法律声明

- 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

数据清洗和特征选择

邹博

主要内容

- 内容
  - 庄家与赔率
  - Nagel-Schreckenberg交通流模型
  - Pandas数据读取和处理
  - Fuzzywuzzy字符串模糊查找
  - 数据清洗和校正
  - 特征提取主成分分析PCA
  - One-hot编码
- 思考：
  - 字符串编辑距离
  - ROC与AUC
  - 分类器：随机森林、Logistic回归

本次说明

- 本PPT后面仅列举使用Python库的效果截图，详细内容请参考该PPT的配套代码。

赔率

- 赔率最早出现在赛马中,1790年由英国人奥格登发明。
  - 中国从2001年发行足彩开始引入赔率。
- 赔率的举例定义：
- 浔阳江畔艄公张横和张顺正进行400米自由泳比赛，宋江开赌场做庄，规定：张横赢赔率为3，张顺赢赔率为2。假定不存在平局。赌徒李逵为张横下注10两。
  - 比赛结束后，若最终张横赢，则宋江付赌徒李逵30两(10×3)，赌资10两归庄家宋江所有，即李逵赚20两。
  - 若张顺赢，赌资10两归庄家宋江所有，即李逵赔10两。

问题分析

X	张横	张顺
P	0.8	0.2
y	1.25	5

- 假定张横赢的概率为0.8，宋江给出的赔率为张横1.25，张顺5，则宋江的盈亏分析如下：
  - 为表述方便，张横赢简称“张横”，张顺赢简称“张顺”。
- 假定所有赌徒中，共有a元买张横，b元买张顺，则开赛前宋江收入为a+b元
- 开赛后的赔付期望为：

$$E(y)=\sum_i p_i y_i = 0.8\times 1.25\times a + 0.2\times 5\times b = a+b$$

赔率分析

X	张横	张顺
P	0.8	0.2
y	1.25	5

- 从上述结论知：使用y=1/p作为赔率，会使得庄家在期望上不赔不赚。
  - 这即“公平赔率”：y_{fair}
  - ——没有利润，这显然是庄家不希望看到的。
- 实际问题中，庄家总是会将公平赔率乘以某小于1的系数α，即得到真实赔率：

$$y = \alpha \cdot y_{fair} = \alpha / p$$

  - 庄家对于α取值不公开。

身边的故事

- 3月12日的拼团人数和赔率

《机器学习》升级版IV，从理论到实践，邹博主讲

原　价　¥899.00

拼团价	¥599.00	100人以上
	¥499.00	200人以上
	¥399.00	300人以上 (当前价)

团长:小象学院　1026 人参团

剩余时间 2天内

参团

XXML升级版IV　3群(494)

3月12日 凌晨00:48

赌博小游戏：
《机器学习·升级版IV期》还有2天多就结束拼团了（截团时间应该是3月14日上午10:00）。最终参团人数是素数还是合数?
我坐庄，我开出的赔率是：
素数：5.5
合数：1.1

例如：
1、如果最终参团人数是素数，且你为素数下注1元，则我返还你5.5元。
2、如果最终参团人数是素数，且你为合数下注1元，则1元不返还。
3、如果最终参团人数是合数，且你为素数下注1元，则1元不返还。
4、如果最终参团人数是合数，且你为合数下注1元，则我返还你1.1元。

计算赔率

```
p = np.array(filter(is_prime3, range(2, b+1)))
p = p[p >= a]
print p
p_rate = float(len(p)) / float(b-a+1)
print '素数的概率：', p_rate, '\t',
print '公正赔率：', 1/p_rate
print '合数的概率：', 1-p_rate, '\t',
print '公正赔率：', 1 / (1-p_rate)
```

```
Prime
[1181 1187 1193 1201 1213 1217 1223 1229]
素数的概率： 0.156862745098    公正赔率： 6.375
合数的概率： 0.843137254902    公正赔率： 1.18604651163
```

- 拼团人数当时是1026人，尚有两天结束，根据历史先验，假定1天参团人数为100人，则最终参团人数为1226左右。考虑到3月12日为星期日，参团人数或许略低，因此大体参团区间可能是[1180,1230]。
- 计算该区间的素数
  - [1181 1187 1193 1201 1213 1217 1223 1229]
  - 素数的概率：0.157　　公正赔率：6.375
  - 合数的概率：0.843　　公正赔率：1.186

计算庄家的盈亏期望

- 实际给出的赔率为5.5和1.1，带入赔率公式$y=\alpha/p$得到α分别是0.863和0.927，从而庄家盈利期望为：

$$\begin{aligned} E &= (a+b)-E(y) \\ &= a+b-(\alpha_1 \cdot a+\alpha_2 \cdot b) \\ &= (1-\alpha_1)\cdot a+(1-\alpha_2)\cdot b \end{aligned}$$

- 若假定a=b，则庄家盈利率为10.5%
  - 假定赌徒不分析场景，随机选边下注。
- 若假定a/b=0.157/0.843，则庄家盈利率为12.7%
  - 假定赌徒下注前经过了细致分析，下注比例与场景概率相符。
- 结论：无论如何，庄家肯定赚。
  - 留言正值午夜，最终只有1人参与该游戏，事实上没有完成。
  - 人算不如天算，最终报名人数为1673人，区间估计严重偏离。

公路堵车概率模型

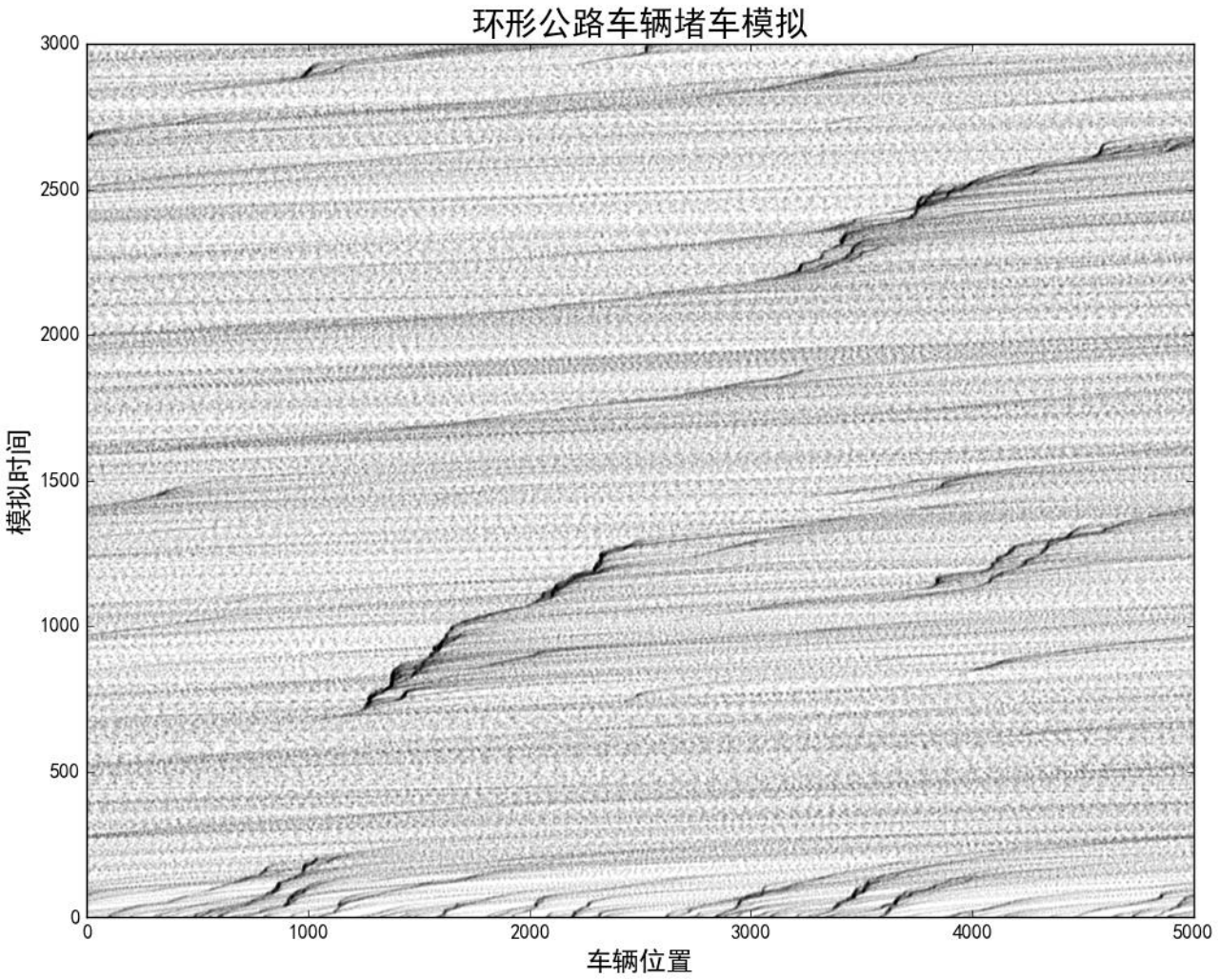
- Nagel-Schreckenberg交通流模型
- 路面上有N辆车，以不同的速度向前行驶，模拟堵车问题。有以下假设：
  - 假设某辆车的当前速度是v。
  - 若前方可见范围内没车，则它在下一秒的车速提高到v+1，直到达到规定的最高限速。
  - 若前方有车，前车的距离为d，且d＜v，则它下一秒的车速降低到d-1。
  - 每辆车会以概率p随机减速v-1。

Nagel-Schreckenberg 模型模拟

```
path = 5000      # 环形公路的长度
n = 100          # 公路中的车辆数目
v0 = 5           # 车辆的初始速度
p = 0.3          # 随机减速概率
Times = 3000

np.random.seed(0)
x = np.random.rand(n) * path
x.sort()
v = np.tile([v0], n).astype(np.float)

plt.figure(figsize=(10, 8), facecolor=
for t in range(Times):
    plt.scatter(x, [t]*n, s=1, c='k',
    for i in range(n):
        if x[(i+1)%n] > x[i]:
            d = x[(i+1) % n] - x[i]
        else:
            d = path - x[i] + x[(i+1)
        if v[i] < d:
            if np.random.rand() > p:
                v[i] += 1
            else:
                v[i] -= 1
        else:
            v[i] = d - 1
    v = v.clip(0, 150)
    x += v
    clip(x, path)
plt.xlim(0, path)
plt.ylim(0, Times)
plt.xlabel(u'车辆位置', fontsize=16)
plt.ylabel(u'模拟时间', fontsize=16)
plt.title(u'环形公路车辆堵车模拟', font
plt.tight_layout(pad=2)
plt.show()
```



Pandas

	A	B	C	D	E	F	G	H	I
1	account	name	street	city	state	postal-code	Jan	Feb	Mar
2	211829	Kerluke, Koepp and Hilpert	34456 Sean Highway	New Jaycob	Texas	28752	10000	62000	35000
3	320563	Walter-Trantow	1311 Alvis Tunnel	Port Khadijah	NorthCarolina	38365	95000	45000	35000
4	648336	Bashirian, Kunde and Price	62184 Schamberger Underpass Apt. 231	New Lilianland	Iowa	76517	91000	120000	35000
5	109996	D'Amore, Gleichner and Bode	155 Fadel Crescent Apt. 144	Hyattburgh	Maine	46021	45000	120000	10000
6	121213	Bauch-Goldner	7274 Marissa Common	Shanahanchester	California	49681	162000	120000	35000
7	132971	Williamson, Schumm and Hettinger	89403 Casimer Spring	Jeremieburgh	Arkansas	62785	150000	120000	35000
8	145068	Casper LLC	340 Consuela Bridge Apt. 400	Lake Gabriellaton	Mississipi	18008	62000	120000	70000
9	205217	Kovacek-Johnston	91971 Cronin Vista Suite 601	Deronville	RhodeIsland	53461	145000	95000	35000
10	209744	Champlin-Morar	26739 Grant Lock	Lake Juliannton	Pennsylvania	64415	70000	95000	35000
11	212303	Gerhold-Maggio	366 Maggio Grove Apt. 998	North Ras	Idaho	46308	70000	120000	35000
12	214098	Goodwin, Homenick and Jerde	649 Cierra Forks Apt. 078	Rosaberg	Tenessee	47743	45000	120000	55000
13	231907	Hahn-Moore	18115 Olivine Throughway	Norbertomouth	NorthDakota	31415	150000	10000	162000
14	242368	Frami, Anderson and Donnelly	182 Bertie Road	East Davian	Iowa	72686	162000	120000	35000
15	268755	Walsh-Haley	2624 Beatty Parkways	Goodwinmouth	RhodeIsland	31919	55000	120000	35000
16	273274	McDermott PLC	8917 Bergstrom Meadow	Kathryneborough	Delaware	27933	150000	120000	70000

- Fuzzywuzzy - Levenshtein distance
- 模糊查询与替换

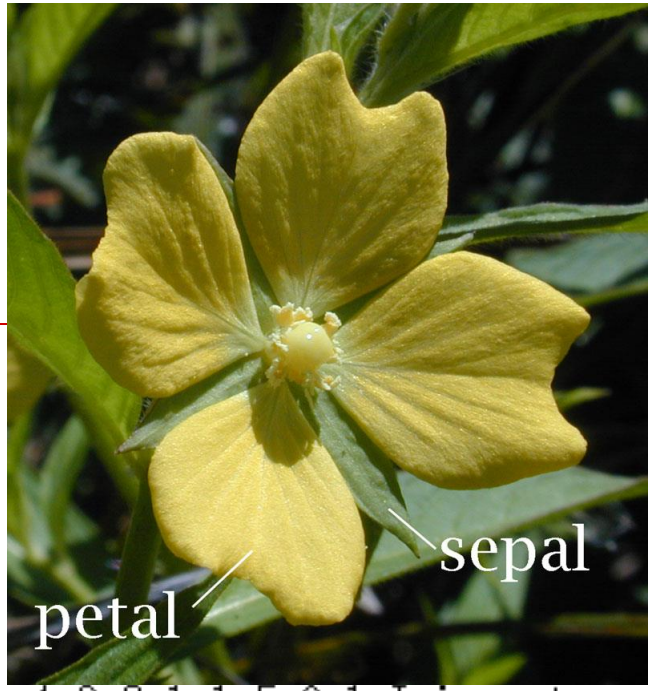
	A	B	C	D	E	F	G	H	I	J	K
1	account	name	street	city	state	SC	postal-code	Jan	Feb	Mar	total
2	211829	Kerluke, Koepp and Hilpert	34456 Sean Highway	New Jaycob	Texas	TX	28752	10000	62000	35000	107000
3	320563	Walter-Trantow	1311 Alvis Tunnel	Port Khadijah	North Carolina	NC	38365	95000	45000	35000	175000
4	648336	Bashirian, Kunde and Price	62184 Schamberger Underpass Apt. 231	New Lilianland	Iowa	IA	76517	91000	120000	35000	246000
5	109996	D'Amore, Gleichner and Bode	155 Fadel Crescent Apt. 144	Hyattburgh	Maine	ME	46021	45000	120000	10000	175000
6	121213	Bauch-Goldner	7274 Marissa Common	Shanahanchester	California	CA	49681	162000	120000	35000	317000
7	132971	Williamson, Schumm and Hettinger	89403 Casimer Spring	Jeremieburgh	Arkansas	AR	62785	150000	120000	35000	305000
8	145068	Casper LLC	340 Consuela Bridge Apt. 400	Lake Gabriellaton	Mississippi	MS	18008	62000	120000	70000	252000
9	205217	Kovacek-Johnston	91971 Cronin Vista Suite 601	Deronville	Rhode Island	RI	53461	145000	95000	35000	275000
10	209744	Champlin-Morar	26739 Grant Lock	Lake Juliannton	Pennsylvania	PA	64415	70000	95000	35000	200000
11	212303	Gerhold-Maggio	366 Maggio Grove Apt. 998	North Ras	Idaho	ID	46308	70000	120000	35000	225000
12	214098	Goodwin, Homenick and Jerde	649 Cierra Forks Apt. 078	Rosaberg	Tennessee	TN	47743	45000	120000	55000	220000
13	231907	Hahn-Moore	18115 Olivine Throughway	Norbertomouth	North Dakota	ND	31415	150000	10000	162000	322000
14	242368	Frami, Anderson and Donnelly	182 Bertie Road	East Davian	Iowa	IA	72686	162000	120000	35000	317000
15	268755	Walsh-Haley	2624 Beatty Parkways	Goodwinmouth	Rhode Island	RI	31919	55000	120000	35000	210000
16	273274	McDermott PLC	8917 Bergstrom Meadow	Kathryneborough	Delaware	DE	27933	150000	120000	70000	340000
17	0	0	0	0	0		0	1462000	1507000	717000	3686000

鸢尾花数据集

- 鸢尾花数据集或许是最有名的模式识别测试数据。
  - 早在1936年，模式识别的先驱Fisher就在论文“The use of multiple measurements in taxonomic problems”中使用了它(直至今日该论文仍然被频繁引用)。
- 该数据集包括3个鸢尾花类别，每个类别有50个样本。其中一个类别是与另外两类线性可分的，而另外两类不能线性可分。
  - 由于Fisher的最原始数据集存在两个错误(35号和38号样本)，实验中我们使用的是修正过的数据。
- 下载链接：http://archive.ics.uci.edu/ml/datasets/Iris

数据描述

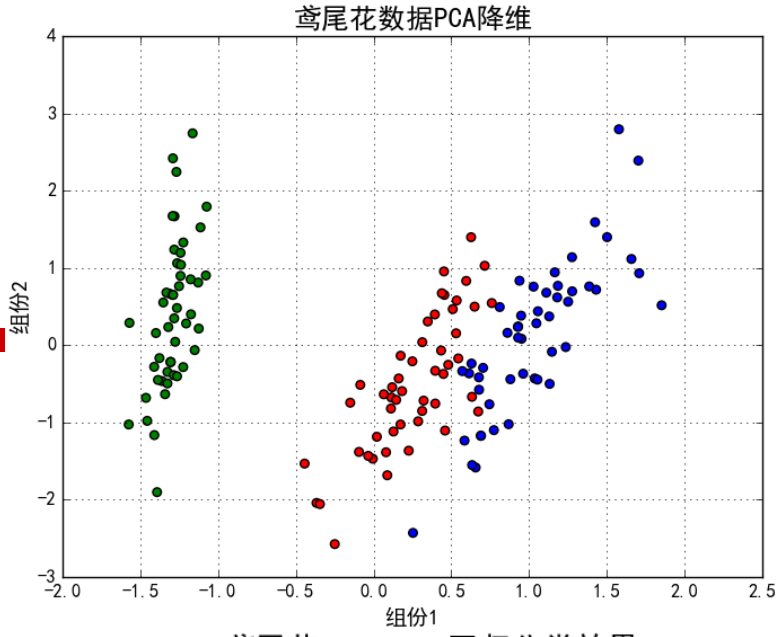
- 该数据集共150行，每行1个样本。每个样本有5个字段，分别是
  - 花萼长度(单位cm)
  - 花萼宽度(单位：cm)
  - 花瓣长度(单位：cm)
  - 花瓣宽度(单位：cm)
  - 类别(共3类)
    - Iris Setosa
    - Iris Versicolour
    - Iris Virginica

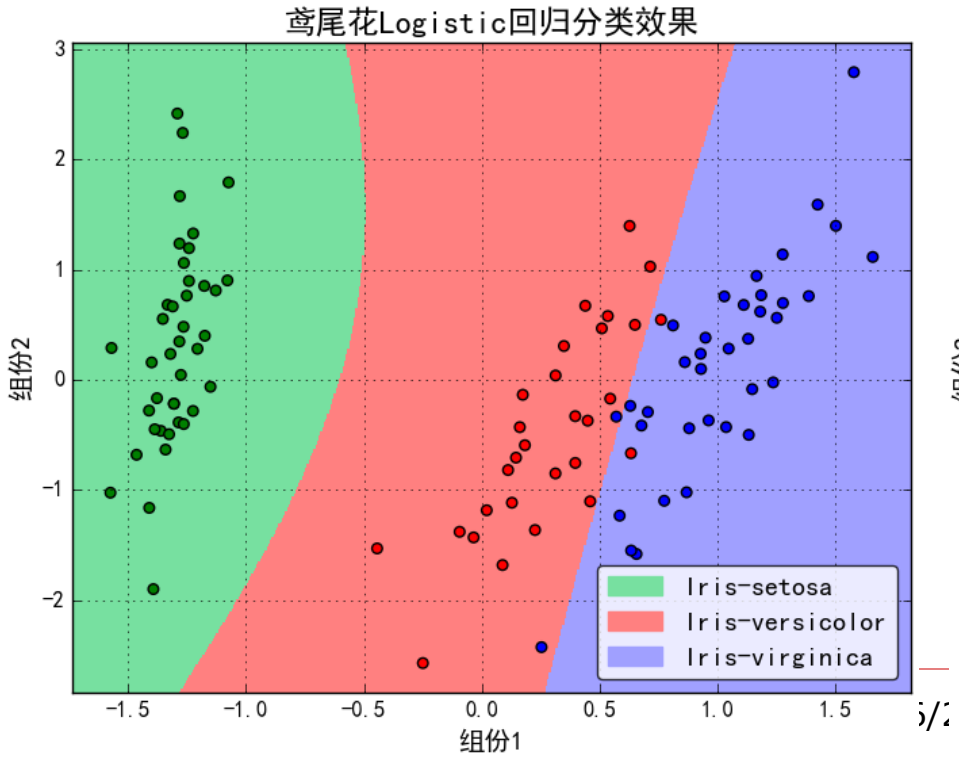


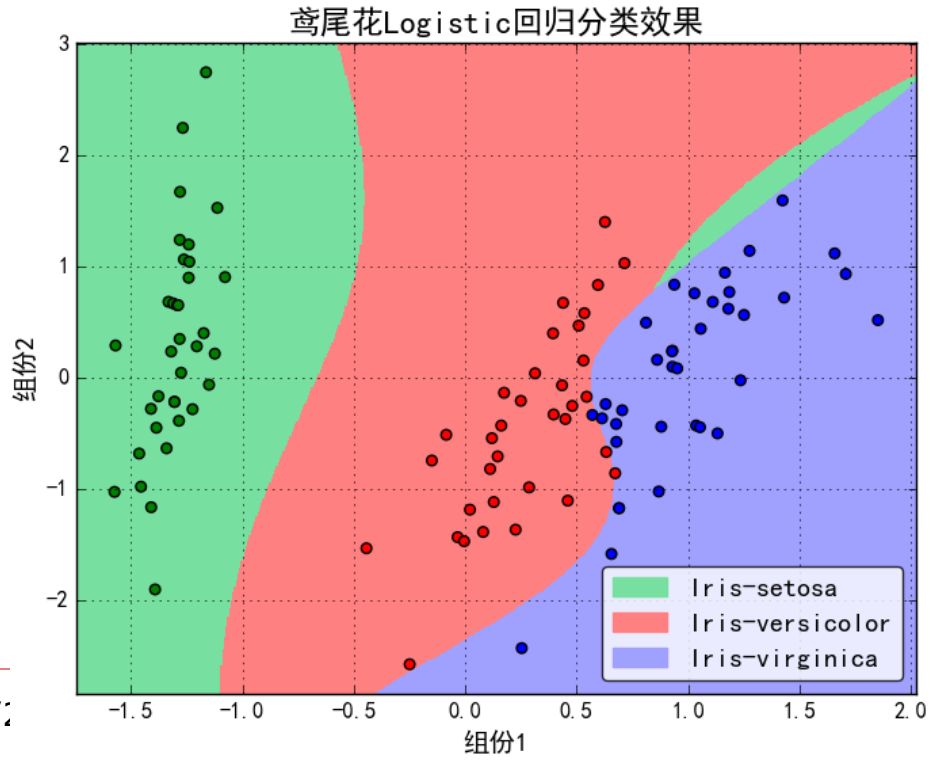

```
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
```

主成分分析PCA

- 多项式特征：2/3
- 管道Pipeline

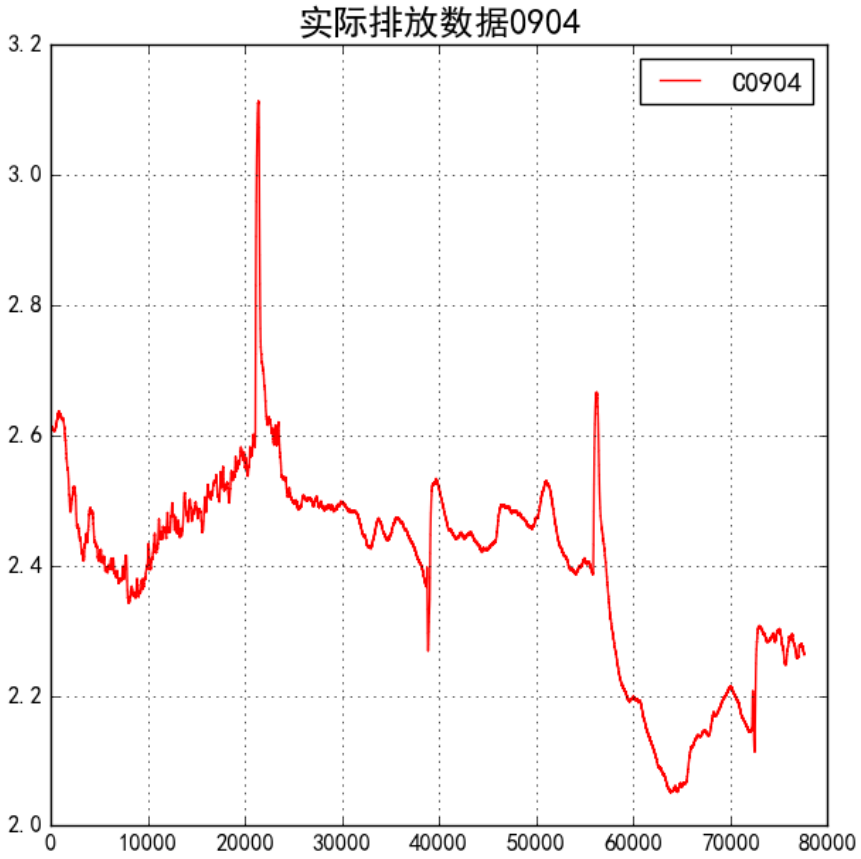


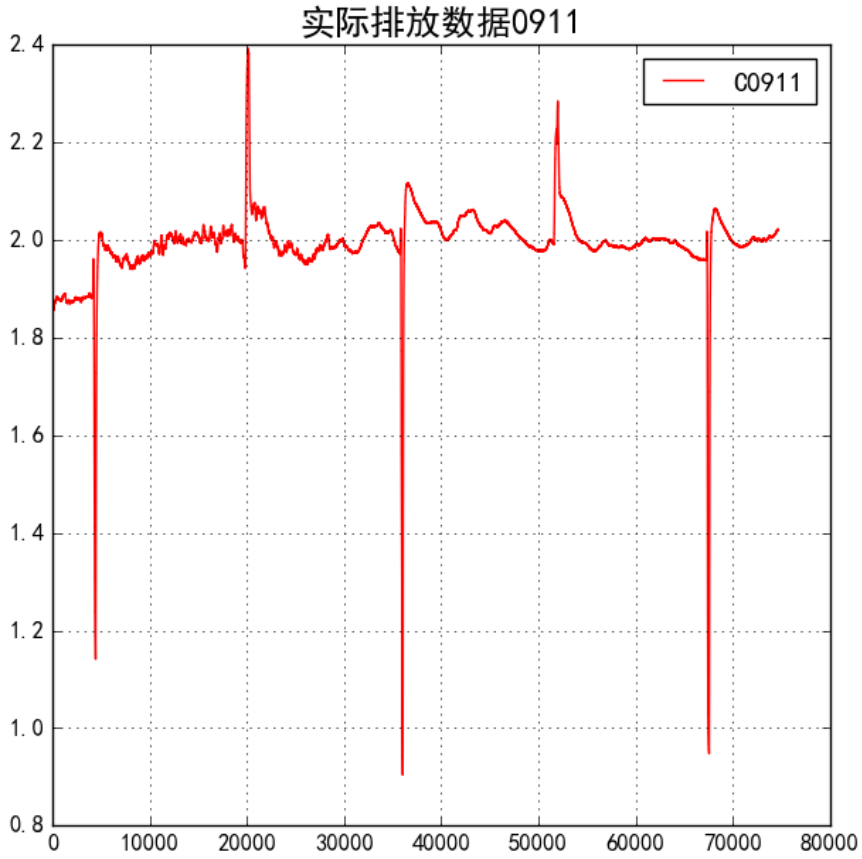


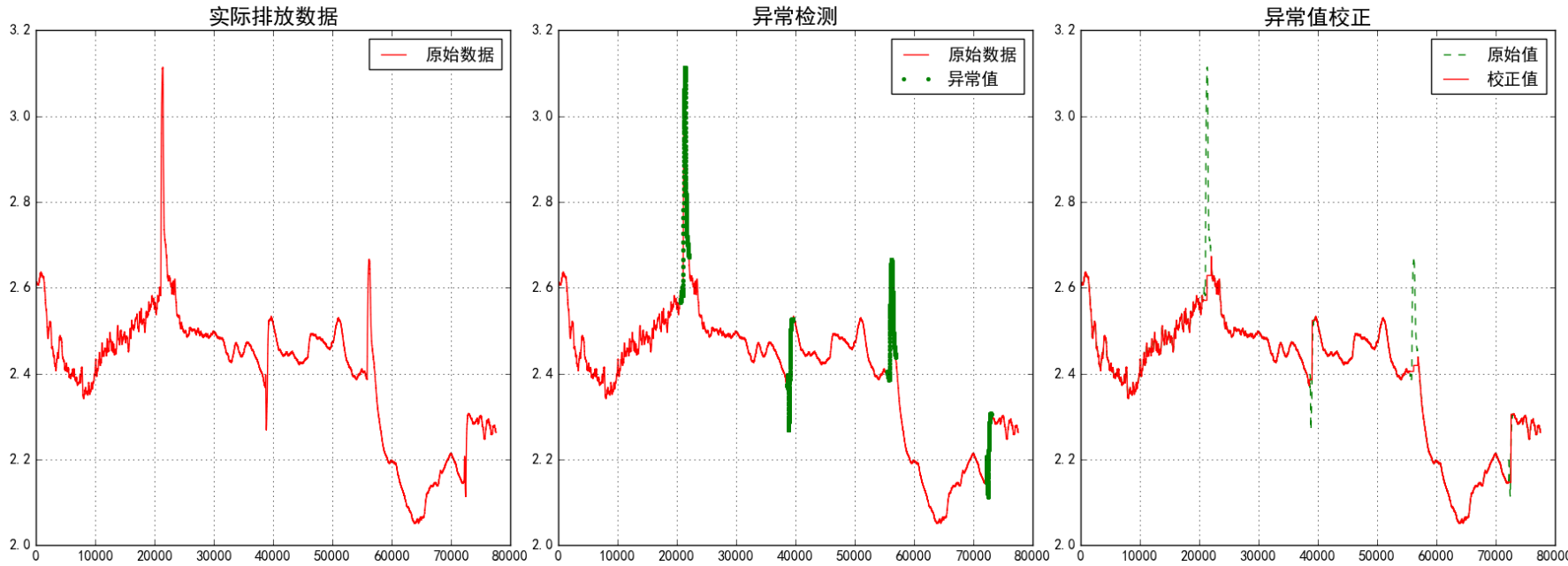


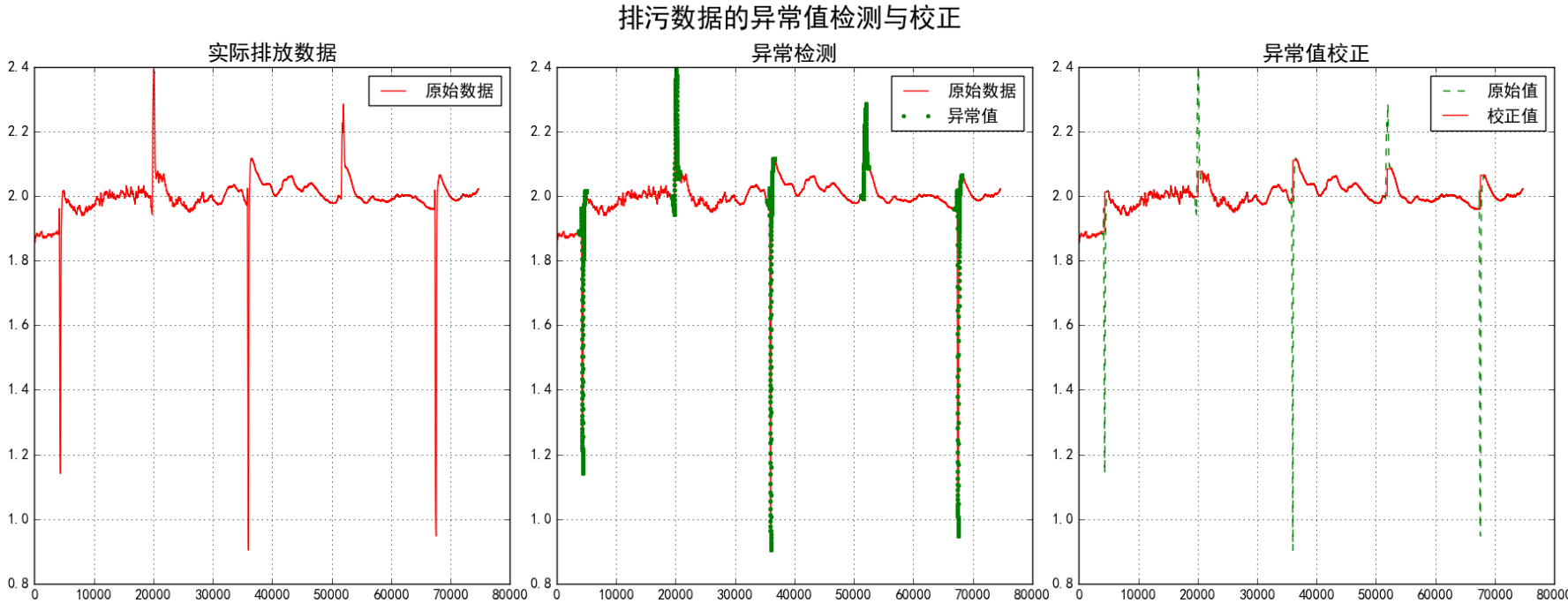
数据清洗和数据处理

如何找到下图中的异常值






排污数据的异常值检测与校正
实际排放数据
原始数据
异常检测
原始数据
异常值
异常值校正
原始值
校正值


排污数据的异常值检测与校正
实际排放数据
原始数据
异常检测
原始数据
异常值
异常值校正
原始值
校正值

车辆数据描述

- 该数据共1728个样本，每行为1个样本。每个样本有7个特征：
  - 购买价格：low/med/high/vhigh
  - 维护价格：low/med/high/vhigh
  - 车门数量：2/3/4/5more
  - 载人数目：2/4/more
  - 后备箱大小：small/med/big
  - 安全程度：low/med/high
  - 接受程度：unacc/acc/good/vgood

	A	B	C	D	E	F	G
1	buy	maintain	doors	persons	boot	safety	accept
2	vhigh	vhigh	2	2	small	low	unacc
3	vhigh	vhigh	2	2	small	med	unacc
4	vhigh	vhigh	2	2	small	high	unacc
5	vhigh	vhigh	2	2	med	low	unacc
6	vhigh	vhigh	2	2	med	med	unacc
7	vhigh	vhigh	2	2	med	high	unacc
8	vhigh	vhigh	2	2	big	low	unacc
9	vhigh	vhigh	2	2	big	med	unacc
10	vhigh	vhigh	2	2	big	high	unacc
11	vhigh	vhigh	2	4	small	low	unacc
12	vhigh	vhigh	2	4	small	med	unacc
13	vhigh	vhigh	2	4	small	high	unacc
14	vhigh	vhigh	2	4	med	low	unacc
15	vhigh	vhigh	2	4	med	med	unacc
16	vhigh	vhigh	2	4	med	high	unacc
17	vhigh	vhigh	2	4	big	low	unacc
18	vhigh	vhigh	2	4	big	med	unacc
19	vhigh	vhigh	2	4	big	high	unacc
20	vhigh	vhigh	2	more	small	low	unacc
21	vhigh	vhigh	2	more	small	med	unacc
22	vhigh	vhigh	2	more	small	high	unacc
23	vhigh	vhigh	2	more	med	low	unacc
24	vhigh	vhigh	2	more	med	med	unacc
25	vhigh	vhigh	2	more	med	high	unacc
26	vhigh	vhigh	2	more	big	low	unacc
27	vhigh	vhigh	2	more	big	med	unacc
28	vhigh	vhigh	2	more	big	high	unacc
29	vhigh	vhigh	3	2	small	low	unacc
30	vhigh	vhigh	3	2	small	med	unacc
31	vhigh	vhigh	3	2	small	high	unacc
32	vhigh	vhigh	3	2	med	low	unacc
33	vhigh	vhigh	3	2	med	med	unacc
34	vhigh	vhigh	3	2	med	high	unacc
35	vhigh	vhigh	3	2	big	low	unacc
36	vhigh	vhigh	3	2	big	med	unacc

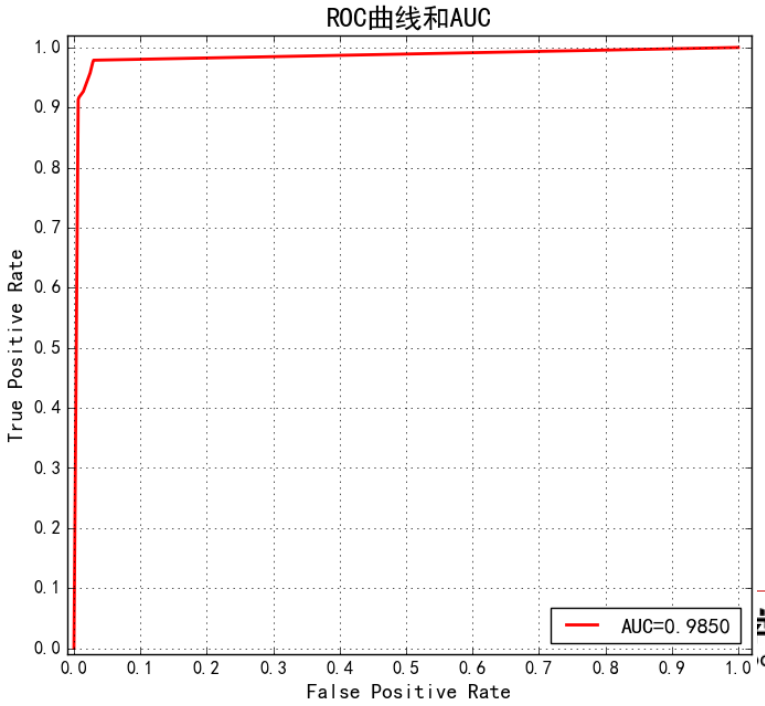
决策树和随机森林分类

```
x = data.loc[:, columns[:-1]]
y = data['accept']
x, x_test, y, y_test = train_test_split(x, y, train_size=0.7)
if random_forest:
    clf = RandomForestClassifier(n_estimators=100, criterion='gini', max_depth=12, min_samples_split=5
else:
    clf = DecisionTreeClassifier(criterion='gini', max_depth=12, min_samples_split=5, max_features=5)
if cross_validation:
    model = GridSearchCV(clf, param_grid={'max_depth': np.arange(10,20),
                                          'min_samples_split': np.arange(5, 20),
                                          'max_features': np.arange(1, 7)
                                          }, cv=3)
    model.fit(x, y)
    print model.best_params_
```

3.pca 1.RF

1720	1	1	3	2	2	2	0
1721	1	1	3	2	2	0	1
1722	1	1	3	2	1	1	2
1723	1	1	3	2	1	2	1
1724	1	1	3	2	1	0	3
1725	1	1	3	2	0	1	2
1726	1	1	3	2	0	2	1
1727	1	1	3	2	0	0	3

[1728 rows x 7 columns]
训练集精确度： 0.988420181969
测试集精确度： 0.967244701349
Micro AUC： 0.991795397255
Micro AUC(System)： 0.991795397255
Macro AUC： 0.987885354719



One-hot编码

	buy	maintain	doors	persons	boot	safety	accept
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc
5	vhigh	vhigh	2	2	med	high	unacc
6	vhigh	vhigh	2	2	big	low	unacc
7	vhigh	vhigh	2	2	big	med	unacc
8	vhigh	vhigh	2	2	big	high	unacc
9	vhigh	vhigh	2	4	small	low	unacc

	buy_high	buy_low	buy_med	buy_vhigh	maintain_high	maintain_low	maintain_med	maintain_vhigh	doors_2	doors_3
0	0	0	0	1	0	0	0	1	1	0
1	0	0	0	1	0	0	0	1	1	0
2	0	0	0	1	0	0	0	1	1	0
3	0	0	0	1	0	0	0	1	1	0
4	0	0	0	1	0	0	0	1	1	0
5	0	0	0	1	0	0	0	1	1	0
6	0	0	0	1	0	0	0	1	1	0
7	0	0	0	1	0	0	0	1	1	0
8	0	0	0	1	0	0	0	1	1	0
9	0	0	0	1	0	0	0	1	1	0

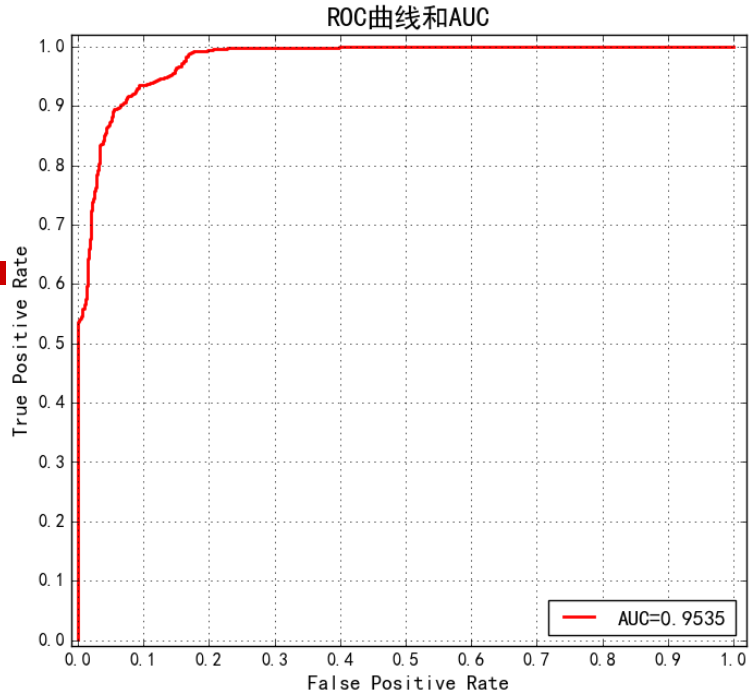
doors_4	doors_5more	persons_2	persons_4	persons_more	boot_big	boot_med	boot_small	safety_high	safety_low	safety_med
0	0	1	0	0	0	0	1	0	1	0
0	0	1	0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	1	1	0	0
0	0	1	0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	0	0	1
0	0	1	0	0	1	0	0	1	0	0
0	0	0	1	0	0	0	1	0	1	0

Logistic回归

```
# one-hot编码
x = pd.DataFrame()
for col in columns[:-1]:
    t = pd.get_dummies(data[col])
    t = t.rename(columns=lambda x: col+'_'+str(x))
    x = pd.concat((x, t), axis=1)
print x.head(10)
# print x.columns
y = pd.Categorical(data['accept']).codes

x, x_test, y, y_test = train_test_split(x, y, train_size=0.7)
clf = LogisticRegressionCV(Cs=np.logspace(-3, 4, 8), cv=5)
clf.fit(x, y)
print clf.C_
y_hat = clf.predict(x)
print '训练集精确度：', metrics.accuracy_score(y, y_hat)
y_test_hat = clf.predict(x_test)
print '测试集精确度：', metrics.accuracy_score(y_test, y_test_hat)
n_class = len(data['accept'].unique())
y_test_one_hot = label_binarize(y_test, classes=np.arange(n_class))
y_test_one_hot_hat = clf.predict_proba(x_test)
fpr, tpr, _ = metrics.roc_curve(y_test_one_hot.ravel(), y_test_one_hot_hat.ravel())
print 'Micro AUC:\t', metrics.auc(fpr, tpr)
print 'Micro AUC(System):\t', metrics.roc_auc_score(y_test_one_hot, y_test_one_hot_hat, average='micro')
auc = metrics.roc_auc_score(y_test_one_hot, y_test_one_hot_hat, average='macro')
print 'Macro AUC:\t', auc

mpl.rcParams['font.sans-serif'] = u'SimHei'
mpl.rcParams['axes.unicode_minus'] = False
plt.figure(figsize=(8, 7), dpi=80, facecolor='w')
plt.plot(fpr, tpr, 'r-', lw=2, label='AUC=%.4f' % auc)
plt.legend(loc='lower right')
plt.xlim((-0.01, 1.02))
plt.ylim((-0.01, 1.02))
plt.xticks(np.arange(0, 1.1, 0.1))
plt.yticks(np.arange(0, 1.1, 0.1))
plt.xlabel('False Positive Rate', fontsize=14)
plt.ylabel('True Positive Rate', fontsize=14)
plt.grid(b=True, ls=':')
plt.title(u'ROC曲线和AUC', fontsize=18)
```



训练集精确度：0.9206
测试集精确度：0.8651
Micro AUC:　0.9776
Macro AUC:　0.9535

作业

- 除准确率(accuracy)外，还有哪些评价分类模型性能的指标？为什么有这些指标？
- 什么是混淆矩阵？TPR、FPR是什么含义？
  - Precision
  - Recall
  - F1-measure
  - AUC
  - AIC/BIC

我们在这里

- http://wenda.ChinaHadoop.cn
  - 视频/课程/社区
- 微博
  - @ChinaHadoop
  - @邹博_机器学习
- 微信公众号
  - 小象
  - 大数据分析挖掘

感谢大家！

恳请大家批评指正！