



对外经济贸易大学

成绩

2017 — 2018 学年第一学期期末考试

报告题目 P2P 逾期还款预测建模

课程代码及课序号 CMP245-3

课程名称 Python 与大数据分析

学 号 201519040

姓 名 孙晔

学 院 国际经济与贸易学院

专 业 国际组织人才基地班

考试时间 2018-1-15

对外经济贸易大学

本科课程期末论文或其他方式考试评阅表

评分要求：*下列仅为建议指标项，任课教师可根据课程需要进行调整或更改。

请选择总分计算方式：1. 总分为各项指标平均分（ ） 2. 总分为各项指标分总和（ ）

指标项*	选题	观点	材料	文字水平	格式与框架	总分
分数						

论文评语：此页用黑笔填写，论文中用红笔批改。

任课教师签字：

年 月 日

教务处制表

P2P 逾期还款预测建模

孙晔

ID: 201519040

January 19, 2018

Abstract

本报告使用了逻辑回归模型，对给定的数据进行学习，预测客户是否会逾期还款。本文采取默认参数和最优参数两种设置进行比较，f1score 可达到 0.96。最后绘制 ROC 曲线。

关键词：逻辑回归，机器学习，python

Contents

1 前期工作	2
2 数据清洗与整理	2
3 采用逻辑回归模型进行建模	5
3.1 采用默认参数设置	6
3.2 调试以获取最优参数	6
3.3 以最优参数再次建模	7
4 ROC 曲线	7

1 前期工作

本文用到的库有 numpy、pandas 以及 matplotlib.pyplot，使用 ggplot 作图风格，并导入数据

```
In [1]: # 导入所需库
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
plt.style.use("ggplot")

# 导入数据
riskdat = pd.read_sas('riskdat.sas7bdat')
riskdat_c = riskdat.copy()

# 数据的维度：2380 个样本，1033 个自变量，1 个因变量
riskdat.shape
```

Out[1]: (2380, 1034)

2 数据清洗与整理

原有数据中有很多我们无法使用的信息，如一些非数值信息，或者是无效的信息，缺失的信息，为了后面建模的方便，我们将数据进行规整

```
In [2]: # 查看数据
riskdat.head()
```

```
Out[2]:
```

	customer_id	CDTB234	CDCT024	CDTP144	CDTB242	CDTB229	CDTB228	\
0	12344.0	NaN	NaN	NaN	NaN	NaN	NaN	
1	12637.0	200.00	0.0	200.0	0.00	b's24'	200.00	
2	12770.0	38.15	0.0	0.0	504.00	b'S22'	504.00	
3	15819.0	NaN	NaN	NaN	NaN	NaN	NaN	
4	15819.0	100.00	0.0	200.0	916.67	b's22'	916.67	

		CDTB262	CDTB263	CDTB264	...	CDTP175	CDTP177	CDMC337	CDMC335	\
0	b' .'	NaN	b' .'	...	NaN	NaN	NaN	NaN		
1	b'5810'	400.0	b'"null"'	...	0.0	0.0	0.0	0.0		
2	b' 1910'	921.0	b' 1000'	...	0.0	0.0	0.0	0.0		
3	b' .'	NaN	b' .'	...	NaN	NaN	NaN	NaN		
4	b'1910'	800.0	b'"null"'	...	0.0	0.0	0.0	0.0		

	CDMC333	CDMC331	CDTB301	CDTB302	SignMth	y
0	NaN	NaN	NaN	NaN	b' 201704'	0.0
1	0.0	0.0	0.0	0.0	b' 201707'	0.0
2	0.0	0.0	0.0	0.0	b' 201704'	0.0
3	NaN	NaN	NaN	NaN	b' 201707'	0.0


```
4      0.0      0.0      0.0      0.0 b' 201707'  0.0
```

```
[5 rows x 1034 columns]
```

```
In [3]: # 目标变量在正负样本上的分布是不均匀的
```

```
riskdat.y.value_counts()
```

```
Out[3]: 0.0    2334
        1.0     46
        Name: y, dtype: int64
```

```
In [4]: # 每个样本的缺失变量数
```

```
riskdat.missing_var = riskdat.isnull().sum(axis=1)
```

```
In [5]: # 部分样本有 90% 以上的变量是缺失的，直接删去这些样本
        # 剩余 2060 个样本
```

```
riskdat = riskdat.loc[riskdat.missing_var < 100, :]
riskdat.shape
```

```
Out[5]: (2060, 1034)
```

```
In [6]: riskdat.head()
```

```
Out[6]:
```

	customer_id	CDTB234	CDCT024	CDTP144	CDTB242	CDTB229	CDTB228	\
1	12637.0	200.00	0.0	200.0	0.00	b's24'	200.00	
2	12770.0	38.15	0.0	0.0	504.00	b'S22'	504.00	
4	15819.0	100.00	0.0	200.0	916.67	b's22'	916.67	
5	16218.0	27.50	0.0	500.0	48.00	b'S24'	500.00	
6	17451.0	100.00	0.0	1500.0	3178.00	b'S22'	3178.00	

		CDTB262	CDTB263	CDTB264	...	CDTP175	CDTP177	CDMC337	CDMC335	\
1		b'5810'	400.0	b'"null"'	...	0.0	0.0	0.0	0.0	
2	b'	1910'	921.0	b' 1000'	...	0.0	0.0	0.0	0.0	
4		b'1910'	800.0	b'"null"'	...	0.0	0.0	0.0	0.0	
5	b'	7910'	1648.0	b' 7930'	...	0.0	0.0	0.0	0.0	
6	b'	5810'	7913.0	b' .'	...	0.0	0.0	0.0	0.0	

	CDMC333	CDMC331	CDTB301	CDTB302	SignMth	y
1	0.0	0.0	0.0	0.0	b' 201707'	0.0
2	0.0	0.0	0.0	0.0	b' 201704'	0.0
4	0.0	0.0	0.0	0.0	b' 201707'	0.0
5	0.0	0.0	0.0	0.0	b' 201706'	0.0
6	0.0	0.0	0.0	0.0	b' 201706'	0.0

```
[5 rows x 1034 columns]
```

```
In [7]: # 某些变量全部为 0，不能提供有效信息，因此删除
```

```
riskdat = riskdat.loc[:, ~(riskdat == 0).all(axis=0)]
riskdat.shape
```

Out[7]: (2060, 977)

In [8]: # 非数值型变量

```
riskdat_sub2 = riskdat.select_dtypes(include=['object'])
riskdat_sub2.head(3)
```

```
Out[8]:
```

	CDTB229	CDTB262	CDTB264	CDTB266	\
1	b's24'	b'5810'	b'"null"'	b'"null"'	
2	b'S22'	b'1910'	b'1000'	b'.'	
4	b's22'	b'1910'	b'"null"'	b'"null"'	

	CDTB268	CDTB269	\
1	b'\xb9\xe3\xd6\xdd\xca\xd0'	b'"null"'	
2	b'\xba\xf4\xba\xcd\xba\xcc'	b'\xb1\xb1\xbe\xa9\xca\xd0'	
4	b'\xba\xf4\xba\xcd\xba\xcc\xcd\xca\xd0'	b'"null"'	

	CDTB270	CDCT018	CDMC292	CDMC299	...	\
1	b'"null"'	b'"null"'	b'6011'	b'"null"'	...	
2	NaN	NaN	b'4511'	b'7311'	...	
4	b'"null"'	b'"null"'	b'5398'	b'6011'	...	

	CDTB153	CDCA006	\
1	b'"null"'	b'cup'	
2	NaN	b'CUP'	
4	b'\xba\xf4\xba\xcd\xba\xcc\xcd\xca\xd0'	b'cup'	

	CDCA007	CDTP090	CDCA001	CDCA003	\
1	b'"null"'	b'"null"'	b'debit'	b'01'	
2	NaN	NaN	b'debit'	b'1'	
4	b'\xc2\xc3\xd3\xce\xbf\xa8'	b'"null"'	b'debit'	b'01'	

	CDCA005	\
1	b'\xd6\xd0\xb9\xfa\xc5\xa9\xd2\xb5\xd2\xf8\xd0...	
2	b'\xbd\xa8\xc9\xe8\xd2\xf8\xd0\xd0'	
4	b'\xb9\xa4\xc9\xcc\xd2\xf8\xd0\xd0'	

	CDCA004	CDCA002	SignMth
1	b'\xbd\xf0\xcb\xeb\xcd\xa8\xb1\xa6\xbf\xa8(\xd...	b'"null"'	b'201707'
2	b'\xbd\xe1\xcb\xe3\xcd\xa8\xbd\xe8\xbc\xc7\xbf...	NaN	b'201704'
4	b'\xb8\xa3\xc5\xa9\xc1\xe9\xcd\xa8\xbf\xa8'	b'"null"'	b'201707'

[3 rows x 74 columns]

In [9]: # 仅抽取数值型变量

```
riskdat_sub1 = riskdat.select_dtypes(exclude=['object'])
riskdat_sub1.head(3)
```

```
Out[9]:
```

	customer_id	CDTB234	CDCT024	CDTP144	CDTB242	CDTB228	CDTB263	CDTB265	\
1	12637.0	200.00	0.0	200.0	0.00	200.00	400.0	0.0	

2	12770.0	38.15	0.0	0.0	504.00	504.00	921.0	160.0
4	15819.0	100.00	0.0	200.0	916.67	916.67	800.0	0.0

	CDTB267	CDTB260	...	CDTP173	CDTP171	CDTB303	CDTB304	CDTP172	\
1	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	
2	0.0	2.0	...	0.0	0.0	0.0	0.0	0.0	
4	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	

	CDTP175	CDTP177	CDTB301	CDTB302	y
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0

[3 rows x 903 columns]

```
In [10]: # 用均值填补缺失的数值变量
riskdat_sub1 = riskdat_sub1.fillna(riskdat_sub1.mean())
riskdat_sub1.shape
```

Out[10]: (2060, 903)

3 采用逻辑回归模型进行建模

对于二分类（逾期 / 不逾期），逻辑回归模型可以很好的达到目标。

```
In [11]: # 定义 X 与 y 方便建模
X1, y1 = riskdat_sub1.iloc[:, 1:-1], riskdat_sub1.y
X1.shape, y1.shape
```

Out[11]: ((2060, 901), (2060,))

```
In [12]: # 建模所需库
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.cross_validation import train_test_split
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
```

```
In [13]: # 正负样本比例
np.mean(y1 == 0), np.mean(y1 == 1)
```

Out[13]: (0.98640776699029131, 0.013592233009708738)

```
In [14]: # 训练集和测试集拆分
X1_train, X1_val, y1_train, y1_val = train_test_split(X1.values, y1.values, test_size=0.3)
X1_train.shape, y1_train.shape, X1_val.shape, y1_val.shape
```

Out[14]: ((1442, 901), (1442,)), (618, 901), (618,))

```
In [15]: # 训练集中的正负样本比例
np.mean(y1_train == 0), np.mean(y1_train == 1)
```

Out[15]: (0.9868238557558946, 0.013176144244105409)

3.1 采用默认参数设置

```
In [16]: # 构建一个逻辑回归模型，采用默认参数设置
lr_clf1 = LogisticRegression(class_weight={0: 0.03, 1: 0.97})
lr_clf1.fit(X1_train, y1_train)
y1_train_pred = lr_clf1.predict(X1_train)
print("Confusion matrix (training):\n {0}\n".format(confusion_matrix(y1_train, y1_train_pred)))
print("Classification report (training):\n {0}".format(classification_report(y1_train, y1_train_pred)))

y1_val_pred = lr_clf1.predict(X1_val)
print("Confusion matrix (validation):\n {0}\n".format(confusion_matrix(y1_val, y1_val_pred)))
print("Classification report (validation):\n {0}".format(classification_report(y1_val, y1_val_pred)))
```

Confusion matrix (training):

```
[[1370  53]
 [  1 18]]
```

Classification report (training):

	precision	recall	f1-score	support
0.0	1.00	0.96	0.98	1423
1.0	0.25	0.95	0.40	19
avg / total	0.99	0.96	0.97	1442

Confusion matrix (validation):

```
[[565  44]
 [  6  3]]
```

Classification report (validation):

	precision	recall	f1-score	support
0.0	0.99	0.93	0.96	609
1.0	0.06	0.33	0.11	9
avg / total	0.98	0.92	0.95	618

3.2 调试以获取最优参数

```
In [17]: # parameter tuning
# lr_clf_tuned = LogisticRegression(class_weight={0: 0.03, 1: 0.97})
# lr_clf_params = {
#     "penalty": ["l1", "l2"],
#     "C": [1, 1.3, 1.5, 1.7, 2]
# }
# lr_clf_cv = GridSearchCV(lr_clf_tuned, lr_clf_params, cv=5)
```

```
# lr_clf_cv.fit(X1_train, y1_train)
# print(lr_clf_cv.best_params_)
```

3.3 以最优参数再次建模

In [18]: # 采用最优参数再次构建逻辑回归模型

```
lr_clf2 = LogisticRegression(penalty="l1", C=1.5, class_weight={0: 0.02, 1: 0.98})
lr_clf2.fit(X1_train, y1_train)
y1_train_pred = lr_clf2.predict(X1_train)
print("Confusion matrix (training):\n {0}\n".format(confusion_matrix(y1_train, y1_train_pred)))
print("Classification report (training):\n {0}".format(classification_report(y1_train, y1_train_pred)))

y1_val_pred = lr_clf2.predict(X1_val)
print("Confusion matrix (validation):\n {0}\n".format(confusion_matrix(y1_val, y1_val_pred)))
print("Classification report (validation):\n {0}".format(classification_report(y1_val, y1_val_pred)))
```

Confusion matrix (training):

```
[[1423   0]
 [   0  19]]
```

Classification report (training):

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	1423
1.0	1.00	1.00	1.00	19
avg / total	1.00	1.00	1.00	1442

Confusion matrix (validation):

```
[[586  23]
 [   7   2]]
```

Classification report (validation):

	precision	recall	f1-score	support
0.0	0.99	0.96	0.98	609
1.0	0.08	0.22	0.12	9
avg / total	0.97	0.95	0.96	618

4 ROC 曲线

In [19]: # 绘制 ROC 曲线

```
y1_valid_score_lr1 = lr_clf1.predict_proba(X1_val)
y1_valid_score_lr2 = lr_clf2.predict_proba(X1_val)
```

```

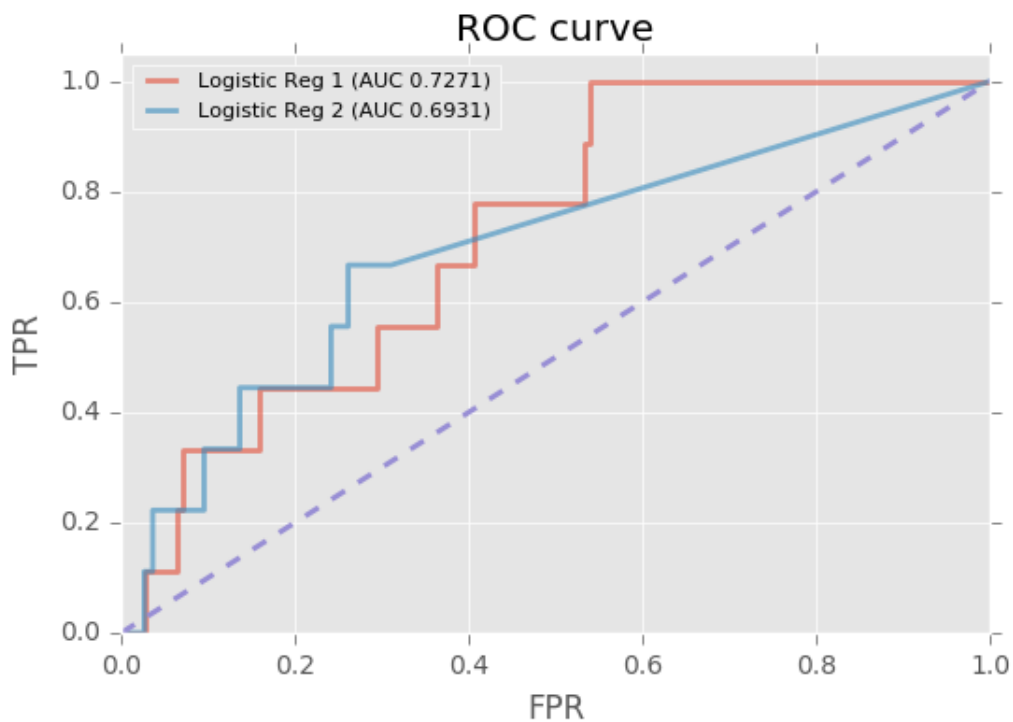
fpr_lr1, tpr_lr1, thresholds_lr1 = roc_curve(y1_val, y1_valid_score_lr1[:, 1])
fpr_lr2, tpr_lr2, thresholds_lr2 = roc_curve(y1_val, y1_valid_score_lr2[:, 1])

roc_auc_lr1 = auc(fpr_lr1, tpr_lr1)
roc_auc_lr2 = auc(fpr_lr2, tpr_lr2)

plt.plot(fpr_lr1, tpr_lr1, fpr_lr2, tpr_lr2, lw=2, alpha=.6)
plt.plot([0, 1], [0, 1], lw=2, linestyle="--")
plt.xlim([0, 1])
plt.ylim([0, 1.05])
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC curve")
plt.legend(["Logistic Reg 1 (AUC {:.4f})".format(roc_auc_lr1),
           "Logistic Reg 2 (AUC {:.4f})".format(roc_auc_lr2)], fontsize=8, loc=2)

```

Out[19]: <matplotlib.legend.Legend at 0x11b3bcf8>



References

- [1] Fang Z, Zhang J, Zhiyuan F, et al., *Study on P2P E-Finance Platform System: A Case in China*[C], international conference on e-business engineering, 2014: 331-337.

- [2] Hosmer D W, Lemeshow S. *Applied logistic regression*[J]. Technometrics, 2000.
- [3] Wesleyan University, *Python Lesson 3: Logistic Regression for a Binary Response Variable, pt. 1*, Coursera, <https://www.coursera.org/learn/regression-modeling-practice/lecture/UCzPC/python-lesson-3-logistic-regression-for-a-binary-response-variable-pt-1>.