

# MLTools 说明文档

## 一、概述

本框架参考 SAS 评分卡建模方法，基于 Python 3.5 开发。实现样本数据加载，输出变量统计分布，自动进行变量特征预处理，根据 IV 值挑选变量（如是评分卡），遍历预定的机器学习算法，根据训练准确率及 KS 值选择最佳算法，并输出保存。

## 二、逻辑框架

包含以下，如图 2.1：



图 2.1-逻辑框架

### 1. 数据导入


导入特定数据类型和文件格式的样本数据集。

### 2. 特征探索

探索变量统计分布/值域，绘制箱线图及更多可视化结果，如图 2.2。



图 2.2-特征探索

（符号  表示未实现,下同）

### 3. 特征处理

对不同类型变量分别进行标准化/归一化/二值化/ONE-HOT/变量交叉，并根据 IV 或 RF 进行特征选择，如图 2.3。

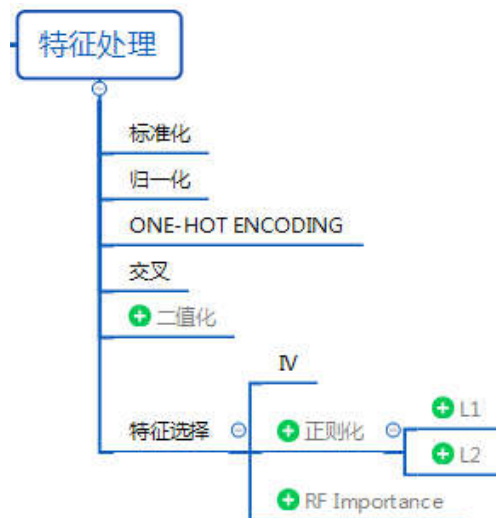


图 2.3-特征处理

#### 4. 算法探索

通过遍历预先设定核心算法（包括：深度学习/强化学习/有监督学习/无监督学习/半监督学习，并结合超参数优化），在多模型结果中进行模型评估，目前主要评估指标为 KS/准确度/精度/召回率/F1/AUC，得到最佳模型结果，如图 2.4。



图 2.4-算法探索

5. 结果导出  
保存过程结果及模型文件供后续调用。

三、代码结构

1. 算法模块(arithmetic)

当前分类算法(classifier)包括基础算法(Base Classifier)/择优算法(Classifier(GridSearchCV))，如图 3.1。

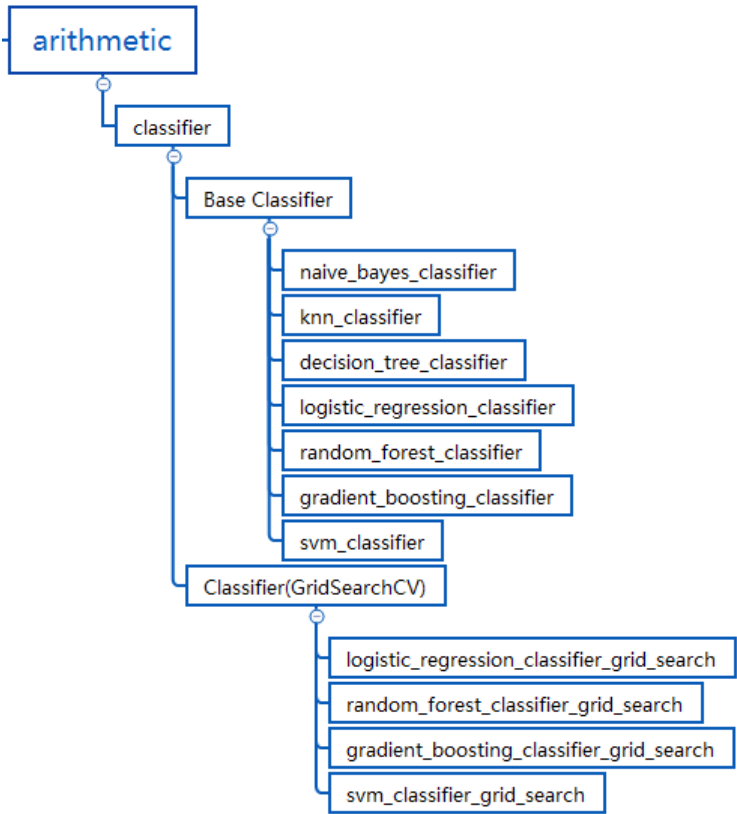


图 3.1-算法模块

2. 计算模块(calculate)

包括 KS 计算模块/WOE 计算模块等，如图 3.2。

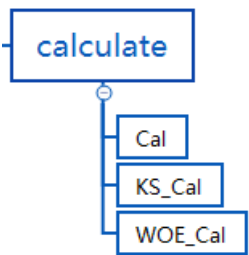


图 3.2-计算模块

3. 核心模块(core)

核心代码，base 用以初始化，common 用以功能重构，如图 3.3。

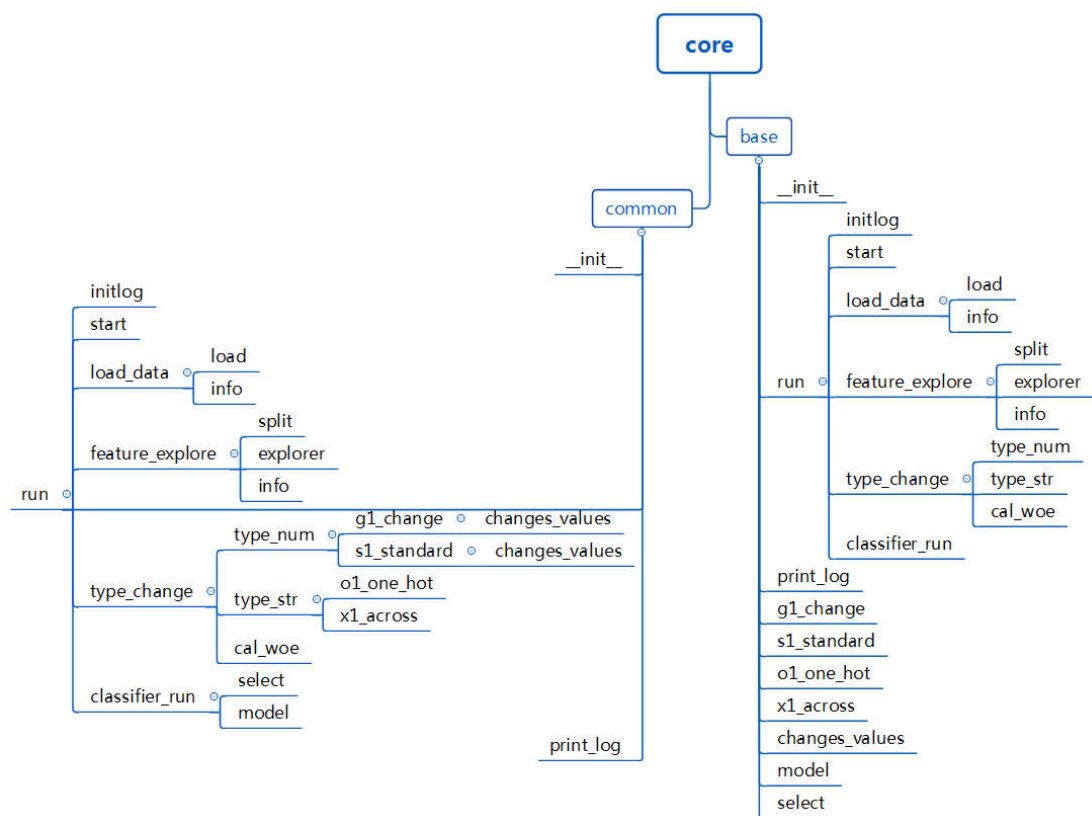


图 3.3-核心模块

#### 4. 绘图模块(drawing)

包括箱线图绘制模块/决策树绘制模块等，如图 3.4。

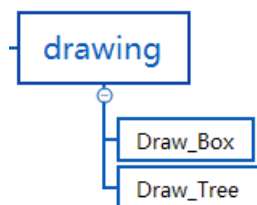


图 3.4-绘图模块

#### 5. 测试模块(test)

仅供测试。

#### 6. 系统模块(util)

所有公用功能函数，包括：获取当前日期(get\_days), 获取当前时间(get\_time), 获取随机数(randoms), 创建文件路径(mk\_dir), 导入切片函数 (cut\_split), 归一化函数(g1\_do), 标准化函数(s1\_do), 交叉指向函数(directing), 如图 3.5。

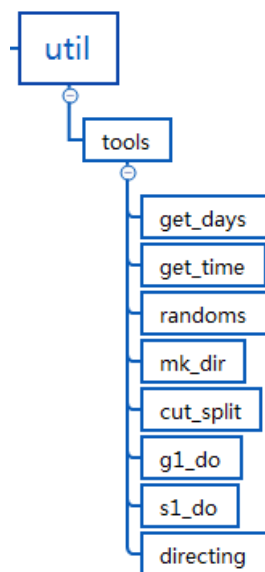


图 3.5-系统模块

#### 四、使用说明

##### 1. 代码路径

开发代码路径：140|G:\OUT\07.AI\MLTools\code\develop;

生产代码路径：140|G:\OUT\07.AI\MLTools\code\product;

##### 2. 参数设计

通过调用类 `core.common.Run()`，`Run(data, size, deletions, path, typed)`即可运行，需要赋调用参数，如表 4.1:

参数名	数据类型	说明	默认值	示例
<b>data</b>	str	数据集路径	无，需赋值	u'G:\\OUT\\...\\' (数据集目标变量 target)
<b>size</b>	float/int	训练数占比	1	0.23
<b>deletions</b>	dict	特殊值	None	{'R1': [-9992.0]}
<b>path</b>	str	输出路径	None	u'G:\\OUT\\...\\'
<b>typed</b>	str	模型类别	'model'	'model' 或 'scorecards'

表 4.1-调用参数

其中默认值通过配置文件获取，所有配置参数(包括测试参数)，如有特别需求可进行修改，如表 4.2:

参数名	参数值	说明
<b>main_path_data</b>	u'G:\\OUT\\07.AI\\MLTools\\data\\'	测试数据集路径
<b>main_path_out</b>	u'G:\\OUT\\07.AI\\MLTools\\out\\'	测试输出路径
<b>main_path_test</b>	u'G:\\OUT\\07.AI\\MLTools\\out\\test\\'	测试路径
<b>data_Name</b>	u'mlm2_sample'	测试数据集名
<b>data_Tame</b>	u'mlm2_sample_T200(MISS)'	200 条测试数据，含缺失
<b>data_type</b>	u'.csv'	数据类型
<b>out_Name</b>	u'fields'	输出文件名
<b>out_type</b>	u'.log'	日志格式
<b>size</b>	0.005	训练数占比默认值

<b>times</b>	5	WOE 分段次数
<b>deletions</b>	{ 'R_1': [-99000792.0, 85], 'R_2': [] }	特殊值
<b>special</b>	[-99000792, -99000784, -99000776]	全局特殊值，评分卡用
<b>data</b>	u'G:\\OUT\\07.AI\\MLTools\\data\\m1m2_sample.csv'	测试数据
<b>data_T</b>	u'G:\\OUT\\07.AI\\MLTools\\data\\m1m2_sample_T200(MISS).csv'	测试缺失数据
<b>pro_gra_dot</b>	'C:\\ProgramFiles(x86)\\Graphviz2.38\\bin\\dot.exe'	Graphviz 程序路径
<b>min_cpu</b>	2	并发线程数
<b>log_mode</b>	'w'	日志写入格式
<b>log_level</b>	NOTSET	日志警告类型
<b>log_format</b>	'%(levelname)s  %(message)s'	日志输出格式
<b>log_encode</b>	'utf-8'	日志编码格式

表 4.2-配置参数

## 五、代码运行

### 1. 测试案例

引用代码可至生产代码路径(140|G:\\OUT\\07.AI\\MLTools\\code\\product)调用，或直接调用由 Python 3.5 系统包(140|D:\\Python35\\python.exe)直接 import。

#### 1) 仅提供数据集

```
from MLTools.code.product.core.common import Run
if __name__ == '__main__':
    t1 = Run(
        data=u'G:\\OUT\\07.AI\\MLTools\\data\\m1m2_sample_T200(MISS).csv'
    )
```

#### 2) 需切 50%数据集

```
from MLTools.code.product.core.common import Run
if __name__ == '__main__':
    t2 = Run(
        data=u'G:\\OUT\\07.AI\\MLTools\\data\\m1m2_sample_T200(MISS).csv',
        size=0.5
    )
```

#### 3) 带特殊值的数据集

```
from MLTools.code.product.core.common import Run
if __name__ == '__main__':
    t3 = Run(
        data=u'G:\\OUT\\07.AI\\MLTools\\data\\m1m2_sample_T200(MISS).csv',
```

```

        deletions={ 'R_POS_CNT_16_Pct_Avg_POS_CNT_1N': [-99000792.0]}
    )

```

#### 4) 自定义输出路径

```

from MLTools.code.product.core.common import Run
if __name__ == '__main__':
    t4 = Run(
        data=u'G:\\OUT\\07.AI\\MLTools\\data\\m1m2_sample_T200(MISS).csv',
        path=u'G:\\OUT\\07.AI\\MLTools\\out\\test\\'
    )

```

#### 5) 评分卡模式

```

from MLTools.code.product.core.common import Run
if __name__ == '__main__':
    t5 = Run(
        data=u'G:\\OUT\\07.AI\\MLTools\\data\\m1m2_sample_T200(MISS).csv',
        typed='scorecards'
    )

```

## 2. 运行结果

如图 5.1:

```

***** RESULT *****

```

CLASSIFIER	TRAIN KS	TEST KS	ACCURACY	PRECISE	RECALL	F1	AUC	SCORE
logistic_regression_classifier	50.78%	43.16%	0.8320	0.4412	0.3846	0.4110	0.6485	43.4503
random_forest_classifier	91.19%	48.17%	0.8672	0.6316	0.3077	0.4138	0.6377	52.6695
gradient_boosting_classifier	77.72%	54.69%	0.8711	0.7143	0.2564	0.3774	0.6190	59.7081

```

BEST CLASSIFIER: gradient_boosting_classifier
SAVE PATH: G:\OUT\07.AI\MLTools\out\test\T2173261299744211\save\
***** END! *****

```

图 5.1-运行结果

## 3. 结果调用

调用结果 model 文件案例，通过重构 core.common.Run 实现，如下：

```

import pandas as pd
import joblib
from sklearn import metrics
from MLTools.code.develop.core.common import Run
from MLTools.code.develop.util.tools import cut_split

path1 = u'G:\\OUT\\07.AI\\MLTools\\out\\model\\logistic_regression_classifier.model'

class RunModel(Run):

```

```
def __init__(self, data, size, deletions=None, path=None, typed='model',
model_path=''):
    self.model_path = model_path
    super(Run, self).__init__(data, size, deletions, path, typed)

def load(self):
    self.print_log(self.model_path)
    data_temp = pd.read_csv(self.data, encoding='gbk')
    if self.size == 1:
        self.dataX, self.dataY = data_temp.drop('target', axis=1), data_temp['target']
    else:
        self.dataX, _, self.dataY, _ = \
            cut_split(data_temp.drop('target', axis=1), data_temp['target'], self.size)

def classifier_run(self):
    m = joblib.load(self.model_path)
    self.print_log(metrics.classification_report(m.predict(self.dataX), self.dataY))
```

## 六、版本说明

**MLTools v0.2.7.**