文件解释

hr_model.ipynb在xinxian的模型上，进行的调参
check_real_value.ipynb 就是用来检查真实值的，没啥用，我就是用它来检查，当某个时间窗口real_value的值很离谱的时候，回到具体的日期时间看下那个时间窗口的具体数值
ratio-Copy1.ipynb/ ratio.py 算时间窗口的ratio以及把ratio乘到小时模型预测出来的小时平均值，得到具体时间窗口的值

ratio算法：
1. 空值的填充：
首先如果一个小时的三个时间窗口，只有一个时间窗口有值，或者三个时间窗口都没有值，那么我直接不考虑这一行

然后的三个时间窗口都有值，可以直接用，这没问题

剩下的两个时间窗口有值，看缺的那个时间窗口的前后值，如果前后都有值，那么用前后时间窗口的值来填这个空，然后用这一行

如果前后某一个没值，填充不了，不考虑该行

2. 空值填充完后，可以算出具体的某天，具体的8，9，17，18点的分别的12个时间窗口的ratio

算出所有天的(去除空值后的）ratio，去掉top-5，last-5，然后取平均，then done

B1

Ratio

```
print(B_1_ratio_8)
print(B_1_ratio_9)
print(B_1_ratio_17)
```

```
print(B_1_ratio_18)
```

```
[1.0031420374343412, 0.9934482062307293,
0.9873723329424676]
[1.0143332259275053, 0.9560381921561566,
0.9965976774126419]
[0.9289831850456122, 1.000556880529891,
1.0657150770402124]
[1.0174539659659674, 1.0053514813377316,
0.9790868479720023]
```

Model

```
gdbt = GradientBoostingRegressor(n_estimators=30,
learning_rate=0.1, max_depth=5, random_state=0,
loss='ls').fit(train_x, train_y)
```

Result

```
mape_list=[]
for i in range(len(pred_value)):
    mape_list.append(np.abs((real_value1[i] -
pred_value[i]) / real_value1[i]))

np.mean(mape_list)
0.24692491475343145
```

筛掉一个太离谱的值

```
mape_list.remove(5.374733704123933)

np.mean(mape_list)
0.17570534823439676
```

test_compare(left:real value, right:prediction)

```
[[88.71, 127.80725530695705],
 [136.07, 126.5721939564116],
 [108.32, 125.79808554545149],
 [109.72, 137.70427874941532],
 [121.11, 129.79023691881665],
 [103.39, 135.29652865897407],
 [120.0, 103.2424137055729],
 [150.77, 111.196746139762],
 [154.23, 118.43809301096218],
```

[131.32, 112.43402669214903],
[131.49, 111.09663834314347],
[175.32, 108.19425790365271],
[92.31, 132.60364983990164],
[135.66, 131.32223868319392],
[153.98, 130.51907926615974],
[117.8, 137.18671241745025],
[137.92, 129.30241578895044],
[142.04, 134.78801193965657],
[120.2, 116.64362152944541],
[115.66, 133.81175042301638],
[141.59, 135.53748141312667],
[108.92, 122.30655894992937],
[138.17, 121.12465340384381],
[83.41, 120.38386184415234],
[119.25, 121.68164049366598],
[105.32, 114.68844027048968],
[132.5, 119.55404515992325],
[87.61, 134.1110467724574],
[173.74, 144.44365922151746],
[140.4, 153.85010928484732],
[21.7, 138.33172137948935],
[143.75, 136.68628326868182],
[137.51, 133.1153777865521],
[126.64, 123.85917434350579],
[163.0, 122.6622651478992],
[118.02, 121.91206964136397],
[123.06, 126.98808537479188],
[115.15, 119.68991694625271],
[97.77, 124.76770720773064],
[174.46, 189.7454774360067],
[221.76, 204.36445573415216],
[392.65, 115.81761996638274],
[188.54, 111.45025943738653],
[105.17, 130.7129245392432],
[116.39, 129.44978434640004],
[107.39, 128.65807675465183],
[130.69, 135.85226332491945],
[144.58, 128.0446592003491],
[126.19, 133.4768956001267],
[160.13, 113.79963094128678],

```
 [155.09, 122.56734629107007],
 [111.85, 130.54916860501254],
 [115.5, 120.1974194247009],
 [175.15, 118.76768651333524],
 [139.86, 106.52999653499705],
 [111.58, 105.50054729850325],
 [189.01, 119.14273628349557],
 [107.28, 117.05953351981076],
 [84.84, 127.48154496584147],
 [94.28, 137.3033861209131],
 [136.49, 146.24484780938903],
 [125.39, 153.61679220939322],
 [104.97, 151.78954013849682],
 [124.76, 147.8240646858864],
 [112.63, 121.14293132412202],
 [95.66, 126.19794519718785],
 [124.09, 118.94518713987596],
 [111.86, 123.99138257820807],
 [114.14, 114.32697085875732],
 [128.66, 123.13531521806125],
 [123.62, 131.1541247654902],
 [146.24, 138.9431548058852],
 [125.79, 133.70375470208782]]
```

A2

Ratio

```
print(A_2_ratio_8)
print(A_2_ratio_9)
print(A_2_ratio_17)
print(A_2_ratio_18)
```

```
[0.9594377262857952, 1.0144147364368832,
1.004964586594884]
[1.000155698203997, 1.0194005415982434,
0.9602155822869323]
[1.0152742931390726, 0.9915728865263712,
0.9819021063140361]
[1.0065583872672326, 0.969453070970589,
1.0087499022367605]
```

Model

```
gdbt = GradientBoostingRegressor(n_estimators=20,
learning_rate=0.1,
                                 max_depth=5,
random_state=0, loss='ls').fit(train_x, train_y)
```

Result

```
mape_list=[]
for i in range(len(pred_value)):
    mape_list.append(np.abs((real_value1[i] -
pred_value[i]) / real_value1[i]))

np.mean(mape_list)
0.133475149336903
```

test_compare

```
[[77.77, 82.44221939830102],
 [70.3, 87.16626412633636],
 [77.68, 86.35423505422872],
 [79.28, 76.39502671813483],
 [83.27, 77.86500816995266],
 [77.67, 73.34427549201202],
 [75.0, 77.72956328786886],
 [63.3, 75.91497978293505],
 [68.39, 75.17458329339762],
 [88.83, 78.4388443881954],
 [59.33, 75.54731005915455],
 [52.23, 78.60962424939741],
 [91.72, 98.96720846762604],
 [92.28, 104.63815622742759],
 [103.94, 103.6633613826553],
 [79.27, 79.25959149316411],
 [81.22, 80.78469246346002],
 [79.82, 76.09444702869645],
 [60.0, 68.0370839622574],
 [56.98, 66.44876974743953],
 [66.91, 65.80069691654687],
 [63.8, 69.79375269646481],
 [55.74, 67.2209071446404],
```

[84.02, 69.94571015442122],
[87.24, 89.31857018843752],
[75.48, 94.43663862101829],
[79.6, 93.55687972804515],
[103.35, 78.03214298788075],
[97.05, 79.53362558125684],
[67.95, 74.91601532716801],
[96.52, 69.4718611889569],
[54.93, 67.85005234251241],
[82.28, 67.18831284507768],
[75.25, 62.648695888729954],
[50.34, 60.33924250189289],
[57.25, 62.785096873112295],
[58.58, 70.46657155138286],
[71.5, 74.50439632453416],
[55.92, 73.81032349232393],
[99.53, 73.67620860719377],
[83.67, 75.09387497561362],
[73.3, 70.73403038695734],
[101.05, 73.27112108872407],
[82.62, 71.56061916266532],
[62.0, 70.86269061985784],
[76.75, 71.68570263496265],
[81.04, 69.04311309036702],
[70.41, 71.84177931408324],
[71.43, 68.43939480634508],
[56.06, 72.3610597564702],
[62.2, 71.6869539564831],
[79.54, 76.79073432438263],
[64.81, 78.26832992160384],
[70.94, 73.72418095096296],
[71.32, 75.1427171755674],
[63.61, 73.3885231554915],
[77.17, 72.67276712062159],
[69.61, 68.97118190644589],
[66.35, 66.42865923477045],
[58.64, 69.12134843381882],
[56.62, 64.79180267152985],
[65.06, 68.50445592206394],
[65.96, 67.86627771934772],
[83.79, 66.4605044545947],

```
 [64.03, 67.73932734429883],
 [66.64, 63.80647743001268],
 [94.08, 72.18591399035061],
 [67.87, 70.50074604041176],
 [63.09, 69.81315440793891],
 [63.97, 70.67586552992667],
 [60.32, 68.07050216680688],
 [69.44, 70.82974355553598],
 [70.59, 89.91339757132512],
 [85.32, 95.06555037454426],
 [63.33, 94.17993262513426],
 [79.18, 76.48002852417821],
 [76.7, 77.95164556778286],
 [51.01, 73.42588284457918],
 [85.03, 74.30074462062095],
 [68.98, 72.56620630739825],
 [53.84, 71.85847030374426],
 [65.13, 68.27920872288792],
 [61.35, 65.76219463985423],
 [62.85, 68.42786866146335]]
```

A3

Ratio

```
print(A_3_ratio_8)
print(A_3_ratio_9)
print(A_3_ratio_17)
print(A_3_ratio_18)
```

```
[0.9702602614629005, 0.978968720287374,
1.0444125878050396]
[1.057196991804728, 1.0448782719176082,
0.8779080952581202]
[1.0131716009078262, 1.008282644002323,
0.977339452004021]
[0.9618902905147334, 1.0074499810834463,
1.007627041410743 6]
```

Model

```
gdbt = GradientBoostingRegressor(n_estimators=28,
learning_rate=0.1,
                                 max_depth=6,
random_state=0, loss='ls').fit(train_x, train_y)
```

Result

```
np.mean(mape_list)
0.2152616858735426
```

test_compare

```
[[197.5, 205.8902948984465],
 [166.9, 207.7382394414689],
 [253.82, 221.62550012572217],
 [240.16, 226.86798487526028],
 [179.19, 224.22446320550713],
 [136.98, 188.39368823485623],
 [136.34, 136.28690841999193],
 [143.7, 135.62926975201842],
 [100.13, 131.4669422940477],
 [122.91, 130.17800785157345],
 [112.75, 136.34385630129157],
 [120.58, 136.36781886844128],
 [319.67, 233.28152146564813],
 [328.02, 235.3753127965753],
 [326.7, 251.11010643029468],
 [339.07, 250.19178541261485],
 [316.84, 247.2764890710172],
 [199.89, 207.76203061821346],
 [152.27, 124.32741492479265],
 [123.18, 123.7274855809428],
 [112.64, 119.93041204747291],
 [110.19, 122.1918655530961],
 [100.43, 127.97945239071117],
 [150.64, 128.00194490562816],
 [183.76, 158.11280785876633],
 [242.81, 159.53193108945905],
 [224.49, 170.19660948693752],
 [201.63, 183.96683486108827],
```

[263.23, 181.82320796398076],
[182.37, 152.76809793779142],
[139.66, 133.32668330299862],
[126.32, 132.68332890139678],
[124.92, 128.61140943953572],
[139.16, 130.72448424858902],
[128.43, 136.91621641477317],
[95.71, 136.94027957477704],
[258.88, 151.22874759288482],
[270.64, 152.58608373638893],
[240.81, 162.78643359654825],
[237.97, 160.11920714234333],
[250.89, 158.25345868049135],
[194.2, 132.9647636592399],
[135.93, 122.13810140852388],
[121.93, 121.54873637522525],
[107.05, 117.81852648895756],
[113.41, 115.14837465720495],
[123.13, 120.60234832822012],
[107.53, 120.62354430982592],
[120.94, 149.97693115534278],
[113.81, 151.32303176510908],
[195.13, 161.43894684797326],
[116.25, 165.61953700699328],
[124.06, 163.68969734604156],
[147.83, 137.532298520007],
[110.48, 122.48352041041343],
[113.84, 121.89248859271909],
[124.66, 118.15172929262481],
[88.7, 113.42285412479463],
[103.83, 118.79509895178069],
[170.18, 118.81597730752704],
[91.06, 114.86109237743399],
[90.74, 115.89201483528545],
[111.3, 123.63937336478921],
[116.89, 158.0689784679873],
[103.09, 156.22711977591305],
[95.62, 131.26223105245023],
[112.65, 125.73934808635393],
[107.82, 125.13260560209021],
[112.35, 121.29241033201833],

```
 [98.51, 122.67247614105469],
 [103.75, 128.4828269777322],
 [102.92, 128.50540796122908],
 [265.38, 242.3275604347299],
 [234.99, 244.5025434427933],
 [181.75, 260.8474906604177],
 [262.43, 243.0318993746955],
 [146.56, 240.20003178971422],
 [92.95, 201.8163819240333],
 [121.97, 147.12298000572804],
 [124.4, 146.41305297222974],
 [95.07, 141.91978192752174],
 [109.61, 145.44911117604494],
 [109.73, 152.33827157616588],
 [165.64, 152.36504517756708]]
```

B3

Ratio

```
print(B_3_ratio_8)
print(B_3_ratio_9)
print(B_3_ratio_17)
print(B_3_ratio_18)
```

```
[0.9588587319159432, 0.9877176009992981,
1.0498548948656439]
[1.0254018000740626, 1.0251925941467648,
0.9276552578492043]
[0.9887454541008326, 0.9964707338380407,
1.0048398973940167]
[1.0134617557170276, 0.9490581825784564,
1.0192924016085947]
```

Model

```
gdbt = GradientBoostingRegressor(n_estimators=30,
learning_rate=0.1,
```

```
                                    max_depth=5,
random_state=0, loss='ls').fit(train_x, train_y)
```

Result

```
np.mean(mape_list)
0.21764746772789725

#筛掉一些比较离谱的时间窗口
for i in mape_list:
    if i>1:
        mape_list.remove(i)

np.mean(mape_list)
0.20489378899015584
```

test_compare

```
[[126.4, 98.85493230726598],
 [115.52, 101.83017929072818],
 [98.86, 108.23631376544952],
 [131.15, 110.21469211177342],
 [151.95, 110.19220573924763],
 [117.69, 99.70846415749726],
 [80.49, 102.47639263304252],
 [97.56, 103.27706260959394],
 [104.47, 104.14446653748585],
 [125.47, 109.76881455340525],
 [111.18, 102.7932145008677],
 [117.39, 110.40033625019112],
 [70.3, 91.79771917393968],
 [145.63, 94.5605644937063],
 [120.0, 100.50936765178339],
 [129.17, 102.10365042783124],
 [108.18, 102.08281889733587],
 [112.14, 92.37060843674509],
 [106.34, 102.65383943065054],
 [105.73, 103.45589583698963],
 [101.06, 104.32480175031723],
 [87.42, 111.83093895535063],
```

[110.38, 104.72429480668184],
[80.84, 112.47432446160369],
[93.88, 103.83988143881308],
[87.78, 106.96516094488375],
[103.01, 113.69433700934667],
[94.06, 115.56475929499359],
[114.88, 115.54118138374992],
[70.85, 104.54853558315908],
[116.79, 135.2560432480628],
[93.09, 136.3128276468234],
[87.7, 137.45769252906612],
[99.22, 115.87530642329394],
[80.24, 108.5116503898139],
[106.83, 116.54195997535919],
[85.64, 110.15290668096502],
[99.17, 113.46818995184381],
[102.09, 120.6064714367407],
[103.95, 123.20235085227564],
[82.41, 123.17721469389025],
[107.51, 111.45807286396516],
[84.76, 102.72225152199903],
[108.29, 103.52484244664197],
[130.19, 104.39432742911151],
[137.44, 110.90804460227919],
[120.71, 103.86004864001893],
[60.02, 111.5461205148177],
[103.23, 116.93848759369558],
[111.49, 120.45799718561305],
[65.55, 128.03600729913254],
[92.04, 139.6501326083589],
[116.15, 139.62164071816767],
[114.06, 126.33796796936169],
[87.86, 116.53664101795869],
[103.59, 117.44716672279446],
[120.99, 118.43358259444094],
[95.22, 121.82896725032658],
[72.77, 114.08696735892292],
[88.6, 122.52987339045953],
[106.67, 95.82149351311458],
[85.55, 98.70544277969844],
[91.14, 104.9150001450878],

```
 [64.23, 107.392778040124],
 [120.49, 107.37086740400717],
 [105.35, 97.15554936295023],
 [88.62, 99.56700846956706],
 [104.55, 100.34494680529764],
 [90.96, 101.18772446379909],
 [109.01, 105.35916159305945],
 [81.63, 98.66378662582511],
 [104.62, 105.96531368436051],
 [120.81, 108.17883464420156],
 [98.33, 111.43470406758247],
 [107.49, 118.44505899752521],
 [147.68, 123.84359276031195],
 [124.16, 123.81832577359374],
 [136.56, 112.03818831480616],
 [110.46, 100.70133776909384],
 [57.83, 101.48813886228805],
 [118.39, 102.34051796815363],
 [104.99, 104.83060264597079],
 [129.71, 98.16881659772116],
 [46.32, 105.43371383312731]]
```

C1

Ratio

```
[1.0085446631871167, 1.0135826326873116,
0.9693400182600777]
[0.9649815049634551, 1.0068986583841009,
1.0131470826840883]
[0.9455038549039972, 0.996692971037903 1,
1.0406337231620946]
[0.95167652707838, 1.0515667415053014,
0.9895062988612441]
```

Model

```
gdbt = GradientBoostingRegressor(n_estimators=8,
learning_rate=0.1,
                                 max_depth=5,
random_state=0, loss='ls').fit(train_x, train_y)
```

Result

```
np.mean(mape_list)
0.2185103004349994
```

test_compare

```
[[134.51, 197.94241618604457],
 [183.06, 198.9311952574531],
 [158.02, 187.61230200495416],
 [130.53, 195.76186094084278],
 [151.99, 196.97668346417055],
 [417.44, 192.7990887722987],
 [185.55, 203.2371371149768],
 [238.49, 212.1971608373272],
 [179.15, 178.11743860130912],
 [263.94, 196.81306534927597],
 [390.85, 185.19772466607287],
 [158.18, 191.66255811536826],
 [189.34, 192.61996749678642],
 [140.53, 184.21215674893227],
 [158.96, 170.58642710081486],
 [250.19, 177.99641102225328],
 [220.91, 179.1009880228095],
 [233.4, 161.76173101749887],
 [190.28, 170.51943199579594],
 [198.43, 178.03704505358908],
 [283.97, 196.53798579169725],
 [314.9, 217.1670766489008],
 [200.43, 180.32622464876644],
 [178.94, 181.2270057971143],
 [200.76, 173.3164948207908],
 [138.53, 201.88072087643408],
 [182.56, 210.65007563206592],
 [152.07, 211.95728866727436],
 [226.48, 205.22865770183458],
 [312.25, 216.33963682541682],
 [174.53, 212.43138006017577],
 [149.01, 234.72867909135184],
```

```
[165.61, 220.87566800732762],
[147.11, 162.3306972782727],
[193.38, 163.14158561240848],
[149.93, 160.24528011699644],
[182.18, 167.20606222219632],
[173.31, 168.2436784843642],
[265.45, 202.90009229780023],
[465.17, 213.88500402961637],
[482.11, 223.31445544368646],
[487.41, 177.0572063484536],
[490.26, 195.641548616803],
[317.67, 184.09534747949132],
[169.57, 200.99867784246774],
[173.51, 192.22513859920852],
[187.36, 185.85458581402187],
[187.97, 193.92779255158723],
[331.19, 167.20132601799241],
[240.21, 176.2535240083931],
[229.13, 184.02393338672263],
[271.24, 140.7015022189796],
[245.61, 155.4698639752492],
[233.02, 146.29448005020836],
[159.65, 173.29093092827037],
[186.79, 174.156568769155],
[153.82, 166.554680503178],
[211.64, 176.34656084326565],
[153.05, 184.00675516621317],
[447.96, 185.148630041871],
[227.86, 195.18544854851953],
[229.92, 205.75269324199905],
[221.27, 214.82361914931906],
[200.4, 202.47754098829702],
[224.13, 223.7300615774604],
[122.38, 210.52615724476578],
[159.83, 193.17823810551832],
[156.72, 194.14321874266878],
[157.63, 185.66891847992557],
[136.74, 197.509619313636],
[205.9, 198.7352877505978],
[190.15, 171.03482302147526],
[154.65, 180.29456466417568],
```

```
 [160.04, 188.24312957378643],
 [271.2, 169.76433849525284],
 [260.35, 187.58320414112018]]
```

C3

Ratio

```
[0.9148328918283646, 1.0075582153926481,
1.0225619584493677]
[1.040282893182057, 0.9801969562756571,
0.9588049191875893]
[0.9750481330349163, 0.9561918653994332,
1.064546493530613]
[1.0017435848975271, 0.980500959518688,
0.9999905668560394]
```

Model

```
gdbt = GradientBoostingRegressor(n_estimators=30,
learning_rate=0.1,
                                 max_depth=5,
random_state=0, loss='ls').fit(train_x, train_y)
```

Result

```
0.24126218559987836

#筛掉一些比较离谱的时间窗口
for i in mape_list:
    if i>1:
        mape_list.remove(i)

np.mean(mape_list)
0.2126357427919870 5
```

test_compare

```
[[233.55, 191.5642774981399],
 [155.31, 194.41689797655874],
```

[216.72, 183.7651254732655],
[178.4, 173.15099367589593],
[144.3, 169.37210775420888],
[189.83, 163.82328594339432],
[84.47, 175.5222981705879],
[204.89, 175.21513997160838],
[188.63, 181.9788691532709],
[175.91, 200.42382196905274],
[202.04, 203.40837162715613],
[209.15, 209.2902540144762],
[143.38, 197.20181049564442],
[168.16, 192.8980341811518],
[108.94, 176.2511500539585],
[185.13, 172.8426630840541],
[273.95, 192.33285265136354],
[123.48, 188.25430920121073],
[170.41, 191.99627653972001],
[141.54, 190.89443366341493],
[194.8, 193.73707936853452],
[148.35, 183.30368743782486],
[209.36, 172.71620794519413],
[143.44, 189.12081974700564],
[138.56, 185.4634487190835],
[173.81, 206.47996616193763],
[211.71, 193.7257218052302],
[124.56, 176.7188875326968],
[214.37, 179.35044245558197],
[106.63, 183.55214792448436],
[155.75, 172.9503175459331],
[131.44, 169.17581122488386],
[193.78, 171.01640802010868],
[254.98, 190.39580243443808],
[273.59, 197.39789645558395],
[216.57, 193.2119454515595],
[242.64, 197.05245668531202],
[196.43, 155.32882748402875],
[134.16, 180.39952747808366],
[174.1, 169.97979002298018],
[135.19, 166.2701131574146],
[199.61, 183.55751352686968],
[167.41, 180.00773020407806],

```
 [245.12, 200.40601152479695],
 [167.08, 183.10430295086627],
 [224.5, 179.22145690974426],
 [241.57, 182.78387649504847],
 [93.53, 193.13321797623385],
 [211.69, 181.97799238720836],
 [198.63, 172.84767537697928],
 [200.92, 192.43458702779927],
 [218.88, 180.57350418142735],
 [138.99, 180.25750653949876],
 [141.63, 186.10412699956055],
 [142.25, 179.42419522048726],
 [169.27, 175.50840154853265],
 [161.79, 172.17719267674556],
 [165.87, 168.8474911821809],
 [111.53, 187.98110628597104],
 [109.12, 189.3160060553432]]
```

不知道当时比赛的时候，评估用的是全部的时间窗口还是怎样，其实phase2的数据，有的时间窗口是没数值的，另外有些时间窗口的车流量只有1且这个1的值很离谱
如果按照我们现在的模型，去掉这些离谱的值，我们的mape是：
summ=0.17570534823439676+0.133475149336903+0.2152616858735426+0.21764746772789725+0.2185103004349994+0.21263574279198705
summ/6=0.19553928239995433
大概就能排在当时的比赛的rank 135/368

其实先算小时平均时间，一个很明显的问题就是，即使小时的平均时间拟合的很好，但时间窗口里依然可能会有一些离谱的情况，比如均值为100的一个小时，三个时间窗口的值可能分别是[50,150,100], 这样的话第一第二时间窗口的误差就会很离谱，但其实想想，这个问题其实好像也没办法避免，即使建模型对具体的每个时间窗口进行拟合，这些离谱的值依然很难被预测准确。

查数据可知，这些离谱的值，大部分是因为，该时间窗口的车流量只有1，然后这个1很快或很慢，把这个1的值作为这个时间窗口的平均时间，这个平均时间就会很离谱。

一个trick，对A_3根据结果进行ratio的调整，调整至：

```
[0.9702602614629005, 0.978968720287374,
1.2444125878050396]
[1.157196991804728, 1.0448782719176082,
0.8779080952581202]
[1.0131716009078262, 1.008282644002323,
0.977339452004021]
[0.9618902905147334, 1.0074499810834463,
1.007627041410743 6]
```

Mape可以从0.2152616858735426降至0.2126686853866618

对C_1根据结果进行ratio的调整，调整至：

```
C_1_ratio_8=[0.95, 1.0135826326873116,
0.9693400182600777]
C_1_ratio_9=[0.9649815049634551, 1.0068986583841009,
1.0131470826840883]
C_1_ratio_17=[1.2, 0.996692971037903 1,
1.0406337231620946]
C_1_ratio_18=[0.95167652707838, 1.0515667415053014,
0.9895062988612441]
```

Mape可以从0.2185103004349994降至0.20240195003038308

总的Mape可以从0.19553928239995433降到0.1924223905847048

排名	参与者	组织	MAPE	最优成绩提交日
101	学霸好强!	复旦大学	0.1921	2017-05-29
102	sang	Other Overseas regions-sg	0.1922	2017-06-01
103	Green Hand Team	东南大学	0.1927	2017-05-31

但不是每一个都能进行调整，需要刚好那条路的那个时间窗口 大部分 的预测值比真实值低了/高了，这时候调整ratio才会有降低mape的作用