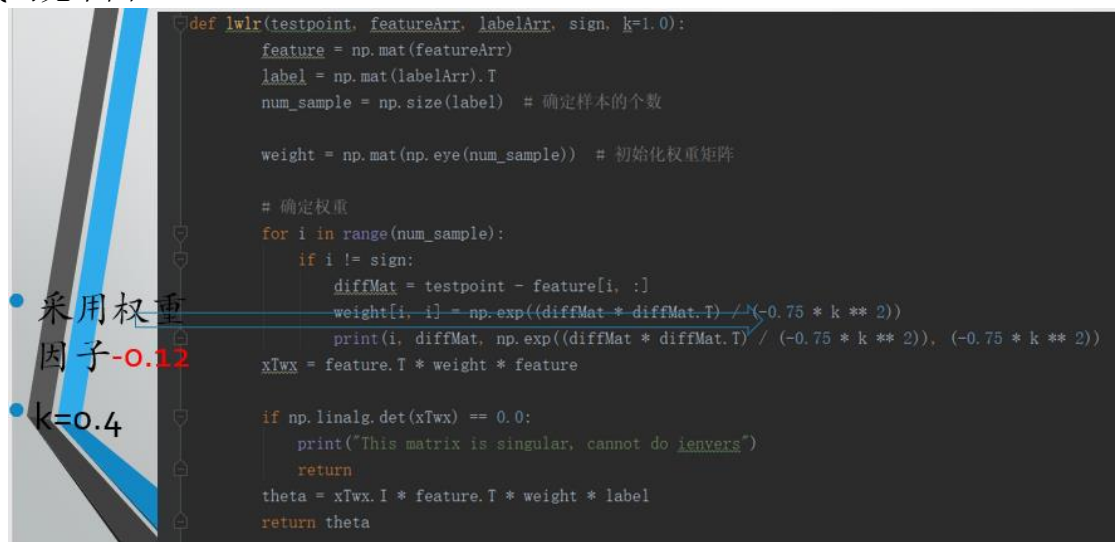


```

# 使用训练样本来进行测试性能
def lwlrtest(featureArr, labelArr, k):
    feature = np.mat(featureArr)
    label = np.mat(labelArr)
    num_sample = np.size(feature[:, 1])
    predict = np.zeros(num_sample)
    # print(predict)
    for i in range(num_sample):
        testpoint = feature[i, :]
        theta = lwlr(testpoint, feature, label, i, k)
        predict[i] = testpoint * theta
    return predict

```

在较粗的颗粒度下我们采用权重因子-0.12 得到了较好的拟合效果，主要拟合代码见下图：



```

def lwlr(testpoint, featureArr, labelArr, sign, k=1.0):
    feature = np.mat(featureArr)
    label = np.mat(labelArr).T
    num_sample = np.size(label) # 确定样本的个数

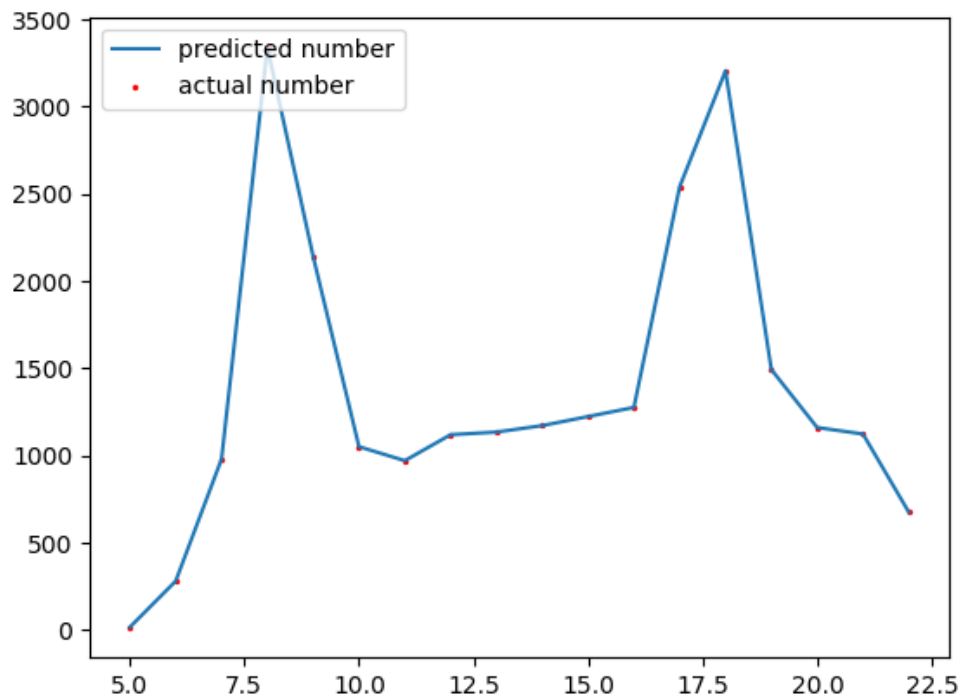
    weight = np.mat(np.eye(num_sample)) # 初始化权重矩阵

    # 确定权重
    for i in range(num_sample):
        if i != sign:
            diffMat = testpoint - feature[i, :]
            weight[i, i] = np.exp((diffMat * diffMat.T) / (-0.75 * k ** 2))
            print(i, diffMat, np.exp((diffMat * diffMat.T) / (-0.75 * k ** 2)), (-0.75 * k ** 2))
    xTwX = feature.T * weight * feature

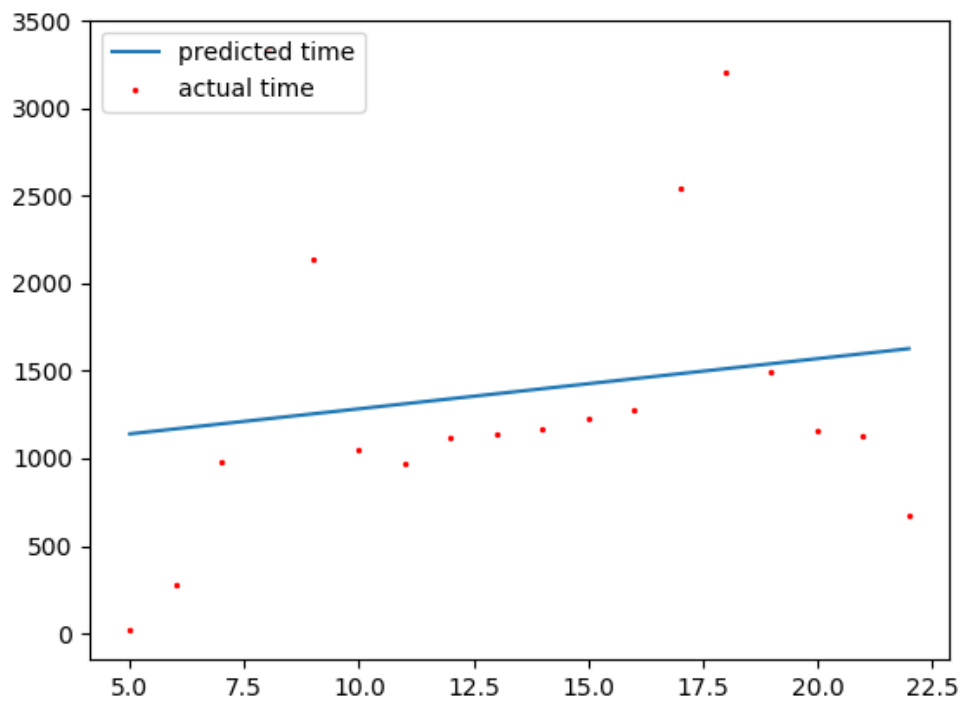
    if np.linalg.det(xTwX) == 0.0:
        print("This matrix is singular, cannot do ienvers")
        return
    theta = xTwX.I * feature.T * weight * label
    return theta

```

拟合结果见下图：



我们看到采用局部加权线性回归的拟合结果非常出色，笔者也用线性回归尝试了一下，读者可以比较一下两者的差距，线性回归图如下：



# 5.2 局部加权线性预测不同时间段的时间花费

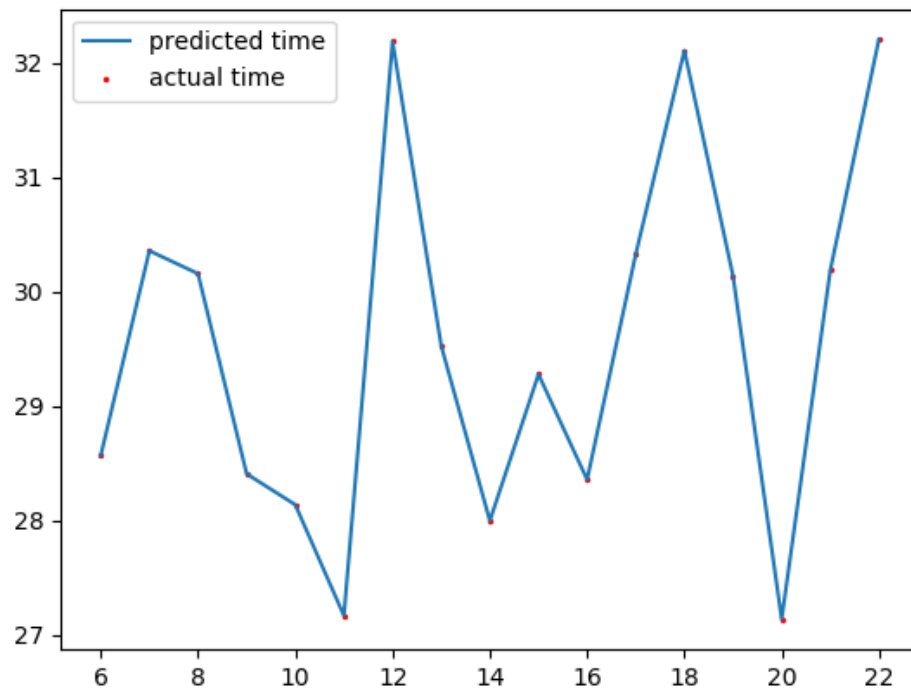
根据清洗出的数据我们便可以进行不同时间段的时间花费的预测，鉴于从人民广场站到张江高科站的样本集较多，我们便以次为例进行数据分析。

局部加权线性回归的代码与上面大同小异，这里着重介绍颗粒度大和颗粒度小的情况下权重因子的调整。

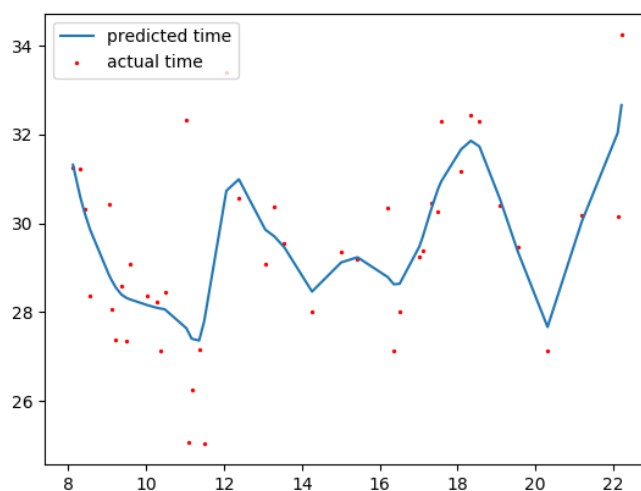
清洗后的数据如下：

	A	B	C	D	E
1	1号线人民广场	2号线张江高科	6:07:53	6:36:36	0:28:43
2	1号线人民广场	2号线张江高科	6:47:23	7:15:35	0:28:12
3	1号线人民广场	2号线张江高科	6:47:25	7:17:20	0:29:55
4	1号线人民广场	2号线张江高科	7:00:15	7:29:30	0:29:15
5	1号线人民广场	2号线张江高科	7:08:26	7:45:46	0:37:20
6	1号线人民广场	2号线张江高科	7:08:41	7:36:26	0:27:45
7	1号线人民广场	2号线张江高科	7:15:24	7:44:54	0:29:30
8	1号线人民广场	2号线张江高科	7:26:55	7:55:20	0:28:25
9	1号线人民广场	2号线张江高科	7:39:21	8:09:53	0:30:32
10	1号线人民广场	2号线张江高科	7:55:06	8:26:31	0:31:25
11	1号线人民广场	2号线张江高科	8:07:21	8:39:55	0:32:34
12	1号线人民广场	2号线张江高科	8:09:57	8:37:00	0:27:03
13	1号线人民广场	2号线张江高科	8:12:56	8:44:21	0:31:25
14	1号线人民广场	2号线张江高科	8:31:33	9:02:56	0:31:23
15	1号线人民广场	2号线张江高科	8:42:44	9:13:15	0:30:31
16	1号线人民广场	2号线张江高科	8:56:02	9:24:40	0:28:38
17	1号线人民广场	2号线张江高科	9:04:12	9:34:55	0:30:43
18	1号线人民广场	2号线张江高科	9:11:55	9:40:02	0:28:07
19	1号线人民广场	2号线张江高科	9:22:23	9:50:01	0:27:38
20	1号线人民广场	2号线张江高科	9:36:07	10:05:05	0:28:58
21	1号线人民广场	2号线张江高科	9:49:21	10:16:55	0:27:34
22	1号线人民广场	2号线张江高科	9:59:21	10:28:29	0:29:08
23	1号线人民广场	2号线张江高科	10:03:12	10:31:49	0:28:37
24	1号线人民广场	2号线张江高科	10:26:55	10:55:17	0:28:22
25	1号线人民广场	2号线张江高科	10:36:19	11:03:31	0:27:12
26	1号线人民广场	2号线张江高科	10:48:09	11:16:55	0:28:46
27	1号线人民广场	2号线张江高科	11:03:26	11:35:59	0:32:33
28	1号线人民广场	2号线张江高科	11:10:29	11:35:36	0:25:07
29	1号线人民广场	2号线张江高科	11:17:55	11:44:21	0:26:26
30	1号线人民广场	2号线张江高科	11:36:05	12:03:22	0:27:17
31	1号线人民广场	2号线张江高科	11:49:01	12:14:05	0:25:04
32	1号线人民广场	2号线张江高科	12:06:11	12:39:52	0:33:41
33	1号线人民广场	2号线张江高科	12:38:21	13:09:17	0:30:56
34	1号线人民广场	2号线张江高科	13:06:55	13:36:04	0:29:09

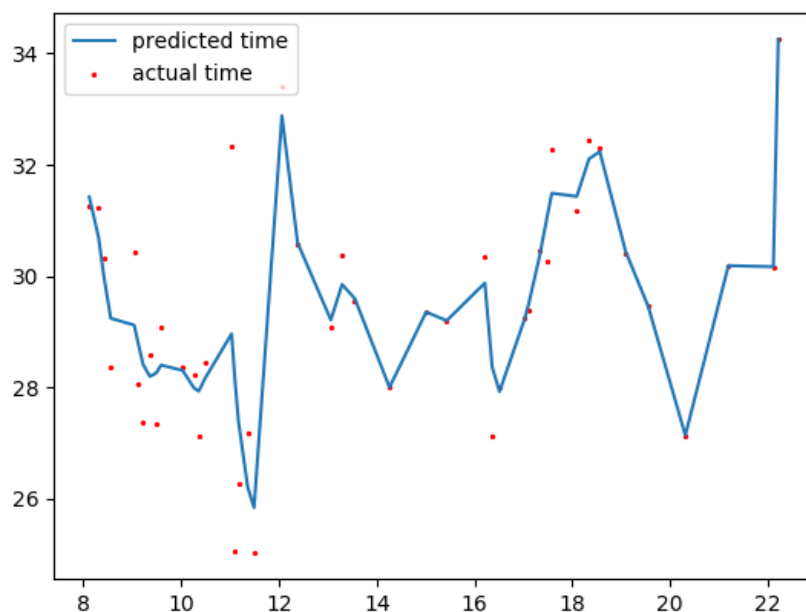
我们将时间分成 6 点到 22 点 15 个范围，对其求均值，得到 15 个对应的数据，这种较粗的颗粒度在权重因子-0.32 的条件下得到了很好的拟合，拟合结果如下：



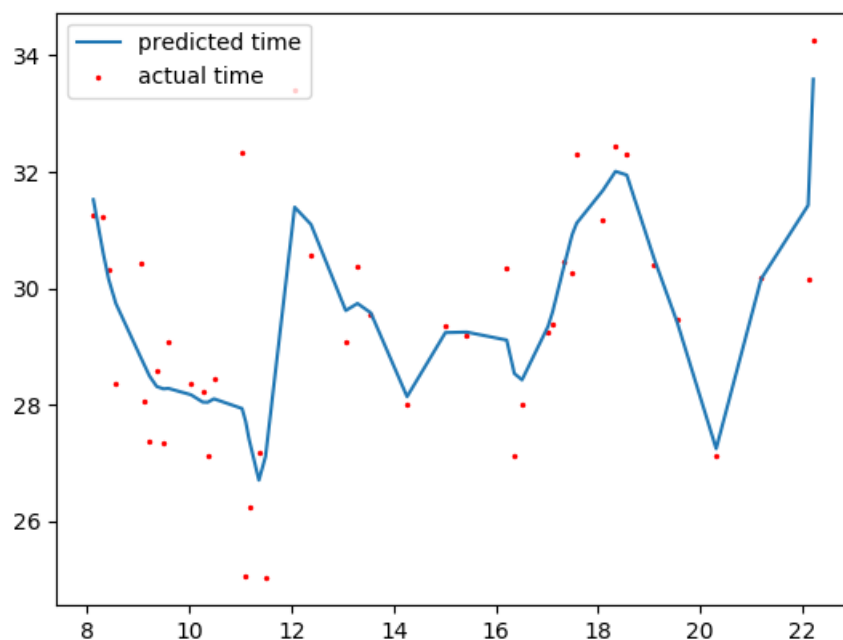
而后，我们尝试对所有样本点进行拟合，这种较细的颗粒的对我们的权重因子提出了挑战，权重因子过小会导致欠拟合，权重因子-0.32 的拟合结果如下：



而权重因子过小会导致过拟合，拟合结果过于取决于某些特殊值，权重因子-0.08 的拟合结果如下：



最后我们采用权重因子-0.18 进行拟合，得到了较好的拟合效果：



最后，我们通过拟合出的数据，将其写成代码用于预测从人民广场站到张江高科站的时间花费，生成结果如下：

\* 作者简介： 印张悦，本科在读， yinzhangyue@126.com

\*\*\*用加权线性回归得到的从人民广场站到张江高科站的花费时间\*\*\*  
请输入出发时间:  
当前时间  
预计用时32分钟

### 5.3 结合局部加权线性预测对软件的改进

通过拟合出的数据，我们可以对所有的路线在不同时间段所需的时间进行预测，告知用户这个时间点到达目的地所需的时间和预计到达时间。如果能有更多的数据集，我们就能不仅仅局限在一天内，能够对工作日、双休日和节假日的客流量和出行时间建立不同的算法模型进行分析和预测，笔者相信用到的算法模型应该依旧是基于局部加权线性回归的，因此局部加权线性回归在不同时间段的时间预测方面可以说是大展身手。

## 6 总结

本文主要分析了数据科学的主要过程，包括数据的清洗和数据的分析，机器学习中的线性回归和局部加权线性回归，本文主要使用局部加权线性回归来预测客流量随时间的变化和不同时间段相同行程所花费的时间，在构建咨询软件时使用到了最短路径算法以及基于实际情况对其做出的改进版本。最后结合数据分析来弥补软件的不足，笔者相信通过数据分析是一个解决不同时间段相同行程所需时间问题的很好方案，由此展现出数据分析的意义和数据的 power。相信大数据在解决多种因素造成的复杂问题上一定能大展身手。

另外，相信读者一定能想到，本文所进行的分析和软件的构想不仅仅适用于地铁，还适用于高铁、火车、飞机、公交车等，如果能获取这些数据集并对其进行有效分析，根据训练出来的算法模型设计出实用的软件，这能大大方便用户的出行，如果你身边有这样的数据集不妨可以通过笔者的思路试一试，本文所有实现均配套相应的代码和数据，供读者参考和使用。本文有不足和纰漏之处，望读者多多指正，有更好的算法也请多多分享。

印张悦

2019 年 1 月 6 日星期日

## 7 附录

### 探索：python 网络爬虫

本文曾提到利用爬虫爬取了地铁站坐标，由于不是本文重点，放进附录进行说明。

所需库函数如下：

```
from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from time import sleep

from tkinter import *

import pandas as pd

import numpy as np
```

初始数据如下：

1	地铁站	经度	纬度
2	0 莘庄		
3	1 外环路		
4	2 莲花路		
5	3 锦江乐园		
6	4 上海南站		
7	5 漕宝路		
8	6 上海体育馆		
9	7 徐家汇		
10	8 衡山路		
11	9 常熟路		
12	10 陕西南路		
13	11 黄陂南路		
14	12 人民广场		
15	13 新闸路		
16	14 汉中路		
17	15 上海火车站		
18	16 中山北路		
19	17 延长路		
20	18 上海马戏城		
21	19 汶水路		
22	20 彭浦新村		
23	21 共康路		
24	22 通河新村		
25	23 呼兰路		
26	24 共富新村		
27	25 宝安公路		
28	26 友谊西路		
29	27 富锦路		
30	28 徐泾东		
31	29 虹桥火车站		
32	30 虹桥2号航站楼		
33	31 淞虹路		
34	32 北新泾		

\* 作者简介： 印张悦，本科在读，yinzhangyue@126.com



爬取代码如下，这里增加了”地铁站”三字，便于得到精确的经纬度，不然可能存在偏差较大甚至找错地点的情况，调用高德地图开放 API 平台爬取，代码如下：

```
In [18]: driver = webdriver.Chrome()
driver.get('https://lbs.amap.com/console/show/picker')
d = 2
record_position = ""
record_address = ""
for i in data.index:
    address = data["地铁站"][i]
    address=address+"地铁站"
    #删除这个关键字对结果影响
    num = 0
    while True:
        try:
            driver.find_element_by_id('txtSearch').send_keys(address)
            driver.find_element_by_id('txtSearch').send_keys(Keys.ENTER)
            if d == 2:
                sleep(2)
            sleep(0.2)
            driver.find_element_by_class_name('picker-copy').click()
            driver.find_element_by_id('txtSearch').send_keys(Keys.CONTROL + 'a')
            driver.find_element_by_id('txtSearch').send_keys(Keys.DELETE)
            position = Tk().clipboard_get()
            if num == 10:
                position = "0,0"
            if position == record_position and address != record_address:
                num += 1
                continue
            long, la = map(float, position.split(','))
            data["经纬度"][i] = long
            data["纬度"][i] = la
            record_position = position
            record_address = address
            break
        except TclError:
            continue
    if d % 100 == 0:
        print(d)
    d += 1
    if data["经纬度"][i] == 0:
        print(data["地铁站"][i])

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:28: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
2000
2100
2200
2300
2400
2500
2600
2700
2800
2900
3000
3100
3200
3300
3400
3500
3600
3700
3800
3900
4000
4100
4200
4300
4400
4500
4600
4700
4800
4900
5000
5100
5200
5300
5400
5500
5600
5700
5800
5900
6000
6100
6200
6300
6400
6500
6600
6700
6800
6900
7000
7100
7200
7300
7400
7500
7600
7700
7800
7900
8000
8100
8200
8300
8400
8500
8600
8700
8800
8900
9000
9100
9200
9300
9400
9500
9600
9700
9800
9900
10000
10100
10200
10300
10400
10500
10600
10700
10800
10900
11000
11100
11200
11300
11400
11500
11600
11700
11800
11900
12000
12100
12200
12300
12400
12500
12600
12700
12800
12900
13000
13100
13200
13300
13400
13500
13600
13700
13800
13900
14000
14100
14200
14300
14400
14500
14600
14700
14800
14900
15000
15100
15200
15300
15400
15500
15600
15700
15800
15900
16000
16100
16200
16300
16400
16500
16600
16700
16800
16900
17000
17100
17200
17300
17400
17500
17600
17700
17800
17900
18000
18100
18200
18300
18400
18500
18600
18700
18800
18900
19000
19100
19200
19300
19400
19500
19600
19700
19800
19900
20000
20100
20200
20300
20400
20500
20600
20700
20800
20900
21000
21100
21200
21300
21400
21500
21600
21700
21800
21900
22000
22100
22200
22300
22400
22500
22600
22700
22800
22900
23000
23100
23200
23300
23400
23500
23600
23700
23800
23900
24000
24100
24200
24300
24400
24500
24600
24700
24800
24900
25000
25100
25200
25300
25400
25500
25600
25700
25800
25900
26000
26100
26200
26300
26400
26500
26600
26700
26800
26900
27000
27100
27200
27300
27400
27500
27600
27700
27800
27900
28000
28100
28200
28300
28400
28500
28600
28700
28800
28900
29000
29100
29200
29300
29400
29500
29600
29700
29800
29900
30000
30100
30200
30300
30400
30500
30600
30700
30800
30900
31000
31100
31200
31300
31400
31500
31600
31700
31800
31900
32000
32100
32200
32300
32400
32500
32600
32700
32800
32900
33000
33100
33200
33300
33400
33500
33600
33700
33800
33900
34000
34100
34200
34300
34400
34500
34600
34700
34800
34900
35000
35100
35200
35300
35400
35500
35600
35700
35800
35900
36000
36100
36200
36300
36400
36500
36600
36700
36800
36900
37000
37100
37200
37300
37400
37500
37600
37700
37800
37900
38000
38100
38200
38300
38400
38500
38600
38700
38800
38900
39000
39100
39200
39300
39400
39500
39600
39700
39800
39900
40000
40100
40200
40300
40400
40500
40600
40700
40800
40900
41000
41100
41200
41300
41400
41500
41600
41700
41800
41900
42000
42100
42200
42300
42400
42500
42600
42700
42800
42900
43000
43100
43200
43300
43400
43500
43600
43700
43800
43900
44000
44100
44200
44300
44400
44500
44600
44700
44800
44900
45000
45100
45200
45300
45400
45500
45600
45700
45800
45900
46000
46100
46200
46300
46400
46500
46600
46700
46800
46900
47000
47100
47200
47300
47400
47500
47600
47700
47800
47900
48000
48100
48200
48300
48400
48500
48600
48700
48800
48900
49000
49100
49200
49300
49400
49500
49600
49700
49800
49900
50000
50100
50200
50300
50400
50500
50600
50700
50800
50900
51000
51100
51200
51300
51400
51500
51600
51700
51800
51900
52000
52100
52200
52300
52400
52500
52600
52700
52800
52900
53000
53100
53200
53300
53400
53500
53600
53700
53800
53900
54000
54100
54200
54300
54400
54500
54600
54700
54800
54900
55000
55100
55200
55300
55400
55500
55600
55700
55800
55900
56000
56100
56200
56300
56400
56500
56600
56700
56800
56900
57000
57100
57200
57300
57400
57500
57600
57700
57800
57900
58000
58100
58200
58300
58400
58500
58600
58700
58800
58900
59000
59100
59200
59300
59400
59500
59600
59700
59800
59900
60000
60100
60200
60300
60400
60500
60600
60700
60800
60900
61000
61100
61200
61300
61400
61500
61600
61700
61800
61900
62000
62100
62200
62300
62400
62500
62600
62700
62800
62900
63000
63100
63200
63300
63400
63500
63600
63700
63800
63900
64000
64100
64200
64300
64400
64500
64600
64700
64800
64900
65000
65100
65200
65300
65400
65500
65600
65700
65800
65900
66000
66100
66200
66300
66400
66500
66600
66700
66800
66900
67000
67100
67200
67300
67400
67500
67600
67700
67800
67900
68000
68100
68200
68300
68400
68500
68600
68700
68800
68900
69000
69100
69200
69300
69400
69500
69600
69700
69800
69900
70000
70100
70200
70300
70400
70500
70600
70700
70800
70900
71000
71100
71200
71300
71400
71500
71600
71700
71800
71900
72000
72100
72200
72300
72400
72500
72600
72700
72800
72900
73000
73100
73200
73300
73400
73500
73600
73700
73800
73900
74000
74100
74200
74300
74400
74500
74600
74700
74800
74900
75000
75100
75200
75300
75400
75500
75600
75700
75800
75900
76000
76100
76200
76300
76400
76500
76600
76700
76800
76900
77000
77100
77200
77300
77400
77500
77600
77700
77800
77900
78000
78100
78200
78300
78400
78500
78600
78700
78800
78900
79000
79100
79200
79300
79400
79500
79600
79700
79800
79900
80000
80100
80200
80300
80400
80500
80600
80700
80800
80900
81000
81100
81200
81300
81400
81500
81600
81700
81800
81900
82000
82100
82200
82300
82400
82500
82600
82700
82800
82900
83000
83100
83200
83300
83400
83500
83600
83700
83800
83900
84000
84100
84200
84300
84400
84500
84600
84700
84800
84900
85000
85100
85200
85300
85400
85500
85600
85700
85800
85900
86000
86100
86200
86300
86400
86500
86600
86700
86800
86900
87000
87100
87200
87300
87400
87500
87600
87700
87800
87900
88000
88100
88200
88300
88400
88500
88600
88700
88800
88900
89000
89100
89200
89300
89400
89500
89600
89700
89800
89900
90000
90100
90200
90300
90400
90500
90600
90700
90800
90900
91000
91100
91200
91300
91400
91500
91600
91700
91800
91900
92000
92100
92200
92300
92400
92500
92600
92700
92800
92900
93000
93100
93200
93300
93400
93500
93600
93700
93800
93900
94000
94100
94200
94300
94400
94500
94600
94700
94800
94900
95000
95100
95200
95300
95400
95500
95600
95700
95800
95900
96000
96100
96200
96300
96400
96500
96600
96700
96800
96900
97000
97100
97200
97300
97400
97500
97600
97700
97800
97900
98000
98100
98200
98300
98400
98500
98600
98700
98800
98900
99000
99100
99200
99300
99400
99500
99600
99700
99800
99900
100000
100100
100200
100300
100400
100500
100600
100700
100800
100900
101000
101100
101200
101300
101400
101500
101600
101700
101800
101900
102000
102100
102200
102300
102400
102500
102600
102700
102800
102900
103000
103100
103200
103300
103400
103500
103600
103700
103800
103900
104000
104100
104200
104300
104400
104500
104600
104700
104800
104900
105000
105100
105200
105300
105400
105500
105600
105700
105800
105900
106000
106100
106200
106300
106400
106500
106600
106700
106800
106900
107000
107100
107200
107300
107400
107500
107600
107700
107800
107900
108000
108100
108200
108300
108400
108500
108600
108700
108800
108900
109000
109100
109200
109300
109400
109500
109600
109700
109800
109900
110000
110100
110200
110300
110400
110500
110600
110700
110800
110900
111000
111100
111200
111300
111400
111500
111600
111700
111800
111900
112000
112100
112200
112300
112400
112500
112600
112700
112800
112900
113000
113100
113200
113300
113400
113500
113600
113700
113800
113900
114000
114100
114200
114300
114400
114500
114600
114700
114800
114900
115000
115100
115200
115300
115400
115500
115600
115700
115800
115900
116000
116100
116200
116300
116400
116500
116600
116700
116800
116900
117000
117100
117200
117300
117400
117500
117600
117700
117800
117900
118000
118100
118200
118300
118400
118500
118600
118700
118800
118900
119000
119100
119200
119300
119400
119500
119600
119700
119800
119900
120000
120100
120200
120300
120400
120500
120600
120700
120800
120900
121000
121100
121200
121300
121400
121500
121600
121700
121800
121900
122000
122100
122200
122300
122400
122500
122600
122700
122800
122900
123000
123100
123200
123300
123400
123500
123600
123700
123800
123900
124000
124100
124200
124300
124400
124500
124600
124700
124800
124900
125000
125100
125200
125300
125400
125500
125600
125700
125800
125900
126000
126100
126200
126300
126400
126500
126600
126700
126800
126900
127000
127100
127200
127300
127400
127500
127600
127700
127800
127900
128000
128100
128200
128300
128400
128500
128600
128700
128800
128900
129000
129100
129200
129300
129400
129500
129600
129700
129800
129900
130000
130100
130200
130300
130400
130500
130600
130700
130800
130900
131000
131100
131200
131300
131400
131500
131600
131700
131800
131900
132000
132100
132200
132300
132400
132500
132600
132700
132800
132900
133000
133100
133200
133300
133400
133500
133600
133700
133800
133900
134000
134100
134200
134300
134400
134500
134600
134700
134800
134900
135000
135100
135200
135300
135400
135500
135600
135700
135800
135900
136000
136100
136200
136300
136400
136500
136600
136700
136800
136900
137000
137100
137200
137300
137400
137500
137600
137700
137800
137900
138000
138100
138200
138300
138400
138500
138600
138700
138800
138900
139000
139100
139200
139300
139400
139500
139600
139700
139800
139900
140000
140100
140200
140300
140400
140500
140600
140700
140800
140900
141000
141100
141200
141300
141400
141500
141600
141700
141800
141900
142000
142100
142200
142300
142400
142500
142600
142700
142800
142900
143000
143100
143200
143300
143400
143500
143600
143700
143800
143900
144000
144100
144200
144300
144400
144500
144600
144700
144800
144900
145000
145100
145200
145300
145400
145500
145600
145700
145800
145900
146000
146100
146200
146300
146400
146500
146600
146700
146800
146900
147000
147100
147200
147300
147400
147500
147600
147700
147800
147900
148000
148100
148200
148300
148400
148500
148600
148700
148800
148900
149000
149100
149200
149300
149400
149500
149600
149700
149800
149900
150000
150100
150200
150300
150400
150500
150600
150700
150800
150900
151000
151100
151200
151300
151400
151500
151600
151700
151800
151900
152000
152100
152200
152300
152400
152500
152600
152700
152800
152900
153000
153100
153200
153300
153400
153500
153600
153700
153800
153900
154000
154100
154200
154300
154400
154500
154600
154700
154800
154900
155000
155100
155200
155300
155400
155500
155600
155700
155800
155900
156000
156100
156200
156300
156400
156500
156600
156700
156800
156900
157000
157100
157200
157300
157400
157500
157600
157700
157800
157900
158000
158100
158200
158300
158400
158500
158600
158700
158800
158900
159000
159100
159200
159300
159400
159500
159600
159700
159800
159900
160000
160100
160200
160300
160400
160500
160600
160700
160800
160900
161000
161100
161200
161300
161400
161500
161600
161700
161800
161900
162000
162100
162200
162300
162400
162500
162600
162700
162800
162900
163000
163100
163200
163300
163400
163500
163600
163700
163800
163900
164000
164100
164200
164300
164400
164500
164600
164700
164800
164900
165000
165100
165200
165300
165400
165500
165600
165700
165800
165900
166000
166100
166200
166300
166400
166500
166600
166700
166800
166900
167000
167100
167200
167300
167400
167500
167600
167700
167800
167900
168000
168100
168200
168300
168400
168500
168600
168700
168800
168900
169000
169100
169200
169300
169400
169500
169600
169700
169800
169900
170000
170100
170200
170300
170400
170500
170600
170700
170800
170900
171000
171100
171200
171300
171400
171500
171600
171700
171800
171900
172000
172100
172200
172300
172400
172500
172600
172700
172800
172900
173000
173100
173200
173300
173400
173500
173600
173700
173800
173900
174000
174100
174200
174300
174400
174500
174600
174700
174800
174900
175000
175100
175200
175300
175400
175500
175600
175700
175800
175900
176000
176100
176200
176300
176400
176500
176600
176700
176800
176900
177000
177100
177200
177300
177400
177500
177600
177700
177800
177900
178000
178100
178200
178300
178400
178500
178600
178700
178800
178900
179000
179100
179200
179300
179400
179500
179600
179700
179800
179900
180000
180100
180200
180300
180400
180500
180600
180700
180800
180900
181000
181100
181200
181300
181400
181500
181600
181700
181800
181900
182000
182100
182200
182300
182400
182500
182600
182700
182800
182900
183000
183100
183200
183300
183
```

最后结果如下：

莘庄	121.385379	31.111193
外环路	121.39302	31.120899
莲花路	121.402943	31.130986
锦江乐园	121.414107	31.142217
上海南站	121.430041	31.154579
漕宝路	121.433143	31.168344
上海体育馆	121.437423	31.182813
徐家汇	121.436837	31.195338
衡山路	121.446424	31.204528
常熟路	121.449141	31.213524
陕西南路	121.458744	31.21515
黄陂南路	121.473306	31.222745
人民广场	121.475137	31.232781
新闻路	121.468151	31.238373
汉中路	121.458699	31.241883
上海火车站	121.457939	31.249632
中山北路	121.459204	31.258891
延长路	121.455329	31.271675
上海马戏城	121.452023	31.279895
汶水路	121.450251	31.292556
彭浦新村	121.448642	31.306604
共康路	121.447063	31.318936
通河新村	121.441546	31.33113
呼兰路	121.437711	31.339703
共富新村	121.434063	31.355082
宝安公路	121.430914	31.369555

## 参考文献

- [1] 机器学习[M]. 周志华. 2016
- [2] 数据科学基础[M]. Avrim Blum, John Hopcroft, Eavindran Kannan 2017
- [3] <https://www.cnblogs.com/zxlovenet/p/4364385.html> [J]
- [4] <https://www.cnblogs.com/liangto/p/6287957.html> [J]
- [5] <https://www.cnblogs.com/smile233/p/8303673.html>
- [6] <https://www.cnblogs.com/GuoJiaSheng/p/3928160.html>