夏令营实验室项目报告

中山大学 曾晖

问题分析

此问题为不平衡数据集的二分类问题。获得对应的分类器对数据进行预测。可以利用数据挖掘的方法以及神经网络的方法进行分析。此次使用数据挖掘的方法进行分析和实验。

导入数据

In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from pandas import Series,DataFrame
%matplotlib inline
obj = pd.read_csv('BigML1.csv')
pd.set_option("max_columns", 1000)
pd.set_option("max_row",300)
obj.describe()
```

Out[1]:

	Temperature	Humidity	Measure1	Measure2	Measure3	Measure4	Measure5	Measure6	Measure7	Meas
count	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000	8784.000000	8784.00
mean	64.026412	83.337090	1090.900387	1.489868	0.999203	1071.629895	1075.822860	1076.023793	1086.897086	1077.27
std	2.868833	4.836256	537.097769	1.115605	0.816473	536.518466	533.158826	534.004966	538.195156	537.18
min	5.000000	65.000000	155.000000	0.000000	0.000000	155.000000	155.000000	155.000000	155.000000	155.00
25%	62.000000	80.000000	629.000000	0.000000	0.000000	608.750000	606.000000	623.000000	621.000000	612.00
50%	64.000000	83.000000	1096.000000	1.000000	1.000000	1058.000000	1077.000000	1072.000000	1089.000000	1074.00
75%	66.000000	87.000000	1555.000000	2.000000	2.000000	1533.000000	1541.000000	1537.000000	1558.000000	1541.00
max	78.000000	122.000000	2011.000000	3.000000	2.000000	2011.000000	2011.000000	2011.000000	2011.000000	2011.00

处理数据

获取各个特征的数据类型以及特征值种类

In [2]:

```
test_data_info = pd.DataFrame()
test_data_info['col'] = obj.columns
unique_num = []
feature = []
for i in obj.columns:
    unique_num.append(len(obj[i].unique()))
    feature.append(obj[i].dtype)
test_data_info['unique_num'] = np.array(unique_num)
test_data_info['feature'] = np.array(feature)
test_data_info
```

Out[2]:

	col	unique_num	feature

	col	unique_num	feature
0	Date	3784	object
1	Temperature	23	int64
2	Humidity	35	int64
3	Operator	8	object
4	Measure1	1843	int64
5	Measure2	4	int64
6	Measure3	3	int64
7	Measure4	1837	int64
8	Measure5	1839	int64
9	Measure6	1843	int64
10	Measure7	1842	int64
11	Measure8	1851	int64
12	Measure9	1839	int64
13	Measure10	1837	int64
14	Measure11	1839	int64
15	Measure12	1842	int64
16	Measure13	1841	int64
17	Measure14	1843	int64
18	Measure15	1837	int64
19	Hours Since Previous Failure	666	int64
20	Failure	2	object
21	Date.year	1	int64
22	Date.month	12	int64
23	Date.day-of-month	31	int64
24	Date.day-of-week	7	int64
25	Date.hour	24	int64
26	Date.minute	1	int64
27	Date.second	1	int64

数据清洗

删除无关列，以及其他类型转化为Int类型

In [3]:

```
obj.drop(columns=[' Date.year', ' Date.minute', ' Date.second'],inplace=True)
```

In [4]:

```
obj.head()
```

Out[4]:

	Date	Temperature	Humidity	Operator	Measure1	Measure2	Measure3	Measure4	Measure5	Measure6	Measure7	Measure8
0	2016-01-01 00:00:00	67	82	Operator1	291	1	1	1041	846	334	706	1086
1	2016-01-01 01:00:00	68	77	Operator1	1180	1	1	1915	1194	637	1093	524
2	2016-01-01 02:00:00	64	76	Operator1	1406	1	1	511	1577	1121	1948	1882

	Date	Temperature	Humidity	Operator	Measure1	Measure2	Measure3	Measure4	Measure5	Measure6	Measure7	Measure8
3	2016-01-01 03:00:00	63	80	Operator1	550	1	1	1754	1834	1413	1151	945
4	2016-01-01 04:00:00	65	81	Operator1	1928	1	2	1326	1082	233	1441	1736

Operator1至8对应于数字1至8

In [5]:

```
Operator2id = dict(zip(sorted(list(set(obj.Operator))), range(1, len(sorted(list(set(obj.Operator))
))+1)))
obj.Operator = obj.Operator.apply(lambda x: Operator2id[x])
```

去除Date列，其相关信息已在后面字段中出现

In [6]:

```
obj.drop(columns=['Date'], inplace=True)
```

将Failure的YES/NO分别对应于易于处理的1/0

In [7]:

```
# Failure为NO, 值为0, 否则为1
obj.Failure = obj.Failure.apply(lambda x: 0  if x == 'No' else 1)
```

提取修改之后的特征数据类型以及特征值种类

In [8]:

```
test_data_info = pd.DataFrame()
test_data_info['col'] = obj.columns
unique_num = []
feature = []
for i in obj.columns:
    unique_num.append(len(obj[i].unique()))
    feature.append(obj[i].dtype)
test_data_info['unique_num'] = np.array(unique_num)
test_data_info['feature'] = np.array(feature)
test_data_info
```

Out[8]:

	col	unique_num	feature
0	Temperature	23	int64
1	Humidity	35	int64
2	Operator	8	int64
3	Measure1	1843	int64
4	Measure2	4	int64
5	Measure3	3	int64
6	Measure4	1837	int64
7	Measure5	1839	int64
8	Measure6	1843	int64
9	Measure7	1842	int64
10	Measure8	1851	int64
11	Measure9	1839	int64
12	Measure10	1837	int64

	col	unique_num	feature
13	Measure11	1839	int64
14	Measure12	1842	int64
15	Measure13	1841	int64
16	Measure14	1843	int64
17	Measure15	1837	int64
18	Hours Since Previous Failure	666	int64
19	Failure	2	int64
20	Date.month	12	int64
21	Date.day-of-month	31	int64
22	Date.day-of-week	7	int64
23	Date.hour	24	int64

In [9]:

```
# 将Failure放到最后一列
temp = obj.Failure
obj.drop(columns=['Failure'], inplace=True)
obj['Failure'] = temp
obj.to_csv('./after_deal.csv', index=False)
```

In [10]:

```
import seaborn as sns
```

选择合适的模型和处理方法

显示样本的正负数据占比。正负数据相差较大

In [11]:

```
obj.Failure.value_counts()
```

Out[11]:

```
0    8703
1      81
Name: Failure, dtype: int64
```

绘制各个特征的数据直方图

In [12]:

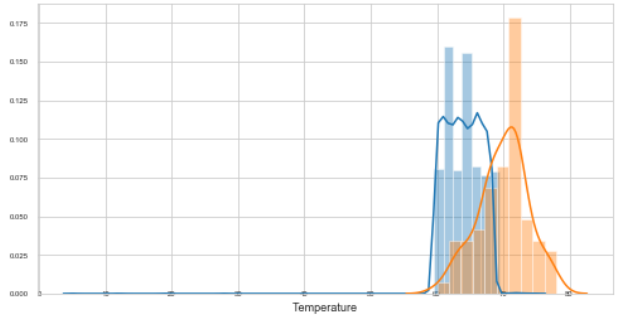
```
def plot_feature_distribution(df1, df2, label1, label2, features):
    i = 0
    sns.set_style('whitegrid')
    plt.figure()
    fig, ax = plt.subplots(12,2,figsize=(18,60))

    for feature in features:
        i += 1
        plt.subplot(12,2,i)
        sns.distplot(df1[feature], label=label1)
        sns.distplot(df2[feature], label=label2)
        plt.xlabel(feature, fontsize=9)
        locs, labels = plt.xticks()
        plt.tick_params(axis='x', which='major', labelsize=6, pad=-6)
        plt.tick_params(axis='y', which='major', labelsize=6)
    plt.show();
```

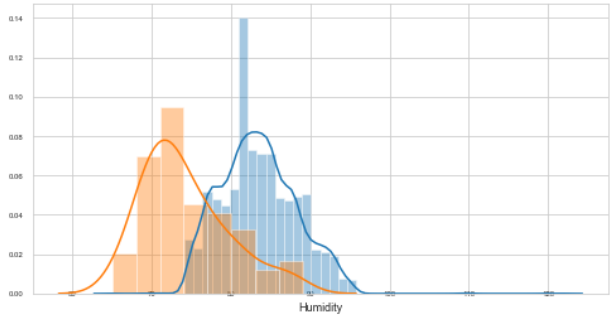
In [ ]:

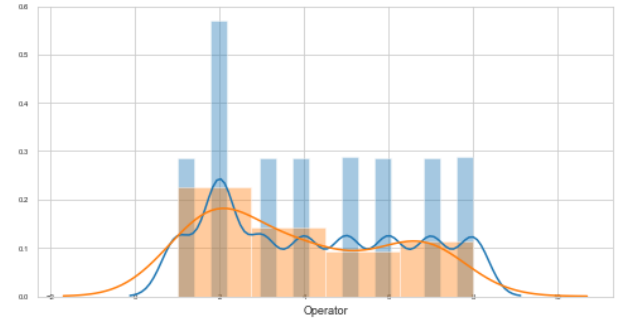
In [13]:

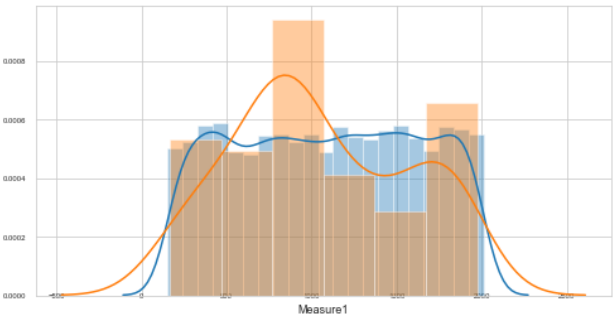
```
t0 = obj.loc[obj['Failure'] == 0]
t1 = obj.loc[obj['Failure'] == 1]
features = obj.columns.values[:-1]
plot_feature_distribution(t0, t1, '0', '1', features)
```

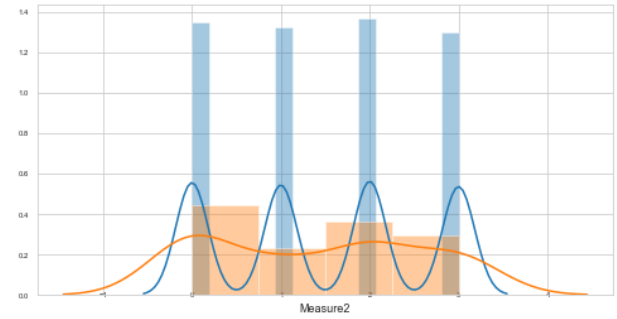
<Figure size 432x288 with 0 Axes>

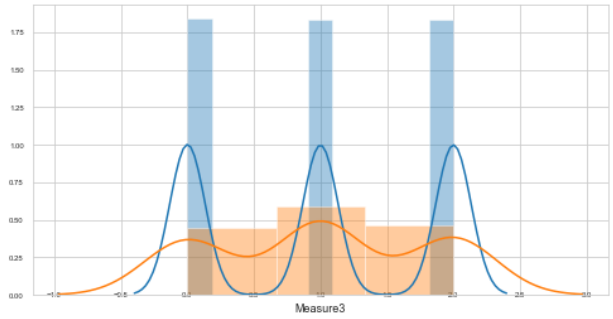


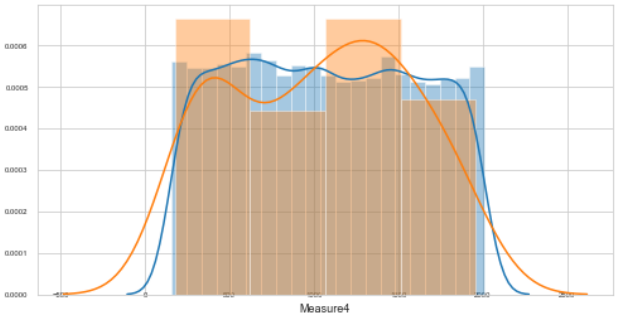


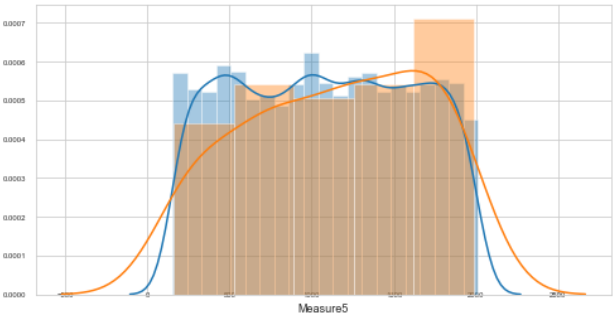


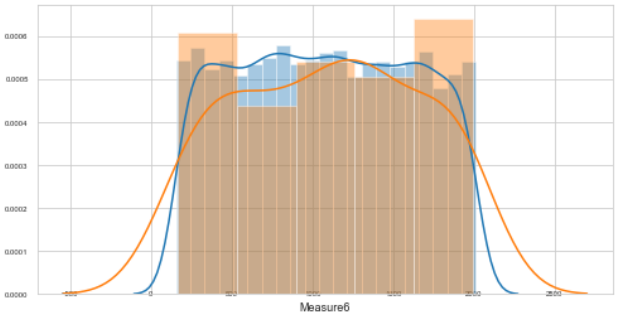


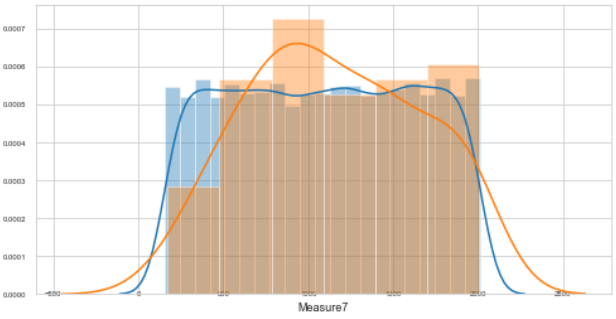


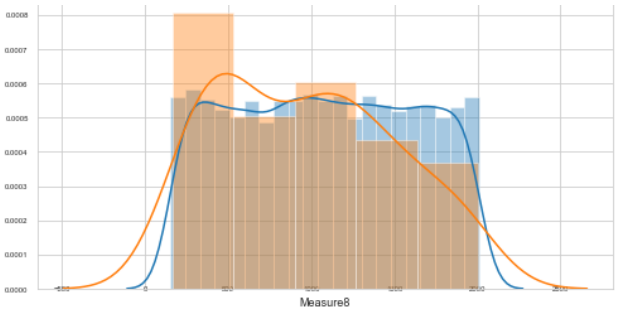


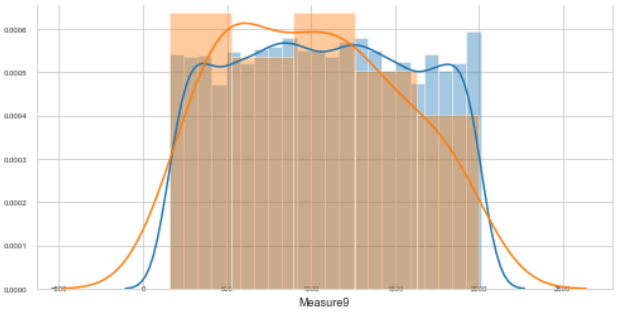


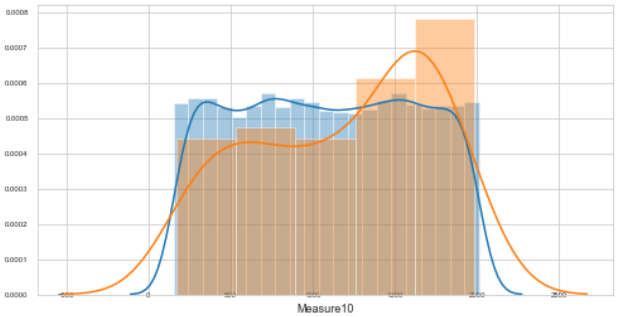


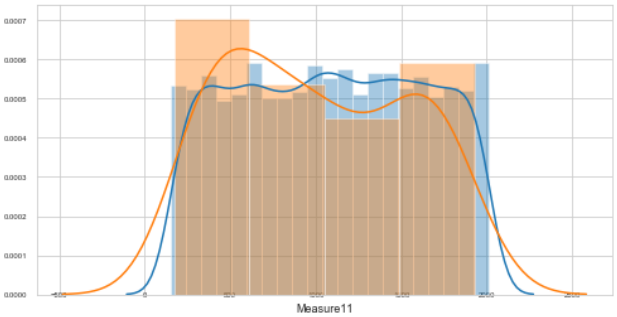


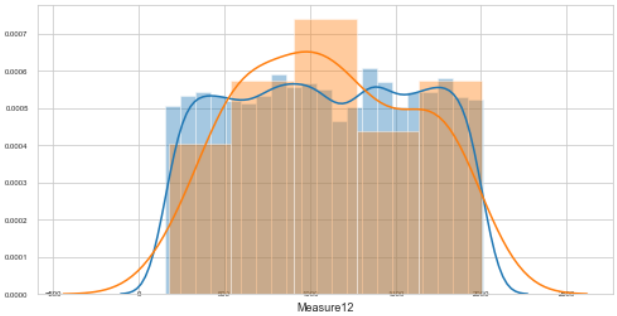


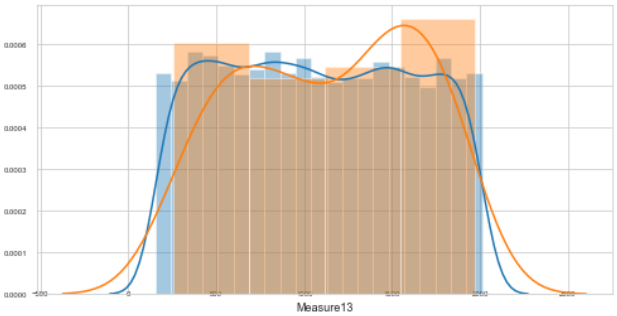

Measure8

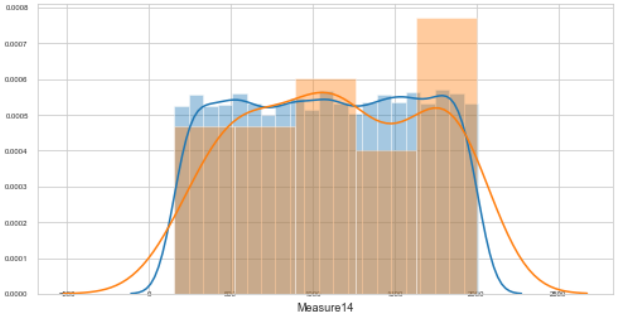

Measure9

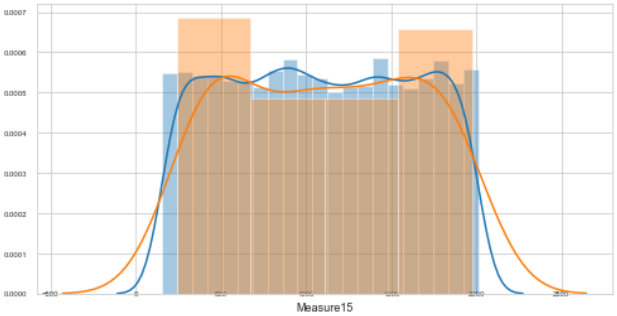

Measure10

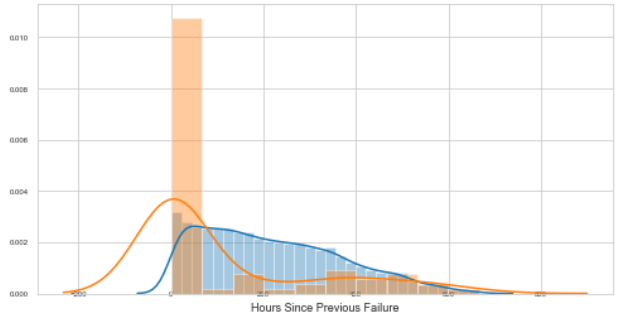

Measure11

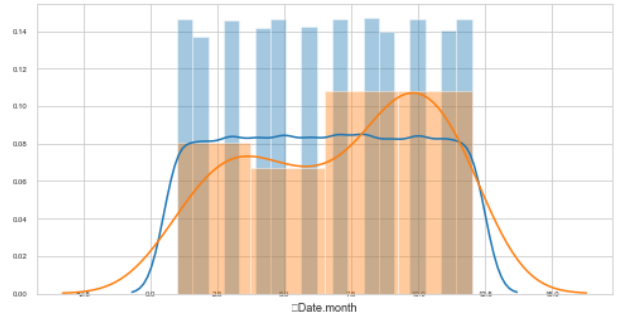

Measure12

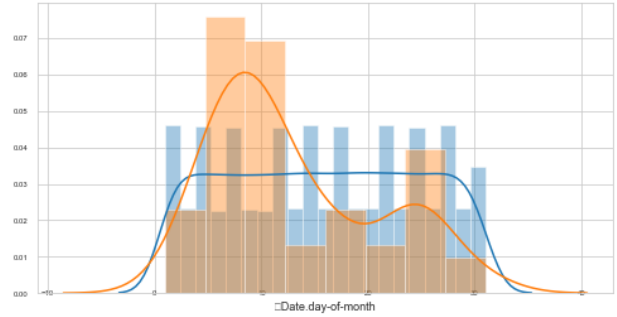

Measure13

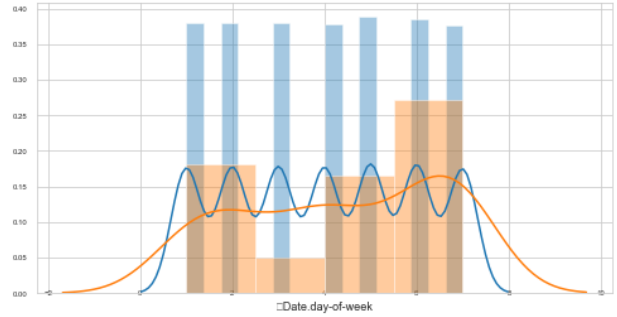

Measure14


Measure15

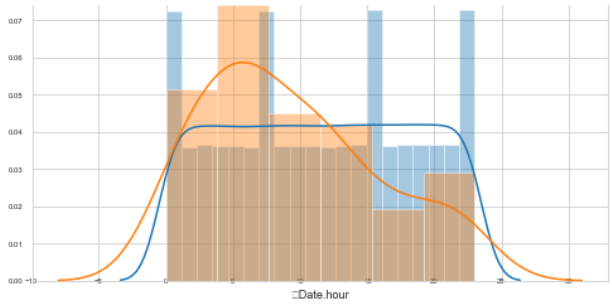

Hours Since Previous Failure

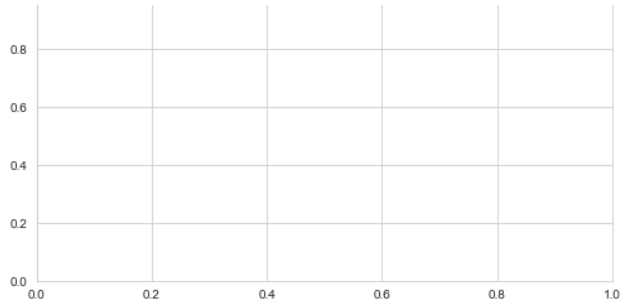

□Date.month


□Date.day-of-month


□Date.day-of-week

1.0





观察各个特征之间的相关性

由绘出的各个特征图可以看出，在各个特征中Temperature、Humidity、Operator与其他特征关联较小，Messure2、Messure3之间可能有着较强的联系，其余特征之间也有着联系。我们可以画出两两特征之间的关系图（图数量过多此处不在画出），观察两两特征之间的相关性，若两个特征之间出现较强的相关性（可以用数学逻辑公式精准表达的），我们可以直接使用两个特征中的一个，综合考量之后得到最后的一些特征进行数据处理。

对数据集进行划分，划分为训练集以及测试集

由于正负样本之间的比例严重失调，相应的负样本在训练的过程中带来的信息会很少，在实现分类的过程中，少数类的判定较为困难，极端出现较大的类包围较小的类，两个类之间的界限难以界定，因此需要解决正负样本之间比例失调的问题。

解决正负样本比例失衡

为什么要解决这个问题：

1. 如果数据存在严重的不平衡，预测得出的结论往往也是有偏的，即分类结果会偏向于较多观测的类。
2. 数据不平衡，在预测时，所有结果都预测为大类，正确率也能很高，分类就没有实际的效果。

思路方法：

- 简单随机过采样

  实现数据集的平衡程度的调整，简单随机过采样是最简单的方法之一。基本思路是将少数类简单复制，使得少数类的数据集大小与较大的类相近。

  优点：简单易实现，在多数类和少数类差别不是很大的时候可以适用

  缺点：在简单随机过采样过程中，只是单纯复制少数类增加数量，并不对其采取任何操作；尤其在少数类与大类的数量相差较为悬殊的时候，会直接导致决策域减小，从而出现过学习/过拟合的现象，即模型学习到的信息过于特别(Specific)而不够泛化(General)。
- SMOTE方法（过采样）

  一种新颖的采样方法，通过一定的策略指导得到人工合成的少数类样本，达到平衡数据的目的。

  基本思路：SMOTE是线性直插的方法，合成主要方式是选取少数类样本和其某一个紧邻的值，计算他们之间的差值，将差值与0-1之间的一个随机数相乘，得到的结果在累积到原来的样本上。通俗的说，把每一个样本看作一个点，将样本点与其最近的临近点连接起来，取连线上的值。

  优点：有效解决过度拟合的现象

  缺点：存在一定的盲目性，新的样本可能会出现样本混叠的情况，SMOTE方法简单假定少数类样本附近围绕的都是少数类样本
- 简单随机欠采样

  与过采样相反，简单随机欠采样主要是随机删除多数类中的部分样本，使得多数类和少数类数量相当。

  优点：简单易实现，同上。

  缺点：随机性和偶然性增加，训练数集大大减少，多数类的一些信息容易在随机删除中丢失。
- 欠采样（Balance Cascade方法）【实现！！！】

  有指导的欠抽样方法。前提假设，少数类的数量虽然少，但是包括了少数类分类中的所有重要信息。不存在少数类分类在样本中信息缺失的情况 基本思路：从多数类中选择数据子集E,使得E的样本数量与少数类的数量相当，将少数类与E共同作为训练样本进行训练得到C1分类器，将训练完的E从原先大数类中去除，再次采样训练，依照此方法得到一系列分类器，最后将所有分类器组合成为联合分类器。

In [14]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import AdaBoostClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
import random
```

数据预处理

In [15]:

```
dataset = obj
obj_index = obj.index.values.tolist()
obj_columns = obj.columns.values.tolist()
```

单独拿出Failure字段

In [16]:

```
#取出Failure
temp = dataset.Failure
dataset.drop(columns=['Failure'], inplace=True)
```

数据标准化

数据的标准化（normalization）是将数据按比例缩放，使之落入一个小的特定区间。在某些比较和评价的指标处理中经常会用到，去除数据的单位限制，将其转化为无量纲的纯数值，便于不同单位或量级的指标能够进行比较和加权。
为什么要进行数据标准化？
1.提升模型的收敛速度

In [17]:

```
%%html
<img src = "./1.png", width=500,height=300>
<img src = "./2.png", width=450,height=300>
```

如上简图，假设y的取值为0-2000，而x的取值为1-5，假如只有这两个特征，对其进行优化时，会得到一个窄长的椭圆形，导致在梯度下降时，梯度的方向为垂直等高线的方向而走之字形路线，这样会使迭代很慢，相比之下，右图的迭代就会很快（理解：也就是步长走多走少方向总是对的，不会走偏）

2.提升模型的精度
归一化的另一好处是提高精度，这在**涉及到一些距离计算的算法时效果显著**，比如算法要计算欧氏距离，上图中x2的取值范围比较小，涉及到距离计算时其对结果的影响远比x1带来的小，所以这就会造成精度的损失。所以归一化很有必要，他可以让各个特征对结果做出的贡献相同。
在多指标评价体系中，由于各评价指标的性质不同，通常具有不同的量纲和数量级。当各指标间的水平相差很大时，如果直接用原始指标值进行分析，就会突出数值较高的指标在综合分析中的作用，相对削弱数值水平较低指标的作用。因此，为了保证结果的可靠性，需要对原始指标数据进行标准化处理。
3. 部分模型强烈依赖归一化/标准化处理数据：SVM， Logistic Regression

In [18]:

```
# 标准化
scaler = StandardScaler().fit(dataset)
dataset=scaler.transform(dataset)
```

In [19]:

```
# 复原dataframe
dataset = DataFrame(dataset, index=obj_index, columns=obj.columns)
dataset['Failure'] = temp
dataset.head()
```

Out[19]:

	Temperature	Humidity	Operator	Measure1	Measure2	Measure3	Measure4	Measure5	Measure6	Measure7	Measure8	Measure9
0	1.036574	-0.276488	-1.401768	-1.489386	-0.43913	0.000976	-0.057093	-0.431083	-1.389624	-0.707771	0.016239	-1.549883
1	1.385168	-1.310404	-1.401768	0.165900	-0.43913	0.000976	1.572021	0.221667	-0.822181	0.011340	-1.030010	-0.305871
2	-0.009207	-1.517188	-1.401768	0.586704	-0.43913	0.000976	-1.045000	0.940068	0.084229	1.600074	1.498114	0.410891
3	-0.357801	-0.690054	-1.401768	-1.007137	-0.43913	0.000976	1.271921	1.422128	0.631072	0.119114	-0.246254	0.431531
4	0.339387	-0.483271	-1.401768	1.558650	-0.43913	1.225826	0.474139	0.011587	-1.578771	0.657983	1.226313	-0.091968

训练数据

In [20]:

```
# 分离数据a
NoFail = dataset.loc[dataset['Failure'] == 0]
YesFail = dataset.loc[dataset['Failure'] == 1]
X, X_test, Y, Y_test = train_test_split(dataset[dataset.columns[:-1]],dataset.Failure,test_size=0.3
)
X['Failure'] = Y
X.head()
```

Out[20]:

	Temperature	Humidity	Operator	Measure1	Measure2	Measure3	Measure4	Measure5	Measure6	Measure7	Measure8	Measur
5667	-1.403581	-0.069705	1.208421	1.484171	0.457295	0.000976	0.147944	-1.020060	-1.226695	-1.718614	1.284024	0.1538
7138	-1.403581	-0.276488	-0.531705	1.584717	1.353721	1.225826	-0.633062	-0.434835	0.524325	-0.611146	1.090412	-1.5723
3949	-0.009207	-0.483271	-0.966736	0.601600	-1.335556	1.225826	1.517965	1.018848	0.329559	-1.198327	0.152139	0.0844
1881	-0.706394	0.964212	-0.531705	0.311134	-1.335556	1.225826	-1.041272	-0.975042	1.161059	-0.051837	-0.471514	0.0337
1356	0.687980	-1.310404	-0.966736	1.458104	0.457295	-1.223874	-0.396337	1.525293	1.655465	-1.248498	-0.357953	1.1708

利用逻辑回归构造分类器组

In [21]:

```
def BalanceCascade(big_dataset, small_dataset, num):
    classifiers={}
    big_dataset.drop(columns=['Failure'], inplace=True)
    small_dataset.drop(columns=['Failure'], inplace=True)
    Big_class = big_dataset.values.tolist()
    Small_class = small_dataset.values.tolist()
    # 构建训练集，同时训练数据，num训练次数
    for i in range(num):
        classifiers[i]=LogisticRegression(solver='lbfgs',max_iter=200)
        # 在大类中随机采样
        if(len(Big_class)>=len(Small_class)):
```

```
            E = random.sample(Big_class, len(Small_class))
            H = E + Small_class
            H_test = [0] * len(Small_class) + [1] *  len(Small_class)
            classifiers[i].fit(H, H_test)
            Big_class = [item for item in Big_class if item not in E]
            print("classfier No.{} finished!".format(i) )
        else:
            Big_class = big_dataset.values.tolist()
            i = i - 1
    return classifiers
```

In [22]:

```
# 生成指定数量的分类器
X_NoFail = X.loc[X['Failure'] == 0]
X_YesFail = X.loc[X['Failure'] == 1]
classifiers = BalanceCascade(NoFail, YesFail, 100)
```

```
e:\python3.7\lib\site-packages\pandas\core\frame.py:3940: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  errors=errors)
```

```
classfier No.0 finished!
classfier No.1 finished!
classfier No.2 finished!
classfier No.3 finished!
classfier No.4 finished!
classfier No.5 finished!
classfier No.6 finished!
classfier No.7 finished!
classfier No.8 finished!
classfier No.9 finished!
classfier No.10 finished!
classfier No.11 finished!
classfier No.12 finished!
classfier No.13 finished!
classfier No.14 finished!
classfier No.15 finished!
classfier No.16 finished!
classfier No.17 finished!
classfier No.18 finished!
classfier No.19 finished!
classfier No.20 finished!
classfier No.21 finished!
classfier No.22 finished!
classfier No.23 finished!
classfier No.24 finished!
classfier No.25 finished!
classfier No.26 finished!
classfier No.27 finished!
classfier No.28 finished!
classfier No.29 finished!
classfier No.30 finished!
classfier No.31 finished!
classfier No.32 finished!
classfier No.33 finished!
classfier No.34 finished!
classfier No.35 finished!
classfier No.36 finished!
classfier No.37 finished!
classfier No.38 finished!
classfier No.39 finished!
classfier No.40 finished!
classfier No.41 finished!
classfier No.42 finished!
classfier No.43 finished!
classfier No.44 finished!
classfier No.45 finished!
classfier No.46 finished!
classfier No.47 finished!
classfier No.48 finished!
classfier No.49 finished!
```

```
classfier No.50 finished!
classfier No.51 finished!
classfier No.52 finished!
classfier No.53 finished!
classfier No.54 finished!
classfier No.55 finished!
classfier No.56 finished!
classfier No.57 finished!
classfier No.58 finished!
classfier No.59 finished!
classfier No.60 finished!
classfier No.61 finished!
classfier No.62 finished!
classfier No.63 finished!
classfier No.64 finished!
classfier No.65 finished!
classfier No.66 finished!
classfier No.67 finished!
classfier No.68 finished!
classfier No.69 finished!
classfier No.70 finished!
classfier No.71 finished!
classfier No.72 finished!
classfier No.73 finished!
classfier No.74 finished!
classfier No.75 finished!
classfier No.76 finished!
classfier No.77 finished!
classfier No.78 finished!
classfier No.79 finished!
classfier No.80 finished!
classfier No.81 finished!
classfier No.82 finished!
classfier No.83 finished!
classfier No.84 finished!
classfier No.85 finished!
classfier No.86 finished!
classfier No.87 finished!
classfier No.88 finished!
classfier No.89 finished!
classfier No.90 finished!
classfier No.91 finished!
classfier No.92 finished!
classfier No.93 finished!
classfier No.94 finished!
classfier No.95 finished!
classfier No.96 finished!
classfier No.97 finished!
classfier No.98 finished!
classfier No.99 finished!
```

In [23]:

```
classifiers
```

Out[23]:

```
{0: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                       intercept_scaling=1, l1_ratio=None, max_iter=200,
                       multi_class='warn', n_jobs=None, penalty='l2',
                       random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                       warm_start=False),
 1: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                       intercept_scaling=1, l1_ratio=None, max_iter=200,
                       multi_class='warn', n_jobs=None, penalty='l2',
                       random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                       warm_start=False),
 2: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                       intercept_scaling=1, l1_ratio=None, max_iter=200,
                       multi_class='warn', n_jobs=None, penalty='l2',
                       random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                       warm_start=False),
 3: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                       intercept_scaling=1, l1_ratio=None, max_iter=200,
                       multi_class='warn', n_jobs=None, penalty='l2',
```

```
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
 4: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
 5: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
 6: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
 7: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
 8: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
 9: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
10: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
11: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
12: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
13: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
14: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
15: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
16: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
17: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
18: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
```

```
19: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
20: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
21: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
22: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
23: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
24: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
25: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
26: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
27: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
28: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
29: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
30: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
31: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
32: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
33: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
34: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
```

```
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
35: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
36: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
37: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
38: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
39: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
40: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
41: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
42: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
43: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
44: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
45: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
46: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
47: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
48: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
49: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
```

```
                   warm_start=False),
50: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
51: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
52: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
53: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
54: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
55: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
56: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
57: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
58: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
59: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
60: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
61: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
62: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
63: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
64: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
65: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
```

```
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
66: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
67: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
68: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
69: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
70: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
71: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
72: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
73: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
74: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
75: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
76: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
77: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
78: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
79: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False),
80: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='warn', n_jobs=None, penalty='l2',
```

```
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 81: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 82: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 83: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 84: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 85: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 86: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 87: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 88: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 89: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 90: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 91: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 92: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 93: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 94: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
 95: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=200,
                  multi_class='warn', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                  warm_start=False),
```

```
 96: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=200,
                    multi_class='warn', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False),
 97: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=200,
                    multi_class='warn', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False),
 98: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=200,
                    multi_class='warn', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False),
 99: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=200,
                    multi_class='warn', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False)}
```

利用组分类器进行对每一个测试样本进行预测，并且投票，票数多者选择的标签为预测标签，与测试样本的结果比对，计算准确率

In [24]:

```
def use_classifier_to_predict(classifiers, X_test, Y_test):
    X_test = X_test.values.tolist()
    Y_test = Y_test.values.tolist()
    votes_list = []
    correct_num = 0
    classifier_result=[]
    for i in range(len(X_test)):
        votes = 0
        for c in range(len(classifiers)):
            t = classifiers[c].predict([X_test[i]])
            if(t == Y_test[i]):
                votes = votes + 1
        if(votes>len(classifiers)/2):
            correct_num=correct_num+1
            classifier_result.append(Y_test[i])
        else:
            if(Y_test[i] == 0):
                classifier_result.append(1)
            else:
                classifier_result.append(0)
        votes_list.append(votes)
    Accuracy = correct_num/len(X_test)
    return votes_list,Accuracy,classifier_result
```

In [25]:

```
X_test.head()
```

Out[25]:

	Temperature	Humidity	Operator	Measure1	Measure2	Measure3	Measure4	Measure5	Measure6	Measure7	Measure8	Measur
5475	-0.706394	2.204911	-1.401768	0.411680	1.353721	-1.223874	-1.113967	-1.650302	0.129175	1.191277	0.358783	-0.3227
6867	1.036574	-0.276488	-0.966736	0.823174	-1.335556	1.225826	-0.457848	-1.335181	-1.073129	1.064921	0.206127	0.6810
4393	0.339387	1.377778	-1.401768	0.785934	0.457295	0.000976	-0.340418	-1.061326	-1.445806	-0.679898	0.427664	0.7129
6605	0.339387	1.791345	1.208421	-0.195321	1.353721	1.225826	-1.371196	0.034095	0.059883	-1.674018	1.049456	0.0112
604	-1.054988	-0.690054	-0.966736	0.910686	-1.335556	0.000976	1.301744	-0.671176	-0.842781	-1.174171	0.941480	0.7430

In [26]:

```
x_test = X_test.values.tolist()
```

In [27]:

```
countYT =  pd.value_counts(Y_test)
countYT
```

Out[27]:

```
0    2613
1      23
Name: Failure, dtype: int64
```

In [28]:

```
count = pd.value_counts(classifiers[0].predict(x_test))
count
```

Out[28]:

```
0    2148
1     488
dtype: int64
```

In [29]:

```
votes_list,Accuracy,classifier_result = use_classifier_to_predict(classifiers, X_test, Y_test)
Y_test_index = Y_test.index.values.tolist()
Y_test_values = Y_test.values.tolist()
result_data = pd.DataFrame({"序号":Series(Y_test_index),"Failure/0-No|1-Yes":Series(Y_test_values),
"votes/100":Series(votes_list),"classifier_result":Series(classifier_result)})
result_data[result_data["Failure/0-No|1-Yes"]==1]
```

Out[29]:

	序号	Failure/0-No|1-Yes	votes/100	classifier_result
139	7415	1	2	0
299	2817	1	8	0
561	517	1	100	1
759	3774	1	100	1
768	7510	1	100	1
778	6748	1	100	1
837	1113	1	0	0
991	6450	1	100	1
1063	1832	1	97	1
1140	2428	1	100	1
1189	7512	1	100	1
1465	6446	1	100	1
1498	3773	1	100	1
1515	6890	1	79	1
1526	6445	1	100	1
1772	1632	1	99	1
2074	1829	1	100	1
2114	148	1	7	0
2133	1828	1	100	1
2186	8182	1	100	1
2213	1092	1	94	1
2248	3775	1	100	1
2576	5934	1	100	1

序号 Failure/0-No1-Yes votes/100 classifier result

In [30]:

```
Accuracy
```

Out[30]:

0.872154779969651

利用SVM构造分类器组

In [31]:

```
from sklearn.linear_model import LogisticRegression
def BalanceCascadeSVM(big_dataset, small_dataset, num):
    classifiers={}
    Big_class = big_dataset.values.tolist()
    Small_class = small_dataset.values.tolist()
    # 构建训练集，同时训练数据，num训练次数
    for i in range(num):
        classifiers[i]=SVC(gamma='auto')
        # 在大类中随机采样
        if(len(Big_class)>=len(Small_class)):
            E = random.sample(Big_class, len(Small_class))
            H = E + Small_class
            H_test = [0] * len(Small_class) + [1] *  len(Small_class)
            classifiers[i].fit(H, H_test)
            Big_class = [item for item in Big_class if item not in E]
            print("classfier No.{} finished!".format(i) )
        else:
            Big_class = big_dataset.values.tolist()
            i = i - 1
    return classifiers
```

In [32]:

```
classifiers_svc = BalanceCascadeSVM(NoFail, YesFail, 100)
```

```
classfier No.0 finished!
classfier No.1 finished!
classfier No.2 finished!
classfier No.3 finished!
classfier No.4 finished!
classfier No.5 finished!
classfier No.6 finished!
classfier No.7 finished!
classfier No.8 finished!
classfier No.9 finished!
classfier No.10 finished!
classfier No.11 finished!
classfier No.12 finished!
classfier No.13 finished!
classfier No.14 finished!
classfier No.15 finished!
classfier No.16 finished!
classfier No.17 finished!
classfier No.18 finished!
classfier No.19 finished!
classfier No.20 finished!
classfier No.21 finished!
classfier No.22 finished!
classfier No.23 finished!
classfier No.24 finished!
classfier No.25 finished!
classfier No.26 finished!
classfier No.27 finished!
classfier No.28 finished!
classfier No.29 finished!
classfier No.30 finished!
classfier No.31 finished!
classfier No.32 finished!
```

```
classfier No.33 finished!
classfier No.34 finished!
classfier No.35 finished!
classfier No.36 finished!
classfier No.37 finished!
classfier No.38 finished!
classfier No.39 finished!
classfier No.40 finished!
classfier No.41 finished!
classfier No.42 finished!
classfier No.43 finished!
classfier No.44 finished!
classfier No.45 finished!
classfier No.46 finished!
classfier No.47 finished!
classfier No.48 finished!
classfier No.49 finished!
classfier No.50 finished!
classfier No.51 finished!
classfier No.52 finished!
classfier No.53 finished!
classfier No.54 finished!
classfier No.55 finished!
classfier No.56 finished!
classfier No.57 finished!
classfier No.58 finished!
classfier No.59 finished!
classfier No.60 finished!
classfier No.61 finished!
classfier No.62 finished!
classfier No.63 finished!
classfier No.64 finished!
classfier No.65 finished!
classfier No.66 finished!
classfier No.67 finished!
classfier No.68 finished!
classfier No.69 finished!
classfier No.70 finished!
classfier No.71 finished!
classfier No.72 finished!
classfier No.73 finished!
classfier No.74 finished!
classfier No.75 finished!
classfier No.76 finished!
classfier No.77 finished!
classfier No.78 finished!
classfier No.79 finished!
classfier No.80 finished!
classfier No.81 finished!
classfier No.82 finished!
classfier No.83 finished!
classfier No.84 finished!
classfier No.85 finished!
classfier No.86 finished!
classfier No.87 finished!
classfier No.88 finished!
classfier No.89 finished!
classfier No.90 finished!
classfier No.91 finished!
classfier No.92 finished!
classfier No.93 finished!
classfier No.94 finished!
classfier No.95 finished!
classfier No.96 finished!
classfier No.97 finished!
classfier No.98 finished!
classfier No.99 finished!
```

In [33]:

```
votes_list_svc,Accuracy_svc,classifier_result_svc = use_classifier_to_predict(classifiers_svc,
X_test, Y_test)
Y_test_index = Y_test.index.values.tolist()
Y_test_values = Y_test.values.tolist()
result_data_svc = pd.DataFrame({"序号":Series(Y_test_index),"Failure/0-No|1-
Yes":Series(Y_test_values),"votes/100":Series(votes_list),"classifier_result":Series(classifier_res
```

```
ult_svc)})
result_data_svc[result_data_svc["Failure/0-No|1-Yes"]==1]
```

Out[33]:

	序号	Failure/0-No|1-Yes	votes/100	classifier_result
139	7415	1	2	1
299	2817	1	8	0
561	517	1	100	1
759	3774	1	100	1
768	7510	1	100	1
778	6748	1	100	1
837	1113	1	0	0
991	6450	1	100	1
1063	1832	1	97	1
1140	2428	1	100	1
1189	7512	1	100	1
1465	6446	1	100	1
1498	3773	1	100	1
1515	6890	1	79	1
1526	6445	1	100	1
1772	1632	1	99	1
2074	1829	1	100	1
2114	148	1	7	0
2133	1828	1	100	1
2186	8182	1	100	1
2213	1092	1	94	1
2248	3775	1	100	1
2576	5934	1	100	1

In [34]:

```
Accuracy_svc
```

Out[34]:

```
0.9707890743550834
```

思考与延伸

1. SVM与逻辑回归 上面实现了两种模型下的错误预测，发现SVM结果比逻辑回归高出了10个百分点，Why?

   SVM与逻辑回归的区别

   * 都是分类器
   * SVM不直接依赖数据发布，分类平面不受影响，LR受所有数据点的影响，如果数据不同类别之间有strongly unbalance,如本测试集，要先进行数据平衡化。
   * SVM依赖数据表达的距离测度，依据距离算loss值不断的迭代。所以要先做标准化，但是LR不受影响。

2. Balance Cascade深度理解

   网上有周志华老师的实现方法和思路，使用的分类器是Adaboost分类器，并且移除样本处只移除被分类器正确分类的样本。
3. 组分类器的决策 由于利用Balance Cascade生成了一个组分类器，组分类器的决策对整个模型的预测判决也至关重要。 当前使用的决策机制是，所有分类器一人一票，对同一个样本预测时，投给自己预测出来的标签。最后统计，票数多者为组分类器对此样本的预测值。 两种改进的思路：
   A. 给每一个分类器添加权重，有权重的进行投票。对于承载信息量较大的样本训练出来的分类器，其分类效果和分类的准确率要比普通的高，提高其权重，表示对此分类器的结果更加信赖。同时也可以避免大量冗余数据训练出大量质量不高的分类器，影响分类效果。

B. 借鉴区块链的一些共识机制，如POW，POS，DPOS等。

- 选择最好的分类器，代理群体实现决策。假设测试样本和预测样本足够大，利用小部分测试样本对每一个单独的分类器进行预测，在每一个标签中选择最好的分类器（预测到正确的类的概率均值最大，分类器会给出每一个测试用例各个分类标签的概率。在随机的小部分测试样本中，获取平均预测概率最高的，在POS中，谁的股权高，谁越容易发掘下一个区块，次数，谁在此处小部分的测试样本中平均正确概率大，谁掌控接下来一段时间数据的分类）。在一段预测时间过后，还可以再次采集小部分测试样本，重新选择新的最好的分类器

总结

此次项目为对不平衡数据集的分类问题，实验的基本过程为：数据分析 => 数据进行预处理 => 数据集分离得到训练集与测试集 => 对数据集进行欠采样（Balance Cascade方法以及实现） => 利用处理后的训练集训练分类器（SVM & LR） => 得到组分类器 => 利用组分类器对测试集进行预测以及验证。

收获：本次项目学习到了新的数据采样、数据分类方法，在实验过程中，也更加细致的了解了SVM与LR之间的不同，同时提出了对此项目的一些思考和想法。在实验过程得到了同学的很大帮助。

不足：在数据分析阶段还可以做更多的工作，如提取最相关的某些特征，去除不相干特征的影响，对相关特征进行评级，通过评级获得最好的数据训练集以及测试集。在数据预处理阶段，只进行了数据的标准化，其实还可以通过数据预处理的其他方法，如分桶等，获得更好的训练数集，消除部分冗余的大类样本。欠采样的过程中，单纯的从大量类中随机采样与少数类合并，其随机性较高。组分类决策过程，可以采取更好的决策方法，获得更好的决策结果。

参考资料

[1]闫欣. 综合过采样和欠采样的不平衡数据集的学习研究[D].东北电力大学,2016.

[2]谈森鹏,杨超.区块链DPoS共识机制的研究与改进[J].现代计算机(专业版),2019(06):11-14.

[3]徐义田,王来生,张好治,孙宝山.基于SVM的分类算法与聚类分析[J].烟台大学学报(自然科学与工程版),2004(01):9-13.

In [ ]: