Task1 赛题理解

Tip:本次新人赛是Datawhale与天池联合发起的0基础入门系列赛事第四场 —— 零基础入门金融风控之贷款违约预测挑战赛。 赛题以金融风控中的个人信贷为背景，要求选手根据贷款申请人的数据信息预测其是否有违约的可能，以此判断是否通过此项贷款，这是一个典型的分类问题。通过这道赛题来引导大家了解金融风控中的一些业务背景，解决实际问题，帮助竞赛新人进行自我练习、自我提高。

项目地址：<https://github.com/datawhalechina/team-learning-data-mining/tree/master/FinancialRiskControl>

比赛地址：<https://tianchi.aliyun.com/competition/entrance/531830/introduction>

1.1 学习目标

理解赛题数据和目标，清楚评分体系。

完成相应报名，下载数据和结果提交打卡（可提交示例结果），熟悉比赛流程

1.2 了解赛题

* 赛题概况
* 数据概况
* 预测指标
* 分析赛题

1.2.1 赛题概况

比赛要求参赛选手根据给定的数据集，建立模型，预测金融风险。

赛题以预测金融风险为任务，数据集报名后可见并可下载，该数据来自某信贷平台的贷款记录，总数据量超过120w，包含47列变量信息，其中15列为匿名变量。为了保证比赛的公平性，将会从中抽取80万条作为训练集，20万条作为测试集A，20万条作为测试集B，同时会对employmentTitle、purpose、postCode和title等信息进行脱敏。

通过这道赛题来引导大家走进金融风控数据竞赛的世界，主要针对于于竞赛新人进行自我练习、自我提高。

1.2.2 数据概况

|  |  |
| --- | --- |
| **Field** | **Description** |
| id | 为贷款清单分配的唯一信用证标识 |
| loanAmnt | 贷款金额 |
| term | 贷款期限（year） |
| interestRate | 贷款利率 |
| installment | 分期付款金额 |
| grade | 贷款等级 |
| subGrade | 贷款等级之子级 |
| employmentTitle | 就业职称 |
| employmentLength | 就业年限（年） |
| homeOwnership | 借款人在登记时提供的房屋所有权状况 |
| annualIncome | 年收入 |
| verificationStatus | 验证状态 |
| issueDate | 贷款发放的月份 |
| purpose | 借款人在贷款申请时的贷款用途类别 |
| postCode | 借款人在贷款申请中提供的邮政编码的前3位数字 |
| regionCode | 地区编码 |
| dti | 债务收入比 |
| delinquency\_2years | 借款人过去2年信用档案中逾期30天以上的违约事件数 |
| ficoRangeLow | 借款人在贷款发放时的fico所属的下限范围 |
| ficoRangeHigh | 借款人在贷款发放时的fico所属的上限范围 |
| openAcc | 借款人信用档案中未结信用额度的数量 |
| pubRec | 贬损公共记录的数量 |
| pubRecBankruptcies | 公开记录清除的数量 |
| revolBal | 信贷周转余额合计 |
| revolUtil | 循环额度利用率，或借款人使用的相对于所有可用循环信贷的信贷金额 |
| totalAcc | 借款人信用档案中当前的信用额度总数 |
| initialListStatus | 贷款的初始列表状态 |
| applicationType | 表明贷款是个人申请还是与两个共同借款人的联合申请 |
| earliesCreditLine | 借款人最早报告的信用额度开立的月份 |
| title | 借款人提供的贷款名称 |
| policyCode | 公开可用的策略\_代码=1新产品不公开可用的策略\_代码=2 |
| n系列匿名特征 | 匿名特征n0-n14，为一些贷款人行为计数特征的处理 |

1.2.3 预测指标

提交结果为每个测试样本是1的概率，也就是y为1的概率。评价方法为AUC评估模型效果（越大越好）。

1.3 代码示例

本部分为对于数据读取和指标评价的示例。

1.3.1 数据读取pandas

**import** numpy **as** np

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn **import** metrics

*## 1)* 载入训练集和测试集*:*

path = **'~/PycharmProjects/ml/data/task1/'**

train\_file = **'train.csv'**

test\_file = **'testA.csv'**

Train\_data = pd.read\_csv(path + train\_file)

Test\_data = pd.read\_csv(path + test\_file)

print(**'Traind data shape: '**, Train\_data.shape)

print(**'Test data shape: '**, Test\_data.shape)

print(Train\_data.head())

print(Train\_data.info())

print(Train\_data.columns)

print(Train\_data.describe())

Traind data shape: (800000, 47)

Test data shape: (200000, 46)

id loanAmnt term interestRate installment ... n10 n11 n12 n13 n14

0 0 35000.0 5 19.52 917.97 ... 7.0 0.0 0.0 0.0 2.0

1 1 18000.0 5 18.49 461.90 ... 13.0 NaN NaN NaN NaN

2 2 12000.0 5 16.99 298.17 ... 11.0 0.0 0.0 0.0 4.0

3 3 11000.0 3 7.26 340.96 ... 9.0 0.0 0.0 0.0 1.0

4 4 3000.0 3 12.99 101.07 ... 12.0 0.0 0.0 0.0 4.0

[5 rows x 47 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 800000 entries, 0 to 799999

Data columns (total 47 columns):

id 800000 non-null int64

loanAmnt 800000 non-null float64

term 800000 non-null int64

interestRate 800000 non-null float64

installment 800000 non-null float64

grade 800000 non-null object

subGrade 800000 non-null object

employmentTitle 799999 non-null float64

employmentLength 753201 non-null object

homeOwnership 800000 non-null int64

annualIncome 800000 non-null float64

verificationStatus 800000 non-null int64

issueDate 800000 non-null object

isDefault 800000 non-null int64

purpose 800000 non-null int64

postCode 799999 non-null float64

regionCode 800000 non-null int64

dti 799761 non-null float64

delinquency\_2years 800000 non-null float64

ficoRangeLow 800000 non-null float64

ficoRangeHigh 800000 non-null float64

openAcc 800000 non-null float64

pubRec 800000 non-null float64

pubRecBankruptcies 799595 non-null float64

revolBal 800000 non-null float64

revolUtil 799469 non-null float64

totalAcc 800000 non-null float64

initialListStatus 800000 non-null int64

applicationType 800000 non-null int64

earliesCreditLine 800000 non-null object

title 799999 non-null float64

policyCode 800000 non-null float64

n0 759730 non-null float64

n1 759730 non-null float64

n2 759730 non-null float64

n3 759730 non-null float64

n4 766761 non-null float64

n5 759730 non-null float64

n6 759730 non-null float64

n7 759730 non-null float64

n8 759729 non-null float64

n9 759730 non-null float64

n10 766761 non-null float64

n11 730248 non-null float64

n12 759730 non-null float64

n13 759730 non-null float64

n14 759730 non-null float64

dtypes: float64(33), int64(9), object(5)

memory usage: 286.9+ MB

None

Index(['id', 'loanAmnt', 'term', 'interestRate', 'installment', 'grade',

'subGrade', 'employmentTitle', 'employmentLength', 'homeOwnership',

'annualIncome', 'verificationStatus', 'issueDate', 'isDefault',

'purpose', 'postCode', 'regionCode', 'dti', 'delinquency\_2years',

'ficoRangeLow', 'ficoRangeHigh', 'openAcc', 'pubRec',

'pubRecBankruptcies', 'revolBal', 'revolUtil', 'totalAcc',

'initialListStatus', 'applicationType', 'earliesCreditLine', 'title',

'policyCode', 'n0', 'n1', 'n2', 'n3', 'n4', 'n5', 'n6', 'n7', 'n8',

'n9', 'n10', 'n11', 'n12', 'n13', 'n14'],

dtype='object')

id loanAmnt ... n13 n14

count 800000.000000 800000.000000 ... 759730.000000 759730.000000

mean 399999.500000 14416.818875 ... 0.089366 2.178606

std 230940.252013 8716.086178 ... 0.509069 1.844377

min 0.000000 500.000000 ... 0.000000 0.000000

25% 199999.750000 8000.000000 ... 0.000000 1.000000

50% 399999.500000 12000.000000 ... 0.000000 2.000000

75% 599999.250000 20000.000000 ... 0.000000 3.000000

max 799999.000000 40000.000000 ... 39.000000 30.000000

[8 rows x 42 columns]

*##* 混淆矩阵

**import** numpy **as** np

**from** sklearn.metrics **import** confusion\_matrix

y\_pred = [0, 1, 0, 1]

y\_true = [0, 1, 1, 0]

print(**'**混淆矩阵**:\n'**,confusion\_matrix(y\_true, y\_pred))

混淆矩阵:

[[1 1]

[1 1]]

*## P-R*曲线

**import** matplotlib.pyplot **as** plt

**from** sklearn.metrics **import** precision\_recall\_curve

y\_pred = [0, 1, 1, 0, 1, 1, 0, 1, 1, 1]

y\_true = [0, 1, 1, 0, 1, 0, 1, 1, 0, 1]

precision, recall, thresholds = precision\_recall\_curve(y\_true, y\_pred)

plt.plot(precision, recall)

*## ROC*曲线

**from** sklearn.metrics **import** roc\_curve

y\_pred = [0, 1, 1, 0, 1, 1, 0, 1, 1, 1]

y\_true = [0, 1, 1, 0, 1, 0, 1, 1, 0, 1]

FPR,TPR,thresholds=roc\_curve(y\_true, y\_pred)

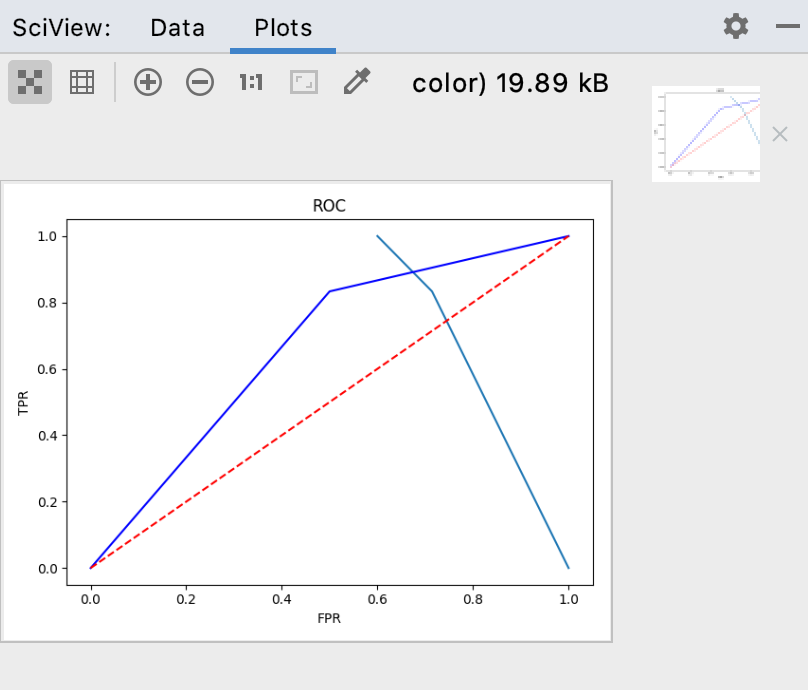
plt.title(**'ROC'**)

plt.plot(FPR, TPR,**'b'**)

plt.plot([0,1],[0,1],**'r--'**)

plt.ylabel(**'TPR'**)

plt.xlabel(**'FPR'**)

plt.show()

*## accuracy*

**from** sklearn.metrics **import** accuracy\_score

y\_pred = [0, 1, 0, 1]

y\_true = [0, 1, 1, 0]

print(**'ACC:'**,accuracy\_score(y\_true, y\_pred))

ACC: 0.5

*## Precision,Recall,F1-score*

**from** sklearn **import** metrics

y\_pred = [0, 1, 0, 1]

y\_true = [0, 1, 1, 0]

print(**'Precision'**,metrics.precision\_score(y\_true, y\_pred))

print(**'Recall'**,metrics.recall\_score(y\_true, y\_pred))

print(**'F1-score:'**,metrics.f1\_score(y\_true, y\_pred))

Precision 0.5

Recall 0.5

F1-score: 0.5

*## AUC*

**import** numpy **as** np

**from** sklearn.metrics **import** roc\_auc\_score

y\_true = np.array([0, 0, 1, 1])

y\_scores = np.array([0.1, 0.4, 0.35, 0.8])

print(**'AUC socre:'**,roc\_auc\_score(y\_true, y\_scores))

AUC socre: 0.75

*## KS*值在实际操作时往往使用*ROC*曲线配合求出*KS*值

**from** sklearn.metrics **import** roc\_curve

y\_pred = [0, 1, 1, 0, 1, 1, 0, 1, 1, 1]

y\_true = [0, 1, 1, 0, 1, 0, 1, 1, 1, 1]

FPR,TPR,thresholds=roc\_curve(y\_true, y\_pred)

KS=abs(FPR-TPR).max()

print(**'KS**值：**'**,KS)

KS值： 0.5238095238095237

1.4 经验总结

1. 复习了AUC，ROC等相关概念和代码
2. 学习了数据，初步了解数据是什么样的结构
3. 环境搭建和使用，使用pycharm 调用matplotlib来生成图形，没能生成，需要在完善一下环境。