

Code

Issues2

Pull requests

Actions

Projects

Wiki

Security

Insights

 master ▾



WOE-and-IV / WOE_IV.ipynb



Sundar0989 Add files via upload



 1 contributor

1955 lines (1955 sloc) | 65.9 KB



```
In [1]: #import packages
import os
import pandas as pd
import numpy as np

df = pd.read_excel('bank.xlsx') #Read excel file
```

```
In [2]: df.head()
```

```
Out[2]:
```

	age	job	marital	education	default	balance	housing	loan	contact
0	30	unemployed	married	primary	no	1787	no	no	cellular
1	33	services	married	secondary	no	4789	yes	yes	cellular
2	35	management	single	tertiary	no	1350	yes	no	cellular
3	30	management	married	tertiary	no	1476	yes	yes	unknown
4	59	blue-collar	married	secondary	no	0	yes	no	unknown

```
In [3]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
age                4521 non-null int64
job                4521 non-null object
marital            4521 non-null object
education          4521 non-null object
default            4521 non-null object
balance            4521 non-null int64
housing            4521 non-null object
loan               4521 non-null object
contact            4521 non-null object
day                4521 non-null int64
month              4521 non-null object
duration           4521 non-null int64
campaign           4521 non-null int64
pdays             4521 non-null int64
previous           4521 non-null int64
poutcome           4521 non-null object
y                  4521 non-null object
dtypes: int64(7), object(10)
memory usage: 600.5+ KB
```

```
In [4]: df['y'].value_counts()
```

```
Out[4]: no      4000
yes       521
Name: y, dtype: int64
```

```
In [5]: df['target'] = df['y'].apply(lambda x : 1 if x == 'yes' else 0) # Convert y to target
df = df.drop('y',axis=1)
```

WOE and IV

```
In [6]: # import packages
import pandas as pd
import numpy as np
import pandas.core.algorithms as algos
from pandas import Series
import scipy.stats.stats as stats
import re
import traceback
import string

max_bin = 20
force_bin = 3

# define a binning function
def mono_bin(Y, X, n = max_bin):

    df1 = pd.DataFrame({"X": X, "Y": Y})
    justmiss = df1[['X', 'Y']][df1.X.isnull()]
    notmiss = df1[['X', 'Y']][df1.X.notnull()]
    r = 0
    while np.abs(r) < 1:
        try:
            d1 = pd.DataFrame({"X": notmiss.X, "Y": notmiss.Y, "Bucket": p
            d2 = d1.groupby('Bucket', as_index=True)
            r, p = stats.spearmanr(d2.mean().X, d2.mean().Y)
            n = n - 1
        except Exception as e:
            n = n - 1

    if len(d2) == 1:
        n = force_bin
        bins = algos.quantile(notmiss.X, np.linspace(0, 1, n))
        if len(np.unique(bins)) == 2:
            bins = np.insert(bins, 0, 1)
            bins[1] = bins[1] - (bins[1]/2)
        d1 = pd.DataFrame({"X": notmiss.X, "Y": notmiss.Y, "Bucket": pd.cu
        d2 = d1.groupby('Bucket', as_index=True)

    d3 = pd.DataFrame({}, index=[])
    d3["MIN_VALUE"] = d2.min().X
    d3["MAX_VALUE"] = d2.max().X
    d3["COUNT"] = d2.count().Y
    d3["EVENT"] = d2.sum().Y
    d3["NONEVENT"] = d2.count().Y - d2.sum().Y
    d3=d3.reset_index(drop=True)

    if len(justmiss.index) > 0:
        d4 = pd.DataFrame({'MIN_VALUE':np.nan}, index=[0])
        d4["MAX_VALUE"] = np.nan
        d4["COUNT"] = justmiss.count().Y
        d4["EVENT"] = justmiss.sum().Y
        d4["NONEVENT"] = justmiss.count().Y - justmiss.sum().Y
        d3 = d3.append(d4, ignore_index=True)

    d3["EVENT_RATE"] = d3.EVENT/d3.COUNT
    d3["NON_EVENT_RATE"] = d3.NONEVENT/d3.COUNT
    d3["DIST_EVENT"] = d3.EVENT/d3.sum().EVENT
    d3["DIST_NON_EVENT"] = d3.NONEVENT/d3.sum().NONEVENT
    d3["WOE"] = np.log(d3.DIST_EVENT/d3.DIST_NON_EVENT)
    d3["IV"] = (d3.DIST_EVENT-d3.DIST_NON_EVENT)*np.log(d3.DIST_EVENT/d3.
    d3["VAR_NAME"] = "VAR"
```

```

d3 = d3[['VAR_NAME', 'MIN_VALUE', 'MAX_VALUE', 'COUNT', 'EVENT', 'EVENT']]
d3 = d3.replace([np.inf, -np.inf], 0)
d3.IV = d3.IV.sum()

return(d3)

def char_bin(Y, X):

    df1 = pd.DataFrame({'X': X, 'Y': Y})
    justmiss = df1[['X', 'Y']][df1.X.isnull()]
    notmiss = df1[['X', 'Y']][df1.X.notnull()]
    df2 = notmiss.groupby('X', as_index=True)

    d3 = pd.DataFrame({}, index=[])
    d3["COUNT"] = df2.count().Y
    d3["MIN_VALUE"] = df2.sum().Y.index
    d3["MAX_VALUE"] = d3["MIN_VALUE"]
    d3["EVENT"] = df2.sum().Y
    d3["NONEVENT"] = df2.count().Y - df2.sum().Y

    if len(justmiss.index) > 0:
        d4 = pd.DataFrame({'MIN_VALUE': np.nan}, index=[0])
        d4["MAX_VALUE"] = np.nan
        d4["COUNT"] = justmiss.count().Y
        d4["EVENT"] = justmiss.sum().Y
        d4["NONEVENT"] = justmiss.count().Y - justmiss.sum().Y
        d3 = d3.append(d4, ignore_index=True)

    d3["EVENT_RATE"] = d3.EVENT/d3.COUNT
    d3["NON_EVENT_RATE"] = d3.NONEVENT/d3.COUNT
    d3["DIST_EVENT"] = d3.EVENT/d3.sum().EVENT
    d3["DIST_NON_EVENT"] = d3.NONEVENT/d3.sum().NONEVENT
    d3["WOE"] = np.log(d3.DIST_EVENT/d3.DIST_NON_EVENT)
    d3["IV"] = (d3.DIST_EVENT-d3.DIST_NON_EVENT)*np.log(d3.DIST_EVENT/d3.DIST_NON_EVENT)
    d3["VAR_NAME"] = "VAR"
    d3 = d3[['VAR_NAME', 'MIN_VALUE', 'MAX_VALUE', 'COUNT', 'EVENT', 'EVENT']]
    d3 = d3.replace([np.inf, -np.inf], 0)
    d3.IV = d3.IV.sum()
    d3 = d3.reset_index(drop=True)

    return(d3)

def data_vars(df1, target):

    stack = traceback.extract_stack()
    filename, lineno, function_name, code = stack[-2]
    vars_name = re.compile(r'\((.*)\).*$').search(code).groups()[0]
    final = (re.findall(r'[\w']+', vars_name))[-1]

    x = df1.dtypes.index
    count = -1

    for i in x:
        if i.upper() not in (final.upper()):
            if np.issubdtype(df1[i], np.number) and len(Series.unique(df1[i])) > 1:
                conv = mono_bin(target, df1[i])
                conv["VAR_NAME"] = i
                count = count + 1
            else:
                conv = char_bin(target, df1[i])
                conv["VAR_NAME"] = i
                count = count + 1

    if count == 0:
        iv_df = conv

```

```
        else:
            iv_df = iv_df.append(conv, ignore_index=True)

    iv = pd.DataFrame({'IV':iv_df.groupby('VAR_NAME').IV.max()})
    iv = iv.reset_index()
    return(iv_df, iv)
```

In [7]: final_iv, IV = data_vars(df, df.target)

In [8]: final_iv

Out[8]:

	VAR_NAME	MIN_VALUE	MAX_VALUE	COUNT	EVENT	EVENT_RATE	NONEVENT
0	age	19	39	2290	259	0.113100	2031
1	age	40	87	2231	262	0.117436	1969
2	job	admin.	admin.	478	58	0.121339	420
3	job	blue-collar	blue-collar	946	69	0.072939	877
4	job	entrepreneur	entrepreneur	168	15	0.089286	153
5	job	housemaid	housemaid	112	14	0.125000	98
6	job	management	management	969	131	0.135191	838
7	job	retired	retired	230	54	0.234783	176
8	job	self-employed	self-employed	183	20	0.109290	163
9	job	services	services	417	38	0.091127	379
10	job	student	student	84	19	0.226190	65
11	job	technician	technician	768	83	0.108073	685
12	job	unemployed	unemployed	128	13	0.101562	115
13	job	unknown	unknown	38	7	0.184211	31
14	marital	divorced	divorced	528	77	0.145833	451
15	marital	married	married	2797	277	0.099035	2520
16	marital	single	single	1196	167	0.139632	1029
17	education	primary	primary	678	64	0.094395	614
18	education	secondary	secondary	2306	245	0.106245	2061
19	education	tertiary	tertiary	1350	193	0.142963	1157
20	education	unknown	unknown	187	19	0.101604	168
21	default	no	no	4445	512	0.115186	3933
22	default	yes	yes	76	9	0.118421	67
23	balance	-3313	69	1133	94	0.082966	1039
24	balance	70	444	1128	106	0.093972	1022
25	balance	445	1480	1131	149	0.131742	982
26	balance	1482	71188	1129	172	0.152347	957
27	balance	1482	71188	1129	172	0.152347	957

27	nousing	no	no	1962	301	0.153415	1661
28	housing	yes	yes	2559	220	0.085971	2339
29	loan	no	no	3830	478	0.124804	3352
...
38	month	dec	dec	20	9	0.450000	11
39	month	feb	feb	222	38	0.171171	184
40	month	jan	jan	148	16	0.108108	132
41	month	jul	jul	706	61	0.086402	645
42	month	jun	jun	531	55	0.103578	476
43	month	mar	mar	49	21	0.428571	28
44	month	may	may	1398	93	0.066524	1305
45	month	nov	nov	389	39	0.100257	350
46	month	oct	oct	80	37	0.462500	43
47	month	sep	sep	52	17	0.326923	35
48	duration	4	62	510	1	0.001961	509
49	duration	63	96	502	10	0.019920	492
50	duration	97	128	509	16	0.031434	493
51	duration	129	163	490	20	0.040816	470
52	duration	164	208	505	26	0.051485	479
53	duration	209	261	498	61	0.122490	437
54	duration	262	354	505	73	0.144554	432
55	duration	355	546	500	101	0.202000	399
56	duration	547	3025	502	213	0.424303	289
57	campaign	1	2	2998	378	0.126084	2620
58	campaign	3	50	1523	143	0.093894	1380
59	pdays	-1	-1	3705	337	0.090958	3368
60	pdays	1	1	2	2	1.000000	0
61	pdays	2	871	814	182	0.223587	632
62	previous	0	1	3991	388	0.097219	3603
63	previous	2	25	530	133	0.250943	397
64	poutcome	failure	failure	490	63	0.128571	427
65	poutcome	other	other	197	38	0.192893	159
66	poutcome	success	success	129	83	0.643411	46
67	poutcome	unknown	unknown	3705	337	0.090958	3368

68 rows × 12 columns

In [9]: IV.sort_values('IV')

```
Out[9]:
```

	VAR_NAME	IV
5	default	0.000016
0	age	0.000452
4	day	0.004581
2	campaign	0.023342
7	education	0.031812
11	marital	0.040090
10	loan	0.060791
1	balance	0.076208
8	housing	0.106556
9	job	0.132519
15	previous	0.177081
13	pdays	0.203267
3	contact	0.247762
12	month	0.379533
14	poutcome	0.461890
6	duration	1.651501

```
In [10]: IV.to_csv('test.csv')
```

Apply WOE values to your dataframe columns

The below code snippet can be used to apply the WOE values to your dataframe columns.

```
In [11]: transform_vars_list = df.columns.difference(['target'])
transform_prefix = 'new_' # leave this value blank if you need replace the
```

```
In [12]: transform_vars_list
```

```
Out[12]: Index(['age', 'balance', 'campaign', 'contact', 'day', 'default', 'duration',
               'education', 'housing', 'job', 'loan', 'marital', 'month', 'pdays',
               'poutcome', 'previous'],
              dtype='object')
```

```
In [13]: for var in transform_vars_list:
          small_df = final_iv[final_iv['VAR_NAME'] == var]
          transform_dict = dict(zip(small_df.MAX_VALUE, small_df.WOE))
          replace_cmd = ''
          replace_cmd1 = ''
```

```
for i in sorted(transform_dict.items()):
    replace_cmd = replace_cmd + str(i[1]) + str(' if x <= ') + str(i[0])
    replace_cmd1 = replace_cmd1 + str(i[1]) + str(' if x == "') + str(i[0])
replace_cmd = replace_cmd + '0'
replace_cmd1 = replace_cmd1 + '0'
if replace_cmd != '0':
    try:
        df[transform_prefix + var] = df[var].apply(lambda x: eval(replace_cmd))
    except:
        df[transform_prefix + var] = df[var].apply(lambda x: eval(replace_cmd1))
```

In [14]: df['contact'].value_counts()

Out[14]:

cellular	2896
unknown	1324
telephone	301

Name: contact, dtype: int64

In [15]: df['new_contact'].value_counts()

Out[15]:

0.252971	2896
-0.992072	1324
0.273413	301

Name: new_contact, dtype: int64

In [16]: small_df = final_iv[final_iv['VAR_NAME'] == 'contact']

In [17]: small_df

Out[17]:

	VAR_NAME	MIN_VALUE	MAX_VALUE	COUNT	EVENT	EVENT_RATE	NONEVENT
31	contact	cellular	cellular	2896	416	0.143646	2480
32	contact	telephone	telephone	301	44	0.146179	257