天池算法大赛总结

LZM

本文结构如下：

1、赛题简介及背景。

2、数据探索部分。

3、数据预处理部分。

4、机器学习部分。

赛题简介及背景：

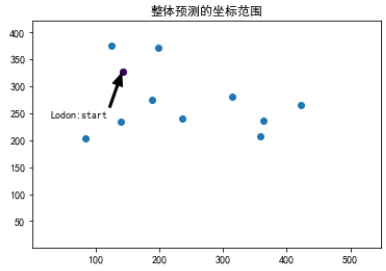
问题描述

目标是为无人运输飞行器寻找一个可以避开危险气象区域的有效航行算法。在飞行器飞行之前，需要根据英国气象局预测的天气数据，计划无人机航行路线。英国气象局每天会运行 10 个不同的预测模型，得出稍有不同但基本准确的预测结果。然而，天气预测的准确率通常为 90% 到 95%。优胜的算法需要基于我们提供的每日天气预测数据，确保无人运输飞行器航线安全且最短。

为简化挑战，我们根据天气预报所覆盖的最小范围，对覆盖区域进行了区域块的划分，每一个区域块都可以用（x,y）唯一表示，x 表示 X 轴方向的坐标值，y 表示 Y 轴方向的坐标值。同时我们假设无人运输飞行器在所有天气条件下的飞行速度均保持不变，在每个区域块的飞行时长固定，限定为 2 分钟飞越一个区域块，且只能从当前区域块上下左右地飞越到下一个区域块，或者停留在当前区域。

每天 3 点钟 10 架推进式无人运输飞行器将从伦敦海德公园飞往英国其他 10 个目的地城市，限定最大飞行时长为 18 个小时 [03:00-21:00)。选手需要基于我们提供的天气预报数据，预测每个区域块（x,y）的天气情况，规划无人机的飞行轨迹。飞行的城市分布如下：

In [50]:
```
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
cityData=pd.read_csv('H:/TIANCHI/CityData.csv')
x_id=list(cityData.xid)
y_id=list(cityData.yid)
plt.scatter(x_id,y_id,marker=(9, 3, 30))
plt.scatter(142,328,c=0)
plt.annotate('Lodon:start', xy=(142, 328), xytext=(30, 240),
             arrowprops=dict(facecolor='black', shrink=0.05,width=2,headwidth=10),
             )
plt.ylim([1,421])
plt.xlim([1,548])
plt.title('整体预测的坐标范围')
plt.show()
```



同时赛题暂且只考虑影响无人飞行器坠毁的一个天气因素：风速。当风速值≥15 时，无人机坠毁。

目标函数

我们将根据一天中每小时的实际天气状况评估参赛者提交内容中所描述的飞行器路线。如果有任何一时刻飞行器进入极恶劣的天气环境后损毁,那么将导致 24 小时的延时处罚。

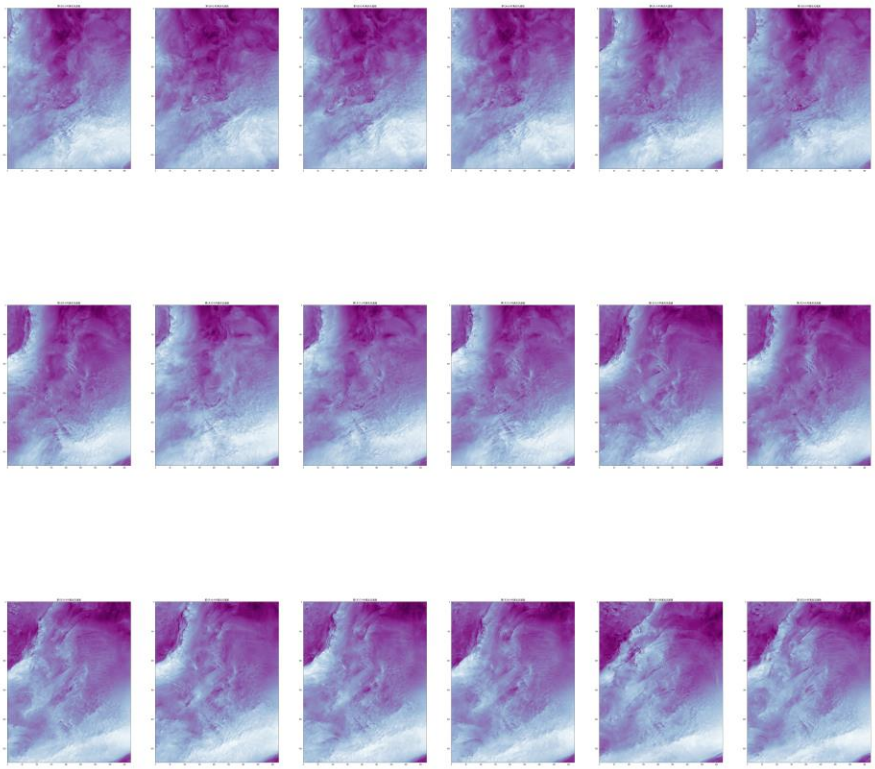
最终得分将是飞行器成功航行时间总时长（分钟）加上处罚（分钟）总数。分数最低者赢得比赛。

目标函数值 = 24*60*飞行器坠毁数 + 顺利到达的飞行器总飞行时长（分钟）

数据格式

xid	yid	date_id	预测器1-10的预测值	风速真实值	hour
1	1	1	16,14,13,15...	15	3
1	2	1	16,14,13,15...	12	4
1	3	1	16,14,13,15...	8	5

以上，背景介绍完毕。我根据真实风速绘制了一下风速图并做了一下简单统计：



数据探索部分：

初赛（复赛）阶段，我们将官方数据整理如下，其中 wind_1 至 wind_10 代表十个分类器对于风速的预测，wind 代表风速真值。

回归算法

```
data.head()
#线性回归的计算基于一些假设，其中一个假设就是 误差符合相互独立、均值为 0 的正态分布。
#我们使用了6种机器学习模型：XGboost、 Lasso、Ridge、Extra Trees、Random Forest、GBM。
```

	Unnamed: 0	xid	yid	date_id	hour	wind_1	wind_2	wind_3	wind_4	wind_5	wind_6	wind_7	wind_8	wind_9	wind_10	wind
0	0	1	1	1	3	13.8	12.8	10.40	11.1	13.2	13.1	12.6	11.2	14.1	12.8	12.8
1	1	1	2	1	3	13.7	12.2	9.84	11.0	13.2	13.0	12.6	13.0	13.9	12.5	12.8
2	2	1	3	1	3	13.3	11.7	12.30	11.0	13.1	12.7	12.7	13.9	13.8	12.0	12.6
3	3	1	4	1	3	12.8	11.2	14.70	11.2	12.5	12.5	12.7	14.9	13.9	11.6	12.1
4	4	1	5	1	3	12.4	11.5	16.80	11.1	12.2	12.1	12.4	14.4	13.8	10.9	11.6

以初赛为例，认为当风速大于等于 15 时，算做飞机坠毁。据此有两种方法对于（X,Y）的状态做预测，分别是分类与回归。
由于正负类比例分布不均匀，分类主要的参考指标是 F1 值：

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

如果考虑回归算法，评价一个模型的好坏主要考虑其 RMSE/MSE：

S={[(x1-x)^2+(x2-x)^2+.....(xn-x)^2]/N}^0.5

结合算法，由于在真实的路径搜索中需要比较路径节点之间的好坏，如果光采用分类算法，节点被 0/1 所描述，是无法比较出其之间的好坏的。最终我们采用 RMSE 为判断标准，采取回归算法。

赛题给出的数据中，已经给出了十个预测器对于当前坐标值在某时刻的风速预测，且据官方所述，这十个预测器的准确程度已经达到了 90%到 95%。十个预测器与目标风速值具有强相关性，问题可以从十个特征的角度考虑，也可以从如何综合十个专家的意见大小着手来综合看待。

对于数据的相关统计处理如下：
所有预测一共 20763720 次

在所有大于 15 的预测之中：（一共 5156997 次）　占所有次数的 0.2483 的比例
第 1 个分类器一共命中了 4377210 次
第 2 个分类器一共命中了 4268447 次
第 3 个分类器一共命中了 4295131 次
第 4 个分类器一共命中了 4264893 次
第 5 个分类器一共命中了 4261991 次
第 6 个分类器一共命中了 4344716 次
第 7 个分类器一共命中了 4459629 次（最厉害）
第 8 个分类器一共命中了 4391250 次
第 9 个分类器一共命中了 4210325 次
第 10 个分类器一共命中了 4296869 次

记大于 15 为一个赞成票，则：
大于 15 的预测中，0 个赞成的一共 16,7225 次(0.0324)，1 个赞成的一共 10,7859 次(0.0524)，2 个赞成的一共 100206 次(0.0718)，3 个赞成的一共 117555 次(0.0946)，
4 个赞成的一共 136428 次(0.121)，5 个赞成的一共 159729 次(0.152),6 次赞成的一共 190962 次(0.189),7 次赞成的一共 235211 次(0.234),8 次赞成的一共 295493 次(0.291),

9 次赞成的一共 454315 次(0.379),10 次赞成的一共 319,2014 次(1)

小于 15 的风速一共有：1560,6723 次
小于 15 的预测中，0 个赞成的一共 1307,0824 次，1 个赞成的一共 77,6511 次，2 个赞成的一共 42,7354 次，3 个赞成的一共 29,4998 次，
4 个赞成的一共 21,7353 次,5 个赞成的一共 16,8267 次_次赞成的一共 14,1597 次,7 次赞成的一共 11,7166 次,8 次赞成的一共 10,0619 次,
9 次赞成的一共 10,0721 次,10 次赞成的一共 19,1313 次

数据一共有 20763720 条　其中每个分类器预测小于等于 13.5　但真实风速大于 15 的点一共有 70781 条　占全部数据的 0.0034088785631861728　占所有负例（5156997）的 0.013725235830077077

数据预处理部分：

一：归一化操作：

原理：对于单个分类器而言，其预测数值大小可能遵循均值不同的同一分布，即其预测可能整体偏大，也有可能整体偏小，故我们将数据其预测整体进行（max-min）归一化操作，将其预测值映射到 0-1 之间。

随后为了配合我们算法在概率上的运用，我们只想知道在 14.5 附近的敏感值，即由 14.5 开始，大于 14.5 的点，我们让它的概率迅速趋近于 1，小于 14.5 的点，我们让它的概率快速趋近于 0,信号具有明显的**阶跃特性**。由此我们想到了 **sigmod 函数**。我们将数据通过映射关系到 sigmod 之上：

```
def sigmoid(x,k):
    x=-(x-k/34.1)*(5./(1.5/34.1))
    x=1./(1+math.exp(x))
    return x
```

（其中 34.1 为归一化因子）

二：其它方法的尝试：

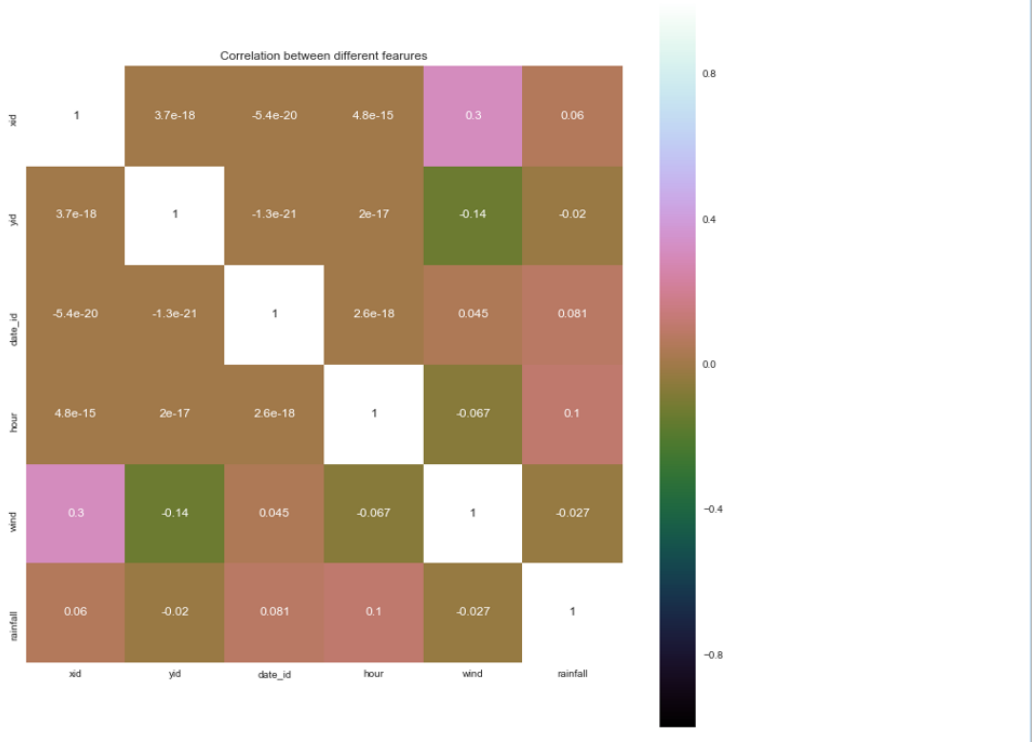
在文章写作的同时，我们看到了比赛中第三名的做法，其基本思路如下，偏向于保守及稳健的预测策略都是希望预测整体性偏大的，这样可以更多地排除存在的可能危险点。其做对于每一行数据单独进行处理，去掉了每次预测分类器的一个最小值，

并将其用均值作为替代。我认为这种做法实际上是删去了一部分有用的信息量而不可取的。

三：协方差矩阵：

依照数据的协方差矩阵，判断数据特征之间是正/负相关或强/弱相关。协方差矩阵如下图所示：在复赛中由于加入了降雨，依照协方差矩阵判断降雨于风速之间的相关性并不大，故在随后的特征筛选中，并没有纳入降雨特征：

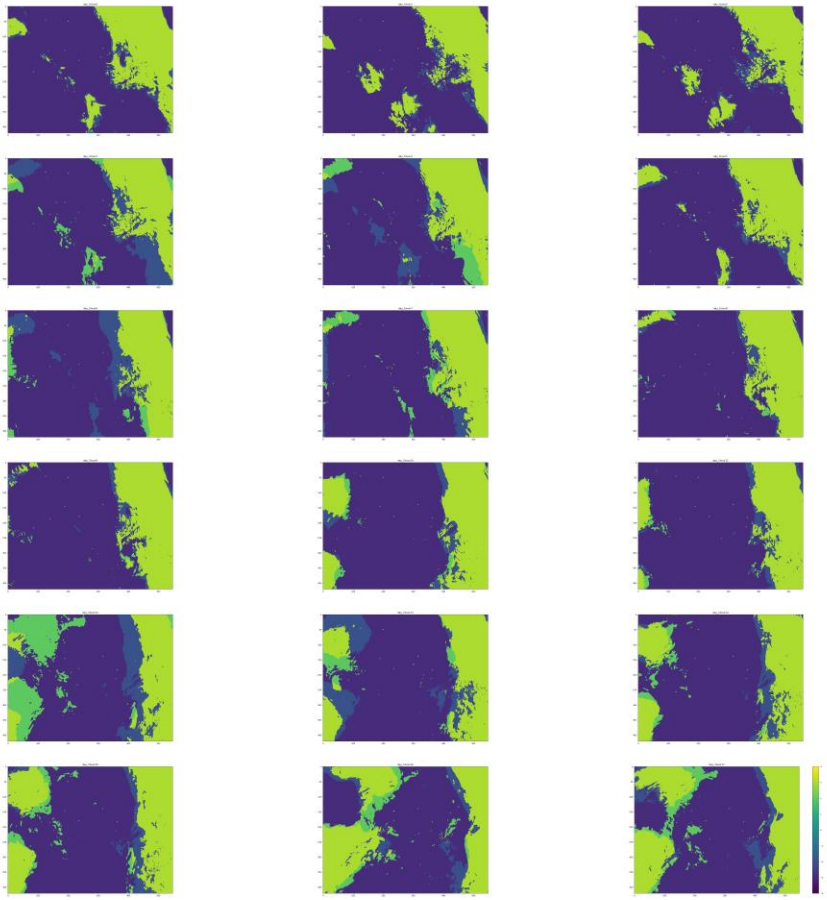
```
In [9]: #归一化处理并判断数据之间相关性
data=(data-data.min())/(data.max()-data.min())#注意数据标准化模块所做的操作就是这样
corr=data.corr()
plt.figure(figsize=(13,13))
sns.heatmap(corr, vmax=1, square=True, annot=True,cmap='cubehelix')
plt.title('Correlation between different fearures')
plt.show()
```



四： 预测的四色图：

在文章的开始，我放上了风速的梯度图形，由于其色泽变化莫测，我们并不能从图中反映出多少信息量特征，故我们采用了四色图，将我们采取的预测算法结果进行可视化。

这么做的目的在于当时我们的分数提升遇到了瓶颈，我们尝试了包括 xgboost,lr.randomforest 在内的各种算法，均无有效的提高，于是我们想结合混淆矩阵看看究竟是预测的那一部分出了问题。依照混淆矩阵的四种情况：正-正，正-负，负-负，负-正。我们将预测的风速可视化为四色图，作图如下：



我们通过图中发现地 6，7 小时以及 12，13 小时的标签被官方弄反了，不知道是不是官方有意为之。更换过标签重新训练过后 RMSE 下降了接近 30 个百分点。

4、机器学习部分。

机器学习部分我们拟采用了一下模型：

1、 xgboost
2、 adboost
3、 randomforest
4、 decisiontree
5、 LinerRegression
6、 others

4.1 各类学习算法特点比较

树模型：

剪枝处理：

目的：防止过拟合。

基本方：预剪枝、后剪枝。

预剪枝：在决策树生成过程中，在节点划分前进行估计，如果不划分能带来泛化性的提升则停止生长。后剪枝：在决策树生成之后自底向上对所有**非叶子节点**进行考察，如果替换为叶节点能带来性能的提升则替换成叶节点。

随机森林：

1、 借鉴于 bagging 方法，但扰动不仅来源于自然扰动，还来源于自属性本身的扰动。

2、 与 bagging 不同的是，其所有采样都是采取的有放回的采样。最终采取投票表决的方式选取结果。

Boost 模型：

提升树模型可以表示为决策树的加法模型：$f_M(x) = T(x;\Theta_m)$

其中，$T(x;\Theta_m)$表示决策树；Θ_m为决策树的参数；M 为树的个数

GDBT：

借鉴思想：利用已经产生的决策树，对随后基分类器的决策树产生造成积极的影响。

如果每一步的弱预测模型生成都是依据损失函数的梯度方向

XGBOOST 总结

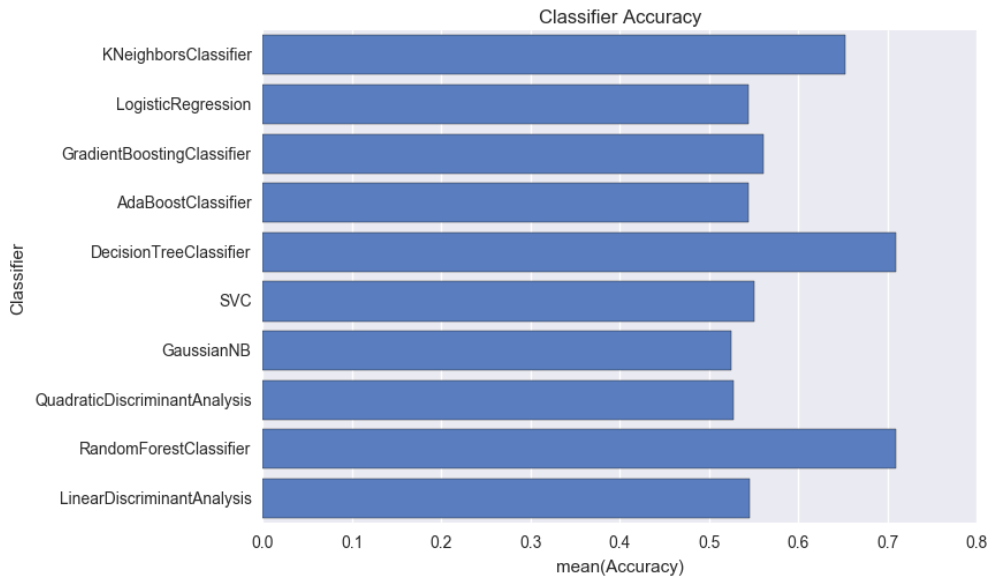
1、XGBOOST 与 GBDT 的区别 xgboost 还支持线性分类器，在代价函数里加入了正则化因子（L1 会产生稀疏的特, L2 会产生更多地特征但是都会接近于 0），有效控制模型的复杂度

2、相对于传统的 GBDT，XGBoost 使用了**二阶信息**，可以更快的在训练集上收敛

3、由于“随机森林”本身具备防止过拟合的优势，因此 XGBoost 仍然一定程度的具有该特性

4、XGBoost 的实现中使用了并行/多核计算, 因此训练速度快; 同时它的原生语言为 C/C++, 这是它速度快的实践原因

实际使用过程之中，我们参考其理论知识，但更多的是依照机器学习算法在训练/测试数据集上的表现来选择最合适的模型，随后采用 sklearn 中 grid_search_cv 模块进行范围内参数的自动选取，依照**特定步长**，采取**随机游走**的方式获取最佳参数值。



4.2 分类器在模型中的应用

分类模型主要用于寻找天气预测之中的异常点，使问题变成一个在**固定障碍地图**下的**带权重的单边寻路**问题。

对于异常点的二分判断采取投票法:1、少数服从多数 2、一票否决（*）

将二分判断的结果与回归地图做对比，相较于回归地图上绝对安全的点，一旦被判决为异常，则标记为 1（障碍），其余点保留其原来的回归值，不做改变。