Calculating K-S Statistic with Python – Yet Another Blog in Statistical Computing

K-S statistic is a measure to evaluate the predictiveness of a statistical model for binary outcomes and has been widely used in direct marketing and risk modeling.

Below is a demonstration on how to calculate K-S statistic with less than 20 lines of python codes. In this piece of code snippet, I am also trying to show how to do data munging effectively with pandas and numpy packages.

```
In [1]: # IMPORT PACKAGES

In [2]: import pandas as pd

In [3]: import numpy as np

In [4]: # LOAD DATA FROM CSV FILE

In [5]: data = pd.read_csv('c:\\projects\\data.csv')

In [6]: data.describe()
Out[6]:
               bad        score
count  5522.000000  5522.000000
mean      0.197573   693.466135
std       0.398205    57.829769
min       0.000000   443.000000
25%       0.000000   653.000000
50%       0.000000   692.500000
75%       0.000000   735.000000
max       1.000000   848.000000

In [7]: data['good'] = 1 - data.bad

In [8]: # DEFINE 10 BUCKETS WITH EQUAL SIZE

In [9]: data['bucket'] = pd.qcut(data.score, 10)

In [10]: # GROUP THE DATA FRAME BY BUCKETS

In [11]: grouped = data.groupby('bucket', as_index = False)

In [12]: # CREATE A SUMMARY DATA FRAME

In [13]: agg1 = grouped.min().score

In [14]: agg1 = pd.DataFrame(grouped.min().score, columns = ['min_scr'])

In [15]: agg1['max_scr'] = grouped.max().score
In [16]: agg1['bads'] = grouped.sum().bad

In [17]: agg1['goods'] = grouped.sum().good

In [18]: agg1['total'] = agg1.bads + agg1.goods

In [19]: agg1
Out[19]:
   min_scr  max_scr  bads  goods  total
0      621      645   201    365    566
1      646      661   173    359    532
2      662      677   125    441    566
3      678      692    99    436    535
4      693      708    89    469    558
5      709      725    66    492    558
6      726      747    42    520    562
7      748      772    30    507    537
8      773      848    14    532    546
9      443      620   252    310    562

In [20]: # SORT THE DATA FRAME BY SCORE

In [21]: agg2 = (agg1.sort_index(by = 'min_scr')).reset_index(drop = True)

In [22]: agg2['odds'] = (agg2.goods / agg2.bads).apply('{0:.2f}'.format)

In [23]: agg2['bad_rate'] = (agg2.bads / agg2.total).apply('{0:.2%}'.format)

In [24]: # CALCULATE KS STATISTIC

In [25]: agg2['ks'] = np.round(((agg2.bads / data.bad.sum()).cumsum() -
(agg2.goods / data.good.sum()).cumsum()), 4) * 100

In [26]: # DEFINE A FUNCTION TO FLAG MAX KS

In [27]: flag = lambda x: '<----' if x == agg2.ks.max() else ''

In [28]: # FLAG OUT MAX KS

In [29]: agg2['max_ks'] = agg2.ks.apply(flag)

In [30]: agg2
Out[30]:
   min_scr  max_scr  bads  goods  total   odds bad_rate     ks max_ks
0      443      620   252    310    562   1.23   44.84%  16.10
1      621      645   201    365    566   1.82   35.51%  26.29
2      646      661   173    359    532   2.08   32.52%  34.04
3      662      677   125    441    566   3.53   22.08%  35.55  <----
4      678      692    99    436    535   4.40   18.50%  34.78
5      693      708    89    469    558   5.27   15.95%  32.36
6      709      725    66    492    558   7.45   11.83%  27.30
7      726      747    42    520    562  12.38    7.47%  19.42
8      748      772    30    507    537  16.90    5.59%  10.72
9      773      848    14    532    546  38.00    2.56%  -0.00
```