助学金精准预测-DataMining

序言

你所看到的这份代码，是Data Castle数据挖掘公开赛《助学金精准预测》的参赛作品。本程序以大学生的行为数据以及历史获助学金情况作为训练数据集，对代码内的模型进行训练，后可根据新的大学生行为数据进行助学金获得情况预测。赛题的详细描述可查看比赛网址：

大学生助学金精准资助预测 训练数据集以及测试数据集均可在该地址中找到。

本文档将详细介绍代码的运行方法，但不会对机器学习的规范化流程和分类器模型的原理做过多的赘述。相关信息可自行google或baidu之。

运行环境

- Linux (kernel >= 3.2) / Unix (macOS)
- Python 2.7
- numpy 1.12.0
- scikit-learn 0.18.1
- scipy 0.18.1
- xgboost 0.6a2

若您的电脑缺少以上环境或程序包，请按照上面的版本安装之。
如何安装请google或百度之。

如果您不确定您是否拥有这些依赖包，请在命令行输入下面命令查看之：

```
$ python --version
```

这将输出您的python版本号, 请确保您的版本号以2.7开头。

```
$ pip freeze
```

输入回车后，您应该能看到以下信息：

```
numpy==1.12.0
scikit-learn==0.18.1
scipy==0.18.1
xgboost==0.6a2
```

包含的文件目录

```
-blending             #融合
-learning_algorithm   #各种分类器的模型
-online               #对test集合进行预测，产生线上提交数据
-studentForm          #由原始训练/测试数据作invert处理后得到的表存放在这里
-data_analysis        #手工地对数据集进行观察的脚本
-original_data        #存放数据的位置，包括处理后得到的模型输入文件。
-preprocess           #划分CV
-commons              #保存有各个分类器模型的参数，以及一些通用的功能模块
-feature_importance   #查看feature的重要度排名脚本
-offline              #对train集合进行训练，线下调参
-processing           #数据预处理，提取feature
-data                 #请将数据集解压后放置在本目录下
```

如何运行

注意 目前，这些代码的编写是只保证可以work，能够生成出结果的。因为在参赛期间只需要团队内部成员可以运行，所以我们并没有提供傻瓜式的一键操作可以让您直接从输入得到输出。要运行这些代码可能需要一些专业知识，和对代码中一些路径名的修改。我们尽可能的将运行代码的步骤详细写出。 尽管如此，您可能还是会在某个环节出现问题，需要你动点脑筋去解决，如果有任何疑问可以通过本文档末尾所列出的成员信息联系我们。

预处理

将打包文件解压后进入项目根目录下：

```
$ unzip datacastle_subsidy.zip
$ cd datacastle_subsidy
```

在当前目录下，您应该可以看到本文档上一节中所描述的所有文件夹目录名，现在我们首先要做的是对原始数据进行预处理，并提取feature值。在此之前，请先确保您已经把下载下来的数据集解压并按照格式放在了 `data` 目录下。如下所示：

```
-data
    -train
        borrow_train.txt
        card_train.txt
        dorm_train.txt
        library_train.txt
        score_train.txt
        subsidy_train.txt
    -test
        borrow_final_test.txt
        card_final_test.txt
        dorm_final_test.txt
        library_final_test.txt
        score_final_test.txt
        subsidy_final_test.txt
```

现在，进入根目录下的 `processing` 文件夹内，并运行脚本以invert测试集数据：

```
$ cd processing
$ python createStudentForms.py borrow_final_test.txt
$ python createStudentForms.py card_final_test.txt
$ python createStudentForms.py dorm_final_test.txt
$ python createStudentForms.py library_final_test.txt
$ python createStudentForms.py score_final_test.txt
```

然后用您喜欢的文字编辑器打开 `creatStudentForms.py` 文件，修改代码中的路径名，以供继续invert训练集数据。修改好后，请运行：

```
$ python createStudentForms.py borrow_train.txt
$ python createStudentForms.py card_train.txt
$ python createStudentForms.py dorm_train.txt
$ python createStudentForms.py library_train.txt
$ python createStudentForms.py score_train.txt
```

上面运行的这些命令将对数据表作invert处理，并将生成出的表存放在根目录的 `studentForm` 下。

注意 上面命令在处理一些大文件的时候可能需要一些时间，请耐心等待不要关闭他们。

提取feature

现在，我们将从 `studentForm` 文件夹下的文件中提取特征值，在 `processing` 文件目录下运行：

```
$ python processBorrow.py train
$ python processCard.py train
$ python processDorm.py train
$ python processLibrary.py train
$ python processScore.py train
```

上面对训练集提取了训练集中的feature值，保存在名为 `*Processed.txt` 文件中并存放在 `processing/trainProcessed` 文件夹下。然后我们对测试集做同样的处理：

```
$ python processBorrow.py test
$ python processCard.py test
$ python processDorm.py test
$ python processLibrary.py test
$ python processScore.py test
```

同样的，上面的命令提取了测试集中的feature值，保存在名为 `*Processed.txt` 文件中并存放在 `processing/testProcessed` 文件夹下。

提取后的特征值以不同表的文件名存放在不同文件中，现在我们将这些feature合并为一个文件，可供训练模型作为输入。运行：

```
$ sh refresh_train.sh
$ sh refresh_test.sh
```

上面命令运行了我们写好的两个命令行脚本，他们会自动的调用一系列我们编写好的python脚本，将提取feature后的文件合并在一起，并且对数据做标准的**归一化处理**。上面命令运行起来可能需要一些时间，请耐心等待不要关闭他们。

请注意观察命令行中是否有错误提示信息出现，如果有请不要继续往下。如果您成功运行，那么恭喜您，特征提取完成了。您现在可以打开 `processing/trainProcessed` 目录下的 `collumInfo.txt` 文件或 `processing/testProcessed` 目录下的 `collumInfo.txt` 文件，他们的内容应该是完全一样的，分别列出了刚刚抽取feature的过程中所抽取的所有1151个feature的编号，名字以及他们所属的表。我们在这个步骤中生成出的以下两个文件将作为模型的输入文件：

```
processing/trainProcessed/idexamples_train_std.txt
processing/trainProcessed/idexamples_test_std.txt
```

将他们拷贝到项目根目录下的original_data目录（这个文件夹的命名有点名不副实，请不要在意）下：

```
$ cd processing
$ cp trainProcessed/idexamples_train_std.txt ../original_data
$ cp testProcessed/idexamples_test_std.txt ../original_data
```

划分Cross Validation集

在本项目中，在训练模型时对模型精度进行校验的方法采用5分法Cross Validation，后简称此法为CV，简称此法产生出的5个examples子集为CV0，CV1，..., CV4。

现在进入项目根目录下的preprocess目录下，运行:

```
python stratified_split_cross_validation.py
```

现在，您应该可以在项目根目录下的original_data下找到5个划分好的CV子集。在对线下的训练集训练时，他们将作为真正的模型输入文件。

注意 5个CV文件实际上分别是 `idexamples_train_std.txt` 的子集。他们只在线下训练时用到。而对 `idexampoles_test_std.txt` 文件我们将不会做划分处理。在预测训练集并做线上提交使用时，我们将使用 `idexamples_train_std.txt` 的全集。

训练模型&调参

到现在为止，一切进展顺利。我们获得了我们需要的模型输入文件。接下来，我们将对各个模型进行训练，调参。

现在，进入根目录下的 `offline` 文件夹下，并开始模型的训练:

```
$ cd offline
$ python learning_validation.py
```

一切正常的话，在等待一段时间后，您应该可以看到类似下面的输出信息:

```
random_seed = 0
n_estimators= 1000 max_depth= 50 min_samples_split= 50 min_samples_leaf= 20 max_featu
res= 200 max_leaf_nodes= None criterion= gini min_impurity_split= 1e-07 class_weight=
 {'0': 5.6, '1500': 95, 'min_weight_fraction_leaf': 0, '2000': 130, '1000': 60}
1000case :     right_num: 75   prediction_num:  385   acture_num:149    prcise: 0.194
805194805   recall: 0.503355704698    f1: 0.280898876404
1500case :     right_num: 15   prediction_num:  110   acture_num:94    prcise: 0.1363
63636364   recall: 0.159574468085    f1: 0.147058823529
2000case :     right_num: 13   prediction_num:  75   acture_num:71    prcise: 0.17333
3333333   recall: 0.183098591549    f1: 0.178082191781
```

```
macro F1:0.0313400447764
one cv done  time: 38.2924408913
n_estimators= 1000 max_depth= 50 min_samples_split= 50 min_samples_leaf= 20 max_featu
res= 200 max_leaf_nodes= None criterion= gini min_impurity_split= 1e-07 class_weight=
 {'0': 5.6, '1500': 95, 'min_weight_fraction_leaf': 0, '2000': 130, '1000': 60}
1000case :     right_num: 70   prediction_num:  386   acture_num:145    prcise: 0.181
347150259   recall: 0.48275862069    f1: 0.263653483992
1500case :     right_num: 16   prediction_num:  129   acture_num:94    prcise: 0.1240
31007752   recall: 0.170212765957    f1: 0.143497757848
2000case :     right_num: 6   prediction_num:  70   acture_num:71    prcise: 0.085714
2857143   recall: 0.0845070422535    f1: 0.0851063829787
macro F1:0.0266057566136
one cv done  time: 37.4284248352
n_estimators= 1000 max_depth= 50 min_samples_split= 50 min_samples_leaf= 20 max_featu
res= 200 max_leaf_nodes= None criterion= gini min_impurity_split= 1e-07 class_weight=
 {'0': 5.6, '1500': 95, 'min_weight_fraction_leaf': 0, '2000': 130, '1000': 60}
1000case :     right_num: 62   prediction_num:  374   acture_num:149    prcise: 0.165
77540107   recall: 0.41610738255    f1: 0.237093690249
1500case :     right_num: 16   prediction_num:  103   acture_num:89    prcise: 0.1553
39805825   recall: 0.179775280899    f1: 0.166666666667
2000case :     right_num: 7   prediction_num:  71   acture_num:71    prcise: 0.098591
5492958   recall: 0.0985915492958    f1: 0.0985915492958
macro F1:0.0262805945657
one cv done  time: 41.1695289612
n_estimators= 1000 max_depth= 50 min_samples_split= 50 min_samples_leaf= 20 max_featu
res= 200 max_leaf_nodes= None criterion= gini min_impurity_split= 1e-07 class_weight=
 {'0': 5.6, '1500': 95, 'min_weight_fraction_leaf': 0, '2000': 130, '1000': 60}
1000case :     right_num: 68   prediction_num:  401   acture_num:149    prcise: 0.169
57605985   recall: 0.456375838926    f1: 0.247272727273
1500case :     right_num: 14   prediction_num:  114   acture_num:94    prcise: 0.1228
07017544   recall: 0.148936170213    f1: 0.134615384615
2000case :     right_num: 13   prediction_num:  70   acture_num:71    prcise: 0.18571
4285714   recall: 0.183098591549    f1: 0.184397163121
macro F1:0.0287108628895
one cv done  time: 38.5329420567
n_estimators= 1000 max_depth= 50 min_samples_split= 50 min_samples_leaf= 20 max_featu
res= 200 max_leaf_nodes= None criterion= gini min_impurity_split= 1e-07 class_weight=
 {'0': 5.6, '1500': 95, 'min_weight_fraction_leaf': 0, '2000': 130, '1000': 60}
1000case :     right_num: 56   prediction_num:  349   acture_num:149    prcise: 0.160
458452722   recall: 0.375838926174    f1: 0.224899598394
1500case :     right_num: 12   prediction_num:  105   acture_num:94    prcise: 0.1142
85714286   recall: 0.127659574468    f1: 0.120603015075
2000case :     right_num: 10   prediction_num:  97   acture_num:70    prcise: 0.10309
2783505   recall: 0.142857142857    f1: 0.119760479042
macro F1:0.0244286173064
```

```
one cv done  time: 39.1871788502
1000case :     right_num: 331   prediction_num:  1895   acture_num:741    prcise: 0.1
74670184697   recall: 0.44669365722    f1: 0.251138088012   weighted_f1:  0.017096308
9772
1500case :     right_num: 73   prediction_num:  561   acture_num:465    prcise: 0.130
124777184   recall: 0.156989247312    f1: 0.142300194932   weighted_f1:  0.0060789702
015
2000case :     right_num: 49   prediction_num:  383   acture_num:354    prcise: 0.127
937336815   recall: 0.138418079096    f1: 0.132971506106   weighted_f1:  0.0043244752
5599
avg macro F1:0.0274997544347
time:194.953258038
```

通过上面的输出信息，您可以查看训练模型在当前参数和CV上的训练精度信息，并根据这些信息调整模型的参数。

模型的参数存放在项目根目录下的 `commons/variables.py` 文件中。用您喜欢的文本编辑器打开它您可以看到类似下面的信息：

```
root_loc = "../original_data/"

# print validation detail option
if_validation_detail = True

# print single validation result
if_display_single_validation_result = True

# write cross validation prediction result
if_write_cv_prediction = True
write_cv_location = "../original_data/cv_prediction_with_"

prob_mode = False

random_seed = 0

cv_num = 5

######################################### gradient boosting parameters


n_estimators_gdbt = 200
learning_rate_gdbt = 0.1
max_depth_gdbt = 7
min_samples_split_gdbt = 200
min_samples_leaf_gdbt = 250
subsample_gdbt = 1.0
max_feature_gdbt = 'sqrt'

weight_0_gdbt = 1
weight_1000_gdbt = 40
weight_1500_gdbt = 50
weight_2000_gdbt = 90

######################################### AdaBoosting parameters

n_estimators_ada = 700
learning_rate_ada = 0.09
...
...
```

直接使用文本编辑器对上面的参数值进行修改并保存，再次运行 `learning_validation.py` 脚本, 即可调参。

训练不同的模型

在参赛过程中，我队尝试了很多不同的分类模型，有：ExtraTree，RandomForest, GDBT, XGB, SVM, Nerual Network, AdaBoosting, logitic Regression, knn, decisionTree. 关于他们的代码可在项目根目录下的learning_algorithms中找到。

要使用不同的训练模型进行训练，要修改代码 `learning_validation.py` 中的模型名字段。它在代码文件的第150行左右：

```
147     return avg_macro_f1
148
149
150 learning_model = "et"
151
152 #variables.min_weight_fraction_leaf = 0.0001 * float(sys.argv[1])
```

在修改时，请使用模型的简称：gdbt, svm，ada, rf, et, knnBag, nn, xgb等。

预测

在您调整好模型参数后（实际上在比赛中我们已经调参至最佳，所以您不需要动参数），就可以对测试集进行预测了，这将输出一个可以用于线上提交的csv文件，内包含有学生id和对应的获奖情况。

首先，进入项目根目录下的 `online` 下，并运行：

```
$ cd online
$ python add_subsidy_info.py
```

上面命令在训练集 `idexamples_train_std.txt` 的每一行前加上了训练集中学生的获奖情况。并生产一个名为 `training_examples.txt` 的文件，同样存放在 `original_data` 下。

接下来，我们运行：

```
$ python prediction_for_test.py
```

等待一段时间后，您应该能看到以下关于预测结果的输出信息：

```
n_estimators= 1000 max_depth= 50 min_samples_split= 50 min_samples_leaf= 20 max_featu
res= 200 max_leaf_nodes= None criterion= gini min_impurity_split= 1e-07 class_weight=
 {'0': 5.6, '1500': 95, 'min_weight_fraction_leaf': 0, '2000': 130, '1000': 60}
subsidy distribution:-----------------
0 : 8003
1000 : 1983
1500 : 585
2000 : 390
--------------------------------------
```

现在，大功告成。预测完成，输出的文件在 `original_data` 下，您可以将此文件上传至DataCastle网站提交，查看你在线上的分数了。

融合

为了将机器学习分类器的效果发挥到极致，我们要使用融合的办法，将不同模型训练后预测的结果做融合，以得到进一步的精度提升。

关于融合的代码，您可以在项目根目录下的 `blending` 中找到。在融合之前，请确保您已经运行了所有的用于融合的模型，并生成出模型的预测结果。

在我们的测试中，使用以下模型的和参数融合效果最好：

```
name, tree_num, learning_rate, max_depth, select_rate min_split, min_leaf, subsample,
 maxfeature, weight(0:1000:1500:2000), seed
gbdt1, 200, 0.1, 7, 200, 250, 1, sqrt, 0.72:40:65:100, 0
gbdt2, 200, 0.1, 7, 200, 250, 1, sqrt, 1:40:50:90, 0
gbdt3, 200, 0.1, 7, 200, 250, 1, sqrt, 0.72:40:65:200, 0

name, tree_num, learning_rate, max_depth, min_child_weight, gamma, alpha, lambda, col
sample_bytree, subsample, weight, seed
xgb, 1100, 0.06, 3, 6, 0.8, 0, 0.03, 1, 2.1:38:62:79, 0

name, tree_num, learning_rate, weight, seed
ada, 350, 0.1, 8.8:52:75:85

name, tree_num, max_depth, max_leaf_nodes, min_samples_split, min_samples_leaf, max_f
eature, criterion, min_impurity_split, bootstrap, weight
rf,5000, 10, None, 110, 5, sqrt, gini, 5.6:60:95:130
et,1000, 50, None, 50, 20, 200, gini, 1.00E-07, FALSE, 4.7:38:58:75
```

采用以上7个模型参数进行融合，按照

`gbdt1:2, gbdt2:0.5, gbdt3:0.5, xgb:1.5, ada:1.5, rf:1, et:1` 的比例，可以得到线下CV测试3.0的精度，提交至线上后，实际分数为0.03143。为排行榜第一。

队伍信息

队伍名：
Yancy&Zhendong

曾用名：
挖掘技术哪家强，忘记你我做不到

成员：

- 队长：杨轩 yancy100696@gmail.com
- 刘振东 zhendong.nus@gmail.com
- 曹峻许 707109280@qq.com
- 刘一鸣 yimingl816@gmail.com
- 卢健 jluan@connect.ust.hk