Li, Jun A

From: Lin, XiangX
Sent: Friday, January 22, 2016 1:38 PM
To: Li, Jun A
Subject: FW: Migration from C# to ATK Python code

From: Zhang, Lijuan
Sent: Friday, January 22, 2016 1:20 PM
To: Wang, AlbertZQ; Lou, Ming; Li, Carl; Cao, Buddy; Ding, Danny; Gao, Fengqian; Lu, Yuan Y; Lin, XiangX
Subject: Migration from C# to ATK Python code

Hi, Albert

下面是针对从c#转到atk python code所需花费时间的大概评估：

1 存在问题总结

处理步骤	存在问题	相应解决方案
创建frame	1 数据中存在空值，创建frame会直接过滤掉此行； 2 创建frame时，数据类型不支持str类型，包括中文、特定字#￥%#符等； 3创建frame时，必须为定长的数组类型；不支持非定长数组； 4 创建frame前需要对名称进行预处理；	1 可以在创建frame时先给定一个特定的值，创建frame成功后再做处理 2.3.4可以提前预处理
分类量化	5 量化的结果不能回写到frame中； 6 不能通过feature指定到列，目前示例中仅支持数组下标；	5 依赖于波兰team提供相应的方法 6 通过数组下标找到相应的列
数据填补	7 目前未找到直接的均值处理方法；	7 用Pandas或python代码求平均

TAP提供的处理方法：

（1） 求中位数，频次
（2） Feature选取
（3） 数据行的操作
（4） Transform操作
（5） 更改列名
（6） 删除重复行
（7） 拆分列

如果需要查看详细的信息，请参考第三第四部分。

2 时间评估

Solution:如果波兰team能把上面的问题5能够解决，那么我们预估从c#转到atk python code需要大概一个人3周的时间。

注意： 以上的评估是基于不提供界面还有分类量化的配置文件操作，如果需要量化分类数据，需要修改代码。

3数据处理步骤说明

当前基于C#版本量化工具主要功能和处理步骤如下：

（1） 读取csv数据文件

（2） 修改列名

- 检测是否有重复列名；
- 将 ”%”，”#”，”/”等字符按规则处理；

（3） Feature选取

- 根据指导的feature进行选取

（4） 分类量化

- 量化前对字段进行处理：例如去掉某些标记符号；
- 将数值型量化成标称型；
- 将数据库标记量化成标称型；

（5） 数据填补

数据填补支持：均值、中位数、频次

（6） 结果导出和使用

- 处理完毕的数据导出为指定格式
- 将导出的数据通过TAP平台，上传至HDFS
- 在模型脚本中根据HDFS数据地址，创建frame并直接使用；

4 使用ATK功能处理数据

（1） 使用ATK功能与C#工具处理步骤的不同：

- 使用ATK首选要将数据上传到HDFS，并创建frame成功后才可以开始处理；
- 列名的处理需要在创建frame之前，否则frame不能创建；

（2） ATK处理步骤

ATK处理数据包括以下6个步骤，处理完毕后frame可以直接为模型使用；

1） 数据上传hdfs：

将csv文件上传至hdfs，并得到数据的地址；

2） 连接服务器：

连接服务器示例代码如下：

```
import trustedanalytics as ta
ta.server.uri='itflex.demo-gotapaas.com'
ta.connect('/root/demo.creds')
```

3） 创建frame：

创建frame分为三个步骤：创建csv数据结构、判断frame名称重复、创建frame

```
#创建csv数据结构
csv = ta.CsvFile("hdfs://nameservice1/org/intel/hdfsbroker/userspace/b61d4808-e761-45c3-bd54-afcb05b84a8b/02c0e183-b118-4ffa-bd93-54549e55dac5/000000_1",
                 schema=[
                  ("GXY",ta.int32),
                  ("Age",ta.int32),
                  ("Sex",ta.str),
                  ("Height",ta.float64),
                  ("BMI",ta.float64),
                  ("DBP",ta.float64),
                  ("Cr",ta.int32),
                  ("HCT",ta.float64)
                  ], skip_header_lines=1);

#判断frame名称重复
frame_name = "myframe";
if frame_name in ta.get_frame_names():
      ta.drop_frames(frame_name)

#创建frame
my_frame = ta.Frame(csv, frame_name)
```

4） Feature选取：

使用atk提供的drop_columns函数可以删除指定列，由此实现列名选择功能

```
my_frame.drop_columns('BMI')
```

使用atk提供的rename_columns函数可以修改feature名称（备注：目前暂无实际意义，因为需要在创建frame之前处理名称）

```
my_frame.rename_columns({'DBP' : 'DBP_change'})
```

输出结果：

```
[##]  GXY  Age  Sex  Height  DBP_change  Cr  HCT
=================================================
[0]     0   48    1   174.0        80.0   1  38.2
[1]     0   42    2   158.0        70.0   1  40.3
```

5） 分类量化：

Atk文档中提供了Transfrom的功能，按照文档和示例代码综合处理说明如下：

示例： 将HCT列的数据值型转换成标称型，转换依赖Sex列：

偏低：男性-＜40，女性-＜37；

正常：男性-40~50，女性-37~48；

偏高：男性-＞50，女性-＞48

```
count = my_frame.row_count
print count
```

```
results = my_frame.take(count)
for row in results:
   htc =  row[6]
   print htc
   sex = row[2]
   print sex
   if sex == 1:
      if htc < 40.0:
         row[6] = 2;
         print row[6]
      if htc >= 40.0 and htc <= 50.0:
         row[6] = 0;
      if htc >= 50.0:
         row[6] = 1;
   if sex == 2:
      if htc < 37.0:
         row[6] = 2;
      if htc >= 37.0 and htc <= 48.0:
         row[6] = 0;
      if htc >= 48.0:
         row[6] = 1;
```

frame列表：

[##]	GXY	Age	Sex	Height	DBP_change	Cr	**HCT**
[0]	0	48	1	174.0	80.0	1	38.2
[1]	0	42	2	158.0	70.0	1	40.3
[2]	0	42	2	168.0	70.0	1	39.7
[3]	1	53	1	164.5	90.0	1	48.1
[4]	0	47	1	169.0	80.0	0	47.3
[5]	0	28	1	176.0	80.0	1	44.2
[6]	0	27	1	169.5	70.0	1	45.3
[7]	0	24	2	169.5	60.0	1	39.1
[8]	0	35	2	155.0	80.0	1	37.7
[9]	0	49	1	165.0	80.0	0	42.8
[10]	1	65	1	171.5	80.0	1	41.5
[11]	0	33	1	183.0	70.0	1	48.1
[12]	0	37	1	179.5	80.0	1	46.4
[13]	0	57	1	178.0	90.0	1	44.2
[14]	0	39	1	169.5	80.0	1	46.8
[15]	0	51	2	172.0	80.0	1	42.1

```
16    //输出row_count
38.2  //输出HCT值,第一行
1     //输出对应Sex值
2     //输出量化值
40.3
2
39.7
2
48.1
1
47.3
```

```
1
44.2
.
.
.
```

转换后的frame：

[#]	GXY	Age	Sex	Height	DBP_change	Cr	HCT
[0]	0	48	1	174.0	80.0	1	38.2
[1]	0	42	2	158.0	70.0	1	40.3
[2]	0	42	2	168.0	70.0	1	39.7
[3]	1	53	1	164.5	90.0	1	48.1
[4]	0	47	1	169.0	80.0	0	47.3
[5]	0	28	1	176.0	80.0	1	44.2
[6]	0	27	1	169.5	70.0	1	45.3
[7]	0	24	2	169.5	60.0	1	39.1
[8]	0	35	2	155.0	80.0	1	37.7
[9]	0	49	1	165.0	80.0	0	42.8

说明1：atk文档中暂时未找到相关参考代码
输出结果并未改变，row[6] = 2 赋值语句没有更新到frame中，怀疑是没有赋值真实副本中

说明2：atk文档中可以通过列名来确定处理的列，但是测试过以后只能使用列的数组下标；
htc= row.HTC #文档中给出方法
htc = row[6] #实际使用方法

错误提示如下：

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent cal
l last)
<ipython-input-73-8239c48e32f7> in <module>()
      3 results = my_frame.take(count)
      4 for row in results:
----> 5     htc =  row.HTC
      6     print htc
      7     sex = row[2]

AttributeError: 'list' object has no attribute 'HTC'
```

6）数据填补：
目前数据填补支持：中位数和权重两种方式，暂不直接支持均数

```
column_median = my_frame.column_median(data_column="Cr")
```

输出结果

```
[========================] 100.00% Tasks retries:0 Time 0:00:06
1
```

column_mode = my_frame.column_mode(data_column="GXY")

输出结果

```
[========================] 100.00% Tasks retries:0 Time 0:00:07
{u'weight_of_mode': 14.0, u'mode_count': 1, u'modes': [0], u'total_weight': 16.
0}
```

Thanks

Lijuan