# -\*- coding: utf-8 -\*-

"""

Spyder Editor

This is a temporary script file.

"""

import pandas as pd

import numpy as np

''' Python code for Information value and Mutual Information (beta) '''

#%%

"""

Created on Thu Jul 12 12:47:32 2018

@author: Infosys\_Corporation

Computes Information Value and Weights of Evidence for a categorical

field with respect to another categorical field that is modeled or predicted

Usage:

========

infValue( x, y, n, labels=(0,1), names=('Neg', 'Pos'), mins=(1,1), verb=False)

where:

x : Vector containing the categorical predictor values

y : Vector containing the categorical class labels to be predicted

n : Number of groupings (equi-probable) to be created for x ( default is 7 )

labels : List of class labels to be retained (rest are ignored)

names : List of class names corresponding to the items in 'labels'

mins : List of min count for each class (corresponding to 'labels')

verb : Verbosity Flag. Prints more debugging information to STDOUT when True

By default labels other than 0 and 1 are ignored in computing WoE and IV.

However -1 and +1 will work fine. Basically the rarer class label should

appear after its counterpart in ASCIIbetical sorting.

By default a value of x with non-zero counts for both classes is used for computing Information Value

For more details see the infValue method

"""

def infValueEquibin( x, y, n=7, labels=(0,1), names=('Neg', 'Pos'), mins=(1,1), verb=False):

#n = np.min([np.max([n,3]),7])

xx = pd.qcut( x, n, duplicates='drop')

a = pd.crosstab( xx, y )

a = a.sort\_index( 0 ) # To ensure ascending order

labs2keep=[g for g in a.columns if g in labels ]

a = a[labs2keep]

a.columns = names

'''

From this point on merely repeats the code in infValue

We can either call infValue with 'xx' instead of 'x'

But a better aproach is to consolidate the below code into core functionality

That leaves the door open to extend it to compute Mututal information.

That code will not need to know anything about how the grouping is done, which is perfect!

'''

# Create column containing Column Percentages

pct\_cols = ['Pct\_'+ b for b in a.columns]

for i in range(len(a.columns)):

a[pct\_cols[i]]=a[a.columns[i]]/sum(a[a.columns[i]])

# The below line should be a loop to handle any number of class labels

a['use'] = np.where( np.logical\_and( a[a.columns[0]] > mins[0], a[a.columns[1]] > mins[1]), 1, 0)

a['woe'] = np.where( a['use'] == 1, np.log(a[a.columns[len(names)]]/a[a.columns[len(names)+1]]), 0)

a['ivcontrib'] = (a[a.columns[len(names)]]-a[a.columns[len(names)+1]]) \* a['woe' ]

inf\_val = sum( a['ivcontrib'] )

if verb:

print('Inf. value = '+ str(inf\_val) + ', '+ str(sum(a['use'])) + ' of '+str(len(a['use'])) + ' levels contributing')

print(a)

return (inf\_val, a)

#%%

"""

@author: Infosys\_Corporation

Computes Information Value and Weights of Evidence for a categorical

field with respect to another categorical field that is modeled or predicted

Usage:

========

infValue( x, y, labels=(0,1), names=('Neg', 'Pos'), mins=(1,1), verb=False)

where:

x : Vector containing the categorical predictor values

y : Vector containing the categorical class labels to be predicted

labels : List of class labels to be retained (rest are ignored)

names : List of class names corresponding to the items in 'labels'

mins : List of min count for each class (corresponding to 'labels')

offsets : Counts to be added to each category-level count

verb : Verbosity Flag. Prints more debugging information to STDOUT when True

By default labels other than 0 and 1 are ignored in computing WoE and IV.

However -1 and +1 will work fine. Basically the rarer class label should

appear after its counterpart in ASCIIbetical sorting.

By default a value of x with non-zero counts for both classes is used for computing Information Value

If your x values are numeric and continuous, you need infValueEquibin() method

To do list:

--------------

1) Laplace Correction (adds 1 to counts for each class)

2) Bayesian Priors (will add positive values to counts for each class)

3) Be able to show WoE with alternative sign (IV will not change)

4) Allow other common definitions of IV such as Pct\_Target x abs(WoE)

"""

def infValue( x, y, labels=(0,1), names=('Neg', 'Pos'), mins=(1,1), offsets=(0,0), verb=False ):

a = pd.crosstab( x, y )

labs2keep=[g for g in a.columns if g in labels ]

a = a[labs2keep]

a.columns = names

if sum(offsets) > 0 :

a = a + offsets

# Create column containing Column Percentages

pct\_cols = ['Pct\_'+b for b in a.columns]

for i in range(len(a.columns)):

a[pct\_cols[i]]=a[a.columns[i]]/sum(a[a.columns[i]])

a['use'] = np.where( np.logical\_and( a[a.columns[0]] > mins[0], a[a.columns[1]] > mins[1]), 1, 0)

a['woe'] = np.where( a['use'] == 1, np.log(a[a.columns[len(names)]]/a[a.columns[len(names)+1]]), 0)

a['ivcontrib'] = (a[a.columns[len(names)]]-a[a.columns[len(names)+1]]) \* a['woe' ]

inf\_val = sum( a['ivcontrib'] )

if verb:

print('Inf. value = '+ str(inf\_val) + ', '+ str(sum(a['use'])) + ' of '+str(len(a['use'])) + ' levels contributing')

print(a)

return (inf\_val, a)

#%%

'''

Created on Thu Sep 26 13:36:32 2019

@author: Infosys\_Corporation

From the given data of non-factorize numerical data to computes

cross tabuation of the x and y data and mutual information of the data was passed in.

Usage:

========

mutualInf( x, y, bins=(10,10), offset=0, verb=False )

where:

x : A cluster of the data with non-factorize

y : A cluster of the data with non-factorize

bin : Set the size of the data cut which also apply to the cross tabuation

offset : Offset of the cross tabuation

verb: Decide to print out the cross matrix of the data.

By defalut bins value of (10,10) is for cut the data into maximum of 10

for both row and columns. However, if the cut value is less bins value then

it will use the samller value rather then using bin size.

By default a value of offset with zero for the cross tabuation matrix.

by default a value of verb with True to not print out the result of the cross matrix.

Example:

from sklearn.datasets import load\_iris

iris = load\_iris()

df = pd.DataFrame(iris.data, columns=iris.feature\_names)

mi = mutualInf(df['sepal length (cm)'],df['petal length (cm)'])[1]

print(mi)

mi will return 0.8768685231704909.

'''

def mutualInf( x, y, bins=(10,10), offset=0, verb=False ):

# x\_f = x

# y\_f = y

if x.dtype == "object":

xx, qx = pd.factorize(x)

xnames = qx

else:

xx = pd.qcut( x, bins[0], duplicates='drop' )

xnames = xx.values.categories

if y.dtype == "object":

yy, qy = pd.factorize(y)

ynames = qy

else:

yy = pd.qcut( y, bins[1], duplicates='drop' )

ynames = yy.values.categories

xx2=pd.factorize(xx)[0]

yy2=pd.factorize(yy)[0]

a = pd.crosstab( xx2, yy2 )

if offset > 0 : a = a + offset

if verb: print(a)

entx = entropy( a.sum( axis=1 ))

enty = entropy( a.sum( axis=0 ))

entxy = entropy( a )

mutinf = entx+enty-entxy

a.columns = ynames

a.index = xnames

return(mutinf,a)

#%%

def entropy( x, normalize=False ):

if x.sum().sum() > 1:

x = x/x.sum().sum()

ent = -(x\*np.log(x)).sum().sum()

maxent = np.log( np.prod(x.shape) ) # Use if normalizing by max

if normalize:

ent = ent/maxent

return (ent)

#%%

# SAMPLE USAGE

# With Bank Marketing data at https://archive.ics.uci.edu/ml/machine-learning-databases/00222/

pd.set\_option( 'display.max\_columns', 10 )

pd.set\_option('precision', 6)

pd.set\_option('max\_colwidth', 20)

pd.set\_option('display.width', 120)

import pprint

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read\_csv('E:/Projects/Codebase/data/bank-marketing/bank-full.csv', sep=';')

# Examine target values and map to 0 and 1

df.y.value\_counts()

targ\_dict = {'no': 0, 'yes':1 }

df['target'] = [targ\_dict[yy] for yy in df.y]

df.target.value\_counts()

#%%

# Information Value -- Categorical fields

(iv1, a1) = infValue( df['education'], df['target'], verb=True, mins=(1000,300), names=('No\_Deposit','Deposit') )

(iv2, a2) = infValue( df['job'], df['target'], verb=True, mins=( 500,100), names=('No\_Deposit','Deposit') )

(iv3, a3) = infValue( df['contact'], df['target'], verb=True, mins=( 500,100), names=('No\_Deposit','Deposit') )

(iv4, a4) = infValue( df['poutcome'], df['target'], verb=True, mins=( 500,100), names=('No\_Deposit','Deposit') )

ivdict1 = {'education': iv1, 'job': iv2, 'contact': iv3, 'poutcome': iv4}

pprint.pprint(ivdict1)

#%%

# Information Value -- Numerical fields

(iv5, a5) = infValueEquibin( df['duration'], df['target'], n=12, verb=True, mins=(500,100), names=('No\_Deposit','Deposit') )

(iv6, a6) = infValueEquibin( df['campaign'], df['target'], n=10, verb=True, mins=(500,100), names=('No\_Deposit','Deposit') )

(iv7, a7) = infValueEquibin( df['pdays'], df['target'], n=12, verb=True, mins=(500,100), names=('No\_Deposit','Deposit') )

(iv8, a8) = infValueEquibin( df['age'], df['target'], n=12, verb=True, mins=(500,100), names=('No\_Deposit','Deposit') )

ivdict2 = {'duration': iv5, 'campaign': iv6, 'pdays': iv7, 'age': iv8}

pprint.pprint(ivdict2)

#%%

# Mutual Information -- Both Categorical

(mi1, b1) = mutualInf(x=df.marital, y= df.job, verb=True)

b1.T

mi1

# Mutual Information -- Both Numerical

(mi2, b2) = mutualInf(x=df.age, y=df.balance, bins=(5,5), verb=False)

b2.T

mi2

# Mutual Information -- One Numerical, Another Categorical

(mi3, b3) = mutualInf(x=df.job, y=df.age, bins=(5,5), offset=1, verb=False)

b3

mi3

# Reversing the order to make sure no strange effects!

(mi3r, b3r) = mutualInf(y=df.job,x=df.age, bins=(5,5), offset=1, verb=False)

# Mutual information must be identical

print((mi3, mi3r))

#%%

# grid = np.random.rand(4, 4)

grid = b3/b3.sum().sum()

x = b3.columns

y = b3.index

plt.figure(figsize=(12,8))

imgplot = plt.imshow(grid, interpolation='none')

imgplot.set\_cmap('nipy\_spectral')

plt.xticks(range(len(x)), x, fontsize=12, rotation=45)

plt.yticks(range(len(y)), y, fontsize=12)

plt.colorbar()