道路交通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

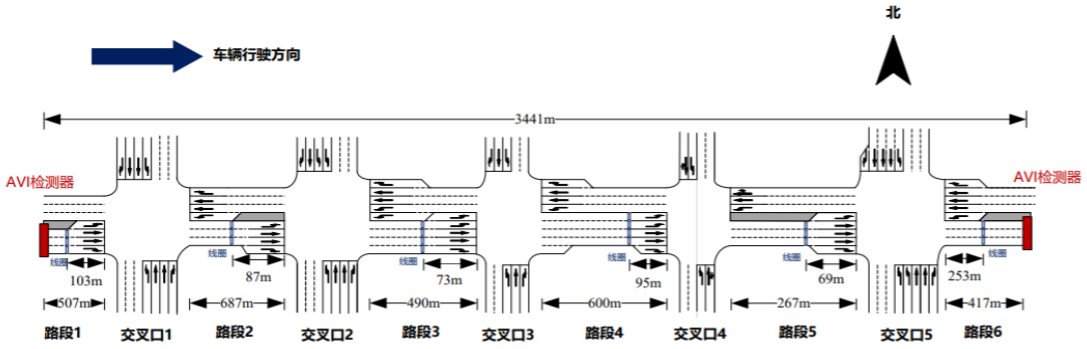


图 1 干道基本示意图

	路段1	路段2	路段3	路段4	路段5	路段6	合计
长度/m	507	687	490	600	267	417	2968
占比	0.1708	0.2315	0.1651	0.2022	0.0900	0.1405	1

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：

```
pingjun = data4.cell(i,3).value*0.17082+data4.cell(i,4).value*0.23147\
          +data4.cell(i,5).value*0.16509+data4.cell(i,6).value*0.20216\
          +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：

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

计算结果如下：

算术平均值	39.69791
中列数	39.43752
中位数	39.62541
标准差	2.45031
变异系数	0.061724
最大值	45.38376
最小值	33.49128
样本数	168

（2）五分位数的计算方法与上述一致：

```
wufen = []
wufen.append(mins)
wufen.append(np.percentile(speed_g,25)) 利用 percentile 函数，25 表示 25%
wufen.append(zhongwei)
wufen.append(np.percentile(speed_g,75))
wufen.append(maxs)
```

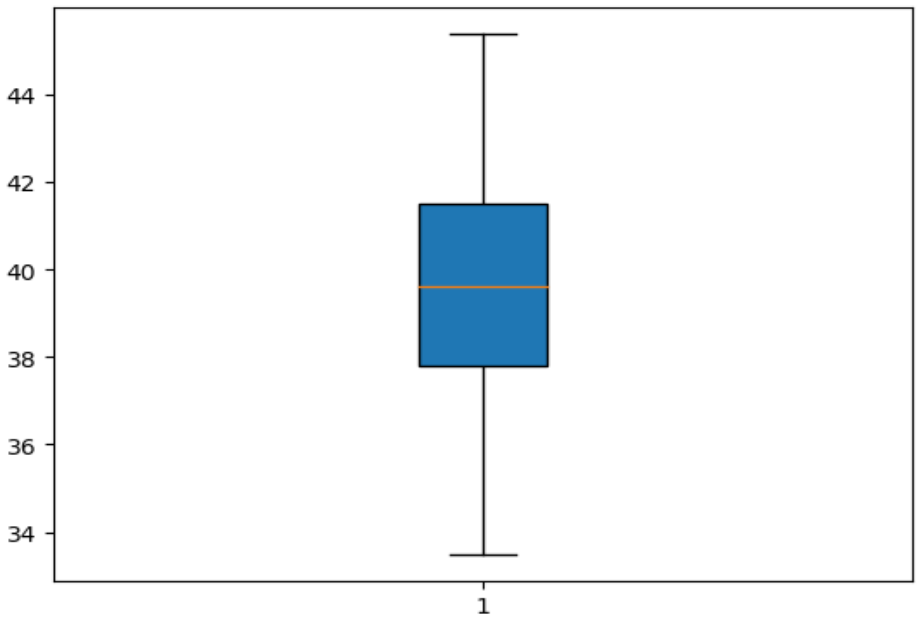
计算结果如下：

五分位数	速度km/h
MIN	33.49128
Q1	37.78937
Median	39.62541
Q3	41.49397
MAX	45.38376

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：

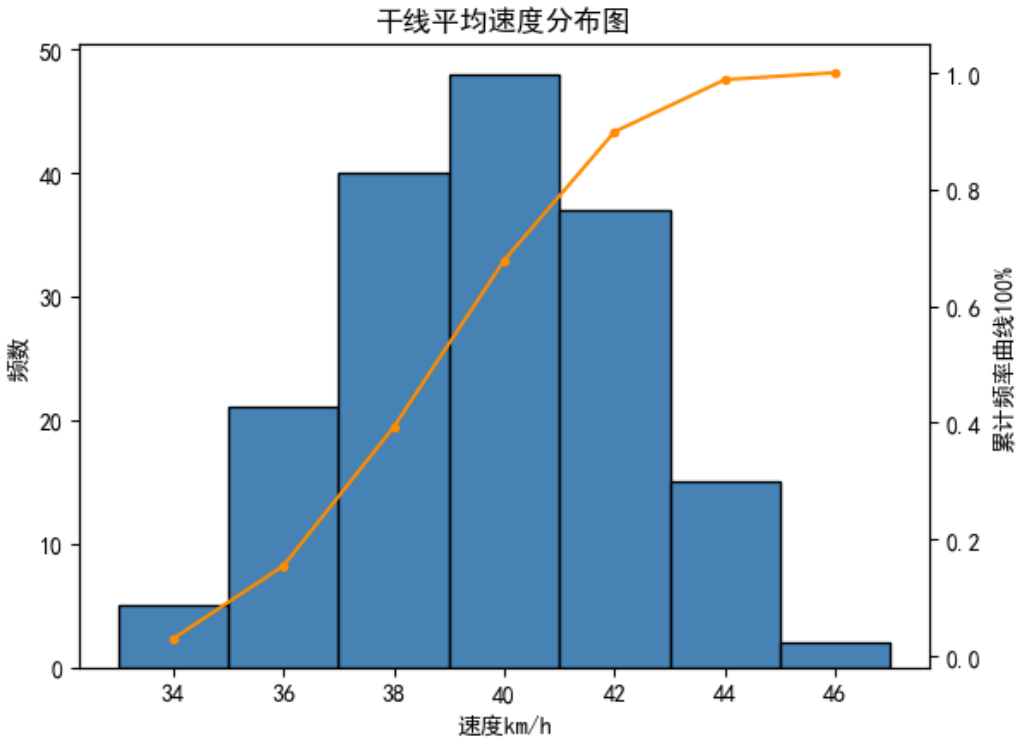
```
1.  plt.rcParams['font.sans-serif']=['SimHei']
2.  plt.rcParams['axes.unicode_minus'] = False  用于消除汉字标题乱码的问题
3.  fig = plt.figure()
```

```
ax = fig.add_subplot(111)
ax.hist(speed_g,bins=[33,35,37,39,41,43,45,47],color='steelblue',edgecolor='black')
ax.set_title('干线平均速度分布图')
ax.set_xlabel('速度 km/h')
ax.set_ylabel('频数')
x=[34,36,38,40,42,44,46]
y=[5/168,26/168,66/168,114/168,151/168,166/168,168/168]
#双 y 轴
ax2 = ax.twinx()
ax2.plot(x,y,'.-',color='darkorange')
ax2.set_ylabel('累计频率曲线 100%')
plt.show()
```

分组后的频数表如下所示：

区间km/h	频数
[33-35)	5
[35-37)	21
[37-39)	40
[39-41)	48
[41-43)	37
[43-45)	15
[45-47)	2
合计	168

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：

```
sudu=[]
liuliang=[]
zhanyoulv=[]
for i in range(3,171):
    for j in range(3,9):
        if data1.cell(i,j).value is not None:
            sudu.append(data1.cell(i,j).value)
    for j in range(9,15):
        if data1.cell(i,j).value is not None:
            liuliang.append(data1.cell(i,j).value)
    for j in range(15,21):
        if data1.cell(i,j).value is not None:
            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。

```
#通过五分位数法进行异常数据剔除
for i in range(3,171):
    for j in range(3,9):
        if data1.cell(i,j).value is not None:
            if data1.cell(i,j).value<sudumin or data1.cell(i,j).value>suduma
x:
                data1.cell(i,j).value=None
    for j in range(9,15):
        if data1.cell(i,j).value is not None:
            if data1.cell(i,j).value<liulmin or data1.cell(i,j).value>liulma
x:
                data1.cell(i,j).value=None
    for j in range(15,21):
        if data1.cell(i,j).value is not None:
            if data1.cell(i,j).value<zhanylmin or data1.cell(i,j).value>zhan
ylmax:
```

```
                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：$S_t = \frac{\sum_{i=t-1}^{t-n} W_i X_i}{\sum_{i=1}^{n} W_i}$，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。

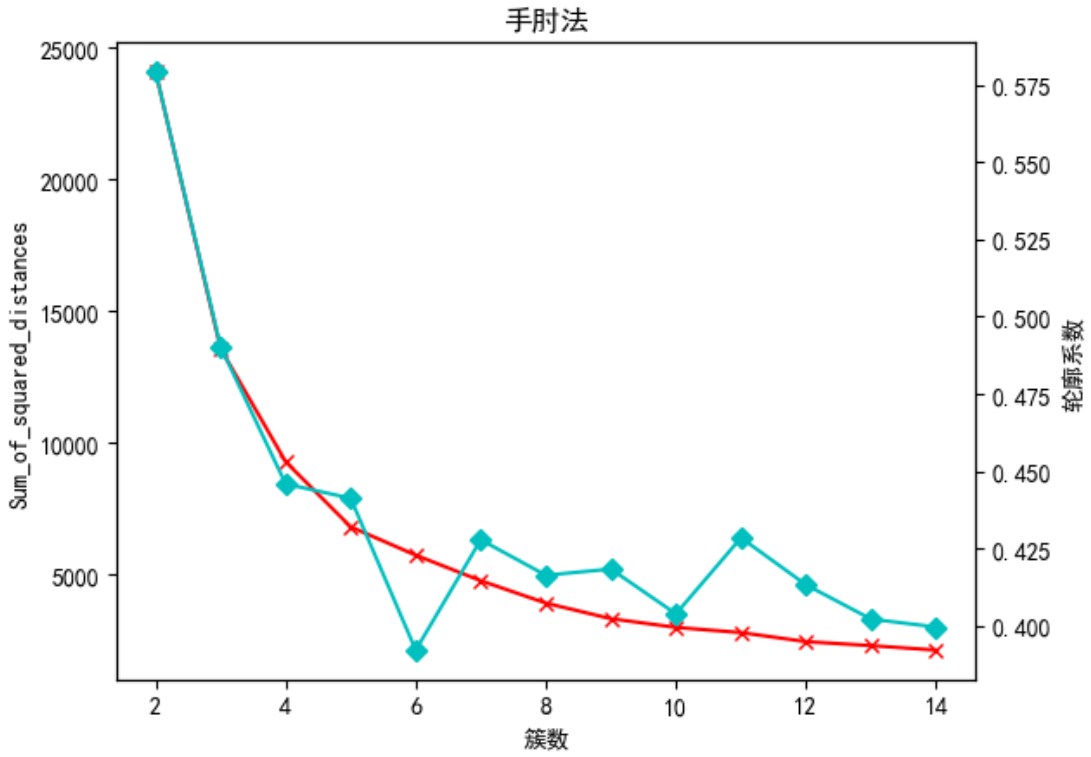
实现的代码如下，主要是循环嵌套循环，同时利用 if 条件进行判定，其中当其内容为空时，进行修补，修补中，利用行数和列数对其完成不同的修补方法，其中，流量转换为整数：

```
for i in range(3,171):
    for j in range(3,21):
        if data1.cell(i,j).value is None:
            if i==3:
                data1.cell(i,j).value=(data1.cell(i+1,j).value+data1.cell(i+2,j).value)/2
            #线性内插法
            if 3<i<8:
                #流量存为整数
                if 8<j & j<15:
                    data1.cell(i,j).value=int(data1.cell(i-1,j).value+data1.cell(i+1,j).value)
                #占有率和速度存为浮点型
                else:
                    data1.cell(i,j).value=(data1.cell(i-1,j).value+data1.cell(i+1,j).value)
            #加权移动平均法
            if i>7:
                if 8<j & j<15:
```

```
                    data1.cell(i,j).value=int((1*data1.cell(i-
5,j).value+2*data1.cell(i-4,j).value\
                            +3*data1.cell(i-3,j).value+4*data1.cell(i-
2,j).value\
                            +5*data1.cell(i-1,j).value)/(5+4+3+2+1))
                else:
                    data1.cell(i,j).value=(1*data1.cell(i-
5,j).value+2*data1.cell(i-4,j).value\
                            +3*data1.cell(i-3,j).value+4*data1.cell(i-
2,j).value\
                            +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：

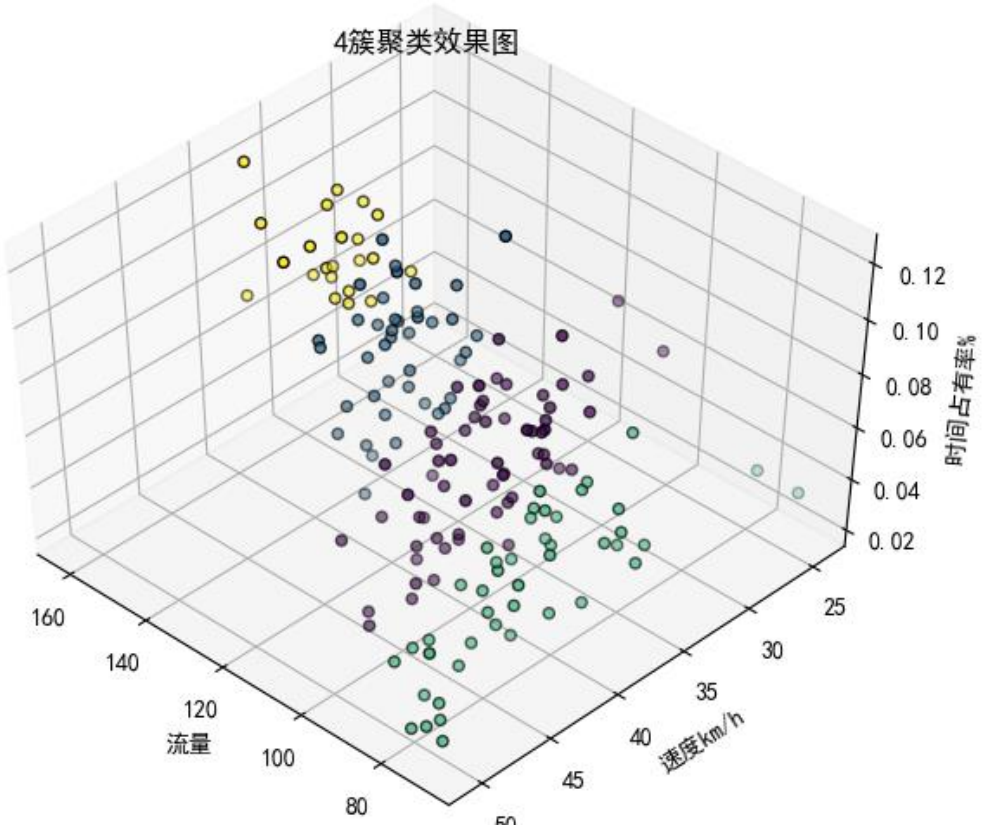


从上图可以看出，选择簇数为 4 最合适，轮廓系数相对较高，且 SSE 以后并

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：

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
estimators = KMeans(n_clusters=4)
estimators.fit(shuzu4)  进行聚类
ax.scatter(shuzu4[:,1],shuzu4[:,0],shuzu4[:,2],
           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：

```
cu1=[]
cu2=[]
cu3=[]
cu4=[]
```

```
for i in range(0,168):
    if estimators.labels_[i]==0:
        cu1.append(shuzu4[i])
    elif estimators.labels_[i]==1:
        cu2.append(shuzu4[i])
    elif estimators.labels_[i]==2:
        cu3.append(shuzu4[i])
    elif estimators.labels_[i]==3:
        cu4.append(shuzu4[i])

#将嵌套列表转换为 numpy 数组，支持多维切片
cu1 = np.array(cu1)
cu2 = np.array(cu2)
cu3 = np.array(cu3)
cu4 = np.array(cu4)
#计算统计特征
tjtz1 = np.zeros([5,3])
for i in range(0,3):
    tjtz1[0][i]=np.mean(cu1[:,i])
    tjtz1[1][i]=np.var(cu1[:,i],ddof=1)
    tjtz1[2][i]=max(cu1[:,i])
    tjtz1[3][i]=min(cu1[:,i])
    tjtz1[4][i]=len(cu1[:,i])
```

得到的统计指标如下表所示：

路段4聚类分析	统计指标	速度km/h	流量pcu	占有率
簇1	均值	38.0568	101.714	0.060955
	方差	18.4481	33.0078	0.000232
	最大值	47.4	112	0.092
	最小值	25.8	93	0.032
	样本数	56	56	56
簇2	均值	37.1652	141.933	0.099189
	方差	4.43461	55.8575	0.000128
	最大值	41.6	162	0.123
	最小值	33.8	133	0.083
	样本数	30	30	30
簇3	均值	37.6568	122.513	0.075557
	方差	7.13052	34.2564	0.000275
	最大值	45.6	131	0.118
	最小值	32.9	113	0.047
	样本数	39	39	39
簇4	均值	40.6698	83.2326	0.041161
	方差	36.8474	36.1351	0.000154
	最大值	50.5	92	0.068
	最小值	24.4	70	0.021
	样本数	43	43	43

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：

```
#线圈路段 3 和 6 欧几里得距离
pf13 = 0
pf16 = 0
for i in range(3,171):
    pf13 = pf13 +pow((data1.cell(i,5).value-data4.cell(i,5).value),2)
    pf16 = pf16 +pow((data1.cell(i,8).value-data4.cell(i,8).value),2)
xq3_ojld = pow(pf13,0.5)
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：

```
def dtw_distance(ts_a,ts_b,d=lambda x,y:abs(x-y),mww=10000):
    import numpy as np
    #Create cost matrix via broadcasting with large int
    ts_a,ts_b=np.array(ts_a),np.array(ts_b)
    M,N=len(ts_a),len(ts_b)
    cost=np.ones((M,N))

    #Initialize the first row and column
    cost[0,0]=d(ts_a[0],ts_b[0])
    for i in range(1,M):
        cost[i,0] = cost[i-1,0] + d(ts_a[i],ts_b[0])

    for j in range(1,N):
        cost[0,j] = cost[0,j-1] + d(ts_a[0],ts_b[j])

    #Populate rest of cost matrix within window
    for i in range(1,M):
        for j in range(max(1,i-mww),min(N,i+mww)):
            choices = cost[i-1,j-1],cost[i,j-1],cost[i-1,j]
```

```
            cost[i,j] = min(choices)+d(ts_a[i],ts_b[j])


    #Return DTW distance given window
    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：

```
#提出各检测器各路段的速度序列
xq3sudu = []
xq6sudu = []
fdc3sudu = []
fdc6sudu = []
zs3sudu = []
zs6sudu = []
for i in range(3,171):
    xq3sudu.append(data1.cell(i,5).value)
    xq6sudu.append(data1.cell(i,8).value)
    fdc3sudu.append(data2.cell(i,5).value)
    fdc6sudu.append(data2.cell(i,8).value)
    zs3sudu.append(data4.cell(i,5).value)
    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：

$$Cov(X,Y) = E(XY) - E(X)E(Y)$$

4.相关系数，利用相关系数的公式如下：

$$r(X,Y) = \frac{Cov(X,Y)}{\sqrt{E(X^2) - E^2(X)}\sigma_Y}$$

主要利用 numpy 的函数，代码如下：

```
#计算相关系数
```

```
y3sigma = pow(qw[0][7]-pow(qw[0][2],2),0.5)
y6sigma = pow(qw[1][7]-pow(qw[1][2],2),0.5)
xq3_xgxs = xq3_cov/(pow(qw[0][5]-pow(qw[0][0],2),0.5)*y3sigma)
xq6_xgxs = xq6_cov/(pow(qw[1][5]-pow(qw[1][0],2),0.5)*y6sigma)
fdc3_xgxs = fdc3_cov/(pow(qw[0][6]-pow(qw[0][1],2),0.5)*y3sigma)
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：

与真实数据分析		相异性		相关性	
		欧几里得距离	DTW距离	协方差	相关系数
路段3	线圈	159.8228354	1408.633	49.10525	0.755421
	浮动车	78.14300992	564.3	69.82874	0.829612
路段6	线圈	8.804942148	42.972	0.057054	0.354823
	浮动车	7.848566748	59.7	0.049945	0.023128

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：

```
#定义 RMSE 函数
def rmse(a,b):
    a,b = np.array(a),np.array(b)
    pingfang = 0
    for i in range(0,len(a)):
        pingfang = pingfang+pow(a[i]-b[i],2)
    zhi = pow(pingfang/len(a),0.5)
    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：

```
#定义 MAPE 函数
def mape(a,b):
    import numpy as np
    a,b = np.array(a),np.array(b)
```

```
    juedui = 0
    for i in range(0,len(a)):
        juedui = juedui+abs((b[i]-a[i])/b[i])
    zhi = juedui/len(a)
    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：

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

```
#线圈的估计误差
#RMSE
total_ = 0
xq_rmse_ = []
for i in range(0,6):
    total_ = total_ + rmse(xqsudu[i],zssudu[i])
    xq_rmse_.append(rmse(xqsudu[i],zssudu[i]))
xq_rmse = total_/6
```

误差的计算结果如下表：

行程速度km/h		平均估计误差	路段1	路段2	路段3	路段4	路段5	路段6
线圈	RMSE	8.096566634	12.81396	2.409045	12.3306	5.442849	14.90363	0.679316
	MAPE	0.272362555	0.405379	0.033762	0.393364	0.139181	0.656698	0.005792
浮动车	RMSE	5.426162247	7.418578	5.575195	6.028864	4.28744	8.641366	0.60553
	MAPE	0.125735574	0.188561	0.083258	0.13942	0.096727	0.238329	0.008117
AVI	RMSE	4.992558103						
	MAPE	0.109769455						

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：

```
#每路段长度和总长度
length_l = [0.507,0.687,0.490,0.600,0.267,0.417,2.968]
#建立 avi 的空集
avi = np.zeros([178,6])
for i in range(3,171):
    #每行的每个路段的时间及总时间
    t_ = [0,0,0,0,0,0]
    t_sum= 0
    for j in range(0,6):
        t_[j]=length_l[j]/data1.cell(i,j+3).value
        t_sum = t_sum+t_[j]
    #按照线圈数据的时间分配 avi 的时间
    #avi 速度得到的路段行程时间
    t_avi = length_l[6]/data3.cell(i,3).value
    for j in range(0,6):
        avi[i-3,j]=t_avi*(t_[j]/t_sum)
        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：

```
#权重系数的计算
quanzhong1 = []
for i in range(0,6):
    quanzhong1.append(1-mape(xqsudu[i][0:120],zssudu[i][0:120]))
quanzhong2 = []
for i in range(0,6):
    quanzhong2.append(1-mape(fdcsudu[i][0:120],zssudu[i][0:120]))
quanzhong3 = []
for i in range(0,6):
    quanzhong3.append(1-mape(avi[0:120,i],zssudu[i][0:120]))
```

计算结果如下表：

权重	线圈	浮动车	AVI
路段1	0.613506	0.81666	0.859316
路段2	0.96605	0.914006	0.812299
路段3	0.612771	0.86269	0.839397
路段4	0.862651	0.904414	0.888752
路段5	0.327934	0.770993	0.650937
路段6	0.994273	0.992095	0.801053

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：

```
#定义预测函数，预测方法 1 是直接三种数据融合
def yuce1(luduan,xq,fdc,avi):
    zong = quanzhong1[luduan-1]+quanzhong2[luduan-1]+quanzhong3[luduan-1]
    zhi = xq*(quanzhong1[luduan-1]/zong)+fdc*(quanzhong2[luduan-1]/zong)\
        +avi*(quanzhong3[luduan-1]/zong)
    return zhi

#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
def yuce2(luduan,xq,fdc,avi):
    quanzhong = [quanzhong1[luduan-1],quanzhong2[luduan-
1],quanzhong3[luduan-1]]
    paoqi = quanzhong.index(min(quanzhong))
    if paoqi== 0:
        he = quanzhong2[luduan-1]+quanzhong3[luduan-1]
        zhi=fdc*(quanzhong2[luduan-1]/he)+avi*(quanzhong3[luduan-1]/he)
    elif paoqi==1:
        he = quanzhong1[luduan-1]+quanzhong3[luduan-1]
        zhi=xq*(quanzhong2[luduan-1]/he)+avi*(quanzhong3[luduan-1]/he)
```

```
    elif paoqi==2:
        he = quanzhong1[luduan-1]+quanzhong2[luduan-1]
        zhi=xq*(quanzhong2[luduan-1]/he)+fdc*(quanzhong3[luduan-1]/he)
    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：

mape	方法1	方法2
路段1	0.125969	0.085535
路段2	0.070039	0.093016
路段3	0.162474	0.11894
路段4	0.066062	0.088043
路段5	0.195921	0.161277
路段6	0.05665	0.093854
平均	0.112852	0.106778

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：

```
#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
#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
def yuce3(luduan,xq,fdc,avi):
    quanzhong = [quanzhong1[luduan-1],quanzhong2[luduan-
1],quanzhong3[luduan-1]]
    paoqi = quanzhong.index(min(quanzhong))
    if min(quanzhong) >= 0.8:
        he = quanzhong[0]+quanzhong[1]+quanzhong[2]
        zhi=xq*(quanzhong1[luduan-1]/he)+fdc*(quanzhong2[luduan-1]/he)\
        +avi*(quanzhong3[luduan-1]/he)
    elif min(quanzhong) < 0.8  and  paoqi== 0:
        he = quanzhong2[luduan-1]+quanzhong3[luduan-1]
        zhi=fdc*(quanzhong2[luduan-1]/he)+avi*(quanzhong3[luduan-1]/he)
    elif min(quanzhong) < 0.8  and  paoqi==1:
        he = quanzhong1[luduan-1]+quanzhong3[luduan-1]
        zhi=xq*(quanzhong2[luduan-1]/he)+avi*(quanzhong3[luduan-1]/he)
    elif min(quanzhong) < 0.8  and  paoqi==2:
        he = quanzhong1[luduan-1]+quanzhong2[luduan-1]
        zhi=xq*(quanzhong2[luduan-1]/he)+fdc*(quanzhong3[luduan-1]/he)
    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：

mape	方法1	方法2	方法3
路段1	0.125969	0.085535	0.085535
路段2	0.070039	0.093016	0.070039
路段3	0.162474	0.11894	0.11894
路段4	0.066062	0.088043	0.066062
路段5	0.195921	0.161277	0.161277
路段6	0.05665	0.093854	0.05665
平均	0.112852	0.106778	0.093084