MR定位建模——分类

1953871 邓泉

1. **获取数据**

读取2G\_data.csv的MR数据与2G\_gongcan.csv的工参表数据，进行匹配工参，使得在MR数据的基础上增添了邻接基站的GPS经纬度位置。

|  |
| --- |
| 1. *# 读取数据* 2. sf = pd.read\_csv(readpath + "2G\_data.csv", encoding = 'gb2312') 3. *#匹配工参* 4. sf\_eng = pd.read\_csv(readpath + "2G\_gongcan.csv", encoding = 'gb2312') 5. for i in range(1, 8): 6. sf = sf.merge(sf\_eng, left\_on = ['RNCID\_%d'% i, 'CellID\_%d'% i], 7. right\_on = ['RNCID', 'CellID'], how = 'left', suffixes = ('', '%d'% i)) |

1. **数据集转换**
   1. **异常值处理**

数据集存在空缺值NaN，其来源有两种：

* + - * MR中邻接基站的数量为从1到7不等，邻接基站个数不够7的MR数据在剩余基站属性栏中值为NaN
      * 基站在工参表中无匹配项 --> 工参匹配时该基站的坐标为NaN

考虑到：

* RSSI：表示信号的强度，随距离的增大而衰减，通常为负值，该值越接近零说明信号强度越高。
* SignalLevel：表示信号幅度，取值[0,4]，该值越高说明信号幅度越大。
* GPS坐标（Longitude, Latitude）：本数据集中的GPS坐标都位于(121, 31)附近。

因此，用一个绝对值足够大的负数-999填充NaN。

* 1. **划分栅格**

利用所有MR数据的GPS labels，确定一个整体的位置范围，并将该范围转换成一个大的矩形（所有数据的位置标签均落入该矩形）。再将该矩形划分成若干个等大的小正方形栅格。代码中选用了边长为20m的小正方形栅格。

|  |
| --- |
| 1. grider = RoadGrid(sf[['Longitude', 'Latitude']].values, grid\_size=20) 2. sf['grid\_id'] = grider.transform(sf[['Longitude', 'Latitude']].values, False) |

至此完成了MR 数据的 GPS labels到栅格 ID labels的转换，结果中新增一列属性——grid\_id，标识该条MR数据所属的栅格编号，作为模型训练的输出。

* 1. **特征选择**

选择各邻接基站的信号及坐标信息——SignalLevel,RSSI, Longitude,Latitude作为输入的特征；栅格编号grid\_id作为模型的输出。

* 1. **数据集划分**

调用sklearn.model\_selection中的train\_test\_split()，随机选取80%的数据记录作为训练集，剩余20%作为测试集。

* 1. **特征工程**

尝试了以下无量纲化和特征降维方法的单独及组合使用：

* + - 无量纲化：标准化、归一化；
    - 特征降维：PCA降维、层次聚类降维；

结论是：**PCA降维效果最差，其余方法单独使用或组合使用效果差不多，单独使用的效果略微好一点，最终选择采用单纯的归一化。**

部分（RF）交叉验证结果如下：

|  |  |
| --- | --- |
| **归一化** | **标准化** |
| **PCA降维(n\_components=0.95)** | **层次聚类降维(n\_clusters=20)** |
| **归一化+层次聚类降维** | **标准化+层次聚类降维** |

|  |
| --- |
| 1. *# 特征预处理：归一化* 2. from sklearn.preprocessing import MinMaxScaler 3. scaler = MinMaxScaler() 4. X\_train = scaler.fit\_transform(X\_train) 5. X\_test = scaler.transform(X\_test) |

1. **模型训练**
   1. **决策树**

|  |
| --- |
| 1. model ='Classifier\_DecisionTree' 2. from sklearn.tree import DecisionTreeClassifier, export\_graphviz 3. estimator = DecisionTreeClassifier(criterion='entropy', max\_depth=None, 4. min\_samples\_split=2, min\_samples\_leaf=1, max\_features=None, max\_leaf\_nodes=None, 5. min\_impurity\_decrease=0.0) 6. estimator.fit(X\_train, Y\_train['grid\_id']) |

* 1. **高斯朴素贝叶斯**

|  |
| --- |
| 1. model ='Classifier\_GaussianNB' 2. from sklearn.naive\_bayes import GaussianNB 3. estimator = GaussianNB() 4. estimator.fit(X\_train, Y\_train['grid\_id']) |

* 1. **KNN**

|  |
| --- |
| 1. model ='Classifier\_KNN' 2. from sklearn.neighbors import KNeighborsClassifier 3. estimator = KNeighborsClassifier() 4. estimator.fit(X\_train, Y\_train['grid\_id']) |

* 1. **随机森林**

|  |
| --- |
| 1. model ='Classifier\_RF’ 2. from sklearn.ensemble import RandomForestClassifier 3. estimator = RandomForestClassifier(n\_estimators=10) 4. estimator.fit(X\_train, Y\_train['grid\_id']) |

1. **模型评估**
   1. **分类器评估**

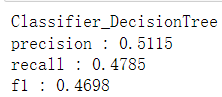
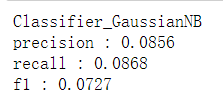
用上述训练好的模型预测测试集数据的栅格编号，记录在pred\_grid。

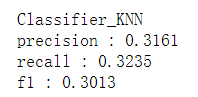
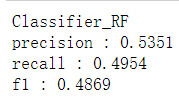
|  |
| --- |
| 1. pred\_grid = estimator.predict(X\_test) |

计算 precision，recall 和 f-measurement 来评价分类器的好坏。

|  |
| --- |
| 1. from sklearn.metrics import precision\_recall\_fscore\_support, precision\_score, recall\_score, f1\_score 2. print(model) 3. print('precision :', precision\_score(Y\_test['grid\_id'], pred\_grid, average='weighted')) 4. print('recall :', recall\_score(Y\_test['grid\_id'], pred\_grid, average='weighted')) 5. print('f1 :', f1\_score(Y\_test['grid\_id'], pred\_grid, average='weighted')) |

四种分类器的评价结果如下：

  ****

**对比分析：**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **precison** | **recall** | **f1** |
| **决策树** | 0.5115 | 0.4785 | 0.4698 |
| **高斯朴素贝叶斯** | 0.0856 | 0.0868 | 0.0727 |
| **K近邻** | 0.3161 | 0.3235 | 0.3013 |
| **随机森林** | 0.5351 | 0.4954 | 0.4869 |

随机森林的precison, recall, f1均为最高，是准确率最高的模型，决策树与其最接近。

**准确性排序：随机森林 > 决策树 > K近邻 >> 高斯朴素贝叶斯**

* 1. **误差分析**

首先从地理坐标系（球面坐标；参考平面：地是椭球面；坐标单位：经纬度）投影到投影坐标系（平面坐标；参考平面地是水平面；坐标单位：米、千米等），即将不规则的地球曲面转换为平面，然后度量两坐标点间的距离。

|  |
| --- |
| 1. def rad(d): 2. return d \* math.pi / 180.0 3. *# gps两点间距离（单位为米）* 4. def distance(true\_pt, pred\_pt): 5. lat1 = float(true\_pt[1]) 6. lng1 = float(true\_pt[0]) 7. lat2 = float(pred\_pt[1]) 8. lng2 = float(pred\_pt[0]) 9. radLat1 = rad(lat1) 10. radLat2 = rad(lat2) 11. a = radLat1 - radLat2 12. b = rad(lng1) - rad(lng2) 13. s = 2 \* math.asin(math.sqrt(math.pow(math.sin(a/2),2) + 14. math.cos(radLat1)\*math.cos(radLat2)\*math.pow(math.sin(b/2),2))) 15. s = s \* 6378.137 16. s = round(s \* 10000) / 10 17. return s |

将分类器预测的栅格编号pred\_grid转换成真实的GPS label——该栅格的中心点坐标pred\_loc，并用该中心点坐标代表预测出的位置进行误差计算。

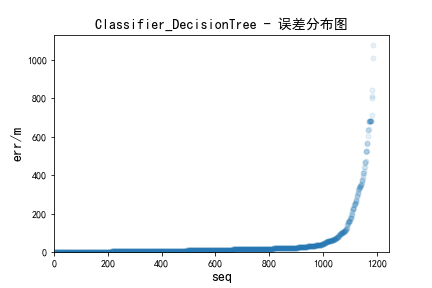
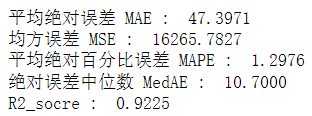
|  |
| --- |
| 1. pred\_loc = np.array([grider.grid\_center[idx] for idx in pred\_grid]) |

调用distance()计算预测坐标与实际坐标的距离作为误差，绘制排序后的误差分布图，计算平均绝对误差MAE、均方误差MSE、平均绝对百分比误差MAPE、绝对误差中位数MedAE、R2\_socre。

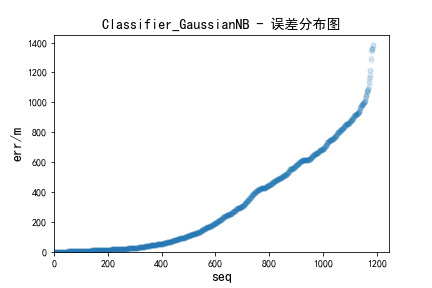
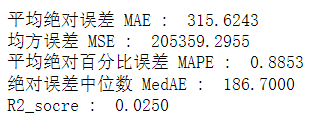
|  |
| --- |
| 1. *# 误差* 2. err = [distance(p,t) for p, t in zip(pred\_loc, Y\_test[['Longitude','Latitude']].values)] 3. *# 绘制误差分布图，其中 x-轴为排序编号，y-轴为对应的误差* 4. err = sorted(err) 5. fig = plt.scatter(range(len(err)), err, s=25, alpha=.1) 6. plt.rcParams['font.sans-serif']=['SimHei'] 7. plt.rcParams['axes.unicode\_minus'] = False 8. plt.title(model + ' - 误差分布图', fontsize=14) 9. plt.xlabel('seq', fontsize=14) 10. plt.ylabel('err/m', fontsize=14) 11. plt.xlim(0, ) 12. plt.ylim(0, ) 13. plt.show() 14. fig.figure.savefig(savepath + model+'/' + model + ' - 误差分布图') 15. from sklearn.metrics import r2\_score 16. print('平均绝对误差 MAE : ', np.mean(err)) 17. print('均方误差 MSE : ', np.std(err)) 18. percentage\_err = np.abs(err - np.mean(err)) / np.mean(err) 19. print('平均绝对百分比误差 MAPE : ', np.mean(percentage\_err)) 20. print('绝对误差中位数 MedAE : ', np.median(err)) 21. print('R2\_socre : ', r2\_score(Y\_test[['Longitude','Latitude']], pred\_loc, multioutput='variance\_weighted')) |

四种模型的误差分析结果如下：

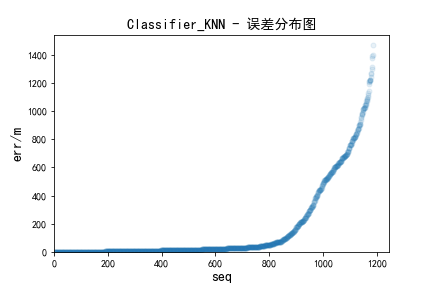
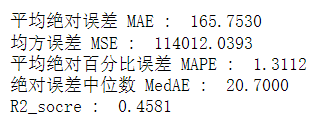
* + 1. **决策树**

 ****

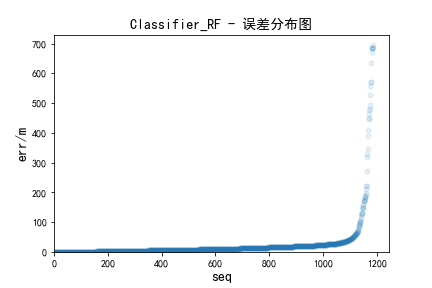
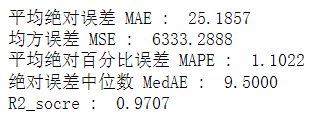
* + 1. **高斯朴素贝叶斯**

 ****

* + 1. **K近邻**

 ****

* + 1. **随机森林**

**对比分析：**

通过观察误差分布图可得到直观的结论：高斯朴素贝叶斯的高误差分布明显最多，K近邻其次；图像上，随机森林和决策树高误差分布的跨度相近，但随机森林的高误差分布部分颜色淡于决策树的，说明此部分分布密度更小，因此随机森林略优于决策树。

**准确性排序：随机森林 > 决策树 > K近邻 > 高斯朴素贝叶斯**

* 1. **地图可视化预测效果**

调用folium库，将实际坐标点（绿色）与预测坐标点（橙色）绘制在地图上。

|  |
| --- |
| 1. import folium 2. *# location 经纬度，list 或者 tuple 格式，顺序为 latitude(纬度), longitude(经度)* 3. *# zoom\_start 缩放值，默认为 10，值越大比例尺越小，地图放大级别越大* 4. tj\_map = folium.Map(location=[31.286363, 121.214786],zoom\_start=14.5) *# 显示地图* 6. *# 添加实际标记：Y\_test[['Longitude','Latitude']].values* 7. for i in Y\_test[['Longitude','Latitude']].values: 8. lat = i[1] 9. lon = i[0] 10. folium.Circle( 11. radius=1, 12. location=[lat, lon], 13. popup='Laurelhurst Park', 14. color='#9AFF9A', *# 绿色* 15. fill=False 16. ).add\_to(tj\_map) 17. *# 添加预测标记：pred\_loc* 18. for i in pred\_loc: 19. lat = i[1] 20. lon = i[0] 21. folium.Circle( 22. radius=1, 23. location=[lat, lon], 24. popup='Laurelhurst Park', 25. color='#FF8C69', *# 橙色* 26. fill=False 27. ).add\_to(tj\_map) 28. tj\_map.save(savepath + model+'/' + model + ' - tj\_map.html') |

四种模型的地图可视化结果如下：

* + 1. **决策树**



[**..\result data\classify\Classifier\_DecisionTree\Classifier\_DecisionTree - tj\_map.html**](../result%20data/classify/Classifier_DecisionTree/Classifier_DecisionTree%20-%20tj_map.html)

* + 1. **高斯朴素贝叶斯**



[**..\result data\classify\Classifier\_GaussianNB\Classifier\_GaussianNB - tj\_map.html**](../result%20data/classify/Classifier_GaussianNB/Classifier_GaussianNB%20-%20tj_map.html)

* + 1. **K近邻**



[**..\result data\classify\Classifier\_KNN\Classifier\_KNN - tj\_map.html**](../result%20data/classify/Classifier_KNN/Classifier_KNN%20-%20tj_map.html)

* + 1. **随机森林**



[**..\result data\classify\Classifier\_RF\Classifier\_RF - tj\_map.html**](file:///D:\学习资料\大三下\数据分析与挖掘（饶卫雄）\1953871+hw1\result%20data\classify\Classifier_RF\Classifier_RF%20-%20tj_map.html)

**对比分析：**

通过观察误差分布图考察模型效果，主要看预测坐标点（橙色）对实际坐标点（绿色）的覆盖效果。随机森林与决策树效果相近，强于K近邻，三者都明显优于高斯朴素贝叶斯。

**准确性排序：随机森林 ≈ 决策树 > K近邻 > 高斯朴素贝叶斯**

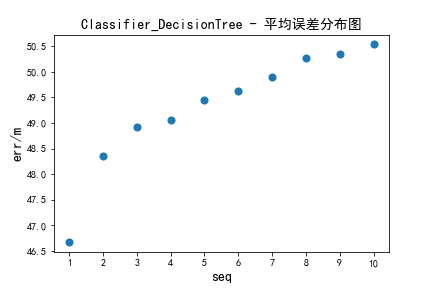
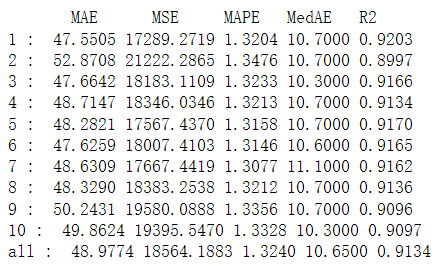
* 1. **交叉验证**

重复10次训练集/测试集的选择和误差计算，求平均误差。绘制平均误差概率分布图，其中x-轴为排序编号，y-轴为对应的平均误差。

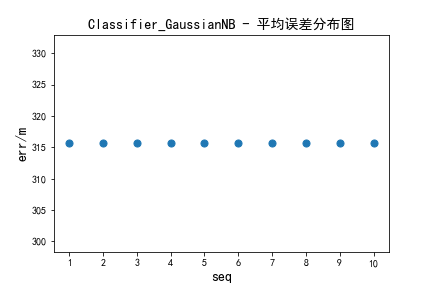
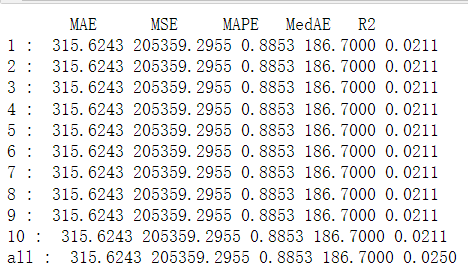
|  |
| --- |
| 1. print('       MAE      MSE     MAPE   MedAE   R2') 2. MAE, MSE, MAPE, MedAE, R2\_score = [], [], [], [], [] 3. *# 10-fold cross validation* 4. for i in range(10): 5. X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=200) 6. *# 特征工程* 7. *# 特征预处理：归一化* 8. from sklearn.preprocessing import MinMaxScaler 9. scaler = MinMaxScaler() 10. X\_train = scaler.fit\_transform(X\_train) 11. X\_test = scaler.transform(X\_test) 12. *# RF* 13. estimator = RandomForestClassifier() 15. estimator.fit(X\_train, Y\_train['grid\_id']) 16. *# 预测的所在栅格编号* 17. pred\_grid= estimator.predict(X\_test) 18. *# 预测的经纬度坐标* 19. pred\_loc = np.array([grider.grid\_center[idx] for idx in pred\_grid]) 20. *# 误差* 21. err = [distance(p,t) for p, t in zip(pred\_loc, Y\_test[['Longitude','Latitude']].values)] 22. err = sorted(err) 23. MAE.append(np.mean(np.abs(err))) 24. MSE.append(np.mean(np.square(err))) 25. percentage\_err = np.abs(err - np.mean(err)) / np.mean(err) 26. MAPE.append(np.mean(percentage\_err)) 27. MedAE.append(np.median(err)) 28. R2\_score.append(r2\_score(Y\_test[['Longitude','Latitude']], pred\_loc, multioutput='uniform\_average')) 29. print('%d : '% (i+1), '%.4f'% np.mean(np.abs(err)), '%.4f'% np.mean(np.square(err)), '%.4f'% np.mean(percentage\_err), 30. '%.4f'% np.median(err), '%.4f'% r2\_score(Y\_test[['Longitude','Latitude']], pred\_loc, multioutput='variance\_weighted')) 32. print('all : ', '%.4f'% np.mean(MAE), '%.4f'% np.mean(MSE), '%.4f'% np.mean(MAPE), 33. '%.4f'% np.mean(MedAE), '%.4f'% np.mean(R2\_score)) 34. *# 绘制平均误差概率分布图，其中 x-轴为排序编号，y-轴为对应的平均误差 Average error distribution plot* 35. MAE = sorted(MAE) 36. fig = plt.scatter(range(1, len(MAE)+1), MAE, s=50) 37. plt.rcParams['font.sans-serif']=['SimHei'] 38. plt.rcParams['axes.unicode\_minus'] = False 39. plt.title(model + ' - 平均误差分布图', fontsize=14) 40. plt.xticks(np.linspace(1,10,10)) 41. plt.xlabel('seq', fontsize=14) 42. plt.ylabel('err/m', fontsize=14) 43. plt.show() 44. fig.figure.savefig(savepath + model+'/' + model + ' - 平均误差分布图') *# 保存* |

四种模型的交叉验证结果如下：

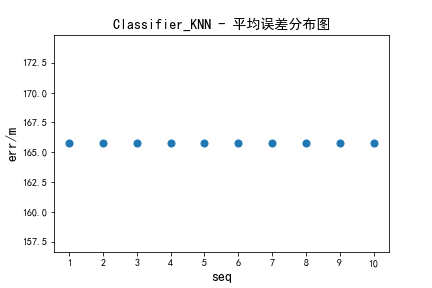
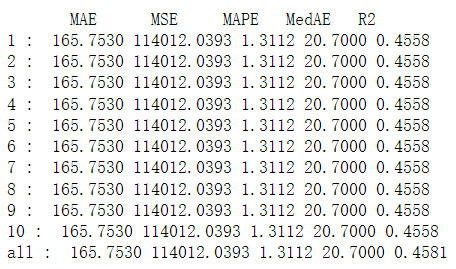
* + 1. **决策树**

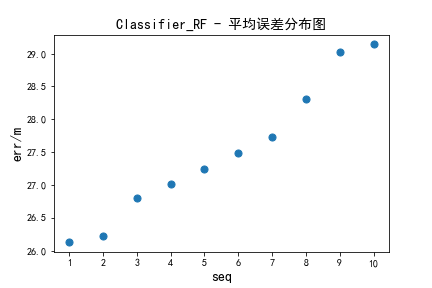
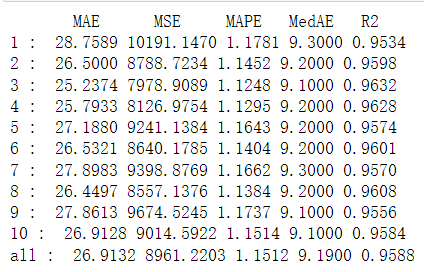
* + 1. **高斯朴素贝叶斯**

****

* + 1. **K近邻**

****

* + 1. **随机森林**

 ****

**对比分析：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **MAE** | **MSE** | **MAPE** | **MedAE** | **R2\_score** |
| **决策树** | 48.9774 | 18564.1883 | 1.3240 | 10.6500 | 0.9134 |
| **高斯朴素贝叶斯** | 315.6243 | 205359.2955 | 0.8853 | 186.7000 | 0.0250 |
| **K近邻** | 165.7530 | 114012.0393 | 1.3112 | 20.7000 | 0.4581 |
| **随机森林** | 26.9132 | 8961.2203 | 1.1512 | 9.1900 | 0.9588 |

随机森林的MAE,MSE,MAPE,MedAE均为最低，同时R2\_score最高，因此在各方面都是准确率最高的模型。

**准确性排序：随机森林 > 决策树 >> K近邻 > 高斯朴素贝叶斯**

从10折交叉验证的平均误差分布图来看，K近邻和高斯朴素贝叶斯的稳定性最好，而随机森林和决策树

**稳定性排序：K近邻 ≈ 高斯朴素贝叶斯 > 随机森林 ≈ 决策树**

* 1. **四种学习模型的性能对比**

根据上述多角度的指标及可视化直观评测，综合结果对比分析，可得模型性能比较图表：

|  |  |  |
| --- | --- | --- |
|  | **准确性** | **稳定性** |
| **决策树** | 较高 | 低 |
| **高斯朴素贝叶斯** | 很低 | 高 |
| **K近邻** | 较低 | 高 |
| **随机森林** | 最高 | 低 |

若要选择一个最优模型，考虑到尽管高斯朴素贝叶斯和K近邻的稳定性高，但它们的准确性远远低于随机森林和决策树，而随机森林相比决策树在性能上表现稍优。综合考虑，这个最优模型应归属于随机森林。