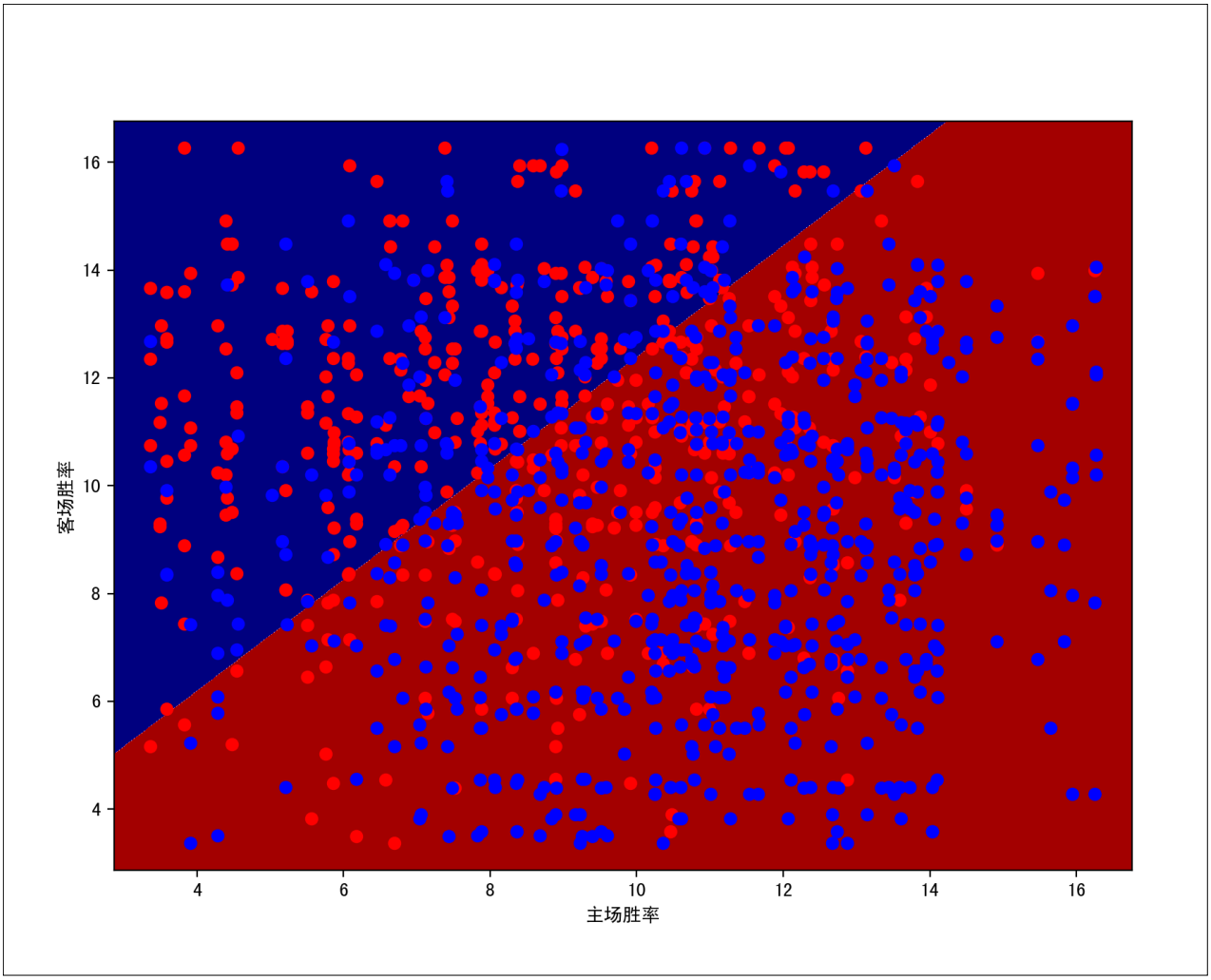


gg 小组种子杯初赛报告

队员： 柳泓鑫 梁志博 洪志远

AUC： 0.7566

2017年10月1日

执行概要

前期分析以及决策

本次比赛一共有5个数据集，其中可供训练的有两个数据集，一个是不同队伍之间的对战成绩，另一个是每个队伍球员的成绩。目前的因此整体的前期数据处理都是分为两个方面，一个方面是通过挖掘比赛队伍之间的成绩来训练模型，另一个是把队员能力映射到球队水平来进行训练。

环境以及工具概要

整个初赛都是由 Python3.6 来写的，环境是 Archlinux和 MacOS。由于数据量比较小，没有使用深度学习框架。用 scikit-learn 方便后期调参以及切换模型，同时使用 XGBoost 的 sklearn 模块与 sklearn 对接，方便日后模型以及参数选择。

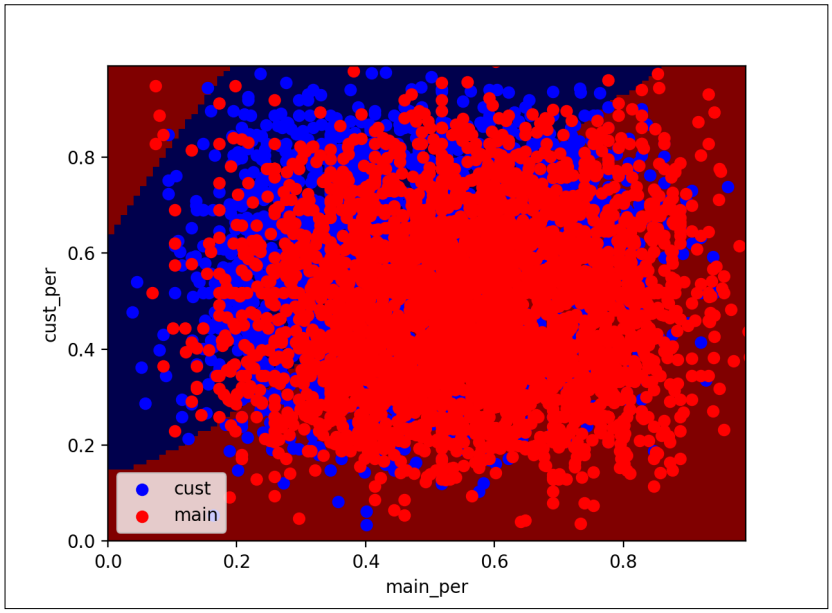
整体思路（数据特征提取思路、模型选取和型参数的优化）

第一阶段

第一天仔细看了一下文档，由于 matchDataTest.csv 数据集中就有比赛的胜负数据，于是就尝试直接处理 test 数据集。直接将胜负率进行处理之后提交，结果0.69999，排名第八。这个主要是一开始还没搭建好结构，不想浪费提交机会。

第二阶段

首先1思路将比赛结果数据进行分类，将数据处理成胜负率，然后清洗掉值为1或0的数据。再手动检查并清洗一遍。在输入模型的时候进行特征缩放，保证后面输入的可行性。把主场球队和客场球队的胜负率作为 X 比赛胜负作为 y 输入。机器学习算法集有 AdaBoostClassifier, RandomForestClassifier, GradientBoostingClassifier,GaussianNB, MultinomialNB, SVC, MLPClassifier,



过拟合训练模型

XGBClassifier。测试算法的时候使用 sklearn 自带的 train_test_split 随机切分数据集，训练样本和测试样本比例为0.2。同时使用 matplotlib 对特征进行可视化来筛选算法模型。AUC 使用 sklearn 自带的 AUC 计算工具进行计算。

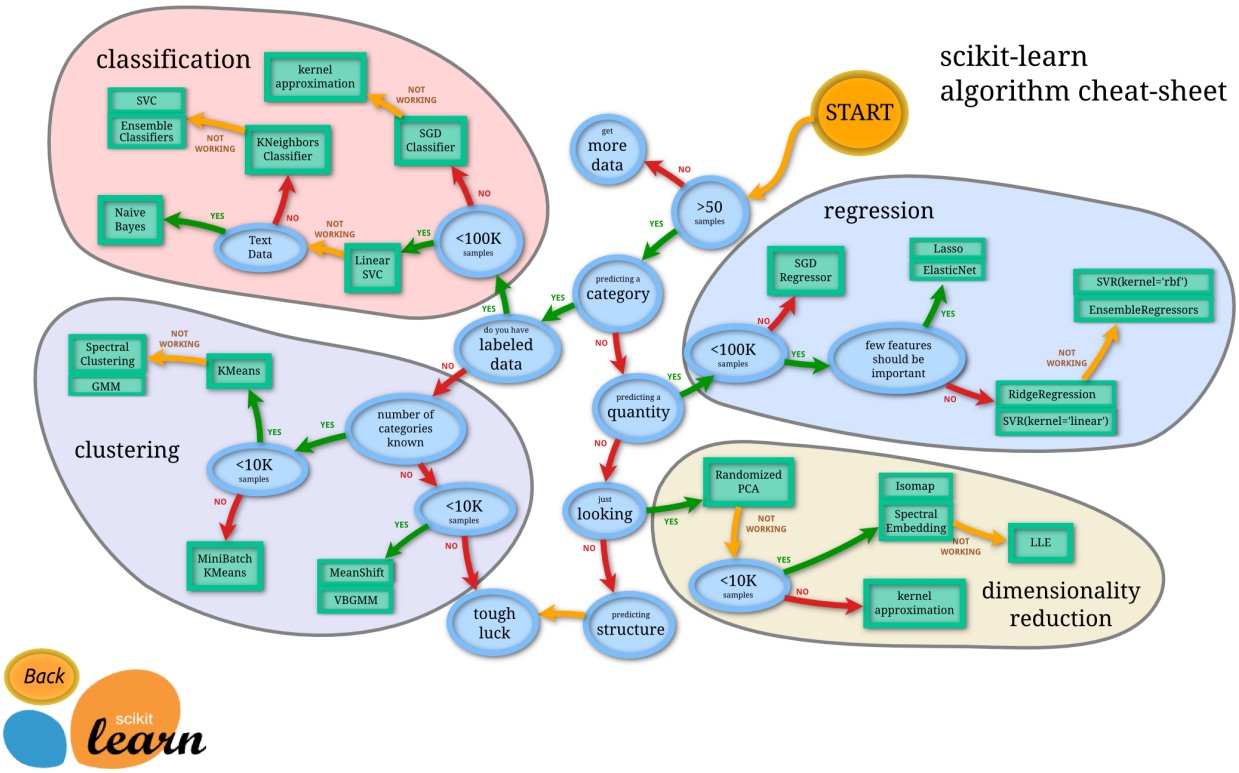
其次，2思路将队员的各项特征按照上场时间进行加权，得到每一队的特征。然后按照 matchDataTest.csv 以及 matchDataTrain.csv 的数据将客场和主场的信息并入文件，进行测试。测试模块与上方相同，输入特征由2个拓展到了14个，进过测试之后的 AUC 基本上维持在0.66左右。将数据可视化之后发现数据相关性非常小，基本上处于一种随机分布状态。同时高维的特征在只有 6k左右 的数据集中很难训练出来，并且最后的比赛胜负也是球员个人特征的表现，因此放弃了从球员信息入手。

第三阶段

因为确定了从胜负入手，所以主要目标放在胜负的样本上面，发现在开始的时候，每个球队的数据量很少，可信度不高。所以对样本进行了处理，将每个队取出了最高的胜率和最大样本的胜率，将这两份数据进行加权，加权系数通过暴力尝试选取特征最明显的系数。

经过测试之后的数据集的掺杂系数在 0.86 的效果最好，因此以这个系数为基础进行算法筛选，最后显示 bayes和 mlp 类的算法拟合程度最高，提交之后发现 mlp 和测试数据集拟合程度最好的算法。

在参数选取方面使用了“暴力调参”使用 sklearn 的 GridSearchCV 和 RandomizedSearchCV 来进行参数选取，这样节约了人力调参选取。



代码相关

目录结构

```
├─── README.md
├─── data
│    ├─── matchDataTest.csv
│    ├─── matchDataTest_2.csv
│    ├─── matchDataTrain.csv
│    ├─── matchDataTrain_2.csv
│    ├─── matchDataTrain_pre.csv
│    ├─── predictPro.csv
│    ├─── predictPro_mlp.csv
│    ├─── predictPro_mnb.csv
│    └─── teamData.csv
└─── src
     ├─── data_io.py
     ├─── pre_process.py
     ├─── scaler.py
     └─── try.py
```

文件说明

src 文件夹是根文件夹，存放所有的机器学习相关的 py 文件。

dataa_io.py 是封装函数模型，数据可视化等工具的文件，主要存放一些“中间件”，进行一些封装模块的存放。

pre_process.py是数据处理相关的问件，将原始数据集处理成期望的最大胜率和最大样本的胜率。进行train和 test 的特征拼接。

scaler.py 是特征缩放测处理文件。

try.py 是模型尝试以及调参的文件，

matchDataTest.csv 是附加相关比率特征之后的 test 文件。

matchDataTrain.csv 是附加相关比率特征之后的 train 文件。

predictPro_mlp.csv 是经过 MLPClassifier 训练之后的结果。

predictPro_mnb.csv 是经过 MultinomialNB 训练之后的结果。

teamData.csv 是每队队员的数据。

data_io.py

data_io.load_csv(file)
传入csv文件路径字符串，返回用numpy解析好的ndarray，默认跳过header。

Parameter	Desc
file	str, csv文件路径。

Returns: ndarray, 用numpy格式化好的ndarray。

data_io.save_csv(file, hypothesis, header='')
传入文件路径, 预测值和头，以%f的格式保存csv。

Parameter	Desc
file	str, csv文件路径。
hypothesis	ndarray, float, 要保存的预测值。
header	str, optional, default '', 要保存的csv文件的头。

Returns: **None**。

data_io.plot_decision_boundary(X, y, predict_func, x_label='', y_label='')
画出点和决策边界。

Parameter	Desc
X	ndarray, float, 要画的X, 限定二维数组。
y	ndarray, float, 要画的y，限定值为0,1。
predict_func	functoin, 预测函数。
x_label	str, optional, default '', x轴标签名。
y_label	str, optional, default '', y轴标签名。

Returns: **None**。

class data_io.Model(clf, scale=False, grid_search=False, random_search=False, search_params=None)

基于 sk-learn 抽象的分类器模型。

Parameter	Desc
clf	object, sk-learn 分类器。
scale	boolean, optional, default False, 是否进行 feature scaling，使用 sklearn.preprocessing.StandardScaler 进行缩放。
grid_search	boolean, optional, default False，是否进行网格搜索，使用 sklearn.grid_search.GridSearchCV 进行搜索。
random_search	boolean, optional, default False, 是否进行随机搜索，使用 sklearn.grid_search.RandomizedSearchCV 进行搜索。
search_params	dict, optional, default None，搜索参数，参数与 sklearn 的参数一致。
Attribute	**Desc**
clf	object, 初始化时传入的分类器。
scale	boolean，初始化时传入的参数。
random_search	boolean，初始化时传入的参数。
grid_search	boolean，初始化时传入的参数。
x	ndarray, float, fit 时传入的 x。
y	ndarray, float, fit 时传入的 y。

Returns: **self**, 返回分类器模型。

Method	Desc
fit(x, y)	基于 sklearn 封装
predict(x)	返回预测值
predict_prob(x)	返回 y 为 1 的置信度
auc(x, y_true)	返回 auc
plot(x, y, x_label='', y_label='')	画出点和决策边界

try.py

try.run(model)
运行该模型，返回 auc

Parameter	Desc
model	object, data_io.Model 的实例

Returns: auc, float
try.save(model)
保存该模型预测的置信度，输出到 csv。

Parameter	Desc
model	object, data_io.Model 的实例

Returns: **None**