Icics Documentation

Release 0.3.4

XuJing

Dec 15, 2017

Contents:

CHAPTER 1

icics

Xujing

Inter Credit Intelligent classification system (icics). It is a python package to Inter-credit. You can use it to assign cases to your salasman.

CHAPTER 2

Introduction

Firstly simply describe the process of the algorithm:

1. First, define the target legal ‘preference’ cases(eg. the law operates the case with a high recovery rate and considers the law ‘preference’ in the case)
2. The sequence of cases randomly disrupting which ready to be allocated (the operation mainly weakens the shortcomings of the use of the algorithm)
3. The clustering algorithm is performed on the target legal ‘preference’ case, which brings together the similar cases. And record the number of the centers of each cluster and the number of cases in each cluster
4. The similarity between each case and target law is calculated for each case, and the case is assigned to a cluster of high similarity, and the similarity is recorded.
5. The case similarity of each cluster sort, select $k.(N_i/N)$ with high similarity in each cluster as the object of the legal case to be allocated cases (where the k said the case for the allocation of the legal distribution of cases of households, N_i said the goal of forensic in each cluster “liking” the number of cases, N said the goal of all legal history “liking” the number of products)
6. Delete the object of law and the case assigned by the objective of the law, and carry out the assignment of the next target legal case.

more details you can pip this package in your equipment, and there are much more information about icics.

CHAPTER 3

API

CHAPTER 4

Demo

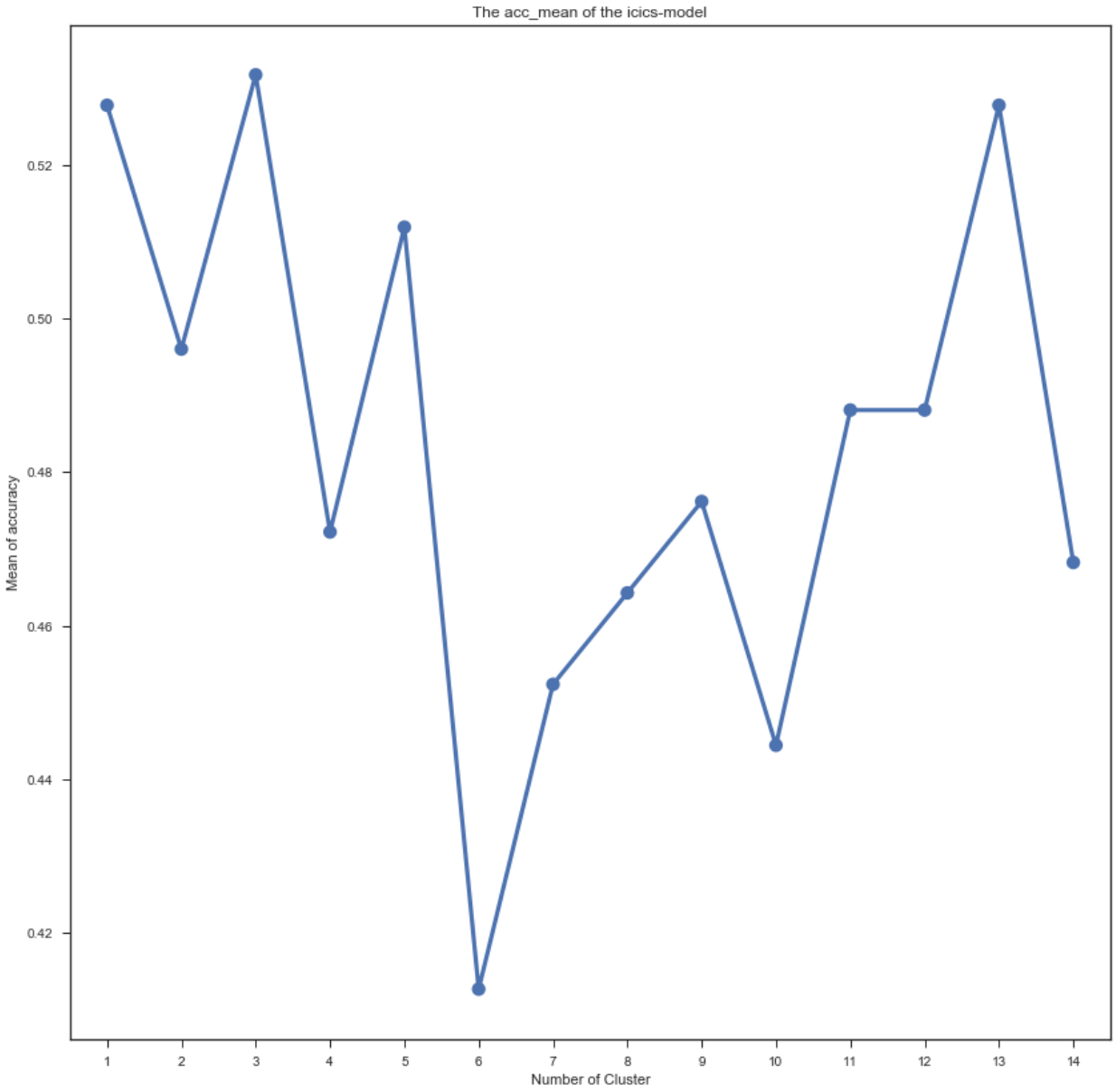
```
import numpy as np
import pandas as pd

from  sklearn.cluster import KMeans
from sklearn.cross_validation import StratifiedKFold

import random
import matplotlib.pyplot as plt

import seaborn as sns

from icics import *

#test the model

train = pd.read_csv(u"C:/Users/Administrator.USER-20170417DX/Desktop/test1.csv")
df = train.copy()

dfold0 = train.iloc[range(1000)]
df0 = train.iloc[range(1000,2000)].drop('ywy0',axis=1)

topn0 = pd.DataFrame({'ywy0':np.unique(dfold0.ywy0),'topn_ywy':
     [30,50,40,40,40,40,40,40,40,40,40,40,40,40,40,40,40,40,40,40,40,20,60]})

bhywy0 = pd.DataFrame({'ywy0':dfold0.ywy0})

icic(dfold0,df0,topn0,bhywy0,ncluster=2,shuffle=1,epsilon=0.001,init='k-means++',
↪random_state=123,max_iter=1000,algorithm="auto",path=0)
```

then you can training the hyper-parameters use the method of acc_mean:

```
#train the model

train = pd.read_excel(u"C:/Users/Administrator.USER-20170417DX/Desktop/test2.xlsx")
```

```
bhywy = pd.DataFrame({'ywy0':train.ywy0})
topn = pd.DataFrame({'ywy0':['cc12','cd17'],'topn_ywy':[30,40]})

pre_mean=[]
for j in np.arange(1,15):
    means = acc_mean(train,bhywy,topn,j)
    pre_mean.append(means)

precies = pd.DataFrame({'ncluster' : np.arange(1,15),'acc_mean' : pre_mean})


plt.figure(1,figsize=(14,14))

with sns.axes_style("ticks"):
    plt.title('The acc_mean of the icics-model')
    sns.pointplot(x='ncluster',y='acc_mean',data=precies)
    plt.xlabel('Number of Cluster')
    plt.ylabel('Mean of accuracy')
plt.show()
```

the result like this:


The acc_mean of the icics-model
Mean of accuracy
0.52
0.50
0.48
0.46
0.44
0.42
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Number of Cluster

CHAPTER 5

Supports

Tested on Python 2.7, 3.5, 3.6

- pip install Icics
- Download: https://pypi.python.org/pypi/Icics
- Documentation: https://github.com/DataXujing/Icics

you can log in Xujing's home page: https://dataxujing.coding.me or https://dataxujing.github.io to learn more.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search