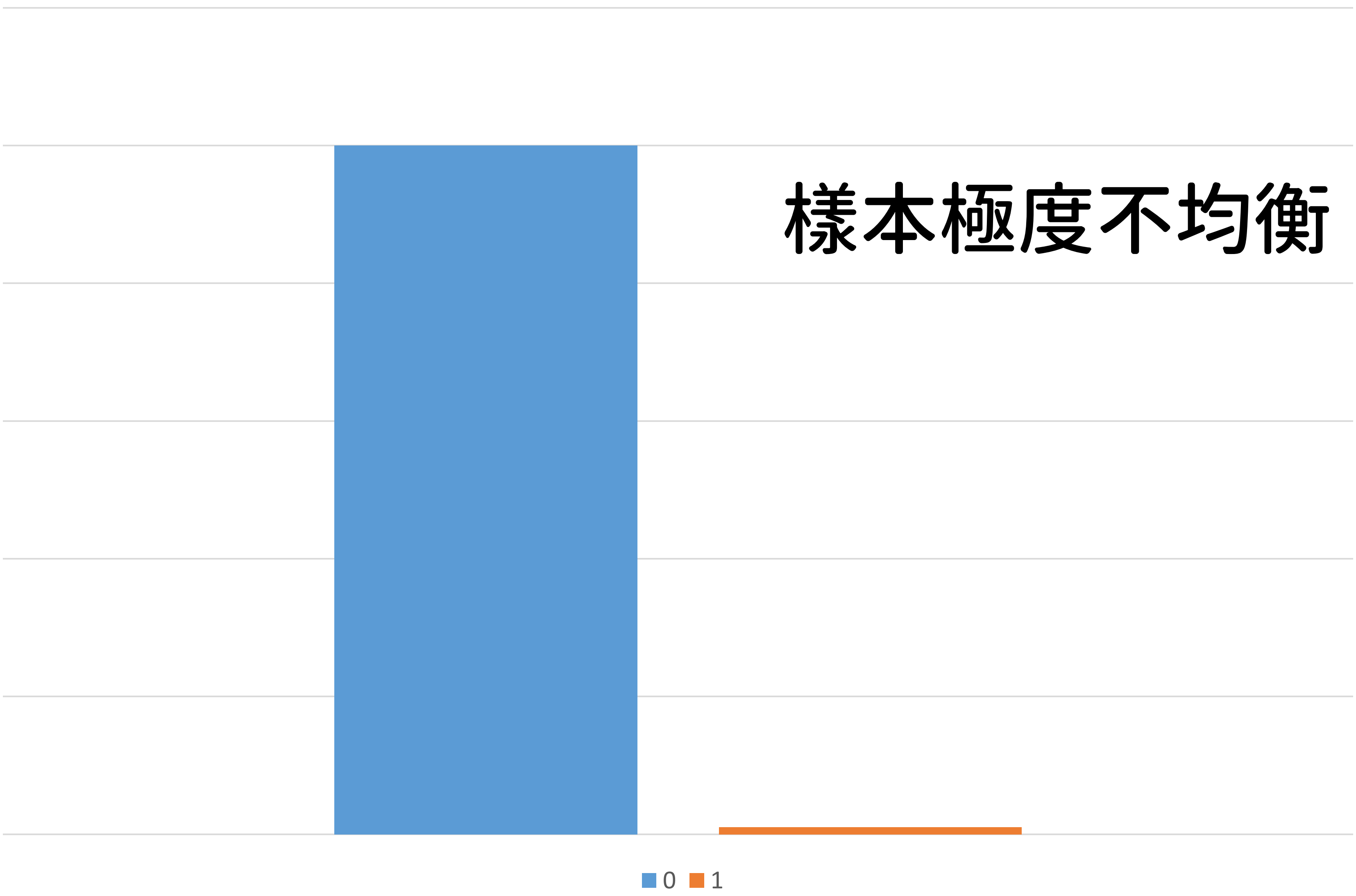
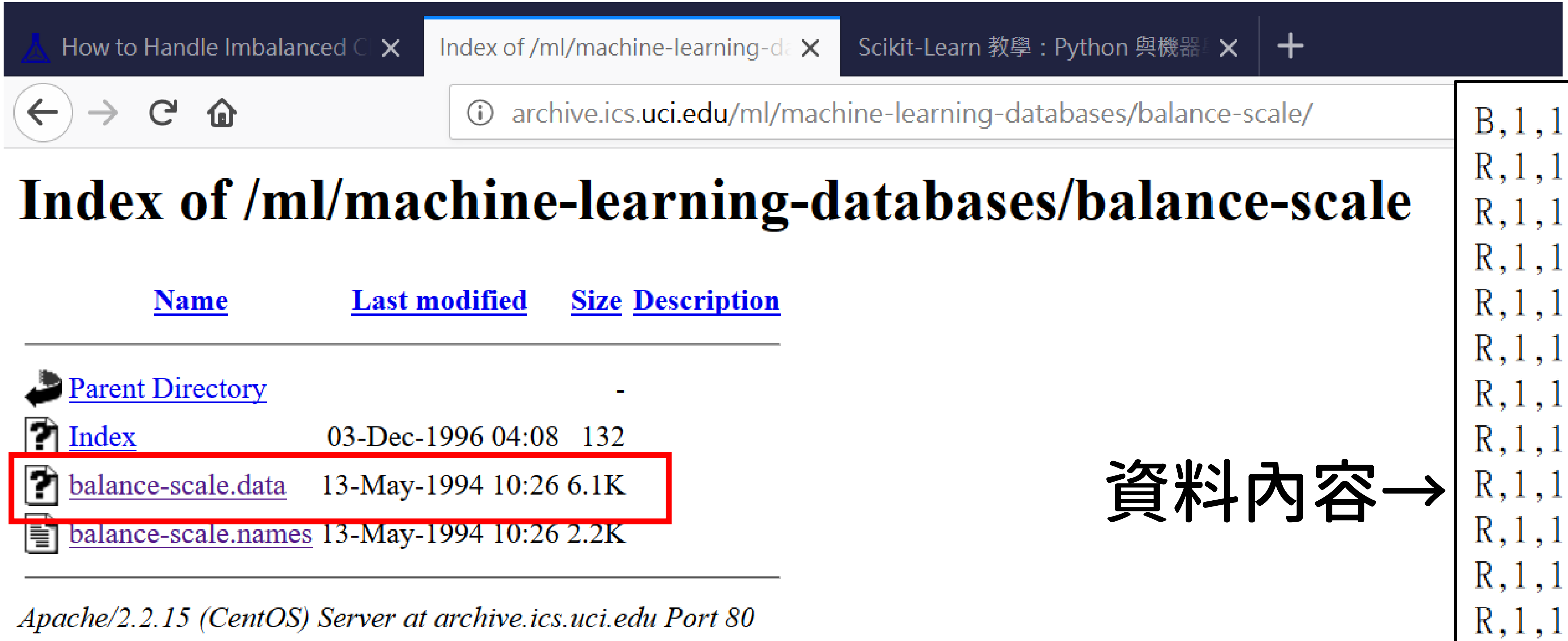
Imbalanced Classes

資料集：balance-scale.data

EX: 信用卡數據樣本中哪些是具有欺詐行為的?



http://archive.ics.uci.edu/ml/machine-learning-databases/balance-scale/



資料內容→

```
B,1,1,1,1
R,1,1,1,2
R,1,1,1,3
R,1,1,1,4
R,1,1,1,5
R,1,1,2,1
R,1,1,2,2
R,1,1,2,3
R,1,1,2,4
R,1,1,2,5
R,1,1,3,1
R,1,1,3,2
R,1,1,3,3
R,1,1,3,4
R,1,1,3,5
R,1,1,4,1
R,1,1,4,2
R,1,1,4,3
```

Imbalanced Classes 2 Last Checkpoint: 10 hours ago (autosaved)

In [22]:
```python
import pandas as pd
import numpy as np

# Read dataset
df = pd.read_csv('C:\\Users\\user\\Desktop\\balance-scale.data',
                 names=['balance', 'var1', 'var2', 'var3', 'var4'])

# Display example observations
df.head()
```

Out[22]:

	balance	var1	var2	var3	var4
0	B	1	1	1	1
1	R	1	1	1	2
2	R	1	1	1	3
3	R	1	1	1	4
4	R	1	1	1	5

In [24]:
```python
df['balance'].value_counts()
```

In [24]:
```python
df['balance'].value_counts()
```
Out[24]:
```
R    288
L    288
B     49
Name: balance, dtype: int64
```
←R 偏右;L 偏左;B平衡

In [25]:
```python
df['balance'] = [1 if b=='B' else 0 for b in df.balance]

df['balance'].value_counts()
```
Out[25]:
```
0    576
1     49
Name: balance, dtype: int64
```
← 0 不平衡 ; 1平衡

```
In [26]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score
```

```
In [27]: # 把balance欄位(y)與var1-4欄位區分開(X)
         y = df.balance
         X = df.drop('balance', axis=1)

         # Train model
         clf_0 = LogisticRegression().fit(X, y)

         # Predict on training set
         pred_y_0 = clf_0.predict(X)
```

X

	var1	var2	var3	var4
0	1	1	1	1
1	1	1	1	2
2	1	1	1	3
3	1	1	1	4
4	1	1	1	5
5	1	1	2	1
6	1	1	2	2

y

0	1
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0

Clf_0：用邏輯思回歸模型分別帶入X,y運算
Pred_y_0：用預測函數predic，預測X每一欄 的結果是0或1

羅吉斯迴歸與線性迴歸的差別在於：前者用於**類別型**資料、後者用於連續型資料

In [28]:
```
print( accuracy_score(pred_y_0, y) )
```

0.9216 ← 將剛剛預測完的值對比y表，發現準確率高達92.16%

In [29]:
```
print( np.unique( pred_y_0 ) )
```

[0] ← 但實際看pred_y_0裡面的數值其實全部都是0，又0在所有資料中占“多數”。

法一：up-sample上採樣

```
In [10]: # up-sample(隨機複製"少數"的樣本，以加強訊號)
         from sklearn.utils import resample
```

In [11]:
```
# Separate majority and minority classes
df_majority = df[df.balance==0]
df_minority = df[df.balance==1]

# Upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples=576,    # to match majority class
                                 random_state=123) # random_state 用來確保每次切分資料的結果都相同

# Combine majority class with upsampled minority class
df_upsampled = pd.concat([df_majority, df_minority_upsampled])

# Display new class counts
df_upsampled.balance.value_counts()
```

Out[11]:
```
1    576
0    576
Name: balance, dtype: int64
```

In [12]:

```
# Separate input features (X) and target variable (y)
y = df_upsampled.balance
X = df_upsampled.drop('balance', axis=1)

# Train model
clf_1 = LogisticRegression().fit(X, y)

# Predict on training set
pred_y_1 = clf_1.predict(X)

# Is our model still predicting just one class?
print( np.unique( pred_y_1 ) )

# How's our accuracy?
print( accuracy_score(y, pred_y_1) )
```

```
[0 1]
0.513888888889
```

與前步驟相同，pred_y_1不再只是全部猜0
準確度 51.4%

法二：down-sample下採樣

In [30]:
```
# down-sample(隨機移除"多數"的樣本)
# Separate majority and minority classes
df_majority = df[df.balance==0]
df_minority = df[df.balance==1]

# Downsample majority class
df_majority_downsampled = resample(df_majority,
                                   replace=False,    # sample without replacement
                                   n_samples=49,     # to match minority class
                                   random_state=123) # reproducible results

# Combine minority class with downsampled majority class
df_downsampled = pd.concat([df_majority_downsampled, df_minority])

# Display new class counts
df_downsampled.balance.value_counts()
```

Out[30]:
```
1    49
0    49
Name: balance, dtype: int64
```

想法與上採樣雷同，下採樣移除“多數”資料，使其與少數資料比為1:1

In [31]:
```python
# Separate input features (X) and target variable (y)
y = df_downsampled.balance
X = df_downsampled.drop('balance', axis=1)

# Train model
clf_2 = LogisticRegression().fit(X, y)

# Predict on training set
pred_y_2 = clf_2.predict(X)

# Is our model still predicting just one class?
print( np.unique( pred_y_2 ) )


# How's our accuracy?
print( accuracy_score(y, pred_y_2) )
# 0.581632653061
```

```
[0 1]
0.581632653061
```

結果準確度58.16%，略高於上採樣

法三：ROC Curve

```
In [84]: from sklearn.metrics import roc_auc_score
```

Note : ROC曲線需要預測類別的機率值，不能只單純用預測類別(0,1)。ROC應該> = 0.5，若否，需要反轉預測，因為Scikit-Learn誤解了正面的類別

In [105]:
```python
# Predict class probabilities
# ROC要有預測"機率"所以這邊用predict_proba()
prob_y_2 = clf_2.predict_proba(X)

# Keep only the positive class
prob_y_2 = [p[1] for p in prob_y_2]

prob_y_2[:5]
```

Out[105]:
```
[0.45419197226479618,
 0.48205962213283882,
 0.46862327066392517,
 0.47868378832689151,
 0.58143856820159723]
```

In [106]:
```python
print( roc_auc_score(y, prob_y_2) )
```

```
0.568096626406
```

法四：Penalize Algorithms (Cost-Sensitive Training)

In [98]:
```
# Separate input features (X) and target variable (y)
y = df.balance
X = df.drop('balance', axis=1)

# Train model
clf_3 = SVC(kernel='linear',
            class_weight='balanced', # penalize
            probability=True)

clf_3.fit(X, y)

# Predict on training set
pred_y_3 = clf_3.predict(X)
```

```python
# Is our model still predicting just one class?
print( np.unique( pred_y_3 ) )


# How's our accuracy?
print( accuracy_score(y, pred_y_3) )


# What about AUROC?
prob_y_3 = clf_3.predict_proba(X)
prob_y_3 = [p[1] for p in prob_y_3]
print( roc_auc_score(y, prob_y_3) )
```

```
[0 1]
0.688
0.4694763322
```

法五：RandomForest

```
In [111]: from sklearn.ensemble import RandomForestClassifier
```

```python
# Separate input features (X) and target variable (y)
y = df.balance
X = df.drop('balance', axis=1)

# Train model
clf_4 = RandomForestClassifier()
clf_4.fit(X, y)

# Predict on training set
pred_y_4 = clf_4.predict(X)

# Is our model still predicting just one class?
print( np.unique( pred_y_4 ) )


# How's our accuracy?
print( accuracy_score(y, pred_y_4) )
```

```python
# What about AUROC?
prob_y_4 = clf_4.predict_proba(X)
prob_y_4 = [p[1] for p in prob_y_4]
print( roc_auc_score(y, prob_y_4) )
```

```
[0 1]
0.9712
0.998635912698
```

Oversample
過採樣

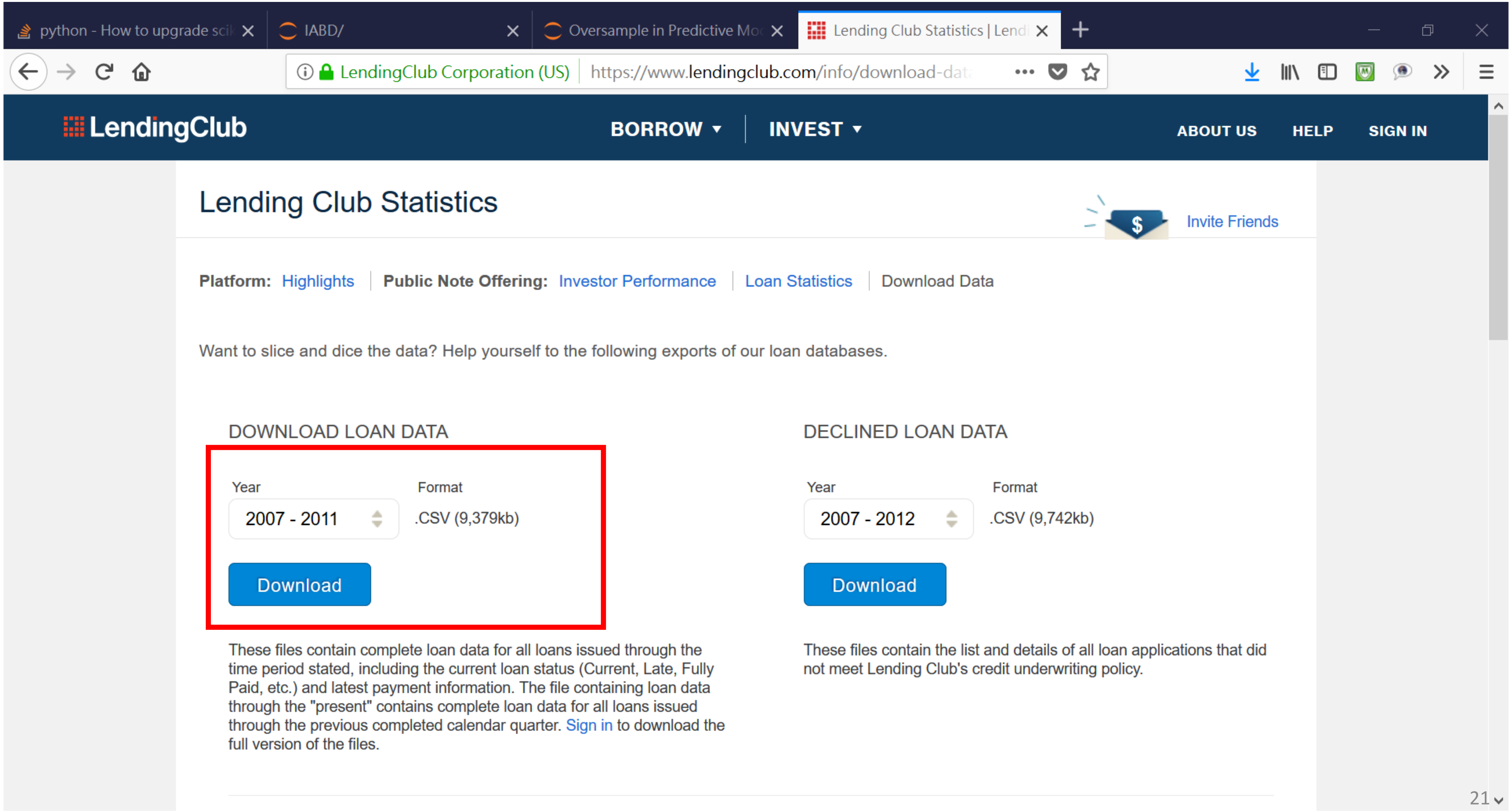
資料集：lending-club-data.csv

過採樣數據生成策略

策略是要讓class為0和1的樣本一樣多，也就是我們需要去進行數據的生成。

利用：SMOTE演算法生成大量異常數據

https://www.lendingclub.com/info/download-data.action



```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from imblearn.over_sampling import SMOTE
```

←所需套件，
先更新↓

```
ImportError: A sklearn version of at least 0.19.0 is requir
ed to use imbalanced-learn. 0.18.1 was found. Please upgrade
sklearn
```

conda install scikit-learn=0.19.0

```
(C:\Users\user\Anaconda3) C:\Users\user>conda install scikit-learn=0.19.0
Fetching package metadata .............
Solving package specifications: .

Package plan for installation in environment C:\Users\user\Anaconda3:

The following packages will be DOWNGRADED:

    scikit-learn: 0.19.1-py36hb0057d7_0 --> 0.19.0-py36he1ad879_2

Proceed ([y]/n)? y

scikit-learn-0 100% |###################################| Time: 0:00:02   1.86 MB/s
```

Scikit-Learn-機器學習庫
非常實用的機器學習演算法庫，包含了基本你覺得你能用上所有機器學習演算法

python - How to upgrade sci | IABD/ | Oversample in Predictive Mo

localhost:8888/notebooks/IABD/Oversample in Predictive Modeling.ipynb

jupyter Oversample in Predictive Modeling Last Checkpoint: 22 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Code

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from imblearn.over_sampling import SMOTE
```

In [2]
```
import sklearn
print(sklearn.__version__)
```
```
0.19.0
```

In [ ]:

IABD/ | Oversample 1 | The Right Way to Oversample in

localhost:8888/notebooks/IABD/Oversample 1.ipynb 120%

jupyter **Oversample 1** Last Checkpoint: 12/13/2017 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Code

In [35]:
```
loans = pd.read_csv("C:/Users/user/Desktop/lending-club-data.csv")
# Pandas 透過使用中括號 [] 與 .iloc
# 可以很靈活地從 data frame 中選擇想要的元素
loans.iloc[0]
```

```
C:\Users\user\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2717: DtypeWarning: Columns (19,47) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[35]:
```
id                    1077501
member_id             1296599
loan_amnt                5000
funded_amnt              5000
funded_amnt_inv          4975
term                36 months
int_rate                10.65
installment            162.87
grade                       B
sub_grade                  B2
emp_title                 NaN
emp_length          10+ years
home_ownership           RENT
```

←第一筆資料內容→

```
In [36]: loans.bad_loans .value_counts()
Out[36]: 0     99457
         1     23150
         Name: bad_loans, dtype: int64
```

不良貸款(bad_loans)：

0 = 好

1 = 不好

```
In [31]: # 刪除payment_inc_ratio欄位中NA的資料
         # ~ --> 排除loans.payment_inc_ratio.isnull()的資料
         loans = loans[~loans.payment_inc_ratio.isnull()]
```

刪除payment_inc_ratio欄位有缺失值的資料

In [6]:
```python
model_variables = ['grade', 'home_ownership','emp_length_num', 'sub_grade','short_emp',
            'dti', 'term', 'purpose', 'int_rate', 'last_delinq_none', 'last_major_derog_none',
            'revol_util', 'total_rec_late_fee', 'payment_inc_ratio', 'bad_loans']

loans_data_relevent = loans[model_variables]
```

建立模型

In [14]:
```python
loans_data_relevent
```

Out[14]:

	grade	home_ownership	emp_length_num	sub_grade	short_emp	dti	term	purpose	int_rate	last_delin
0	B	RENT	11	B2	0	27.65	36 months	credit_card	10.65	
1	C	RENT	1	C4	1	1.00	60 months	car	15.27	
2	C	RENT	11	C5	0	8.72	36 months	small_business	15.96	
3	C	RENT	11	C1	0	20.00	36 months	other	13.49	

用one-hot分類每個特徵

```
In [7]: loans_relevant_enconded = pd.get_dummies(loans_data_relevent)
```

```
In [13]: loans_relevant_enconded
```

use	purpose_major_purchase	purpose_medical	purpose_moving	purpose_other	purpose_small_business	purpose_vacation	purpose_wedding
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0

```
In [11]: # 建立training data & test data
training_features, test_features, \
training_target, test_target, = train_test_split(loans_relevant_enconded.drop(['bad_loans'], axis=1),
                                                  loans_relevant_enconded['bad_loans'],
                                                  test_size = .1,
                                                  random_state=12)
```

↓驗證用，故再從training裡面切

```
In [12]: x_train, x_val, y_train, y_val = train_test_split(training_features, training_target,
                                                  test_size = .1,
                                                  random_state=12)
```

```
In [10]: sm = SMOTE(random_state=12, ratio = 1.0)
x_train_res, y_train_res = sm.fit_sample(x_train, y_train)
```

In [16]:
```
clf_rf = RandomForestClassifier(n_estimators=25, random_state=12)
clf_rf.fit(x_train_res, y_train_res)
```

Out[16]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=25, n_jobs=1,
            oob_score=False, random_state=12, verbose=0, warm_start=False)
```

In [14]:
```
print ('Validation Results')
print (clf_rf.score(x_val, y_val))
print (recall_score(y_val, clf_rf.predict(x_val)))
print ('\nTest Results')
print (clf_rf.score(test_features, test_target))
print (recall_score(test_target, clf_rf.predict(test_features)))
```

```
Validation Results
0.800362483009
0.138195777351

Test Results
0.803278688525
0.142546718818
```

Recall_score:
召回是比率TP/（TP + FN），最好的值是1，最差的值是0。

結論：
沒有哪一個方法比較好或比較不好，試了才知道。

THE END