市场监管-企业活力指数

摘要

企业活力是企业生产经营活动得以迅速发展的内在力量，企业活力指数主要用于推测企业当前的生产经营状况。现基于材料数据，构建独立的企业活力指数体系。该体系主要采用熵权法和主成分分析法，分别对数据（企业公示基本信息和涉诉等其他六个维度的信息）进行分析，并用 Python 语言实现。

关键字：企业活力指数　熵权法　主成分分析法　Python 语言

一、问题重述

现有一份企业数据，需根据数据中所提供的多维度的信息自行设计公式，构建企业活力指数体系，计算每家企业的企业活力指数得分，并根据得分对企业进行排名

二、问题分析

在企业数据中，有六个维度，分别为企业公示信息、开庭公告、税务评级、招投标、抽查检查、行政处罚、经营异常。在企业活力指数体系中，定义开庭公告、税务评级、招投标、抽查检查、行政处罚、经营异常为一级指标，将其细化后，定义出二级指标，如表 1：

表 1

一级指标	二级指标
开庭公告	不要式合同
	要式合同
	其他合同
	人身权利
	财产权利
	其他纠纷
税务评级	国税
	地税
投招标	仅中标
	仅招标
	招标中标
	采购中标
	其他
抽查检查	正常
	未发现异常
	不合格
	弄虚作假
	其他
行政处罚	不合格产品
	未按规定执行
	违反处罚
	违法罚款
经营异常	未公示年度报告
	弄虚作假
	无法联系
	未移除

通过熵权法计算出二级指标的权重，过程如下：

再通过主成分分析和熵权法的综合得分进行归一化后求出最总得分，在对其进行排名。

三、模型假设

1. 假设各指标对企业活力有影响
2. 假设通过息熵计算各指标的权重都较为准确
3.假设计算得出企业活力指数得分能够推测企业当前的生产经营状况

四、建模建立

表 2：二级指标数据

	指标（$X_1, X_2, \cdots, X_i$）			
公司 ID	X_{11}	X_{12}	$\cdots$	X_{1j}
	X_{21}	X_{22}	$\cdots$	X_{2j}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	X_{i1}	X_{i2}	$\cdots$	X_{ij}

（一）熵权法：

4.1.1 数据标椎化

数据中有 i 个指标，j 个公司 ID，假设各指标数据标准化后的值为Y_{ij}，则

$$Y_{ij} = \frac{X_{ij} - \min(X_i)}{max(X_i) - min(X_i)}$$

4.1.2 求个指标的信息熵

根据信息论中信息熵的定义，假设一组数据的信息熵为E_j,则

$$E_j = -ln(n)^{-1} \sum_{i=1}^{n} P_{ij} ln P_{ij}$$

其中，$P_{ij} = {Y_{ij}}\Big/{\sum_{i=1}^{n} Y_i}$　　如果$P_{ij} = 0$,则定义$\lim_{P_{ij} \to 0} P_{ij} \ln P_{ij} = 0$

4. 1. 3 确定各指标权重

根据信息熵的计算公式，计算出各个指标的信息熵为E_j。

通过信息熵计算各指标的权重：$W_i = \frac{1-E_i}{k-\sum E_i}$

经过计算，得出各二级指标的权重，如表 3：

表 3 市场监管评价指标体系

一级指标	权重	二级指标	权重
开庭公告	0.0459182	不要式合同	0.177812
		人身权利	0.0493907
		其他合同	0.0682399
		其他纠纷	0.382907
		要式合同	0.269038
		财产权利	0.0526125
税务评级	0.0773297	国税	0.0979839
		地税	0.902016
招投标	0.169566	中标	0.170651
		其他	0.1700720
		招标	0.285877
		招标中标	0.223041
		采购中标	0.150359
抽查检查	0.149426	不合格	0.168571
		其他	0.153652
		弄虚作假	0.244325
		未发现异常	0.107241
		正常	0.32621
行政处罚	0.154779	不合格产品	0.0657651
		其他	0.511104
		未按规定执行	0.150733
		违反处罚	0.184369
		违法罚款	0.0880292
经营异常	0.402981	未移除	0.319277
		弄虚作假	0.236527
		无法联系	0.268982
		未公示年度报告	0.175215

4.1.4 指标加权计算得分

利用加权求和公式计算样本的分数，其中为$\sum_{i=1}^{781} x_{ij} w_{ij}$综合得分。

（二）主成分分析法：

主成分分析法是一种数学降维方法，其主要目的是找出几个综合变量来代替原来众多变量，使得这些综合变量尽可能地代表原来变量的信息彼此之间互不相关。

主成分分析的一般步骤：

4.2.1.对原数据进行标准化处理

4.2.2.选择重要的主成分，并写主成分表达式

4.2.3.计算主成分得分

4.2.4.依据主成分得分的数据，进一步从事统计分析

（三）综合两种方法得分为：

表 4 市场监管企业活力指数得分排名

ID	name	score	Rank
94878702	飞利浦（中国	2	1
1378470808	广东华润涂	1.293747976	2
864283691	广东万和新电	1.243142906	3
592682847	广州市阳光科密	1.201449647	4
165103609	四川南骏汽车	1.05417621	5
782314337	上海一冷开利空	1.047146033	6
415923145	中建海峡建设	1.047126551	7
22822	北京百度网讯	1.047126551	7
416911296	湘潭市电线	1.03900006	8
189667023	佛山电器照明	1.021685806	9
2349641578	南京富士通电子信	1.008763524	10
…	…	…	…

五、优缺点

（一）熵权法：

优点：

1. 能深刻反映出指标的区分能力，进而确定权重；
2. 是一种客观赋权法，相对主管赋权具有较高的可信度和精确度；
3. 算法简单。

缺点：

1. 若无业务经验指导，权重可能失真；
2. 对样本的依赖性较大，随着建模样本不断变化，权重会发生一定波动。

（二）主成分分析法：

优点：

1. 可消除评价指标之间的相关影响；
2. 可减少指标选择的工作量。

缺点：

1. 变量降维后的信息量必须保持在一个较高水平上，其次对这些被提取的主成分必须都能够给出符合实际背景和意义的解释；
2. 主成分的解释其含义一般多少带点模糊性，不像原始变量的含义那么清楚、确切。

参考文献：

张虹、庄文英，基于创新能力的小微企业活力指数体系研究

六、附录

代码：

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import math
```

```
from numpy import array
from sklearn.preprocessing import StandardScaler

#读取 Excel 工作表数据数据
A0=pd.read_excel('市场监管赛题数据.xls','企业公示信息')
A1=pd.read_excel('市场监管赛题数据.xls','开庭公告')
A2=pd.read_excel('市场监管赛题数据.xls','税务评级')
A3=pd.read_excel('市场监管赛题数据.xls','招投标')
A4=pd.read_excel('市场监管赛题数据.xls','抽查检查')
A5=pd.read_excel('市场监管赛题数据.xls','行政处罚')
A6=pd.read_excel('市场监管赛题数据.xls','经营异常')

#初步数据清理
A0=A0.fillna({'LEGAL':'合法','ENNAME':'无','NSRZZ':'无'})
#A0=A0[-A0.ENT_STATUS.isin(['吊销','注销'])]

A0=A0.dropna()
A1=A1.dropna()
A3=A3.dropna()
#删除一列中为 1 的数值
A4=A4[-A4.TYPES.isin(['1'])]
A4=A4[-A4.JC.isin(['1'])]
A4=A4.dropna()
A5=A5.dropna()
A6=A6.fillna({'YCYY':'未移除','YCDATES':'know'})
A6=A6.dropna()

a=A0.iloc[:,13].values
listday=[]
for i in range(len(a)):
        listday.append(2018-int(a[i][:4]))
num=np.array(listday)

#处理数据开庭公告
Z1=[]
for i in range(len(A1)):
    a=A1.iloc[i,2]
    #要式合同与不要式合同
    if a.find('合同',0,len(a))!=-1:
        #要式合同
        if a.find('租赁',0,len(a))!=-1 or a.find('保证',0,len(a))!=-1:
            Z1.append(1)
        elif a.find('借款',0,len(a))!=-1 or a.find('建造',0,len(a))!=-1:
            Z1.append(1)
```

```
        elif a.find('建设',0,len(a))!=-1 or a.find('转让',0,len(a))!=-1:
            Z1.append(1)
        elif a.find('运输',0,len(a))!=-1 or a.find('劳动',0,len(a))!=-1:
            Z1.append(1)
        #不要式合同
        elif a.find('买卖',0,len(a))!=-1 or a.find('承揽',0,len(a))!=-1:
            Z1.append(2)
        elif a.find('定作',0,len(a))!=-1 or a.find('委托',0,len(a))!=-1:
            Z1.append(2)
        elif a.find('仓储',0,len(a))!=-1 or a.find('居间',0,len(a))!=-1:
            Z1.append(2)
        elif a.find('劳务',0,len(a))!=-1 or a.find('加工',0,len(a))!=-1:
            Z1.append(2)

        else:
            Z1.append(3)

    else:
        #财产权利
        if a.find('专利',0,len(a))!=-1 or a.find('著作',0,len(a))!=-1:
            Z1.append(4)
        elif a.find('商标',0,len(a))!=-1 or a.find('传播',0,len(a))!=-1:
            Z1.append(4)
        elif a.find('侵权',0,len(a))!=-1 or a.find('追偿',0,len(a))!=-1:
            Z1.append(4)
        elif a.find('求偿',0,len(a))!=-1 or a.find('追偿',0,len(a))!=-1:
            Z1.append(4)
        #人身权利
        elif a.find('健康',0,len(a))!=-1 or a.find('生命',0,len(a))!=-1:
            Z1.append(5)
        elif a.find('名誉',0,len(a))!=-1 or a.find('人格',0,len(a))!=-1:
            Z1.append(5)
        elif a.find('肖像',0,len(a))!=-1 or a.find('人身',0,len(a))!=-1:
            Z1.append(5)
        elif a.find('身体',0,len(a))!=-1 or a.find('人身',0,len(a))!=-1:
            Z1.append(5)
        else:
            Z1.append(6)

Z1=np.array(Z1)
z1=[];z2=[];z3=[];z4=[];z5=[];z6=[]
for k in range(len(A0)):
    a=Z1[A1.iloc[:,1].values==A0.iloc[k,1]]
    if len(a)>0:
```

```
            z1.append(len(a[a==1]))
            z2.append(len(a[a==2]))
            z3.append(len(a[a==3]))
            z4.append(len(a[a==4]))
            z5.append(len(a[a==5]))
            z6.append(len(a[a==6]))
        else:
            z1.append(0)
            z2.append(0)
            z3.append(0)
            z4.append(0)
            z5.append(0)
            z6.append(0)

Z1={'ID':A0.iloc[:,1].values,'要式合同':z1,'不要式合同':z2,
   '其他合同':z3,'财产权利':z4,'人身权利':z5,'其他纠纷':z6}
Z1=pd.DataFrame(Z1)
Z1=Z1.set_index(['ID'])
#Z1.to_excel('Z1.xls')

#处理数据税务评级
Z2=[]
for i in range(len(A2)):
    a=A2.iloc[i,3]
    if a.find('国税',0,len(a))!=-1:
        Z2.append(1)
    elif a.find('地税',0,len(a))!=-1:
        Z2.append(2)

Z2=np.array(Z2)
d1=[];d2=[];
for k in range(len(A0)):
    a=Z2[A2.iloc[:,1].values==A0.iloc[k,1]]
    if len(a)>0:
        d1.append(len(a[a==1]))
        d2.append(len(a[a==2]))
    else:
        d1.append(0)
        d2.append(0)
Z2={'ID':A0.iloc[:,1].values,'国税':d1,'地税':d2}
Z2=pd.DataFrame(Z2)
Z2=Z2.set_index(['ID'])
#Z2.to_excel('D.xls')
```

```
#处理数据投招标
Z3=[]
for i in range(len(A3)):
    a=A3.iloc[i,2]
    #要式合同与不要式合同
    if a.find('中标',0,len(a))!=-1 and a.find('招标',0,len(a))!=-1:
        Z3.append(1)
    elif a.find('招标',0,len(a))!=-1:
        Z3.append(2)
    elif a.find('采购',0,len(a))!=-1 and a.find('中标',0,len(a))!=-1:
        Z3.append(3)
    elif a.find('中标',0,len(a))!=-1:
        Z3.append(4)
    else:
        Z3.append(0)

Z3=np.array(Z3)
x1=[];x2=[];x3=[];x4=[];x5=[]
for k in range(len(A0)):

    a=Z3[A3.iloc[:,1].values==A0.iloc[k,1]]

    if len(a)>0:
        x1.append(len(a[a==1]))
        x2.append(len(a[a==2]))
        x3.append(len(a[a==3]))
        x4.append(len(a[a==4]))
        x5.append(len(a[a==0]))
    else:
        x1.append(0)
        x2.append(0)
        x3.append(0)
        x4.append(0)
        x5.append(0)

Z3={'ID':A0.iloc[:,1].values,'招标中招':x1,'招标':x2,
   '采购中标':x3,'中标':x4,'其他':x5}
Z3=pd.DataFrame(Z3)
Z3=Z3.set_index(['ID'])
#Z3.to_excel('X.xls')

#处理数据抽查检查
Z4=[]
for i in range(len(A4)):
```

```
    a=A4.iloc[i,3]
    if a.find('正常',0,len(a))!=-1 or a.find('合格',0,len(a))!=-1 or a.find('符合',0,len(a))!=-1:
        Z4.append(1)
    elif a.find('暂没发现违法违规行为',0,len(a))!=-1 or a.find('未发现异常',0,len(a))!=-1:
        Z4.append(2)
    elif a.find('不合格',0,len(a))!=-1 or a.find('发现问题',0,len(a))!=-1:
        Z4.append(3)
    elif a.find('弄虚作假',0,len(a))!=-1 or a.find('整改',0,len(a))!=-1:
        Z4.append(4)
    else:
        Z4.append(5)

Z4=np.array(Z4)
e1=[];e2=[];e3=[];e4=[];e5=[]
for k in range(len(A0)):
    a=Z4[A4.iloc[:,1].values==A0.iloc[k,1]]
    if len(a)>0:
        e1.append(len(a[a==1]))
        e2.append(len(a[a==2]))
        e3.append(len(a[a==3]))
        e4.append(len(a[a==4]))
        e5.append(len(a[a==5]))
#         E6.append(len(a[a==0]))
    else:
        e1.append(0)
        e2.append(0)
        e3.append(0)
        e4.append(0)
        e5.append(0)
#         E6.append(0)
Z4={'ID':A0.iloc[:,1].values,'正常':e1,'未发现异常':e2,'不合格':e3,'弄虚作假':e4,'其他':e5}
Z4=pd.DataFrame(Z4)
Z4=Z4.set_index(['ID'])
#Z4.to_excel('E.xls')


#处理数据行政处罚
Z5=[]
for i in range(len(A5)):
    a=A5.iloc[i,3]
    if a.find('违法',0,len(a))!=-1 or a.find('罚款',0,len(a))!=-1:
```

```
        Z5.append(1)
    elif a.find('违反',0,len(a))!=-1 or a.find('处罚',0,len(a))!=-1:
        Z5.append(2)
    elif a.find('不合格',0,len(a))!=-1:
        Z5.append(3)
    elif a.find('未',0,len(a))!=-1:
        Z5.append(4)
    else:
        Z5.append(5)

Z5=np.array(Z5)
f1=[];f2=[];f3=[];f4=[];f5=[]
for k in range(len(A0)):
    a=Z5[A5.iloc[:,1].values==A0.iloc[k,1]]
    if len(a)>0:
        f1.append(len(a[a==1]))
        f2.append(len(a[a==2]))
        f3.append(len(a[a==3]))
        f4.append(len(a[a==4]))
        f5.append(len(a[a==5]))
#         E6.append(len(a[a==0]))
    else:
        f1.append(0)
        f2.append(0)
        f3.append(0)
        f4.append(0)
        f5.append(0)
#         E6.append(0)
Z5={'ID':A0.iloc[:,1].values,'违法罚款':f1,'违反处罚':f2,'不合格产品':f3,'未按
规定执行':f4,'其他':f5}
Z5=pd.DataFrame(Z5)
Z5=Z5.set_index(['ID'])
#Z5.to_excel('F.xls')


#处理数据经营异常
Z6=[]
for i in range(len(A6)):
    a=A6.iloc[i,2]
    b=A6.iloc[i,4]
    if b.find('未移除',0,len(b))!=-1:
        Z6.append(0)
    elif a.find('无法',0,len(a))!=-1 and a.find('联系',0,len(a))!=-1:
        Z6.append(1)
```

```
    elif a.find('公示',0,len(a))!=-1 and a.find('年',0,len(a))!=-1 and a.find('报',0,len(a))!=-1:
        Z6.append(2)
    elif a.find('弄虚作假',0,len(a))!=-1:
        Z6.append(3)
    else:
        Z6.append(4)

Z6=np.array(Z6)
g1=[];g2=[];g3=[];g4=[]
for k in range(len(A0)):
    a=Z6[A6.iloc[:,1].values==A0.iloc[k,1]]
    if len(a)>0:
        g1.append(len(a[a==0]))
        g2.append(len(a[a==1]))
        g3.append(len(a[a==2]))
        g4.append(len(a[a==3]))
    else:
        g1.append(0)
        g2.append(0)
        g3.append(0)
        g4.append(0)
Z6={'ID':A0.iloc[:,1].values,'未移除':g1,'无法联系':g2,'未公示年度报告':g3,'弄虚作假':g4}
Z6=pd.DataFrame(Z6)
Z6=Z6.set_index(['ID'])
#Z6.to_excel('G.xls')


S0=A0.iloc[:,1]
S0=pd.DataFrame(S0)
S0=S0.set_index(['ID'])
S1=A1.groupby(['ID'],as_index=False)['ID'].agg({'开庭公告':'count'})
S2=A2.groupby(['ID'],as_index=False)['ID'].agg({'税务评级':'count'})
S3=A3.groupby(['ID'],as_index=False)['ID'].agg({'招投标':'count'})
S4=A4.groupby(['ID'],as_index=False)['ID'].agg({'抽查检查':'count'})
S5=A5.groupby(['ID'],as_index=False)['ID'].agg({'行政处罚':'count'})
S6=A6.groupby(['ID'],as_index=False)['ID'].agg({'经营异常':'count'})


S1=S1.set_index(['ID'])
S2=S2.set_index(['ID'])
S3=S3.set_index(['ID'])
S4=S4.set_index(['ID'])
```

```
S5=S5.set_index(['ID'])
S6=S6.set_index(['ID'])

B=S0.join(S1)
B=B.join(S2)
B=B.join(S3)
B=B.join(S4)
B=B.join(S5)
B=B.join(S6)

B=B.fillna({'开庭公告':0,'税务评级':0,'招投标':0,'抽查检查':0,'行政处罚':0,'经营异常':0})
#B.to_excel('B.xls')
def cmu_z(z,num):
  for j in range(len(z.iloc[0,:])):
      z.iloc[:,j]=z.iloc[:,j]/num
  return z

Z1=cmu_z(Z1,num)
Z2=cmu_z(Z2,num)
Z3=cmu_z(Z3,num)
Z4=cmu_z(Z4,num)
Z5=cmu_z(Z5,num)
Z6=cmu_z(Z6,num)

def cal_weight(x,z):
    '''熵值法计算变量的权重'''
# 标准化
    x = x.apply(lambda x: ((x - np.min(x)) / (np.max(x) - np.min(x))))
    if z==1:
        x=1-x
# 求 k
    rows = x.index.size  # 行
    cols = x.columns.size  # 列
    k= 1.0 / math.log(rows)

#    print(k)
    lnf = [[None] * cols for i in range(rows)]
            # 矩阵计算--
# 信息熵
    x = array(x)
    lnf = [[None] * cols for i in range(rows)]
    lnf = array(lnf)
    for i in range(0, rows):
```

```
        for j in range(0, cols):
            if x[i][j] == 0:
                lnfij = 0.0
            else:
                p = x[i][j] / x.sum(axis=0)[j]
                lnfij = math.log(p) * p * (-k)
            lnf[i][j] = lnfij
    lnf = pd.DataFrame(lnf)
    E = lnf
# 计算冗余度
    d = 1 - E.sum(axis=0)
# 计算各指标的权重
    wb = [[None] * 1 for i in range(cols)]
    for j in range(0, cols):
        wj = d[j] / sum(d)
        wb[j] = wj
    wb = pd.DataFrame(wb)
    return wb

if __name__ == '__main__':
#计算各指标各字段的权重
    W0 =cal_weight(B,0,)
    W1 =cal_weight(Z1,1)
    W2 =cal_weight(Z2,0)
    W3 =cal_weight(Z3,0)# 调用 cal_weight
    W4 =cal_weight(Z4,1)
    W5 =cal_weight(Z5,1)
    W6 =cal_weight(Z6,1)

    W0.index = B.columns
    W1.index = Z1.columns
    W2.index = Z2.columns
    W3.index = Z3.columns
    W4.index = Z4.columns
    W5.index = Z5.columns
    W6.index = Z6.columns

    W0.columns = ['一级指标权重']
    W1.columns = ['开庭公告各指标权重']
    W2.columns = ['税务评级各指标权重']
    W3.columns = ['招投标各指标权重']
    W4.columns = ['抽查检查各指标权重']
    W5.columns = ['行政处罚各指标权重']
    W6.columns = ['经营异常各指标权重']
```

```
#主成分数据规范化处理
scaler=StandardScaler()
scaler.fit(B)
B2=scaler.transform(B)
from sklearn.decomposition import PCA
pca=PCA(n_components=0.95)#贡献率设为95%以上
Y=pca.fit_transform(B2)#满足贡献率为95%的组成分数据
#Y=pd.DataFrame(Y)
gxl=pca.explained_variance_ratio_ #返回主成分方差百分比(贡献率)
#gxl=pd.DataFrame(gxl)
Score2=gxl[0]*Y[:,0]+gxl[1]*Y[:,1]+gxl[2]*Y[:,2]+gxl[3]*Y[:,3]+gxl[4]*Y[:,4]+gx
l[5]*Y[:,5]
tq=B.index.values
Rs=pd.DataFrame(Score2,index=tq)
#Rs=Rs.sort_values(ascending=False)
Rs.columns = ['Score']

Rs['top']=Rs['Score'].rank(ascending=0,method='dense')#添加排序
t1=A0.iloc[:,0]
t1.index=Rs.index
t1=pd.DataFrame(t1)
Top1=t1.join(Rs)
#计算综合得分
i=0;j=0
C=[]
for j in range(0,len(B)):
    i=0;d=0
    for i in range(0,len(W0)):
        m=W0.iloc[i,:]
        n=B.iloc[j,i]
        c=m*n
        d=d+c
        i=i+1
    C.append(d)
    j=j+1
C=pd.DataFrame(C)
C.index = B.index
C.columns = ['Score']


C['top']=C['Score'].rank(ascending=0,method='dense')#添加排序
```

```
t0=A0.iloc[:,[0,1]]
t0.index=B.index
Top=t0.join(C)

f1=Top.iloc[:,2].values
f2=Top1.iloc[:,1].values
f1=(f1-min(f1))/(max(f1)-min(f1))
f2=(f2-min(f2))/(max(f2)-min(f2))
f=f1+f2
#S=pd.Series(f,index=Top.iloc[:,0].values)
#fs=S.sort_values(ascending=False)  #最后得分排名

D={'name':A0.iloc[:,0].values,'id':A0.iloc[:,1].values,'score':f}
Data=pd.DataFrame(D)

Data['top']=Data['score'].rank(ascending=0,method='dense')
Data=Data.sort_values('top',ascending=True)

Data.to_excel('Data.xls')
```