# 期末大作业数据建模及分析报告

1752221-潘帅鑫

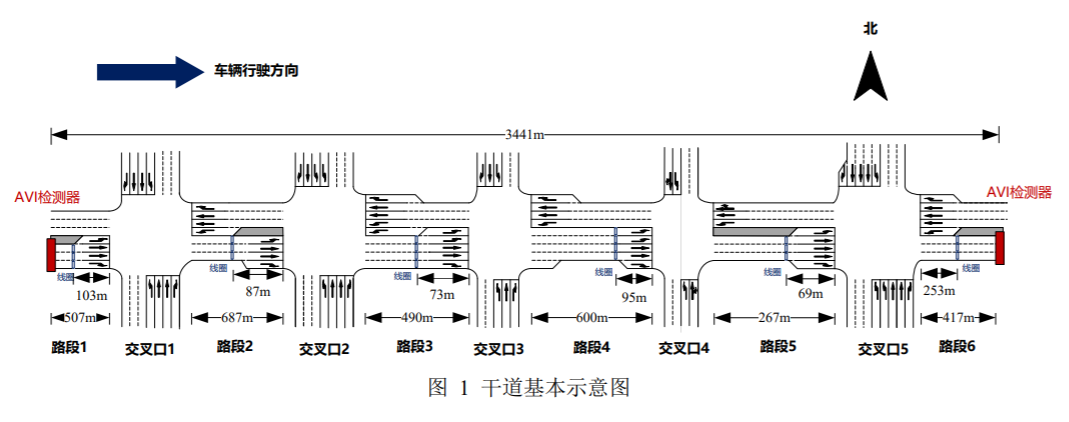
## 1.说明

本次使用的编程语言是python，其中数据集1,2…6指各题处理后的数据集。数据集1包含的是干线平均速度和需要计算的统计分析量。数据集2是Sheet1预处理后的数据。数据集3是聚类分析后的各类簇的统计特征数据。数据集4是路段3，6的线圈和浮动车数据估计得到的行程速度与真实行程速度数据的相关性和相异性参数结果。数据集5是三种检测方式的行程估计误差。数据集6是经建模分析后的最后4小时的按方法3的测试结果，即是最后4小时各路段的预测的行程速度。

## 2.题目部分

### 第1题：

以真实行程数据(sheet4)为基础，计算整条干线的平均行程速度，根据各路段的路段长度加权，根据下图，可以得到各路段的路段长度与其比值，如下表1.1所示





按照上述的路段长度加权，得到干道的平均路程长度，计算代码如下：

1. pingjun = data4.cell(i,3).value\*0.17082+data4.cell(i,4).value\*0.23147\
2. +data4.cell(i,5).value\*0.16509+data4.cell(i,6).value\*0.20216\
3. +data4.cell(i,7).value\*0.08996+data4.cell(i,8).value\*0.1405

循环，可以得到每个时刻的干线平均行程速度km/h。

（1）计算算术平均值、中列数、中位数等值:

计算中使用的是python的numpy包中的函数进行直接调用，每行代码后面的注释部分是粘贴后添加的，用于解释，并不是源代码附带的：

1. junzhi = np.mean(speed\_g)  算术平均值
2. zhonglie = (max(speed\_g)+min(speed\_g))/2  中列数，最大值与最小值的平均
3. zhongwei = np.median(speed\_g)  中位数
4. wbiaozhun = np.std(speed\_g,ddof=1)  标准差，样本标准差，除以n-1
5. CV = wbiaozhun/junzhi  变异系数，标准差与均值之比
6. maxs = max(speed\_g)  最大值
7. mins = min(speed\_g)  最小值
8. shul = len(speed\_g)  样本量即是list长度

计算结果如下：



（2）五分位数的计算方法与上述一致：

1. wufen = []
2. wufen.append(mins)
3. wufen.append(np.percentile(speed\_g,25)) 利用percentile函数，25表示25%
4. wufen.append(zhongwei)
5. wufen.append(np.percentile(speed\_g,75))
6. wufen.append(maxs)

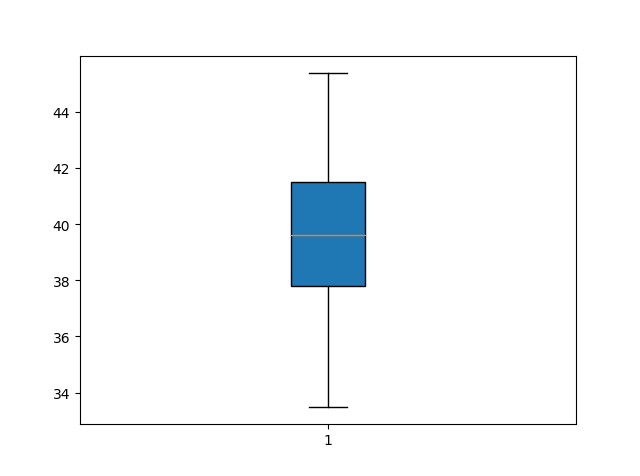
计算结果如下：



画出箱图，用boxplot方法，主要代码如下：

1. **import** matplotlib.pyplot as plt
2. plt.boxplot(speed\_g,patch\_artist=True,showbox=True)
3. plt.show()

结果如下：



（3）计算区间频数和累计分布频率：

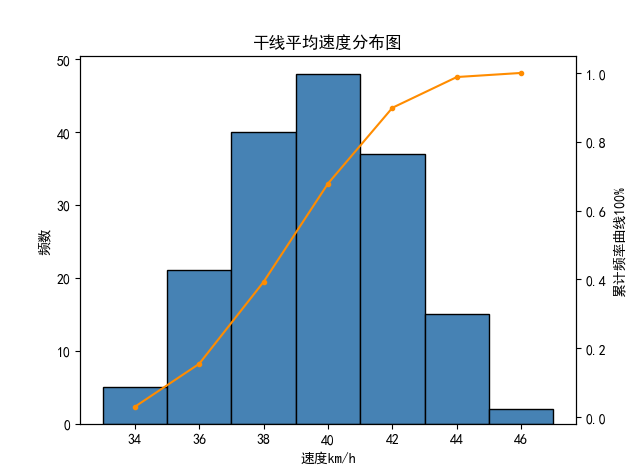
根据得到的干线行程速度的最大值和最小值，选取区间为2km/h分成了7个区间，绘制其直方图和累计分布频率图，直方图使用的是pyplot.hist()函数，使用双y轴的形式，使用的是plot函数：

1. plt.rcParams['font.sans-serif']=['SimHei']
2. plt.rcParams['axes.unicode\_minus'] = False  用于消除汉字标题乱码的问题
3. fig = plt.figure()
4. ax = fig.add\_subplot(111)
5. ax.hist(speed\_g,bins=[33,35,37,39,41,43,45,47],color='steelblue',edgecolor='black')
6. ax.set\_title('干线平均速度分布图')
7. ax.set\_xlabel('速度km/h')
8. ax.set\_ylabel('频数')
9. x=[34,36,38,40,42,44,46]
10. y=[5/168,26/168,66/168,114/168,151/168,166/168,168/168]
11. #双y轴
12. ax2 = ax.twinx()
13. ax2.plot(x,y,'.-',color='darkorange')
14. ax2.set\_ylabel('累计频率曲线100%')
15. plt.show()

分组后的频数表如下所示：



画出的结果图如下所示：



## 第2题：

对线圈数据(Sheet)进行预处理：

（1）自选方法剔除Sheet1中的异常数据：

阈值法：采用的是阈值法中的五分位数原则，分别对速度、流量、占有率进行五分位数的计算，并确定异常区间，将处于异常区间之外的观测值视为异常数据，异常数据就将其赋值为空。具体过程如下：

1. sudu=[]
2. liuliang=[]
3. zhanyoulv=[]
4. **for** i **in** range(3,171):
5. **for** j **in** range(3,9):
6. **if** data1.cell(i,j).value **is** **not** None:
7. sudu.append(data1.cell(i,j).value)
8. **for** j **in** range(9,15):
9. **if** data1.cell(i,j).value **is** **not** None:
10. liuliang.append(data1.cell(i,j).value)
11. **for** j **in** range(15,21):
12. **if** data1.cell(i,j).value **is** **not** None:
13. zhanyoulv.append(data1.cell(i,j).value)

这一部分是将所有的速度、流量、占有率值添加到三个list中，添加的条件是当读到的单元格不为空值时才进行添加。

1. #通过五分位数法进行异常数据剔除
2. **for** i **in** range(3,171):
3. **for** j **in** range(3,9):
4. **if** data1.cell(i,j).value **is** **not** None:
5. **if** data1.cell(i,j).value<sudumin **or** data1.cell(i,j).value>sudumax:
6. data1.cell(i,j).value=None
7. **for** j **in** range(9,15):
8. **if** data1.cell(i,j).value **is** **not** None:
9. **if** data1.cell(i,j).value<liulmin **or** data1.cell(i,j).value>liulmax:
10. data1.cell(i,j).value=None
11. **for** j **in** range(15,21):
12. **if** data1.cell(i,j).value **is** **not** None:
13. **if** data1.cell(i,j).value<zhanylmin **or** data1.cell(i,j).value>zhanylmax:
14. data1.cell(i,j).value=None

先根据上一步得到的list，算出各自的正常区间（代码已省略），然后将非正常区间的值视作异常，并赋值为空，赋为空值的目的是为了下面的数据修补提供方便。上面的步骤是进行异常数据判断并赋值的过程，主要是利用循环，遍历Sheet1的指定行与列的单元格，对其内容进行判断处理。

（2）自选方法补全Sheet1中的缺失数据：

该缺失数据包含上面的异常数据，修补方法主要使用时间序列数据修补的方法，对于前5个周期而言使用的是线性内插法，即是前一个周期数据和后一个周期数据的平均值，其中如果第一个周期缺失，由于缺少它前面周期的数据，因此使用的是后两个周期数据的平均值。

对于第6个周期及以后的缺失数据，使用的是加权移动平均法的修补方法，

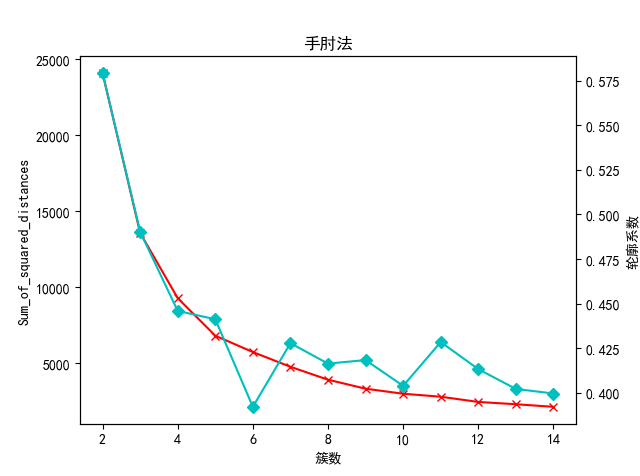
公式：，其中St是待修补数据，n取5，待修补数据前5个周期进行加权平均，Wt-1=5, Wt-2=4, Wt-3=3, Wt-2=2, Wt-5=1。

实现的代码如下，主要是循环嵌套循环，同时利用if条件进行判定，其中当其内容为空时，进行修补，修补中，利用行数和列数对其完成不同的修补方法，其中，流量转换为整数：

1. **for** i **in** range(3,171):
2. **for** j **in** range(3,21):
3. **if** data1.cell(i,j).value **is** None:
4. **if** i==3:
5. data1.cell(i,j).value=(data1.cell(i+1,j).value+data1.cell(i+2,j).value)/2
6. #线性内插法
7. **if** 3<i<8:
8. #流量存为整数
9. **if** 8<j & j<15:
10. data1.cell(i,j).value=int(data1.cell(i-1,j).value+data1.cell(i+1,j).value)
11. #占有率和速度存为浮点型
12. **else**:
13. data1.cell(i,j).value=(data1.cell(i-1,j).value+data1.cell(i+1,j).value)
14. #加权移动平均法
15. **if** i>7:
16. **if** 8<j & j<15:
17. data1.cell(i,j).value=int((1\*data1.cell(i-5,j).value+2\*data1.cell(i-4,j).value\
18. +3\*data1.cell(i-3,j).value+4\*data1.cell(i-2,j).value\
19. +5\*data1.cell(i-1,j).value)/(5+4+3+2+1))
20. **else**:
21. data1.cell(i,j).value=(1\*data1.cell(i-5,j).value+2\*data1.cell(i-4,j).value\
22. +3\*data1.cell(i-3,j).value+4\*data1.cell(i-2,j).value\
23. +5\*data1.cell(i-1,j).value)/(5+4+3+2+1)

## 第3题：

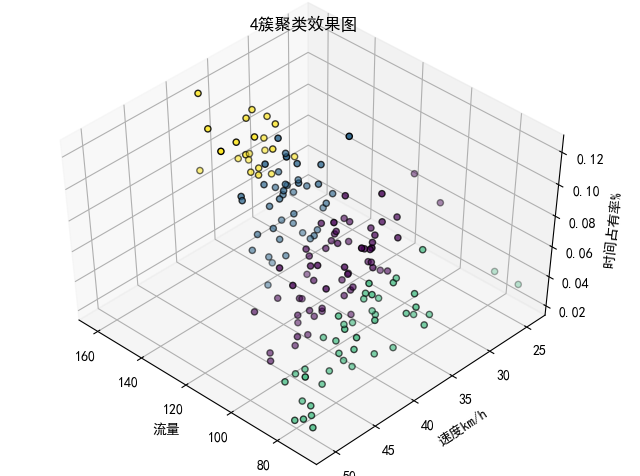
自选方法，对路段 2~5 中的 1 个路段线圈数据进行聚类分析，选取的是路段4，利用平均速度、流量和占有率三个变量，进行聚类，使用的是K-means算法，进行分类。先要确定分类簇数，根据silhouette\_score计算轮廓系数，并计算距离平方和，其中将轮廓系数选为2-15，按照肘方法进行判断，得到下图：



从上图可以看出，选择簇数为4最合适，轮廓系数相对较高，且SSE以后并没有太大的下滑，因此选择将其聚为四类。选用sklearn.cluster中的Kmeans包，进行聚类，并画出3D图：

1. **from** mpl\_toolkits.mplot3d **import** Axes3D
2. fig = plt.figure()
3. ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
4. estimators = KMeans(n\_clusters=4)
5. estimators.fit(shuzu4)  进行聚类
6. ax.scatter(shuzu4[:,1],shuzu4[:,0],shuzu4[:,2],
7. c=estimators.labels\_.astype(np.float),edgecolor='k') 画出效果图

聚类效果如下图所示：



上面不同的颜色表示不同的簇，将不同的簇放到四个数组中，并利用numpy库中的将list转换为numpy数组，其后用统计函数，进行计算下面的统计指标，示例计算cu1的代码如下：

1. cu1=[]
2. cu2=[]
3. cu3=[]
4. cu4=[]
5. **for** i **in** range(0,168):
6. **if** estimators.labels\_[i]==0:
7. cu1.append(shuzu4[i])
8. **elif** estimators.labels\_[i]==1:
9. cu2.append(shuzu4[i])
10. **elif** estimators.labels\_[i]==2:
11. cu3.append(shuzu4[i])
12. **elif** estimators.labels\_[i]==3:
13. cu4.append(shuzu4[i])
15. #将嵌套列表转换为numpy数组，支持多维切片
16. cu1 = np.array(cu1)
17. cu2 = np.array(cu2)
18. cu3 = np.array(cu3)
19. cu4 = np.array(cu4)
20. #计算统计特征
21. tjtz1 = np.zeros([5,3])
22. **for** i **in** range(0,3):
23. tjtz1[0][i]=np.mean(cu1[:,i])
24. tjtz1[1][i]=np.var(cu1[:,i],ddof=1)
25. tjtz1[2][i]=max(cu1[:,i])
26. tjtz1[3][i]=min(cu1[:,i])
27. tjtz1[4][i]=len(cu1[:,i])

得到的统计指标如下表所示：



### 第4题：

选择路段3和6分析线圈和浮动车估计得到的行程速度与真实行程速度的相异性和相关性，其中相异性是计算欧几里得距离和DTW距离，相关性是计算相关系数和协方差。

1.欧几里得距离：

1. #线圈路段3和6欧几里得距离
2. pf13 = 0
3. pf16 = 0
4. **for** i **in** range(3,171):
5. pf13 = pf13 +pow((data1.cell(i,5).value-data4.cell(i,5).value),2)
6. pf16 = pf16 +pow((data1.cell(i,8).value-data4.cell(i,8).value),2)
7. xq3\_ojld = pow(pf13,0.5)
8. xq6\_ojld = pow(pf16,0.5)

浮动车的欧几里得同理。

2.DTW距离，先定义DTW函数，再带入线圈的速度数据和真实的速度数据，定义的DTW函数如下：

1. **def** dtw\_distance(ts\_a,ts\_b,d=**lambda** x,y:abs(x-y),mww=10000):
2. **import** numpy as np
3. #Create cost matrix via broadcasting with large int
4. ts\_a,ts\_b=np.array(ts\_a),np.array(ts\_b)
5. M,N=len(ts\_a),len(ts\_b)
6. cost=np.ones((M,N))
8. #Initialize the first row and column
9. cost[0,0]=d(ts\_a[0],ts\_b[0])
10. **for** i **in** range(1,M):
11. cost[i,0] = cost[i-1,0] + d(ts\_a[i],ts\_b[0])
13. **for** j **in** range(1,N):
14. cost[0,j] = cost[0,j-1] + d(ts\_a[0],ts\_b[j])
16. #Populate rest of cost matrix within window
17. **for** i **in** range(1,M):
18. **for** j **in** range(max(1,i-mww),min(N,i+mww)):
19. choices = cost[i-1,j-1],cost[i,j-1],cost[i-1,j]
20. cost[i,j] = min(choices)+d(ts\_a[i],ts\_b[j])

23. #Return DTW distance given window
24. **return** cost[-1,-1]

提出线圈、浮动车和真实数据的路段3和6的行程速度序列，如下：

1. #提出各检测器各路段的速度序列
2. xq3sudu = []
3. xq6sudu = []
4. fdc3sudu = []
5. fdc6sudu = []
6. zs3sudu = []
7. zs6sudu = []
8. **for** i **in** range(3,171):
9. xq3sudu.append(data1.cell(i,5).value)
10. xq6sudu.append(data1.cell(i,8).value)
11. fdc3sudu.append(data2.cell(i,5).value)
12. fdc6sudu.append(data2.cell(i,8).value)
13. zs3sudu.append(data4.cell(i,5).value)
14. zs6sudu.append(data4.cell(i,8).value)

然后代入函数中的参数，求得DTW距离。

3.协方差，根据协方差和相关系数的关系，协方差是计算相关系数的一个中间变量，协方差需要计算出E(X)、E(Y)、E(XY)，X是指线圈和浮动车路段3、6的行程速度，Y是指真实的路段3、6的行程速度，将X、Y、XY的速度数据得到10个矩阵，利用np.mean计算数学期望，再利用协方差公式，计算出相应的协方差，协方差计算公式如下：

4.相关系数，利用相关系数的公式如下：

主要利用numpy的函数，代码如下：

1. #计算相关系数
2. y3sigma = pow(qw[0][7]-pow(qw[0][2],2),0.5)
3. y6sigma = pow(qw[1][7]-pow(qw[1][2],2),0.5)
4. xq3\_xgxs = xq3\_cov/(pow(qw[0][5]-pow(qw[0][0],2),0.5)\*y3sigma)
5. xq6\_xgxs = xq6\_cov/(pow(qw[1][5]-pow(qw[1][0],2),0.5)\*y6sigma)
6. fdc3\_xgxs = fdc3\_cov/(pow(qw[0][6]-pow(qw[0][1],2),0.5)\*y3sigma)
7. fdc6\_xgxs = fdc6\_cov/(pow(qw[0][6]-pow(qw[0][1],2),0.5)\*y6sigma)

四个计算量的结果如下表所示：



## 第5题：

先将三种检测方式以及真实行程速度读出为表，定义MAPE和RMSE函数，然后代入参数，就能求得线圈、浮动车的各个路段的行程速度误差和平均误差，以及AVI数据的干线行程速度估计误差。

定义的RMSE函数如下：

1. #定义RMSE函数
2. **def** rmse(a,b):
3. a,b = np.array(a),np.array(b)
4. pingfang = 0
5. **for** i **in** range(0,len(a)):
6. pingfang = pingfang+pow(a[i]-b[i],2)
7. zhi = pow(pingfang/len(a),0.5)
8. **return** zhi

参数为需要计算的真实数据和估计数据序列，先将其转换成numpy的array格式，便于计算，然后遍历整个数组，计算出其均方根误差。MAPE函数同理，如下：

1. #定义MAPE函数
2. **def** mape(a,b):
3. **import** numpy as np
4. a,b = np.array(a),np.array(b)
5. juedui = 0
6. **for** i **in** range(0,len(a)):
7. juedui = juedui+abs((b[i]-a[i])/b[i])
8. zhi = juedui/len(a)
9. **return** zhi

不过这里需要注意的是，由于需要比上真实速度，所以a,b有先后顺序。a为估计的行程速度，而b是真实的行程速度。

这里，为简单起见，再定义一个将sheet表中的一列数据读出的函数，如下所示：

1. #定义一个将sheet表中一列数据读出的函数
2. **def** duchu(datax,lie,hangf,hange):
3. dedao = []
4. **for** i **in** range(hangf,hange+1):
5. dedao.append(datax.cell(i,lie).value)
6. **return** dedao

然后读取后，利用上述定义的误差函数，可以计算结果，举例说明，如计算线圈的RMSE误差的代码如下：

1. #线圈的估计误差
2. #RMSE
3. total\_ = 0
4. xq\_rmse\_ = []
5. **for** i **in** range(0,6):
6. total\_ = total\_ + rmse(xqsudu[i],zssudu[i])
7. xq\_rmse\_.append(rmse(xqsudu[i],zssudu[i]))
8. xq\_rmse = total\_/6

误差的计算结果如下表：



## 第6题：

自选方法对线圈、浮动车和AVI检测数据进行融合，估计每个路段的行程速度。AVI数据是干线的行程速度，我们将其按照线圈速度进行加权，分为六个路段的行程速度，根据线圈数据计算出每一路段的时间，AVI数据计算出总的行程时间，利用的是6个路段的总长度，而不是干线的总长度，然后根据线圈计算出的每路段的时间进行加权，计算出AVI数据每个路段的时间，用各路段长度除以时间得到AVI估计出的各路段速度。代码实现如下：

1. #每路段长度和总长度
2. length\_l = [0.507,0.687,0.490,0.600,0.267,0.417,2.968]
3. #建立avi的空集
4. avi = np.zeros([178,6])
5. **for** i **in** range(3,171):
6. #每行的每个路段的时间及总时间
7. t\_ = [0,0,0,0,0,0]
8. t\_sum= 0
9. **for** j **in** range(0,6):
10. t\_[j]=length\_l[j]/data1.cell(i,j+3).value
11. t\_sum = t\_sum+t\_[j]
12. #按照线圈数据的时间分配avi的时间
13. #avi速度得到的路段行程时间
14. t\_avi = length\_l[6]/data3.cell(i,3).value
15. **for** j **in** range(0,6):
16. avi[i-3,j]=t\_avi\*(t\_[j]/t\_sum)
17. avi[i-3,j]=length\_l[j]/avi[i-3,j]

然后对三种数据进行融合，利用的是基于加权平均法的数据级的融合方法。关键在于加权的权重计算，我们利用预测精度来定义权重。前10小时数据作为训练集（就是历史数据），最后4小时作为测试数据。将历史数据（前10小时）三种检测方法估计得到的行程速度数据与真实的行程数据计算MAPE误差，计算出三种检测方式6个路段的平均绝对百分误差，用1减去该误差，作为每种检测方式的权重，即是平均误差越小，权重越大，计算过程如下：

1. #权重系数的计算
2. quanzhong1 = []
3. **for** i **in** range(0,6):
4. quanzhong1.append(1-mape(xqsudu[i][0:120],zssudu[i][0:120]))
5. quanzhong2 = []
6. **for** i **in** range(0,6):
7. quanzhong2.append(1-mape(fdcsudu[i][0:120],zssudu[i][0:120]))
8. quanzhong3 = []
9. **for** i **in** range(0,6):
10. quanzhong3.append(1-mape(avi[0:120,i],zssudu[i][0:120]))

计算结果如下表：



下面对其进行加权融合，选择了两种方法，一种是直接三种数据均用于融合，按照上述的权重进行计算，另一种是对每个路段的权重进行比较，得出较高的两个权重，利用这两种监测数据进行融合，从上表可以看出每个路段的权重比较结果并不完全一样，三种检测数据都得到了利用。先定义这两种方法的预测函数，然后分别计算各路段的mape和平均mape。具体过程如下：

定义函数：

1. #定义预测函数，预测方法1是直接三种数据融合
2. **def** yuce1(luduan,xq,fdc,avi):
3. zong = quanzhong1[luduan-1]+quanzhong2[luduan-1]+quanzhong3[luduan-1]
4. zhi = xq\*(quanzhong1[luduan-1]/zong)+fdc\*(quanzhong2[luduan-1]/zong)\
5. +avi\*(quanzhong3[luduan-1]/zong)
6. **return** zhi
8. #预测方法2是将每个路段的权重进行筛选，选择出最高的两个，将其数据进行融合
9. **def** yuce2(luduan,xq,fdc,avi):
10. quanzhong = [quanzhong1[luduan-1],quanzhong2[luduan-1],quanzhong3[luduan-1]]
11. paoqi = quanzhong.index(min(quanzhong))
12. **if** paoqi== 0:
13. he = quanzhong2[luduan-1]+quanzhong3[luduan-1]
14. zhi=fdc\*(quanzhong2[luduan-1]/he)+avi\*(quanzhong3[luduan-1]/he)
15. **elif** paoqi==1:
16. he = quanzhong1[luduan-1]+quanzhong3[luduan-1]
17. zhi=xq\*(quanzhong2[luduan-1]/he)+avi\*(quanzhong3[luduan-1]/he)
18. **elif** paoqi==2:
19. he = quanzhong1[luduan-1]+quanzhong2[luduan-1]
20. zhi=xq\*(quanzhong2[luduan-1]/he)+fdc\*(quanzhong3[luduan-1]/he)
21. **return** zhi

利用上述的函数对最后四小时进行融合并计算误差，得到如下误差：



从上面的误差和权重比较可以看出，我们可以得到更准确的确定方法，当其计算得出的权重（也就是准确度）大于0.8时，就都使用，如果三个中不是都大于0.8时，就抛弃最小的准确度，利用较高的两种监测数据进行融合，得到第三种融合方法，定义函数如下：

1. #预测方法3是当其准确度都不小于0.8时，采用该检测数据
2. #但如果最小值小于0.8时，就选取较高的两种检测数据融合
3. **def** yuce3(luduan,xq,fdc,avi):
4. quanzhong = [quanzhong1[luduan-1],quanzhong2[luduan-1],quanzhong3[luduan-1]]
5. paoqi = quanzhong.index(min(quanzhong))
6. **if** min(quanzhong) >= 0.8:
7. he = quanzhong[0]+quanzhong[1]+quanzhong[2]
8. zhi=xq\*(quanzhong1[luduan-1]/he)+fdc\*(quanzhong2[luduan-1]/he)\
9. +avi\*(quanzhong3[luduan-1]/he)
10. **elif** min(quanzhong) < 0.8  **and**  paoqi== 0:
11. he = quanzhong2[luduan-1]+quanzhong3[luduan-1]
12. zhi=fdc\*(quanzhong2[luduan-1]/he)+avi\*(quanzhong3[luduan-1]/he)
13. **elif** min(quanzhong) < 0.8  **and**  paoqi==1:
14. he = quanzhong1[luduan-1]+quanzhong3[luduan-1]
15. zhi=xq\*(quanzhong2[luduan-1]/he)+avi\*(quanzhong3[luduan-1]/he)
16. **elif** min(quanzhong) < 0.8  **and**  paoqi==2:
17. he = quanzhong1[luduan-1]+quanzhong2[luduan-1]
18. zhi=xq\*(quanzhong2[luduan-1]/he)+fdc\*(quanzhong3[luduan-1]/he)
19. **return** zhi

方法3既是结合了方法1和方法2，将两种方法中预测较好的合在一起，得到的最佳效果，同时确定判断依据，当其预测精度最小值小于0.8时，再抛弃掉最差的预测精度。得到的误差如下，相比单独用线圈或浮动车的判断的6个路段平均百分误差要小：

