#Website.py

#coding:utf-8

#python2.7

import sys

reload(sys)

sys.setdefaultencoding("utf-8")

from flask import Flask,render\_template,url\_for,redirect,request,flash,jsonify,Response

import flask

from flask\_login import LoginManager, login\_required, login\_user, logout\_user, UserMixin,current\_user,AnonymousUserMixin

from flask\_sqlalchemy import SQLAlchemy

from flask\_wtf import FlaskForm

from wtforms import StringField,PasswordField,SubmitField

from wtforms.validators import DataRequired

from flask\_bootstrap import Bootstrap

import os

import numpy

import json

import csv

app = Flask(\_\_name\_\_)

bootstrap = Bootstrap(app)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///data.sqlite'

app.config['SQLALCHEMY\_COMMIT\_ON\_TEARDOWN'] = True

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS']= True

APP\_ROOT = os.path.dirname(os.path.abspath(\_\_file\_\_))

APP\_STATIC\_DATA = os.path.join(APP\_ROOT, 'static/data')

APP\_STATIC\_TEMPLATE = os.path.join(APP\_ROOT, 'templates')

db = SQLAlchemy(app)

WTF\_CSRF\_ENABLED = False

WTF\_CSRF\_CHECK\_DEFAULT = False

#user models

class User(UserMixin, db.Model):

\_\_tablename\_\_ = 'users'

id = db.Column(db.Integer, primary\_key=True, autoincrement=True)

username = db.Column(db.String(64), unique=True)

password = db.Column(db.String(64))

def \_\_init\_\_(self, username, password):

self.username = username

self.password = password

def \_\_repr\_\_(self):

"""Define the string format for instance of User."""

return "<Model User `{}`>".format(self.username)

def is\_authenticated(self):

"""Check the user whether logged in."""

# Check the User's instance whether Class AnonymousUserMixin's instance.

if isinstance(self, AnonymousUserMixin):

return False

else:

return True

def is\_active(self):

"""Check the user whether pass the activation process."""

return True

def is\_anonymous(self):

"""Check the user's login status whether is anonymous."""

if isinstance(self, AnonymousUserMixin):

return True

else:

return False

def get\_id(self):

"""Get the user's uuid from database."""

return unicode(self.id)

class Total\_data(db.Model):

\_\_tablename\_\_ = 'datas'

id = db.Column(db.Integer, primary\_key=True, autoincrement=True)

sentence = db.Column(db.String(255))

mark\_data = db.Column(db.String(255),default = None)

class U\_D(db.Model):

id = db.Column(db.Integer, primary\_key=True,autoincrement=True)

user\_id = db.Column(db.Integer, db.ForeignKey("users.id"))

user\_name = db.Column(db.Integer, db.ForeignKey("users.username"))

data\_id = db.Column(db.Integer, db.ForeignKey("datas.id"))

# U\_D = db.Table('User\_Data',

# db.Column('user\_id', db.Integer, db.ForeignKey('users.id')),

# db.Column('data\_id', db.Integer, db.ForeignKey('datas.id'))

# )

#login through Form

class LoginForm(FlaskForm):

#the form whose name=Username is the username

username = StringField('Usermame', [DataRequired()])

password = PasswordField('Password', [DataRequired()])

submit = SubmitField('Log In')

class SignForm(FlaskForm):

username = StringField('Usermame', [DataRequired()])

password = PasswordField('Password', [DataRequired()])

submit = SubmitField('Sign up')

db.create\_all()

# # user\_1 = User('wd','abc123')

# db.session.add(user\_1)

# db.session.commit()

#flask-login

app.secret\_key = 's12saf123f'

login\_manager = LoginManager()

login\_manager.session\_protection = 'strong'

#when you are not logged in, the page will redirect to /login to let you log in

login\_manager.login\_view = 'login'

login\_manager.login\_message = 'Please log in'

login\_manager.login\_message\_category = "info"

login\_manager.init\_app(app)

@login\_manager.user\_loader

def load\_user(user\_id):

return db.session.query(User).filter(User.id == user\_id).first()

#Add data

@app.route('/themethodtoadddatas')

def add\_data():

# test data1

test\_data1 = Total\_data(sentence = '很糟糕的购物体验,明明在使用范围内的优惠券不让用,说好返券,联系好多次,都不给解决,典型的黑店'.decode('utf-8'))

db.session.add(test\_data1)

db.session.commit()

html\_sentence = '数据成功添加'

return render\_template('hello\_world.html',user = current\_user, html\_sentence = html\_sentence)

@app.route('/')

def hello\_world():

unmarked\_tuple = Total\_data.query.filter(Total\_data.mark\_data == None).first()

if unmarked\_tuple == None:

return redirect(url\_for('complete'))

post\_sentece = '存在评论未被标注'

return render\_template('hello\_world.html',user = current\_user, html\_sentence = post\_sentece)

@app.route('/signup',methods=['POST','GET'])

def signup():

form = SignForm()

if flask.request.method == "GET" :

return render\_template('signup.html', form=form)

else:

#get input from user

name = form.username.data

password = form.password.data

search\_user = db.session.query(User).filter(User.username==name).all()

if search\_user:

flash ('用户名已被注册')

else:

search\_user = User(username = name, password = password)

db.session.add(search\_user)

db.session.commit()

flash ('注册成功')

return redirect(url\_for('login'))

return render\_template('signup.html', form=form)

@app.route("/login",methods=['POST','GET'])

def login():

form = LoginForm()

if form.validate\_on\_submit():

user = db.session.query(User).filter(User.username == form.username.data).first()

if user is not None and user.password==form.password.data:

login\_user(user)

return redirect(url\_for('hello\_world'))

flash('用户名和密码不匹配！')

return render\_template('login.html', form=form)

@app.route('/logout',methods=['POST','GET'])

@login\_required

def logout():

logout\_user()

flash('成功登出')

return redirect(url\_for('login'))

#test method

@app.route("/test", methods=["GET","POST"])

@login\_required

def test():

if flask.request.method == "POST":

mark\_Option = request.values.getlist("markword")

for t in mark\_Option:

if t == 'A':

tempMark = 0

if t == 'B':

tempMark = 1

if t == 'C':

tempMark = 2

if t == 'D':

tempMark = 3

marked\_data = db.session.query(Total\_data).filter(Total\_data.id == request.values.get("sentenceid")).first()

marked\_data.mark\_data = tempMark

u\_d = U\_D()

u\_d.user\_id = current\_user.id

u\_d.user\_name = current\_user.username

u\_d.data\_id = request.values.get("sentenceid") #don't have test data

db.session.add(u\_d)

db.session.commit()

return redirect(url\_for("test"))

else:

unmarked\_tuple = Total\_data.query.filter(Total\_data.mark\_data == None).first()

print(unmarked\_tuple)

if unmarked\_tuple == None:

return redirect(url\_for('complete'))

else:

unmarked\_sentence = unmarked\_tuple.sentence

print(type(unmarked\_sentence))

unmarked\_id = unmarked\_tuple.id

return render\_template("test.html",sentence = unmarked\_sentence,sentenceid = unmarked\_id)

return render\_template('test.html')

#complete template

@app.route("/complete")

@login\_required

def complete():

complete\_sentence = '所有评论均被标注'

return render\_template('hello\_world.html',user = current\_user,html\_sentence = complete\_sentence)

filename=''

@app.route("/statics")

@login\_required

def statics():

return render\_template('statics.html')

#render graph's html

@app.route("/statics/<filename>")

def statics\_graph(filename):

json\_alldata\_new = {}

filename\_list = filename.encode('utf-8').split('\_')

# print(filename\_list)

#商品品类名称

itemname = '\_'.join(filename\_list[1:-1])

if( itemname == 'phone'):

title\_itemname = '手机'

elif(itemname == 'laptop'):

title\_itemname = '笔记本电脑'

elif(itemname == 'air\_conditioner'):

title\_itemname = '洗衣机'

# print('\_'.join(filename\_list[1:-1]))

html\_file\_name\_prefix = filename\_list[0] + '\_' + filename\_list[-1]

#statics\_3~6

if (filename\_list[0] == 'statics' and int(filename\_list[-1]) >= 3 and int(filename\_list[-1]) <= 6):

templete\_filename = '\_'.join(filename\_list[0:-1]) + '\_3'

with open(os.path.join(APP\_STATIC\_DATA + '/' + itemname, templete\_filename + '.json'), 'r') as f:

json\_alldata = json.loads(f.read(), encoding='utf-8')

index = int(filename\_list[-1]) - 3

json\_alldata\_new = {}

json\_alldata\_new['count'] = json\_alldata['count\_list'][index]

json\_alldata\_new['brand'] = json\_alldata['brand\_list'][index]

json\_alldata\_new['sum'] = json\_alldata['sum\_list'][index]

json\_alldata\_new['factor'] = json\_alldata['factors']

#区分品牌

json\_alldata\_new['index'] = filename\_list[-1]

json\_alldata\_new['index2'] = str(int(filename\_list[-1]) + 4)

json\_alldata\_new['index3'] = str(int(filename\_list[-1]) + 9)

json\_alldata\_new['index4'] = str(int(filename\_list[-1]) + 15)

print(json\_alldata\_new)

#3~6公用模版

html\_file\_name\_prefix = filename\_list[0] + '\_' + '3'

return render\_template(html\_file\_name\_prefix + '.html', itemname=itemname, json\_alldata=json\_alldata\_new, title\_itemname = title\_itemname)

# statics\_7~10

if (filename\_list[0] == 'statics' and int(filename\_list[-1]) >= 7 and int(filename\_list[-1]) <= 10):

templete\_filename = '\_'.join(filename\_list[0:-1]) + '\_7'

with open(os.path.join(APP\_STATIC\_DATA + '/' + itemname, templete\_filename + '.json'), 'r') as f:

json\_alldata = json.loads(f.read(), encoding='utf-8')

# print(json\_alldata)

index = int(filename\_list[-1]) - 7

json\_alldata\_new = {}

json\_alldata\_new['count\_label2'] = json\_alldata['label2\_list\_list'][index]

json\_alldata\_new['count\_label3'] = json\_alldata['label3\_list\_list'][index]

json\_alldata\_new['count\_label1'] = json\_alldata['label1\_list\_list'][index]

json\_alldata\_new['count\_label0'] = json\_alldata['label0\_list\_list'][index]

json\_alldata\_new['count\_p\_label2'] = json\_alldata['label2\_p\_list\_list'][index]

json\_alldata\_new['count\_p\_label3'] = json\_alldata['label3\_p\_list\_list'][index]

json\_alldata\_new['count\_p\_label1'] = json\_alldata['label1\_p\_list\_list'][index]

json\_alldata\_new['count\_p\_label0'] = json\_alldata['label0\_p\_list\_list'][index]

json\_alldata\_new['brand'] = json\_alldata['brand\_list'][index]

json\_alldata\_new['product\_list'] = json\_alldata['product\_list\_list'][index]

print(json\_alldata\_new['product\_list'])

json\_alldata\_new['max'] = json\_alldata['max\_list'][index]

json\_alldata\_new['factor'] = json\_alldata['factors']

# 区分品牌

json\_alldata\_new['index0'] = str(int(filename\_list[-1]) - 4)

json\_alldata\_new['index'] = filename\_list[-1]

json\_alldata\_new['index2'] = str(int(filename\_list[-1]) + 5)

json\_alldata\_new['index3'] = str(int(filename\_list[-1]) + 11)

# print(json\_alldata\_new)

# 7~10公用模版

html\_file\_name\_prefix = filename\_list[0] + '\_' + '7'

return render\_template(html\_file\_name\_prefix + '.html', itemname=itemname, json\_alldata=json\_alldata\_new, title\_itemname = title\_itemname)

# statics\_11

if (filename\_list[0] == 'statics' and int(filename\_list[-1]) == 11):

with open(os.path.join(APP\_STATIC\_DATA + '/' + itemname, filename + '.json'), 'r') as f:

json\_alldata = json.loads(f.read(), encoding='utf-8')

json\_alldata\_new = {}

json\_alldata\_new['user\_list'] = json\_alldata['user\_list']

for i in range(len(json\_alldata['user\_list'])):

json\_alldata\_new['count\_month\_' + str(i)] = json\_alldata['count\_month\_' + str(i)]

json\_alldata\_new['result\_date'] = json\_alldata['result\_date']

json\_alldata\_new['result\_date\_count'] = json\_alldata['result\_date\_count']

json\_alldata\_new['result\_hour\_count'] = json\_alldata['result\_hour\_count']

json\_alldata\_new['result\_hour\_count\_total'] = json\_alldata['result\_hour\_count\_total']

print(json\_alldata\_new)

# 3~6公用模版

return render\_template(html\_file\_name\_prefix + '.html', itemname=itemname, json\_alldata=json\_alldata\_new, title\_itemname = title\_itemname)

# statics\_12~15

if (filename\_list[0] == 'statics' and int(filename\_list[-1]) >= 12 and int(filename\_list[-1]) <= 15):

templete\_filename = '\_'.join(filename\_list[0:-1]) + '\_12'

with open(os.path.join(APP\_STATIC\_DATA + '/' + itemname, templete\_filename + '.json'), 'r') as f:

json\_alldata = json.loads(f.read(), encoding='utf-8')

# print(json\_alldata)

index = int(filename\_list[-1]) - 12

json\_alldata\_new = {}

json\_alldata\_new['brand'] = json\_alldata['brand\_list'][index]

brand\_number = len(json\_alldata['brand\_list'])

product\_number = len(json\_alldata['product\_list'][0])

json\_alldata\_new['product\_list'] = json\_alldata['product\_list'][index]

json\_alldata\_new['all\_date\_list'] = json\_alldata['all\_date\_list'][index]

json\_alldata\_new['all\_count\_list'] = json\_alldata['all\_count\_list'][index]

for i in range(product\_number):

json\_alldata\_new['date\_list\_'+ str(i)] = json\_alldata['date\_list'][product\_number\*index + i]

json\_alldata\_new['count\_list\_'+ str(i)] = json\_alldata['count\_list'][product\_number\*index + i]

# json\_alldata\_new['max'] = json\_alldata['max\_list'][index]

# 区分品牌

json\_alldata\_new['index0'] = str(int(filename\_list[-1]) - 9)

json\_alldata\_new['index'] = filename\_list[-1]

json\_alldata\_new['index2'] = str(int(filename\_list[-1]) - 5)

json\_alldata\_new['index3'] = str(int(filename\_list[-1]) + 6)

# print(json\_alldata\_new)

# 7~10公用模版

html\_file\_name\_prefix = filename\_list[0] + '\_' + '12'

return render\_template(html\_file\_name\_prefix + '.html', itemname=itemname, json\_alldata=json\_alldata\_new, title\_itemname = title\_itemname)

# statics\_18~21

if (filename\_list[0] == 'statics' and int(filename\_list[-1]) >= 18 and int(filename\_list[-1]) <= 21):

templete\_filename = '\_'.join(filename\_list[0:-1]) + '\_18'

with open(os.path.join(APP\_STATIC\_DATA + '/' + itemname, templete\_filename + '.json'), 'r') as f:

json\_alldata = json.loads(f.read(), encoding='utf-8')

# print(json\_alldata)

index = int(filename\_list[-1]) - 18

json\_alldata\_new = {}

json\_alldata\_new['brand'] = json\_alldata['brand\_list'][index]

json\_alldata\_new['user\_list'] = json\_alldata['user\_list\_list'][index]

json\_alldata\_new['count\_list'] = json\_alldata['user\_count\_list'][index]

json\_alldata\_new['totalcount'] = json\_alldata['user\_totalcount\_list'][index]

# json\_alldata\_new['max'] = json\_alldata['max\_list'][index]

# 区分品牌

json\_alldata\_new['index0'] = str(int(filename\_list[-1]) - 15)

json\_alldata\_new['index'] = filename\_list[-1]

json\_alldata\_new['index2'] = str(int(filename\_list[-1]) - 11)

json\_alldata\_new['index3'] = str(int(filename\_list[-1]) - 6)

# print(json\_alldata\_new)

html\_file\_name\_prefix = filename\_list[0] + '\_' + '18'

return render\_template(html\_file\_name\_prefix + '.html', itemname=itemname, json\_alldata=json\_alldata\_new, title\_itemname = title\_itemname)

if (filename\_list[0] == 'statics' and int(filename\_list[-1]) == 26):

with open(os.path.join(APP\_STATIC\_DATA + '/' + itemname, 'statics\_' + itemname + '\_tree\_a\_comments.json'),'r') as f:

json\_alldata = json.loads(f.read(), encoding='utf-8')

json\_alldata\_new = json\_alldata

print(os.path.join(APP\_STATIC\_DATA + '/' + itemname, 'statics\_' + itemname + '\_tree\_a\_comments.json'))

return render\_template(html\_file\_name\_prefix + '.html', itemname=itemname, title\_itemname = title\_itemname,json\_alldata = json\_alldata\_new)

if (filename\_list[0] == 'statics' and int(filename\_list[-1]) == 27):

with open(os.path.join(APP\_STATIC\_DATA + '/' + itemname, 'statics\_' + itemname + '\_tree\_n\_comments.json'),'r') as f:

json\_alldata = json.loads(f.read(), encoding='utf-8')

json\_alldata\_new = json\_alldata

print(os.path.join(APP\_STATIC\_DATA + '/' + itemname, 'statics\_' + itemname + '\_tree\_n\_comments.json'))

return render\_template(html\_file\_name\_prefix + '.html', itemname=itemname, title\_itemname = title\_itemname,json\_alldata = json\_alldata\_new)

#除箱线图外数据均在这里处理

with open(os.path.join(APP\_STATIC\_DATA + '/' + itemname, filename + '.json'), 'r') as f:

json\_alldata = json.loads(f.read(),encoding='utf-8')

json\_alldata\_new = json\_alldata

print('json\_alldata')

print(filename)

print(json\_alldata\_new)

return render\_template(html\_file\_name\_prefix+'.html',itemname = itemname, json\_alldata=json\_alldata\_new, title\_itemname = title\_itemname)

@app.route("/getData/<filename>", methods=["GET"])

@login\_required

def getData(filename):

filename\_list = filename.encode('utf-8').split('\_')

print(filename\_list)

itemname = '\_'.join(filename\_list[1:-2])

print(itemname)

with open(os.path.join(APP\_STATIC\_DATA + '/' + itemname, filename + '.json'), 'r') as f:

json\_alldata = json.loads(f.read(), encoding='utf-8')

return jsonify(json\_alldata)

@app.route("/getIcon/<filename>")

def getIconPath(filename):

iconPath = os.path.join(APP\_STATIC\_TEMPLATE + '/icon', filename)

print(iconPath)

icon = file(iconPath)

return iconPath

resp = Response(icon,mimetype="image/png")

return resp

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

# step\_4\_statics12-15.py

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

'''

目的：从jd\_xx\_time.csv中统计各种品牌差评的数量最多的四个品牌之中的最多的四个品种按照日期统计差评数

格式：

brand\_list=[数量最多的四个品牌]

product\_list\_foreachbrands = 【【差评数量第一的品牌下差评数量前四的品牌id】.....】

outfile\_dic['brand\_list'] = brand\_list

outfile\_dic['product\_list'] = product\_list\_foreachbrands

outfile\_dic['date\_list'].append(product\_time\_date\_list)

outfile\_dic['count\_list'].append(product\_time\_count\_list)

outfile\_dic['all\_date\_list'].append(all\_date\_maplist)

outfile\_dic['all\_count\_list'].append(all\_count\_maplist)

'''

import csv

import time

def is\_date(str):

'''判断是否是一个有效的日期字符串'''

try:

time.strptime(str, "%Y-%m-%d")

return True

except:

return False

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics","itemname")

file\_prefix = config.get("statics","file\_prefix")

file\_path\_prefix = config.get("statics","file\_path\_prefix")

csv\_name = config.get("statics","csv\_name")

csv\_time\_name = config.get("statics","csv\_time\_name")

result\_map={}

result\_brand = []

result\_count = []

flag=0#标志第一行

i=0

j=0

with open(csv\_name,'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

#跳过第一行

if(flag==0):

flag+=1

# print(line)

continue

if(line[0] in result\_map.keys()):

result\_map[line[0]]+=int(line[6])

else:

result\_map[line[0]]=0

result\_map[line[0]]+=int(line[6])

sorted\_result\_list = sorted(result\_map.items(), key = lambda x: x[1], reverse = True)

# print(sorted\_result\_list)

sorted\_result\_list\_brand = []

sorted\_result\_list\_count = []

for items in sorted\_result\_list:

sorted\_result\_list\_brand.append(items[0])

sorted\_result\_list\_count.append(items[1])

#验证两列表长度是否相同

print(len(sorted\_result\_list\_brand)==len(sorted\_result\_list\_count))

#获得数量最多的四个品牌

brand\_list = sorted\_result\_list\_brand[0:4]

#各个品牌下差评数量最多的四个产品

product\_list\_foreachbrands=[]

# print(brand\_list)

for i in range(4):

flag = 0 # 标志着第一行，重新置零

product\_list=[]

# product\_list\_map\_1={'referenceId':sum}

product\_list\_map\_1 = {}

product\_list\_map\_1\_sorted\_list = []

# product\_list\_map\_2={'referenceId':[label2,label3,label1,label0]}

product\_list\_map\_1\_sorted\_list\_mostfourproduct=[]

with open(csv\_name,'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

temp\_list = []

#跳过第一行

if(flag==0):

flag+=1

# print(line)

continue

if(line[0] == brand\_list[i]):

if(line[1] in product\_list\_map\_1.keys()):

product\_list\_map\_1[line[1]] += int(line[6])

else:

product\_list\_map\_1[line[1]] =0

product\_list\_map\_1[line[1]] += int(line[6])

# 对product\_list\_map\_1排序

product\_list\_map\_1\_sorted\_list = sorted(product\_list\_map\_1.items(), key=lambda x: x[1], reverse=True)

# print(product\_list\_map\_1\_sorted\_list)

for j in range(4):

product\_list\_map\_1\_sorted\_list\_mostfourproduct.append(product\_list\_map\_1\_sorted\_list[j][0])

product\_list\_foreachbrands.append(product\_list\_map\_1\_sorted\_list\_mostfourproduct)

# print(brand\_list[i])

# print(product\_list\_map\_1\_sorted\_list\_mostfourproduct)

# print(product\_list\_foreachbrands)

lines=[]

i=0

j=0

import json

#写入文件

outfile\_path\_prefix = config.get("statics","outfile\_path\_prefix")

outfile\_name = 'statics\_'+ itemname + '\_12.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic ={}

outfile\_dic['brand\_list'] = []

outfile\_dic['date\_list'] = []

outfile\_dic['count\_list'] = []

outfile\_dic['all\_date\_list'] = []

outfile\_dic['all\_count\_list'] = []

for i in range(4):

brand\_output = brand\_list[i]

all\_list\_map = {}

all\_date\_maplist = []

all\_count\_maplist = []

for j in range(4):

product\_time\_count\_map = {}

product\_time\_count\_map\_list = []

product\_time\_date\_list = []

product\_time\_count\_list = []

product\_output = product\_list\_foreachbrands[i][j]

print(brand\_output)

print(product\_output)

with open(csv\_time\_name, 'r') as csvfile2:

for line in csvfile2:

line=line.split(',')

if(flag==0):

continue

if(line[1] == product\_output and is\_date(line[3])):

if(line[3] in product\_time\_count\_map.keys()):

product\_time\_count\_map[line[3]] += 1

else:

product\_time\_count\_map[line[3]] = 1

if(line[3] in all\_list\_map.keys()):

all\_list\_map[line[3]] += 1

else:

all\_list\_map[line[3]] = 1

product\_time\_count\_map\_list = sorted(product\_time\_count\_map.items(), key=lambda x:x[0], reverse=False)

for product\_count\_item in product\_time\_count\_map\_list:

product\_time\_date\_list.append(product\_count\_item[0])

product\_time\_count\_list.append(product\_count\_item[1])

print(product\_time\_count\_map\_list)

outfile\_dic['date\_list'].append(product\_time\_date\_list)

outfile\_dic['count\_list'].append(product\_time\_count\_list)

all\_list\_map\_list = sorted(all\_list\_map.items(), key=lambda x: x[0], reverse=False)

for all\_list\_map\_list\_item in all\_list\_map\_list:

all\_date\_maplist.append(all\_list\_map\_list\_item[0])

all\_count\_maplist.append(all\_list\_map\_list\_item[1])

outfile\_dic['all\_date\_list'].append(all\_date\_maplist)

outfile\_dic['all\_count\_list'].append(all\_count\_maplist)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

outfile\_dic['brand\_list'] = brand\_list

outfile\_dic['product\_list'] = product\_list\_foreachbrands

print(product\_list\_foreachbrands)

print(brand\_list)

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

# step\_4\_statics16.py

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

'''

目的：获得某原因下各类用户的占比情况

输入文件 jd\_ac\_time.csv

list中数量代表用户类型的顺序：['PLUS会员','PLUS会员[试用]','注册会员','企业会员','铜牌会员','银牌会员',"金牌会员","钻石会员"]

label： 0 1 2 3

输出：4个factors下各个用户的数量的list

userlever\_list=[排列的用户类型顺序]

xdic\_list\_list[i]

outfile\_dic['factor'] = factors

outfile\_dic['userlevel\_list'] = userlever\_list

outfile\_dic['dic\_list\_list'] = dic\_list\_list

'''

import json

import os

import re

import csv

import codecs

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics","itemname")

file\_prefix = config.get("statics","file\_prefix")

file\_path\_prefix = config.get("statics","file\_path\_prefix")

csv\_time\_name = config.get("statics","csv\_time\_name")

factors = ['物流原因','产品原因','售后原因','营销原因']

timefile = csv\_time\_name

#输出的csv文件使用带bom的utf-8编码

csvfile = codecs.open(file\_path\_prefix+itemname+'brandUserFactors.csv','w+','utf-8-sig')

writer = csv.writer(csvfile)

writer.writerow(['userLevelName','label','count'])

initnumber=0

userlist\_label0=[initnumber]\*8

userlist\_label1=[initnumber]\*8

userlist\_label2=[initnumber]\*8

userlist\_label3=[initnumber]\*8

dic\_list\_list=[]

def countUser(index,label):

if(label == 0):

userlist\_label0[index]+=1

elif(label==1):

userlist\_label1[index]+=1

elif(label==2):

userlist\_label2[index]+=1

elif(label==3):

userlist\_label3[index]+=1

else:

print("Unrecognized Label!!")

def handleDicToList(userdict):

temp\_list = []

keys=['name','value']

for key, value in userdict.items():

values=[key,value]

temp\_dict=dict(zip(keys,values))

temp\_list.append(temp\_dict)

dic\_list\_list.append(temp\_list)

with open(timefile, 'r') as f:

# s\_3 = json.load(f)

reader\_timefile = csv.DictReader(f)

rows\_timefile = [row for row in reader\_timefile]

userLeverName\_list = ['PLUS会员','PLUS会员[试用]','注册会员','企业会员','铜牌会员','银牌会员',"金牌会员","钻石会员"]

# print userLeverName\_list.index('注册会员') #返回 注册会员 在列表中的下标

count = [([0]\*4) for i in range(len(userLeverName\_list))]

count\_echarts = []

for line in rows\_timefile:

line\_userLeverName = line['userLevelName']

if(line['label'] == 'label'):

continue

line\_label = int(line['label'])

level\_index = userLeverName\_list.index(line\_userLeverName)

countUser(level\_index,line\_label)

#echarts绘制饼图需要的dict格式

dic\_label0=dict(zip(userLeverName\_list,userlist\_label0))

dic\_label1=dict(zip(userLeverName\_list,userlist\_label1))

dic\_label2=dict(zip(userLeverName\_list,userlist\_label2))

dic\_label3=dict(zip(userLeverName\_list,userlist\_label3))

handleDicToList(dic\_label0)

handleDicToList(dic\_label1)

handleDicToList(dic\_label2)

handleDicToList(dic\_label3)

userlever\_list = []

for items in dic\_label0:

userlever\_list.append(items)

for i in range(4):

print(factors[i])

print(userlever\_list)

print(dic\_list\_list[i])

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

import json

#写入文件

outfile\_path\_prefix = config.get("statics","outfile\_path\_prefix")

outfile\_name = 'statics\_'+ itemname + '\_16.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic ={}

outfile\_dic['factor'] = factors

outfile\_dic['userlevel\_list'] = userlever\_listoutfile\_dic['dic\_list\_list'] = dic\_list\_list#with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f: f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))# step\_4\_statics17.py

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

'''

目的：从jd\_xx\_time.csv中按照用户类型统计 [金牌会员 银牌会员 钻石会员 PLUS会员 铜牌会员 PLUS会员[试用] 企业会员 注册会员]的数量

顺序：['售后原因','营销原因','产品原因','物流原因']

格式：

user\_list = ['金牌会员', '银牌会员', '钻石会员', 'PLUS会员', '铜牌会员', 'PLUS会员[试用]', '企业会员', '注册会员']

user\_count=[各种类型会员差评的统计]

outfile\_dic['user\_list'] = user\_list

outfile\_dic['user\_count'] = result\_hour\_count

outfile\_dic['sum'] = count\_sum

'''

import csv

import time

def is\_date(str):

'''判断是否是一个有效的日期字符串'''

try:

time.strptime(str, "%Y-%m-%d")

return True

except:

return False

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics", "itemname")

file\_prefix = config.get("statics", "file\_prefix")

file\_path\_prefix = config.get("statics", "file\_path\_prefix")

csv\_time\_name = config.get("statics", "csv\_time\_name")

user\_list = ['金牌会员', '银牌会员', '钻石会员', 'PLUS会员',

'铜牌会员', 'PLUS会员[试用]', '企业会员', '注册会员']

result\_hour\_count = [0 for i in range(len(user\_list))]

flag = 0 # 标志第一行

count\_sum = 0

with open(csv\_time\_name, 'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

# 跳过第一行

if(flag == 0):

flag = 1

continue

else:

if(line[2] == 'userLevelName'):

continue

result\_hour\_count\_index = user\_list.index(line[2])

result\_hour\_count[result\_hour\_count\_index] += 1

count\_sum += 1

print(user\_list)

print(result\_hour\_count)

import json

# 写入文件

outfile\_path\_prefix = config.get("statics", "outfile\_path\_prefix")

outfile\_name = 'statics\_' + itemname + '\_17.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic = {}

outfile\_dic['user\_list'] = user\_list

outfile\_dic['user\_count'] = result\_hour\_count

outfile\_dic['sum'] = count\_sum

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

# step\_4\_statics18-21.py

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

'''

目的：先从从jd\_xx.csv中统计各种品牌差评的数量最多的四个品牌收入 brand\_list=[]

从jd\_xx\_time.csv中分别对那四个品牌按照用户类型统计 [金牌会员 银牌会员 钻石会员 PLUS会员 铜牌会员 PLUS会员[试用] 企业会员 注册会员]的数量

格式：

brand\_output

user\_list\_output = [] 按count降序排列

user\_count=[各种类型会员差评的统计]降序排列

outfile\_dic['user\_list\_list'].append(user\_list\_output)

outfile\_dic['user\_count\_list'].append(user\_count)

outfile\_dic['brand\_list'] = brand\_list

outfile\_dic['user\_totalcount\_list'].append(count\_temp)

'''

import csv

import time

def is\_date(str):

'''判断是否是一个有效的日期字符串'''

try:

time.strptime(str, "%Y-%m-%d")

return True

except:

return False

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics", "itemname")

file\_prefix = config.get("statics", "file\_prefix")

file\_path\_prefix = config.get("statics", "file\_path\_prefix")

csv\_time\_name = config.get("statics", "csv\_time\_name")

csv\_name = config.get("statics", "csv\_name")

user\_list = ['金牌会员', '银牌会员', '钻石会员', 'PLUS会员',

'铜牌会员', 'PLUS会员[试用]', '企业会员', '注册会员']

'''统计数量排名前四的品牌'''

flag = 0 # 标志第一行

result\_map = {}

i = 0

count\_list = []

with open(csv\_name, 'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

# 跳过第一行

if(flag == 0):

flag += 1

# print(line)

continue

if(line[0] in result\_map.keys()):

result\_map[line[0]] += int(line[6])

else:

result\_map[line[0]] = int(line[6])

sorted\_result\_list = sorted(

result\_map.items(), key=lambda x: x[1], reverse=True)

sorted\_result\_list\_brand = []

for items in sorted\_result\_list:

sorted\_result\_list\_brand.append(items[0])

# 获得数量最多的四个品牌

brand\_list = sorted\_result\_list\_brand[0:4]

print(brand\_list)

import json

# 写入文件

outfile\_path\_prefix = config.get("statics", "outfile\_path\_prefix")

outfile\_name = 'statics\_' + itemname + '\_18.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic = {}

outfile\_dic['user\_list\_list'] = []

outfile\_dic['user\_count\_list'] = []

outfile\_dic['user\_totalcount\_list'] = []

'''对四个品牌进行统计'''

count = 0

flag = 0

for brand in brand\_list:

user\_count\_map = {}

brand\_output = brand

user\_count = []

user\_list\_output = []

count\_temp = 0

with open(csv\_time\_name, 'r') as csvfile2:

reader = csv.reader(csvfile2)

for line in reader:

# 跳过第一行

if(flag == 0):

flag = 1

# print(line)

continue

if(brand\_output == line[0]):

count\_temp += 1

if(line[2] in user\_count\_map.keys()):

user\_count\_map[line[2]] += 1

else:

user\_count\_map[line[2]] = 1

user\_count\_map\_list = sorted(

user\_count\_map.items(), key=lambda x: x[1], reverse=True)

flag = 0

# 装入列表输出

for levelcount in user\_count\_map\_list:

user\_list\_output.append(levelcount[0])

user\_count.append(levelcount[1])

print(count\_temp)

print(brand\_output)

print(user\_list\_output)

print(user\_count)

outfile\_dic['user\_list\_list'].append(user\_list\_output)

outfile\_dic['user\_count\_list'].append(user\_count)

outfile\_dic['user\_totalcount\_list'].append(count\_temp)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

outfile\_dic['brand\_list'] = brand\_list

#

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

# step\_4\_statics22-23.py

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

# 运行环境python3

import pickle as pke

import json

#结果输出成json转入文件

'''

用户倾向 作五图

输出

键：map\_key\_value\_i=【】

map\_value\_i=【】

i取1，2，3，4，5

23:将score5 4 3 2 1分别以一行的形式写入statics\_ac\_23\_1.txt

'''

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics","itemname")

file\_prefix = config.get("statics","file\_prefix")

file\_path\_prefix = config.get("statics","file\_path\_prefix")

#给出下表数字列表，返回比该数字小的最大列表内数字

def roundthenumber(xxisNumber,number):

preNumeber = -1

for i in range(len(xxisNumber)):

# print(i)

if (number < xxisNumber[i]):

return preNumeber

elif(number == 1):

return xxisNumber[len(xxisNumber)-1]

else:

preNumeber = xxisNumber[i]

continue

print('数字超出范围')

return

# 给出间隔数字，返回0到1的数字列表

def getTheXxisNumberList(gapNumber):

temp = -1

xxisNumber = []

while (temp<1):

if (temp < 0.00001 and temp > -0.0001):

xxisNumber.append(0)

temp += gapNumber

continue

xxisNumber.append(round(temp, 2))

temp += gapNumber

# 压入1.0

xxisNumber.append(1)

return xxisNumber

filepath = config.get("statics","dump\_filepath")

rf = open(filepath,'rb')

pmapsl = pke.load(rf)

rf.close()

#smapls={}

dlist=[]

count = 0

map\_value1={}

map\_value2={}

map\_value3={}

map\_value4={}

map\_value5={}

l1=[]

l2=[]

l3=[]

l4=[]

l5=[]

score\_5 = []

score\_4 = []

score\_3 = []

score\_2 = []

score\_1 = []

map\_name='map\_value'

# 数值区间统计间隔

gap\_number = 0.01

xxis\_number = getTheXxisNumberList(gap\_number)

print(xxis\_number)

# 初始化每个字典

for i in range(len(xxis\_number)):

map\_value1[xxis\_number[i]] = 0

map\_value2[xxis\_number[i]] = 0

map\_value3[xxis\_number[i]] = 0

map\_value4[xxis\_number[i]] = 0

map\_value5[xxis\_number[i]] = 0

for pid in pmapsl:

sl=pmapsl[pid]

# print('key'+str(pid))

# print('value'+str(sl))

for i in range(0,len(sl[0])):

# 获得每个id中每一对score和情绪倾向得分

s=sl[0][i]

l=sl[1][i]

l\_round = roundthenumber(xxis\_number,l)

# print(l)

# print(l\_round)

# print(l)

# print(l\_round)

# 定位score数值

count+=1

if (s==5):

temp = map\_value5

score\_5.append(l)

# print('map\_value5')

l5.append(l\_round)

elif (s==4):

temp = map\_value4

l4.append(l\_round)

score\_4.append(l)

# print('map\_value4')

elif (s==3):

temp = map\_value3

l3.append(l\_round)

score\_3.append(l)

# print('map\_value3')

elif (s==2):

temp = map\_value2

l2.append(l)

score\_2.append(l)

# print('map\_value2')

elif (s==1):

temp = map\_value1

l1.append(l)

score\_1.append(l)

# print('map\_value1')

else:

print('用户打分异常')

temp[l\_round] = temp.get(l\_round,0) + 1

# print(temp)

map\_value5\_sorted = sorted(map\_value5.items(), key=lambda e:e[0], reverse=False)

map\_value4\_sorted = sorted(map\_value4.items(), key=lambda e:e[0], reverse=False)

map\_value3\_sorted = sorted(map\_value3.items(), key=lambda e:e[0], reverse=False)

map\_value2\_sorted = sorted(map\_value2.items(), key=lambda e:e[0], reverse=False)

map\_value1\_sorted = sorted(map\_value1.items(), key=lambda e:e[0], reverse=False)

map\_key\_value\_5=[]

map\_value\_5=[]

map\_key\_value\_4=[]

map\_value\_4=[]

map\_key\_value\_3=[]

map\_value\_3=[]

map\_key\_value\_2=[]

map\_value\_2=[]

map\_key\_value\_1=[]

map\_value\_1=[]

score=5

print(score)

for item in map\_value5\_sorted:

map\_key\_value\_5.append(item[0])

map\_value\_5.append(item[1])

print('键')

print(map\_key\_value\_5)

print('值')

print(map\_value\_5)

score=4

print(score)

for item in map\_value4\_sorted:

map\_key\_value\_4.append(item[0])

map\_value\_4.append(item[1])

print('键')

print(map\_key\_value\_4)

print('值')

print(map\_value\_4)

# print(l4)

score=3

print(score)

for item in map\_value3\_sorted:

map\_key\_value\_3.append(item[0])

map\_value\_3.append(item[1])

print('键')

print(map\_key\_value\_3)

print('值')

print(map\_value\_3)

# print(l3)

score=2

print(score)

for item in map\_value2\_sorted:

map\_key\_value\_2.append(item[0])

map\_value\_2.append(item[1])

print('键')

print(map\_key\_value\_2)

print('值')

print(map\_value\_2)

# print(l2)

score=1

print(score)

for item in map\_value1\_sorted:

map\_key\_value\_1.append(item[0])

map\_value\_1.append(item[1])

print('键')

print(map\_key\_value\_1)

print('值')

print(map\_value\_1)

# print(l1)

import json

#写入文件

outfile\_path\_prefix =file\_path\_prefix + itemname + '/'

outfile\_name = 'statics\_'+ itemname + '\_23.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic ={}

outfile\_dic['map\_key'] = map\_key\_value\_1

outfile\_dic['score\_5'] = map\_value\_5

outfile\_dic['score\_3'] = map\_value\_3

outfile\_dic['score\_2'] = map\_value\_2

outfile\_dic['score\_4'] = map\_value\_4

outfile\_dic['score\_1'] = map\_value\_1

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

outfile\_path\_prefix = config.get("statics","outfile\_path\_prefix")

outfile\_name = 'statics\_'+ itemname + '\_22.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name)

print(outfile\_name\_withprefix)

print('开始写处理txt')

temp\_dic ={}

outfile\_dic ={}

temp\_dic['score'] = []

temp\_dic['score'].append(score\_1)

temp\_dic['score'].append(score\_2)

temp\_dic['score'].append(score\_3)

temp\_dic['score'].append(score\_4)

temp\_dic['score'].append(score\_5)

import pandas

outfile\_dic['boxData'] = []

outfile\_dic['axisData'] = []

outfile\_dic['outliers'] = []

for i in range(5):

outfile\_dic['axisData'].append(str(i))

x = pandas.Series(temp\_dic['score'][i])

x\_min = x.min()

x\_max = x.max()

print('low',x\_min)

print('max',x\_max)

Q1 = x.quantile(1/4)

Q2 = x.quantile(2/4)

Q3 = x.quantile(3/4)

IQR = Q3 - Q1

print('Q1',Q1)

print('Q2',Q2)

print('Q3',Q3)

print('IQR',IQR)

x\_low = Q1 - 1.5\*IQR

x\_upper = Q3 + 1.5\*IQR

if (x\_low<x\_min):

x\_low = x\_min

if (x\_upper>x\_max):

x\_upper = x\_max

print(x\_low)

print(x\_upper)

outfile\_dic['boxData'].append([x\_low,Q1,Q2,Q3,x\_max])

for dataitem in temp\_dic['score'][i]:

temp\_data = []

if(dataitem < x\_low or dataitem > x\_upper):

temp\_data = [dataitem,i]

outfile\_dic['outliers'].append(temp\_data)

print(outfile\_dic['boxData'])

print(outfile\_dic['axisData'])

print(outfile\_dic['outliers'])

# print('写txt文件')

# with open(outfile\_name\_withprefix, 'a', newline='', encoding='utf-8') as txt\_name\_file:

# for i in range(5):

# txt\_name\_file.write(','.join(str(data) for data in temp\_dic['score'][i]))

# txt\_name\_file.write('\n')

# print('done')

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, ensure\_ascii=False))

# step\_4\_statics24.py

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

import json

import os

import re

import csv

import codecs

'''

目的：获取行业的用户-因素矩阵以绘制热力图

输入文件 jd\_ac\_time.csv

['PLUS会员','PLUS会员[试用]','注册会员','企业会员','铜牌会员','银牌会员',"金牌会员","钻石会员"]

label： 0 1 2 3

输出：可以直接用于echarts绘图的list

x ： ['物流原因', '产品原因', '售后原因', '营销原因']

y : userLeverName\_list

图例需要一个比最大的数值大一些的数 maxround = round(max/100 + 1)\*100

'''

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics", "itemname")

file\_prefix = config.get("statics", "file\_prefix")

file\_path\_prefix = config.get("statics", "file\_path\_prefix")

csv\_time\_name = config.get("statics", "csv\_time\_name")

csv\_name = config.get("statics", "csv\_name")

label\_list = ['物流原因', '产品原因', '售后原因', '营销原因']

max = 0

with open(csv\_time\_name, 'r') as f:

# s\_3 = json.load(f)

reader\_timefile = csv.DictReader(f)

rows\_timefile = [row for row in reader\_timefile]

userLeverName\_list = ['PLUS会员', 'PLUS会员[试用]',

'注册会员', '企业会员', '铜牌会员', '银牌会员', "金牌会员", "钻石会员"]

# print userLeverName\_list.index('注册会员') #返回 注册会员 在列表中的下标

count = [([0] \* 4) for i in range(len(userLeverName\_list))]

count\_echarts = []

for line in rows\_timefile:

if(line['userLevelName'] == 'userLevelName'):

continue

line\_userLEvelName = line['userLevelName']

line\_label = line['label']

level\_index = userLeverName\_list.index(line\_userLEvelName)

count[level\_index][int(line\_label)] += 1

for i in range(len(userLeverName\_list)):

for j in range(4):

if(count[i][j] > max):

max = count[i][j]

Levelname = userLeverName\_list[i]

# 输出echarts热力图绘图需要的格式

count\_echarts.append([i, j, count[i][j]])

maxround = round(max / 100) \* 100

print(count\_echarts)

print(userLeverName\_list)

print(label\_list)

print(maxround)

import json

# 写入文件

outfile\_path\_prefix = config.get("statics", "outfile\_path\_prefix")

outfile\_name = 'statics\_' + itemname + '\_24.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic = {}

outfile\_dic['factor'] = label\_list

outfile\_dic['userlevel\_list'] = userLeverName\_list

outfile\_dic['data'] = count\_echarts

outfile\_dic['max'] = maxround

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

# step\_4\_statics25.py

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

coding = utf - 8

import json

import os

import re

import csv

import codecs

'''

目的：获得某类用户下各类原因的占比情况

输入文件 jd\_ac\_time.csv

list中数量代表用户类型的顺序：['PLUS会员','PLUS会员[试用]','注册会员','企业会员','铜牌会员','银牌会员',"金牌会员","钻石会员"]

label： 0 1 2 3

输出：8类用户下各个原因的数量的list

outfile\_dic['factor'] = factors

outfile\_dic['userlevel\_list'] = userLeverName\_list

outfile\_dic['result'] = label\_list

label\_list.append([])

'''

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics", "itemname")

file\_prefix = config.get("statics", "file\_prefix")

file\_path\_prefix = config.get("statics", "file\_path\_prefix")

csv\_time\_name = config.get("statics", "csv\_time\_name")

csv\_name = config.get("statics", "csv\_name")

# 输出的csv文件使用带bom的utf-8编码

def countUser(userlist, label):

if(label >= 0 and label <= 3):

userlist[label] += 1

else:

print("Unrecognized Label!!")

factors = ['物流原因', '产品原因', '售后原因', '营销原因']

with open(csv\_time\_name, 'r') as f:

# s\_3 = json.load(f)

reader\_timefile = csv.DictReader(f)

rows\_timefile = [row for row in reader\_timefile]

userLeverName\_list = ['PLUS会员', 'PLUS会员[试用]',

'注册会员', '企业会员', '铜牌会员', '银牌会员', "金牌会员", "钻石会员"]

# print userLeverName\_list.index('注册会员') #返回 注册会员 在列表中的下标

count = [([0] \* 4) for i in range(len(userLeverName\_list))]

count\_echarts = []

for line in rows\_timefile:

if(line['userLevelName'] == 'useLevelName' or line['label'] == 'label'):

continue

line\_userLeverName = line['userLevelName']

line\_label = int(line['label'])

level\_index = userLeverName\_list.index(line\_userLeverName)

countUser(count[level\_index], line\_label)

print('userlist\_Plus 会员')

for i in range(8):

temp\_sum = 0

for items in count[i]:

temp\_sum += items

for j in range(len(count[i])):

if(temp\_sum == 0):

count[i][j] == 0

continue

else:

count[i][j] = round(count[i][j] / temp\_sum, 4)

print(count[0])

print('userlist\_Plus 会员[试用]')

print(count[1])

print('userlist\_注册会员')

print(count[2])

print('userlist\_企业会员')

print(count[3])

print('userlist\_铜牌会员')

print(count[4])

print('userlist\_银牌会员')

print(count[5])

print('userlist\_金牌会员')

print(count[6])

print('userlist\_钻石会员')

print(count[7])

label\_list = []

for label in range(4):

label\_list.append([])

for countlist in count:

label\_list[label].append(countlist[label])

print(label\_list)

import json

# 写入文件

outfile\_path\_prefix = config.get("statics", "outfile\_path\_prefix")

outfile\_name = 'statics\_' + itemname + '\_25.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic = {}

outfile\_dic['factor'] = factors

outfile\_dic['userlevel\_list'] = userLeverName\_list

outfile\_dic['result'] = label\_list

#

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

# step\_4\_statics26\_sun.py

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

import json

import os

import jieba.analyse

import re

import csv

import codecs

'''

目的：获得树状图

mh\_S

输出：8类用户下各个原因的数量的list

注意：将多个txt合成一个文件

'''

itemname = 'phone'

itemname\_zh = '手机'

file\_prefix = '/Users/diw/PycharmProjects/jd\_negatives/' + itemname + '/'

file\_path\_prefix = '/users/diw/Desktop/'

csv\_name = file\_path\_prefix + itemname + '.csv'

csv\_name\_time = file\_path\_prefix + itemname + '\_time.csv'

# 输出的csv文件使用带bom的utf-8编码

file\_folder = file\_path\_prefix + '180326\_phone\_词频分类排序/positiveRejectWeight/'

firstfolder\_List = os.listdir(file\_folder)

count\_max = 6

factor = ['物流', '产品', '售后', '营销']

first\_dic = {}

first\_dic['name'] = itemname\_zh + '差评'

first\_dic\_list = []

raw\_comments = []

fileList = os.listdir(file\_prefix)

times = 3

extract\_comments = {}

comment\_length = 20

print('装载所有评论ing')

for file\_name in fileList:

if os.path.splitext(file\_prefix + file\_name)[1] == '.json':

with open(file\_prefix + file\_name, 'r', encoding='utf-8') as temp\_file:

for temp\_line in temp\_file:

if temp\_line != '\r\n':

temp\_dic = json.loads(temp\_line)

temp\_content = temp\_dic['content']

if(len(temp\_content) <= 500):

temp\_content = temp\_content.replace('[', '')

temp\_content = temp\_content.replace('\n', '')

temp\_content = temp\_content.replace(']', '')

raw\_comments.append(

temp\_content.replace('&hellip;', '...'))

print('已加载', len(raw\_comments), '条评论')

def addbr(current\_comments):

new\_comment = ''

for i in range(0, len(current\_comments), comment\_length):

if (i > len(current\_comments) - comment\_length):

new\_comment += current\_comments[i:i + comment\_length]

else:

new\_comment += current\_comments[i:i + comment\_length] + '</br>'

return (new\_comment)

def getFeatureComments(feature\_word, second\_index):

word\_cut = jieba.cut\_for\_search(feature\_word)

word\_cut\_list = list(word\_cut)

first\_index = int(second\_index)

factor\_temp = factor[first\_index]

if(factor\_temp not in word\_cut\_list):

word\_cut\_list.append(factor\_temp)

print(word\_cut\_list)

flag1 = 0

for items in word\_cut\_list:

for temp\_content in raw\_comments:

if(temp\_content.find(items) != -1):

flag1 += 1

if ((len(word\_cut\_list) <= 3) and (flag1 >= len(word\_cut\_list)) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

if((len(word\_cut\_list) > 3) and (flag1 >= 5.0 \* len(word\_cut\_list) / 6.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

# 找不到评论后，降低要求

for items in word\_cut\_list:

for temp\_content in raw\_comments:

if(temp\_content.find(items) != -1):

flag1 += 1

if ((len(word\_cut\_list) <= 3) and (flag1 >= 2.0 \* len(word\_cut\_list) / 3.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

if ((len(word\_cut\_list) > 3) and (flag1 >= len(word\_cut\_list) / 2.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

for firstfolder in firstfolder\_List:

childdren\_1 = {}

childdren\_1['name'] = factor[int(firstfolder)]

print('原因', firstfolder)

firstfolder\_path = file\_folder + firstfolder + '/'

secondfolder\_List = os.listdir(firstfolder\_path)

childdren\_1\_children = []

secondfolder\_List\_dic = {}

for secondfolder in secondfolder\_List:

print('具体原因排序', secondfolder)

secondfolder\_list\_count = 0.0

secondfolder\_path = firstfolder\_path + secondfolder + '/'

leaf\_file\_list = os.listdir(secondfolder\_path)

leaf\_dict = {}

leaf\_dict['name'] = secondfolder

leaf\_children = []

for leaf\_file in leaf\_file\_list:

if(leaf\_file.endswith('a.txt')):

# print(leaf\_file)

leaf\_file\_path = secondfolder\_path + leaf\_file

with open(leaf\_file\_path, 'r', encoding='utf-8') as f:

for line in f:

line\_dic = {}

line\_list = line.split()

# print(line\_list)

# 二级指标计数

secondfolder\_list\_count += float(line\_list[1])

secondfolder\_List\_dic[secondfolder] = secondfolder\_list\_count

secondfolder\_List\_sorted = sorted(

secondfolder\_List\_dic.items(), key=lambda x: x[1], reverse=True)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_')

print(secondfolder\_List\_sorted)

for secondfolder\_items in secondfolder\_List\_sorted:

secondfolder = secondfolder\_items[0]

print(secondfolder)

print('具体原因', secondfolder)

print(secondfolder\_List\_sorted)

secondfolder\_path = firstfolder\_path + secondfolder + '/'

leaf\_file\_list = os.listdir(secondfolder\_path)

leaf\_dict = {}

leaf\_dict['name'] = secondfolder

leaf\_children = []

line\_dic\_count = 0.0

for leaf\_file in leaf\_file\_list:

count = 0

if (leaf\_file.endswith('a.txt')):

# print(leaf\_file)

leaf\_file\_path = secondfolder\_path + leaf\_file

with open(leaf\_file\_path, 'r', encoding='utf-8') as f:

for line in f:

line\_dic = {}

line\_list = line.split()

# print(line\_list)

# 二级指标计数

if (count < count\_max):

line\_dic['name'] = line\_list[0]

print('关键词', line\_list[0])

# 典型评论提取

extract\_comments[line\_list[0]] = getFeatureComments(

line\_list[0], firstfolder)

print(extract\_comments[line\_list[0]])

line\_dic['value'] = line\_list[1]

line\_dic\_count += float(line\_list[1])

leaf\_children.append(line\_dic)

count += 1

# print(leaf\_children)

leaf\_dict['children'] = leaf\_children

leaf\_dict['value'] = line\_dic\_count

childdren\_1\_children.append(leaf\_dict)

childdren\_1['children'] = childdren\_1\_children

first\_dic\_list.append(childdren\_1)

first\_dic['children'] = first\_dic\_list

print(first\_dic)

import json

# 写入文件

outfile\_path\_prefix = file\_path\_prefix + itemname + '/'

outfile\_name = 'statics\_' + itemname + '\_tree\_a.json'

outfile\_name2 = 'statics\_' + itemname + '\_tree\_a\_comments.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

outfile\_name\_withprefix2 = outfile\_path\_prefix + outfile\_name2

# print(outfile\_name\_withprefix)

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(first\_dic, indent=4, ensure\_ascii=False))

with open(outfile\_name\_withprefix2, 'w+', encoding='utf-8') as f:

f.write(json.dumps(extract\_comments, indent=4, ensure\_ascii=False))

# step\_4\_statics26\_wang.py

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

import json

import os

import jieba.analyse

import re

import csv

import codecs

'''

目的：获得树状图

mh\_S

输出：8类用户下各个原因的数量的list

注意：将多个txt合成一个文件

'''

itemname = 'laptop'

itemname\_zh = '笔记本电脑'

file\_prefix='/Users/diw/PycharmProjects/jd\_negatives/'+itemname+'/'

file\_path\_prefix = '/users/diw/Desktop/'

csv\_name = file\_path\_prefix + itemname + '.csv'

csv\_name\_time = file\_path\_prefix + itemname + '\_time.csv'

#输出的csv文件使用带bom的utf-8编码

file\_folder = file\_path\_prefix + '180324\_laptop\_差评原因细分过滤与排序/'

firstfolder\_List = os.listdir(file\_folder)

count\_max = 6

factor = ['物流','产品','售后','营销']

first\_dic = {}

first\_dic['name'] = itemname\_zh + '差评'

first\_dic\_list = []

raw\_comments = []

fileList = os.listdir(file\_prefix)

times = 3

extract\_comments = {}

comment\_length = 20

print('装载所有评论ing')

for file\_name in fileList:

if os.path.splitext(file\_prefix + file\_name)[1] == '.json':

with open(file\_prefix + file\_name,'r',encoding='utf-8') as temp\_file:

for temp\_line in temp\_file:

if temp\_line != '\r\n':

temp\_dic = json.loads(temp\_line)

temp\_content = temp\_dic['content']

if(len(temp\_content) <= 500):

temp\_content = temp\_content.replace('[', '')

temp\_content = temp\_content.replace(']', '')

raw\_comments.append(temp\_content.replace('&hellip;','...'))

print('已加载',len(raw\_comments),'条评论')

def addbr(current\_comments):

new\_comment = ''

for i in range(0, len(current\_comments),comment\_length):

if (i > len(current\_comments) - comment\_length):

new\_comment += current\_comments[i:i + comment\_length]

else:

new\_comment += current\_comments[i:i + comment\_length] + '</br>'

return (new\_comment)

def getFeatureComments(feature\_word, second\_index):

word\_cut=jieba.cut\_for\_search(feature\_word)

word\_cut\_list = list(word\_cut)

first\_index = int(second\_index)

factor\_temp = factor[first\_index]

if(factor\_temp not in word\_cut\_list):

word\_cut\_list.append(factor\_temp)

print(word\_cut\_list)

flag1 = 0

for items in word\_cut\_list:

for temp\_content in raw\_comments:

if(temp\_content.find(items) != -1):

flag1 += 1

if ((len(word\_cut\_list) <= 3) and (flag1 >= len(word\_cut\_list)) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

if((len(word\_cut\_list) > 3) and (flag1 >= 5.0\*len(word\_cut\_list)/6.0) and (len(temp\_content) > times\*len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

#找不到评论后，降低要求

for items in word\_cut\_list:

for temp\_content in raw\_comments:

if(temp\_content.find(items) != -1):

flag1 += 1

if ((len(word\_cut\_list) <= 3) and (flag1 >= 2.0 \* len(word\_cut\_list) / 3.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

if ((len(word\_cut\_list) > 3) and (flag1 >= len(word\_cut\_list) / 2.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

for firstfolder in firstfolder\_List:

childdren\_1 = {}

childdren\_1['name'] = factor[int(firstfolder)]

print('原因',firstfolder)

firstfolder\_path = file\_folder + firstfolder + '/'

secondfolder\_List = os.listdir(firstfolder\_path)

childdren\_1\_children = []

secondfolder\_List\_dic = {}

for secondfolder in secondfolder\_List:

print('具体原因排序',secondfolder)

secondfolder\_list\_count = 0.0

secondfolder\_path = firstfolder\_path + secondfolder + '/'

leaf\_file\_list = os.listdir(secondfolder\_path)

leaf\_dict = {}

leaf\_dict['name'] = secondfolder

leaf\_children = []

for leaf\_file in leaf\_file\_list:

if(leaf\_file.endswith('a.txt')):

# print(leaf\_file)

leaf\_file\_path = secondfolder\_path + leaf\_file

with open(leaf\_file\_path,'r',encoding='utf-8') as f:

for line in f:

line\_dic = {}

line\_list = line.split()

# print(line\_list)

# 二级指标计数

secondfolder\_list\_count += float(line\_list[1])

secondfolder\_List\_dic[secondfolder] = secondfolder\_list\_count

secondfolder\_List\_sorted = sorted(secondfolder\_List\_dic.items(),key = lambda x:x[1],reverse=True)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_')

print(secondfolder\_List\_sorted)

for secondfolder\_items in secondfolder\_List\_sorted:

secondfolder = secondfolder\_items[0]

print(secondfolder)

print('具体原因', secondfolder)

print(secondfolder\_List\_sorted)

secondfolder\_path = firstfolder\_path + secondfolder + '/'

leaf\_file\_list = os.listdir(secondfolder\_path)

leaf\_dict = {}

leaf\_dict['name'] = secondfolder

leaf\_children = []

line\_dic\_count = 0.0

for leaf\_file in leaf\_file\_list:

count = 0

if (leaf\_file.endswith('a.txt')):

# print(leaf\_file)

leaf\_file\_path = secondfolder\_path + leaf\_file

with open(leaf\_file\_path, 'r',encoding='utf-8') as f:

for line in f:

line\_dic = {}

line\_list = line.split()

# print(line\_list)

# 二级指标计数

if (count < count\_max):

line\_dic['name'] = line\_list[0]

print('关键词',line\_list[0])

# 典型评论提取

extract\_comments[line\_list[0]] = getFeatureComments(line\_list[0],firstfolder)

print(extract\_comments[line\_list[0]])

line\_dic['value'] = line\_list[1]

line\_dic\_count += float(line\_list[1])

leaf\_children.append(line\_dic)

count += 1

# print(leaf\_children)

leaf\_dict['children'] = leaf\_children

leaf\_dict['value'] = line\_dic\_count

childdren\_1\_children.append(leaf\_dict)

childdren\_1['children'] = childdren\_1\_children

first\_dic\_list.append(childdren\_1)

first\_dic['children'] = first\_dic\_list

print(first\_dic)

import json

#写入文件

outfile\_path\_prefix =file\_path\_prefix + itemname + '/'

outfile\_name = 'statics\_'+ itemname + '\_tree\_a.json'

outfile\_name2 = 'statics\_'+ itemname + '\_tree\_a\_comments.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

outfile\_name\_withprefix2 = outfile\_path\_prefix + outfile\_name2

# print(outfile\_name\_withprefix)

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(first\_dic, indent=4, ensure\_ascii=False))

with open(outfile\_name\_withprefix2, 'w+', encoding='utf-8') as f:

f.write(json.dumps(extract\_comments, indent=4, ensure\_ascii=False))

# step\_4\_statics27\_sun.py

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

import json

import os

import jieba.analyse

import re

import csv

import codecs

'''

目的：获得树状图

mh\_S

输出：8类用户下各个原因的数量的list

注意：将多个txt合成一个文件

'''

itemname = 'phone'

itemname\_zh = '手机'

file\_prefix='/Users/diw/PycharmProjects/jd\_negatives/'+itemname+'/'

file\_path\_prefix = '/users/diw/Desktop/'

csv\_name = file\_path\_prefix + itemname + '.csv'

csv\_name\_time = file\_path\_prefix + itemname + '\_time.csv'

#输出的csv文件使用带bom的utf-8编码

file\_folder = file\_path\_prefix + '180326\_phone\_词频分类排序/positiveRejectWeight/'

firstfolder\_List = os.listdir(file\_folder)

count\_max = 6

factor = ['物流','产品','售后','营销']

first\_dic = {}

first\_dic['name'] = itemname\_zh + '差评'

first\_dic\_list = []

raw\_comments = []

fileList = os.listdir(file\_prefix)

times = 3

extract\_comments = {}

comment\_length = 20

print('装载所有评论ing')

for file\_name in fileList:

if os.path.splitext(file\_prefix + file\_name)[1] == '.json':

with open(file\_prefix + file\_name,'r',encoding='utf-8') as temp\_file:

for temp\_line in temp\_file:

if temp\_line != '\r\n':

temp\_dic = json.loads(temp\_line)

temp\_content = temp\_dic['content']

if(len(temp\_content) <= 500):

temp\_content = temp\_content.replace('[','')

temp\_content = temp\_content.replace(']','')

raw\_comments.append(temp\_content.replace('&hellip;', '...'))

print('已加载',len(raw\_comments),'条评论')

def addbr(current\_comments):

new\_comment = ''

for i in range(0, len(current\_comments),comment\_length):

if (i > len(current\_comments) - comment\_length):

new\_comment += current\_comments[i:i + comment\_length]

else:

new\_comment += current\_comments[i:i + comment\_length] + '</br>'

return (new\_comment)

def getFeatureComments(feature\_word, second\_index):

word\_cut=jieba.cut\_for\_search(feature\_word)

word\_cut\_list = list(word\_cut)

first\_index = int(second\_index)

factor\_temp = factor[first\_index]

if(factor\_temp not in word\_cut\_list):

word\_cut\_list.append(factor\_temp)

print(word\_cut\_list)

flag1 = 0

for items in word\_cut\_list:

for temp\_content in raw\_comments:

if(temp\_content.find(items) != -1):

flag1 += 1

if ((len(word\_cut\_list) <= 3) and (flag1 >= len(word\_cut\_list)) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

if((len(word\_cut\_list) > 3) and (flag1 >= 5.0\*len(word\_cut\_list)/6.0) and (len(temp\_content) > times\*len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

#找不到评论后，降低要求

for items in word\_cut\_list:

for temp\_content in raw\_comments:

if(temp\_content.find(items) != -1):

flag1 += 1

if ((len(word\_cut\_list) <= 3) and (flag1 >= 2.0 \* len(word\_cut\_list) / 3.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

if ((len(word\_cut\_list) > 3) and (flag1 >= len(word\_cut\_list) / 2.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

for firstfolder in firstfolder\_List:

childdren\_1 = {}

childdren\_1['name'] = factor[int(firstfolder)]

print('原因',firstfolder)

firstfolder\_path = file\_folder + firstfolder + '/'

secondfolder\_List = os.listdir(firstfolder\_path)

childdren\_1\_children = []

secondfolder\_List\_dic = {}

for secondfolder in secondfolder\_List:

print('具体原因排序',secondfolder)

secondfolder\_list\_count = 0.0

secondfolder\_path = firstfolder\_path + secondfolder + '/'

leaf\_file\_list = os.listdir(secondfolder\_path)

leaf\_dict = {}

leaf\_dict['name'] = secondfolder

leaf\_children = []

for leaf\_file in leaf\_file\_list:

if(leaf\_file.endswith('n.txt')):

# print(leaf\_file)

leaf\_file\_path = secondfolder\_path + leaf\_file

with open(leaf\_file\_path,'r',encoding='utf-8') as f:

for line in f:

line\_dic = {}

line\_list = line.split()

# print(line\_list)

# 二级指标计数

secondfolder\_list\_count += float(line\_list[1])

secondfolder\_List\_dic[secondfolder] = secondfolder\_list\_count

secondfolder\_List\_sorted = sorted(secondfolder\_List\_dic.items(),key = lambda x:x[1],reverse=True)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_')

print(secondfolder\_List\_sorted)

for secondfolder\_items in secondfolder\_List\_sorted:

secondfolder = secondfolder\_items[0]

print(secondfolder)

print('具体原因', secondfolder)

print(secondfolder\_List\_sorted)

secondfolder\_path = firstfolder\_path + secondfolder + '/'

leaf\_file\_list = os.listdir(secondfolder\_path)

leaf\_dict = {}

leaf\_dict['name'] = secondfolder

leaf\_children = []

line\_dic\_count = 0.0

for leaf\_file in leaf\_file\_list:

count = 0

if (leaf\_file.endswith('n.txt')):

# print(leaf\_file)

leaf\_file\_path = secondfolder\_path + leaf\_file

with open(leaf\_file\_path, 'r',encoding='utf-8') as f:

for line in f:

line\_dic = {}

line\_list = line.split()

# print(line\_list)

# 二级指标计数

if (count < count\_max):

line\_dic['name'] = line\_list[0]

print('关键词',line\_list[0])

# 典型评论提取

extract\_comments[line\_list[0]] = getFeatureComments(line\_list[0],firstfolder)

print(extract\_comments[line\_list[0]])

line\_dic['value'] = line\_list[1]

line\_dic\_count += float(line\_list[1])

leaf\_children.append(line\_dic)

count += 1

# print(leaf\_children)

leaf\_dict['children'] = leaf\_children

leaf\_dict['value'] = line\_dic\_count

childdren\_1\_children.append(leaf\_dict)

childdren\_1['children'] = childdren\_1\_children

first\_dic\_list.append(childdren\_1)

first\_dic['children'] = first\_dic\_list

print(first\_dic)

import json

#写入文件

outfile\_path\_prefix =file\_path\_prefix + itemname + '/'

outfile\_name = 'statics\_'+ itemname + '\_tree\_n.json'

outfile\_name2 = 'statics\_'+ itemname + '\_tree\_n\_comments.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

outfile\_name\_withprefix2 = outfile\_path\_prefix + outfile\_name2

# print(outfile\_name\_withprefix)

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(first\_dic, indent=4, ensure\_ascii=False))

with open(outfile\_name\_withprefix2, 'w+', encoding='utf-8') as f:

f.write(json.dumps(extract\_comments, indent=4, ensure\_ascii=False))

# step\_1.py

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

'''

从特定品牌的文件夹中按id提取各个factors的数量以及各id产品的差评总计

该步需要手动修改brand列以保证准确性

'''

import json

import os

import re

import csv

import pandas

# 导入配置文件

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics", "itemname")

file\_prefix = config.get("statics", "file\_prefix")

file\_path\_prefix = config.get("statics", "file\_path\_prefix")

fileList = []

fileList = os.listdir(file\_prefix)

print(fileList)

csv\_name = config.get("statics", "csv\_name\_crossvalidation")

csv\_worksheet\_file = open(csv\_name, 'w', encoding='utf-8')

csv\_worksheet\_writer = csv.writer(csv\_worksheet\_file)

csv\_worksheet\_writer.writerow(['referenceID', 'description'])

csv\_worksheet\_file.close()

count\_line = 0

# 转换文件从xls到csv

for file\_name in fileList:

if os.path.splitext(file\_name)[1] == '.json':

file\_holename = file\_prefix + file\_name

referenceId = file\_name = file\_name[6:-5]

count = 0

count\_logistics = 0

count\_product = 0

count\_afterservive = 0

count\_marketing = 0

with open(file\_holename, 'r') as f:

# s\_3 = json.load(f)

for line in f:

if line != '\r\n':

# print line + str(type(line))

line\_dic = json.loads(line)

# print 'line\_dic:' + str(type(line\_dic))

temp\_count = line\_dic['label']

# write line by line

description = line\_dic['referenceName']

# print type(referenceName)

# result\_list = re.findall('[a-zA-Z0-9]+', referenceName)

# 写入csvd

values = [referenceId, description]

with open(csv\_name, 'a', newline='', encoding='utf-8') as csv\_name\_file:

csv\_name\_writer = csv.writer(csv\_name\_file)

csv\_name\_writer.writerow(values)

# step\_4\_statics27\_wang.py

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

import json

import os

import jieba.analyse

import re

import csv

import codecs

'''

目的：获得树状图

mh\_S

输出：8类用户下各个原因的数量的list

注意：将多个txt合成一个文件

'''

itemname = 'laptop'

itemname\_zh = '笔记本电脑'

file\_prefix = '/Users/diw/PycharmProjects/jd\_negatives/' + itemname + '/'

file\_path\_prefix = '/users/diw/Desktop/'

csv\_name = file\_path\_prefix + itemname + '.csv'

csv\_name\_time = file\_path\_prefix + itemname + '\_time.csv'

# 输出的csv文件使用带bom的utf-8编码

file\_folder = file\_path\_prefix + '180324\_laptop\_差评原因细分过滤与排序/'

firstfolder\_List = os.listdir(file\_folder)

count\_max = 6

factor = ['物流', '产品', '售后', '营销']

first\_dic = {}

first\_dic['name'] = itemname\_zh + '差评'

first\_dic\_list = []

raw\_comments = []

fileList = os.listdir(file\_prefix)

times = 3

extract\_comments = {}

comment\_length = 20

print('装载所有评论ing')

for file\_name in fileList:

if os.path.splitext(file\_prefix + file\_name)[1] == '.json':

with open(file\_prefix + file\_name, 'r', encoding='utf-8') as temp\_file:

for temp\_line in temp\_file:

if temp\_line != '\r\n':

temp\_dic = json.loads(temp\_line)

temp\_content = temp\_dic['content']

if(len(temp\_content) <= 500):

temp\_content = temp\_content.replace('[', '')

temp\_content = temp\_content.replace(']', '')

raw\_comments.append(

temp\_content.replace('&hellip;', '...'))

print('已加载', len(raw\_comments), '条评论')

def addbr(current\_comments):

new\_comment = ''

for i in range(0, len(current\_comments), comment\_length):

if (i > len(current\_comments) - comment\_length):

new\_comment += current\_comments[i:i + comment\_length]

else:

new\_comment += current\_comments[i:i + comment\_length] + '</br>'

return (new\_comment)

def getFeatureComments(feature\_word, second\_index):

word\_cut = jieba.cut\_for\_search(feature\_word)

word\_cut\_list = list(word\_cut)

first\_index = int(second\_index)

factor\_temp = factor[first\_index]

if(factor\_temp not in word\_cut\_list):

word\_cut\_list.append(factor\_temp)

print(word\_cut\_list)

flag1 = 0

for items in word\_cut\_list:

for temp\_content in raw\_comments:

if(temp\_content.find(items) != -1):

flag1 += 1

if ((len(word\_cut\_list) <= 3) and (flag1 >= len(word\_cut\_list)) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

if((len(word\_cut\_list) > 3) and (flag1 >= 5.0 \* len(word\_cut\_list) / 6.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

# 找不到评论后，降低要求

for items in word\_cut\_list:

for temp\_content in raw\_comments:

if(temp\_content.find(items) != -1):

flag1 += 1

if ((len(word\_cut\_list) <= 3) and (flag1 >= 2.0 \* len(word\_cut\_list) / 3.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

if ((len(word\_cut\_list) > 3) and (flag1 >= len(word\_cut\_list) / 2.0) and (len(temp\_content) > times \* len(feature\_word))):

temp\_content = addbr(temp\_content)

return temp\_content

for firstfolder in firstfolder\_List:

childdren\_1 = {}

childdren\_1['name'] = factor[int(firstfolder)]

print('原因', firstfolder)

firstfolder\_path = file\_folder + firstfolder + '/'

secondfolder\_List = os.listdir(firstfolder\_path)

childdren\_1\_children = []

secondfolder\_List\_dic = {}

for secondfolder in secondfolder\_List:

print('具体原因排序', secondfolder)

secondfolder\_list\_count = 0.0

secondfolder\_path = firstfolder\_path + secondfolder + '/'

leaf\_file\_list = os.listdir(secondfolder\_path)

leaf\_dict = {}

leaf\_dict['name'] = secondfolder

leaf\_children = []

for leaf\_file in leaf\_file\_list:

if(leaf\_file.endswith('n.txt')):

# print(leaf\_file)

leaf\_file\_path = secondfolder\_path + leaf\_file

with open(leaf\_file\_path, 'r', encoding='utf-8') as f:

for line in f:

line\_dic = {}

line\_list = line.split()

# print(line\_list)

# 二级指标计数

secondfolder\_list\_count += float(line\_list[1])

secondfolder\_List\_dic[secondfolder] = secondfolder\_list\_count

secondfolder\_List\_sorted = sorted(

secondfolder\_List\_dic.items(), key=lambda x: x[1], reverse=True)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_')

print(secondfolder\_List\_sorted)

for secondfolder\_items in secondfolder\_List\_sorted:

secondfolder = secondfolder\_items[0]

print(secondfolder)

print('具体原因', secondfolder)

print(secondfolder\_List\_sorted)

secondfolder\_path = firstfolder\_path + secondfolder + '/'

leaf\_file\_list = os.listdir(secondfolder\_path)

leaf\_dict = {}

leaf\_dict['name'] = secondfolder

leaf\_children = []

line\_dic\_count = 0.0

for leaf\_file in leaf\_file\_list:

count = 0

if (leaf\_file.endswith('n.txt')):

# print(leaf\_file)

leaf\_file\_path = secondfolder\_path + leaf\_file

with open(leaf\_file\_path, 'r', encoding='utf-8') as f:

for line in f:

line\_dic = {}

line\_list = line.split()

# print(line\_list)

# 二级指标计数

if (count < count\_max):

line\_dic['name'] = line\_list[0]

print('关键词', line\_list[0])

# 典型评论提取

extract\_comments[line\_list[0]] = getFeatureComments(

line\_list[0], firstfolder)

print(extract\_comments[line\_list[0]])

line\_dic['value'] = line\_list[1]

line\_dic\_count += float(line\_list[1])

leaf\_children.append(line\_dic)

count += 1

# print(leaf\_children)

leaf\_dict['children'] = leaf\_children

leaf\_dict['value'] = line\_dic\_count

childdren\_1\_children.append(leaf\_dict)

childdren\_1['children'] = childdren\_1\_children

first\_dic\_list.append(childdren\_1)

first\_dic['children'] = first\_dic\_list

print(first\_dic)

import json

# 写入文件

outfile\_path\_prefix = file\_path\_prefix + itemname + '/'

outfile\_name = 'statics\_' + itemname + '\_tree\_n.json'

outfile\_name2 = 'statics\_' + itemname + '\_tree\_n\_comments.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

outfile\_name\_withprefix2 = outfile\_path\_prefix + outfile\_name2

# print(outfile\_name\_withprefix)

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(first\_dic, indent=4, ensure\_ascii=False))

with open(outfile\_name\_withprefix2, 'w+', encoding='utf-8') as f:

f.write(json.dumps(extract\_comments, indent=4, ensure\_ascii=False))

# write\_config.py

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

import configparser

config = configparser.ConfigParser()

config.read("parameter\_ini.ini")

itemname = input('请输入类别：\n')

factor1 = ['售后', '产品', '营销', '物流']

file\_prefix = '/Users/diw/PycharmProjects/jd\_negatives/' + itemname + '/'

file\_path\_prefix = '/users/diw/Desktop/'

outfile\_path\_prefix = file\_path\_prefix + itemname + '/'

csv\_name = file\_path\_prefix + itemname + '.csv'

csv\_name\_pre = file\_path\_prefix + itemname + '\_pre.csv'

csv\_name\_crossvalidation = file\_path\_prefix + itemname + '\_crossvalidation.csv'

csv\_time\_name = file\_path\_prefix + itemname + '\_time.csv'

dump\_filepath = file\_path\_prefix + itemname + '\_pid\_scores\_leanings.dump'

try:

config.add\_section("statics")

config.set("statics", "itemname", itemname)

config.set("statics", "file\_prefix", file\_prefix)

config.set("statics", "file\_path\_prefix", file\_path\_prefix)

config.set("statics", "outfile\_path\_prefix", outfile\_path\_prefix)

config.set("statics", "csv\_name", csv\_name)

config.set("statics", "csv\_name\_pre", csv\_name\_pre)

config.set("statics", "csv\_name\_crossvalidation", csv\_name\_crossvalidation)

config.set("statics", "csv\_time\_name", csv\_time\_name)

config.set("statics", "dump\_filepath", dump\_filepath)

except configparser.DuplicateSectionError:

print('section已存在')

with open("parameter\_ini.ini", "w") as config\_f:

config.write(config\_f)

# step\_2.py

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

'''

从特定品牌的文件夹中按id提取各个factors的数量以及各id产品的差评总计

该步需要手动修改brand列以保证准确性

使用标记前两百条以作为统计的brand

'''

import json

from xlutils.copy import copy

import os

import re

import csv

import pandas

# 导入配置文件

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics", "itemname")

file\_prefix = config.get("statics", "file\_prefix")

file\_path\_prefix = config.get("statics", "file\_path\_prefix")

fileList = []

fileList = os.listdir(file\_prefix)

print(fileList)

csv\_name\_after = config.get("statics", "csv\_name")

print(csv\_name\_after)

csv\_name = config.get("statics", "csv\_name\_pre")

csv\_worksheet\_file = open(csv\_name\_after, 'w', encoding='utf-8')

csv\_worksheet\_writer = csv.writer(csv\_worksheet\_file)

csv\_worksheet\_writer.writerow(

['Brand', 'referenceID', 'Logistics', 'Product', 'Afterservice', 'Marketing', 'Sum'])

csv\_worksheet\_file.close()

count\_line = 0

# 对比前200条标记

count\_max = 200

current\_count = 0

for file\_name in fileList:

if os.path.splitext(file\_name)[1] == '.json':

file\_holename = file\_prefix + file\_name

referenceId = file\_name = file\_name[6:-5]

count = 0

count\_logistics = 0

count\_product = 0

count\_afterservive = 0

count\_marketing = 0

with open(file\_holename, 'r') as f:

# s\_3 = json.load(f)

for line in f:

if line != '\r\n':

# print line + str(type(line))

line\_dic = json.loads(line)

# print 'line\_dic:' + str(type(line\_dic))

temp\_count = line\_dic['label']

# print 'temp\_count' + str(type(temp\_count))

if temp\_count == 0:

count\_logistics += 1

elif temp\_count == 1:

count\_product += 1

elif temp\_count == 2:

count\_afterservive += 1

elif temp\_count == 3:

count\_marketing += 1

else:

print("No label!")

count += 1

# write line by line

referenceName = line\_dic['referenceName'].upper()

current\_count = 0

find = 0

with open(csv\_name, 'r') as f2:

for line2 in f2:

if(line2 != '\r\n'):

line2\_list = line2.split(',')

if (current\_count == 0):

current\_count += 1

continue

if (current\_count <= 200):

current\_count += 1

marked\_referenceName = line2\_list[0].upper()

if ((referenceName.find(marked\_referenceName) != -1) and (len(marked\_referenceName) != 0)):

referenceName = marked\_referenceName

print('Match Successfully!')

find = 1

break

if(find == 0):

print('Unmatch!')

# 写入csv

if(find == 1):

values = [referenceName, referenceId, count\_logistics,

count\_product, count\_afterservive, count\_marketing, count]

with open(csv\_name\_after, 'a', newline='', encoding='utf-8') as csv\_name\_file:

csv\_name\_writer = csv.writer(csv\_name\_file)

csv\_name\_writer.writerow(values)

# step\_3.py

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

'''

从特定品牌的文件夹中按时间提取各个记录的时间信息

'''

import json

from xlrd import open\_workbook

from xlutils.copy import copy

import os

import re

# from a file get the json data.

import csv

# 导入配置文件

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics","itemname")

file\_prefix = config.get("statics","file\_prefix")

print(file\_prefix)

file\_path\_prefix = config.get("statics","file\_path\_prefix")

csv\_time\_name = config.get("statics","csv\_time\_name")

file\_1\_name = file\_path\_prefix + itemname + '.csv'

csv\_worksheet\_file = open(csv\_time\_name,'a',encoding='utf-8-sig')

csv\_worksheet\_writer = csv.writer(csv\_worksheet\_file)

csv\_worksheet\_writer.writerow(['brand','referenceId','userLevelName','creationDate','creationTime','label','month','hour'])

csv\_worksheet\_file.close()

fileList=[]

fileList = os.listdir(file\_prefix)

print(fileList)

m = 1

exist\_brandreferencedic = {}

with open(file\_1\_name, 'r', encoding='utf-8-sig') as file\_1:

file\_1\_reader = csv.DictReader(file\_1)

for row in file\_1\_reader:

if(row != '\n'):

exist\_brandreferencedic[row['referenceID']] = row['Brand']

for file\_name in fileList:

if os.path.splitext(file\_name)[1] == '.json':

file\_holename=file\_prefix+file\_name

file\_name=file\_name[6:-5]

referenceId = file\_name

# 和之前的文件中整理好的品牌信息交叉对比

if (referenceId not in exist\_brandreferencedic.keys()):

continue

brand = exist\_brandreferencedic[referenceId]

with open(file\_holename, 'r') as f:

# print(file\_name)

referenceId\_list = []

userLeverName\_list = []

creationDate\_list = []

creationtime\_list = []

label\_list=[]

count = 0

for line in f:

if line != '\r\n':

line\_dic = json.loads(line)

label = line\_dic['label']

if('userLevelName' in line\_dic):

userLeverName = line\_dic['userLevelName']

else:

continue

creationDate = line\_dic['creationTime'].split()[0]

creationtime = line\_dic['creationTime'].split()[1]

count += 1

referenceId\_list.append(referenceId)

userLeverName\_list.append(userLeverName)

creationDate\_list.append(creationDate)

creationtime\_list.append(creationtime)

label\_list.append(label)

#按每个referencId写文件

print('Writing to csv from file:label-'+referenceId+'.json')

for i in range(0,count):

csv\_worksheet\_file = open(csv\_time\_name, 'a',encoding='utf-8-sig')

csv\_worksheet\_writer = csv.writer(csv\_worksheet\_file)

csv\_worksheet\_writer.writerow([brand, referenceId\_list[i], userLeverName\_list[i], creationDate\_list[i], creationtime\_list[i], label\_list[i], creationDate\_list[i][5:7], creationtime\_list[i][0:2]])

# print(creationDate\_list[i]+' '+creationDate\_list[i][5:7])

# print(creationtime\_list[i] + ' ' +creationtime\_list[i][0:2])

csv\_worksheet\_file.close()

# step\_4\_statics1.py

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

'''

目的：从jd\_xx.csv中统计各种差评因素的直接数据

格式： factor = ['售后','产品','营销','物流']

result = [label2,label1,label3,label0]

sum

'''

import csv

import os

# 导入配置文件

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics", "itemname")

file\_prefix = config.get("statics", "file\_prefix")

file\_path\_prefix = config.get("statics", "file\_path\_prefix")

# 新建文件夹

if(not os.path.exists(file\_path\_prefix + itemname)):

os.mkdir(file\_path\_prefix + itemname)

csv\_name = config.get("statics", "csv\_name")

result = [0, 0, 0, 0]

flag = 0 # 标志第一行

factor = ['售后', '产品', '营销', '物流']

with open(csv\_name, 'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

# 跳过第一行

if(flag == 0):

flag += 1

continue

result[0] += int(line[4])

result[1] += int(line[3])

result[2] += int(line[5])

result[3] += int(line[2])

sum = 0

print(factor)

print(result)

for i in range(len(result)):

sum += result[i]

print(sum)

import json

# 写入文件

outfile\_path\_prefix = config.get("statics", "outfile\_path\_prefix")

outfile\_name = 'statics\_' + itemname + '\_1.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic = {}

outfile\_dic['factor'] = factor

outfile\_dic['result'] = result

outfile\_dic['sum'] = sum

#

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

# step\_4\_statics2.py

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

'''

目的：从jd\_xx.csv中统计各种品牌差评的数量

格式：result\_brand = [品牌1，品牌2，...]

result\_count = [品牌1数量，品牌2数量，...]

输出：按数量降序排列的字典

按数量降序排列好的 品牌列表

计数列表

outfile\_dic['brand'] = sorted\_result\_list\_brand

outfile\_dic['count'] = sorted\_result\_list\_count

outfile\_dic['sum'] = sum

'''

import csv

# 导入配置文件

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics", "itemname")

file\_prefix = config.get("statics", "file\_prefix")

file\_path\_prefix = config.get("statics", "file\_path\_prefix")

csv\_name = config.get("statics", "csv\_name")

result\_map = {}

result\_brand = []

result\_count = []

flag = 0 # 标志第一行

i = 0

with open(csv\_name, 'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

# 跳过第一行

if(flag == 0):

flag += 1

print(line)

continue

if(line[0] in result\_map.keys()):

result\_map[line[0]] += int(line[6])

else:

result\_map[line[0]] = 0

result\_map[line[0]] += int(line[6])

sorted\_result\_list = sorted(

result\_map.items(), key=lambda x: x[1], reverse=True)

# print(sorted\_result\_list)

sorted\_result\_list\_brand = []

sorted\_result\_list\_count = []

for items in sorted\_result\_list:

sorted\_result\_list\_brand.append(items[0])

sorted\_result\_list\_count.append(items[1])

print(sorted\_result\_list\_brand)

print(sorted\_result\_list\_count)

sum = 0

for i in sorted\_result\_list\_count:

sum += i

print(sum)

# 验证两列表长度是否相同

print(len(sorted\_result\_list\_brand) == len(sorted\_result\_list\_count))

import json

# 写入文件

outfile\_path\_prefix = config.get("statics", "outfile\_path\_prefix")

outfile\_name = 'statics\_' + itemname + '\_2.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic = {}

outfile\_dic['brand'] = sorted\_result\_list\_brand

outfile\_dic['count'] = sorted\_result\_list\_count

outfile\_dic['sum'] = sum

#

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

# step\_4\_statics3-6.py

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

'''

目的：从jd\_xx.csv中统计各种品牌差评的数量最多的四个品牌

格式：

['售后','产品','营销','物流']

'brand1' brand1 =[xx,xx,xx,xx]

'brand2' brand3 =[xx,xx,xx,xx]

'brand3' brand4 =[xx,xx,xx,xx]

'brand4' brand5 =[xx,xx,xx,xx]

outfile\_dic['brand\_list'] = brand\_list

outfile\_dic['factors'] = factors

outfile\_dic['count\_list'] = count\_certainBrand\_list

outfile\_dic['sum\_list'] = sum\_list

'''

import csv

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics", "itemname")

file\_prefix = config.get("statics", "file\_prefix")

file\_path\_prefix = config.get("statics", "file\_path\_prefix")

csv\_name = config.get("statics", "csv\_name")

result\_map = {}

result\_brand = []

result\_count = []

flag = 0 # 标志第一行

i = 0

with open(csv\_name, 'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

# 跳过第一行

if(flag == 0):

flag += 1

# print(line)

continue

if(line[0] in result\_map.keys()):

result\_map[line[0]] += int(line[6])

else:

result\_map[line[0]] = 0

result\_map[line[0]] += int(line[6])

sorted\_result\_list = sorted(

result\_map.items(), key=lambda x: x[1], reverse=True)

print(sorted\_result\_list)

sorted\_result\_list\_brand = []

sorted\_result\_list\_count = []

for items in sorted\_result\_list:

sorted\_result\_list\_brand.append(items[0])

sorted\_result\_list\_count.append(items[1])

# 验证两列表长度是否相同

print(len(sorted\_result\_list\_brand) == len(sorted\_result\_list\_count))

import json

# 写入文件

outfile\_path\_prefix = config.get("statics", "outfile\_path\_prefix")

outfile\_name = 'statics\_' + itemname + '\_3.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

# 获得数量最多的四个品牌

factors = ['售后', '产品', '营销', '物流']

brand\_list = sorted\_result\_list\_brand[0:4]

print(brand\_list)

print(factors)

count\_certainBrand\_list = []

sum\_list = []

for i in range(4):

sum\_temp = 0

flag = 0 # 标志着第一行，重新置零

count\_certainBrand = [0, 0, 0, 0]

with open(csv\_name, 'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

# 跳过第一行

if(flag == 0):

flag += 1

# print(line)

continue

if(line[0] == brand\_list[i]):

count\_certainBrand[0] += int(line[4])

count\_certainBrand[1] += int(line[3])

count\_certainBrand[2] += int(line[5])

count\_certainBrand[3] += int(line[2])

# 输出产品名

print(brand\_list[i])

print(factors)

count\_certainBrand\_list.append(count\_certainBrand)

print(count\_certainBrand)

for item in count\_certainBrand:

sum\_temp += item

sum\_list.append(sum\_temp)

outfile\_dic = {}

outfile\_dic['brand\_list'] = brand\_list

outfile\_dic['factors'] = factors

outfile\_dic['count\_list'] = count\_certainBrand\_list

outfile\_dic['sum\_list'] = sum\_list

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

# step\_4\_statics7-10.py

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

'''

目的：从jd\_xx.csv中统计各种品牌差评的数量最多的四个品牌把它们

顺序：['售后原因','营销原因','产品原因','物流原因']

格式：

outfile\_dic['brand\_list'] = brand\_list

outfile\_dic['factors'] = factors

outfile\_dic['count\_list'] = count\_list

temp\_list.append(label2\_list)

temp\_list.append(label3\_list)

temp\_list.append(label1\_list)

temp\_list.append(label0\_list)

temp\_list.append(label2\_p\_list)

temp\_list.append(label3\_p\_list)

temp\_list.append(label1\_p\_list)

temp\_list.append(label0\_p\_list)

'''

import csv

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics","itemname")

file\_prefix = config.get("statics","file\_prefix")

file\_path\_prefix = config.get("statics","file\_path\_prefix")

csv\_name = config.get("statics","csv\_name")

csv\_name\_crossvalidation = config.get("statics","csv\_name\_crossvalidation")

def getdescription(referenceid,csv\_name\_crossvalidation):

flag = 0

with open(csv\_name\_crossvalidation,'r') as f:

for line in f:

if(flag == 0):

flag = 1

continue

line\_list = line.split(',')

if(referenceid == line\_list[0]):

print('Successfully match')

print(line\_list[1])

return line\_list[1]

print('Fail to match')

result\_map={}

result\_brand = []

result\_count = []

flag=0#标志第一行

i=0

with open(csv\_name,'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

#跳过第一行

if(flag==0):

flag+=1

# print(line)

continue

if(line[0] in result\_map.keys()):

result\_map[line[0]]+=int(line[6])

else:

result\_map[line[0]]=0

result\_map[line[0]]+=int(line[6])

sorted\_result\_list = sorted(result\_map.items(), key = lambda x: x[1], reverse = True)

# print(sorted\_result\_list)

sorted\_result\_list\_brand = []

sorted\_result\_list\_count = []

for items in sorted\_result\_list:

sorted\_result\_list\_brand.append(items[0])

sorted\_result\_list\_count.append(items[1])

#验证两列表长度是否相同

print(len(sorted\_result\_list\_brand)==len(sorted\_result\_list\_count))

import json

#写入文件

outfile\_path\_prefix = config.get("statics","outfile\_path\_prefix")

outfile\_name = 'statics\_'+ itemname + '\_7.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic = {}

#获得数量最多的四个品牌

factors = ['售后','营销','产品','物流']

brand\_list = sorted\_result\_list\_brand[0:4]

print('brand\_list', brand\_list)

print(factors)

count\_list = []

maxnumber\_list = []

label2\_list\_list = []

label3\_list\_list = []

label1\_list\_list = []

label0\_list\_list = []

label2\_p\_list\_list = []

label3\_p\_list\_list = []

label1\_p\_list\_list = []

label0\_p\_list\_list = []

product\_list\_list = []

for i in range(4):

flag = 0 # 标志着第一行，重新置零

product\_list=[]

label0\_list=[]

label1\_list=[]

label2\_list=[]

label3\_list=[]

maxnumber = 0

label0\_p\_list = []

label1\_p\_list = []

label2\_p\_list = []

label3\_p\_list = []

count\_temp = 0

# product\_list\_map\_1={'referenceId':sum}

product\_list\_map\_1 = {}

product\_list\_map\_1\_sorted\_list = []

# product\_list\_map\_2={'referenceId':[label2,label3,label1,label0]}

product\_list\_map\_2 = {}

with open(csv\_name,'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

# temp\_list = [sum,label2,label3,label1,label0]

temp\_list = []

#跳过第一行

if(flag==0):

flag+=1

# print(line)

continue

if(line[0] == brand\_list[i]):

product\_list\_map\_1[line[1]] = int(line[6])

temp\_list.append(line[4])

temp\_list.append(line[5])

temp\_list.append(line[3])

temp\_list.append(line[2])

product\_list\_map\_2[line[1]] = temp\_list

# 对product\_list\_map\_1排序

product\_list\_map\_1\_sorted\_list = sorted(product\_list\_map\_1.items(), key=lambda x: x[1], reverse=True)

product\_list\_map\_1\_sorted\_list\_temp = []

for item in product\_list\_map\_1\_sorted\_list:

if(item[1] >= 50 and count\_temp < 25):

count\_temp += 1

product\_list\_map\_1\_sorted\_list\_temp.append(item)

product\_list\_map\_1\_sorted\_list = sorted(product\_list\_map\_1\_sorted\_list\_temp, key=lambda x: x[1], reverse=False)

print(product\_list\_map\_1\_sorted\_list)

#按顺序遍历。把结果装入目标列表中

for productitem in product\_list\_map\_1\_sorted\_list:

if (productitem[1] >= maxnumber):

maxnumber = productitem[1]

product\_list.append(productitem[0])

product\_totalcount = int(productitem[1])

productitem\_inprolismap2 = product\_list\_map\_2[productitem[0]]

#数量

label2\_list.append(int(productitem\_inprolismap2[0]))

label3\_list.append(int(productitem\_inprolismap2[1]))

label1\_list.append(int(productitem\_inprolismap2[2]))

label0\_list.append(int(productitem\_inprolismap2[3]))

#百分比

label2\_p\_list.append(round(int(productitem\_inprolismap2[0]) / product\_totalcount, 4))

label3\_p\_list.append(round(int(productitem\_inprolismap2[1]) / product\_totalcount, 4))

label1\_p\_list.append(round(int(productitem\_inprolismap2[2]) / product\_totalcount, 4))

label0\_p\_list.append(round(int(productitem\_inprolismap2[3]) / product\_totalcount, 4))

maxnumber = round(maxnumber/100 +1)\*100

print(maxnumber)

print(brand\_list[i])

print(product\_list)

print(label2\_list)

print(label3\_list)

print(label1\_list)

print(label0\_list)

print(label2\_p\_list)

print(label3\_p\_list)

print(label1\_p\_list)

print(label0\_p\_list)

product\_list\_list.append(product\_list)

maxnumber\_list.append(maxnumber)

label2\_list\_list.append(label2\_list)

label3\_list\_list.append(label3\_list)

label1\_list\_list.append(label1\_list)

label0\_list\_list.append(label0\_list)

label2\_p\_list\_list.append(label2\_p\_list)

label3\_p\_list\_list.append(label3\_p\_list)

label1\_p\_list\_list.append(label1\_p\_list)

label0\_p\_list\_list.append(label0\_p\_list)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

brand\_list\_d = []

for i in product\_list\_list:

temp = []

for j in i:

temp.append(getdescription(j,csv\_name\_crossvalidation))

brand\_list\_d.append(temp)

print(brand\_list\_d)

print(type(brand\_list[1][1]))

outfile\_dic['brand\_list'] = brand\_list

outfile\_dic['product\_list\_list'] = brand\_list\_d

outfile\_dic['factors'] = ['','','','','售后原因','营销原因','产品原因','物流原因']

outfile\_dic['count\_list'] = count\_list

outfile\_dic['max\_list'] = maxnumber\_list

outfile\_dic['label2\_list\_list'] = label2\_list\_list

outfile\_dic['label3\_list\_list'] = label3\_list\_list

outfile\_dic['label1\_list\_list'] = label1\_list\_list

outfile\_dic['label0\_list\_list'] = label0\_list\_list

outfile\_dic['label2\_p\_list\_list'] = label2\_p\_list\_list

outfile\_dic['label3\_p\_list\_list'] = label3\_p\_list\_list

outfile\_dic['label1\_p\_list\_list'] = label1\_p\_list\_list

outfile\_dic['label0\_p\_list\_list'] = label0\_p\_list\_list

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

# step\_4\_statics11.py

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

'''

目的：从jd\_xx.csv中按照 【月份 日期 小时】 统计 [金牌会员 银牌会员 钻石会员 PLUS会员 铜牌会员 PLUS会员[试用] 企业会员 注册会员]

的数量

顺序：['售后原因','营销原因','产品原因','物流原因']

格式：

user\_list\_reverse = ['注册会员', '企业会员', 'PLUS会员[试用]', '铜牌会员', 'PLUS会员', '钻石会员', '银牌会员', '金牌会员']

月份： result\_months=[1, 2，..., 12] result\_user\_count=[[一月reverse【金牌会员 银牌会员 钻石会员 PLUS会员 铜牌会员 PLUS会员[试用] 企业会员 注册会员】 的差评统计]],...]

日期: result\_result\_date=[日期从小到大排列] result\_date\_count=[每日的差评数量]

时间 result\_hour=[1, 2，..., 23] result\_hour\_count[每小时的差评数量]

user\_list\_reverse = ['注册会员', '企业会员', 'PLUS会员[试用]', '铜牌会员', 'PLUS会员', '钻石会员', '银牌会员', '金牌会员']

outfile\_dic['count\_month\_' + i] = result\_user\_count[i]

outfile\_dic['result\_date'] = result\_date

outfile\_dic['result\_date\_count'] = result\_date\_count

outfile\_dic['result\_hour\_count'] = result\_hour\_count[i]

outfile\_dic['user\_list'] = user\_list\_reverse

'''

import csv

import time

def is\_date(str):

'''判断是否是一个有效的日期字符串'''

try:

time.strptime(str, "%Y-%m-%d")

return True

except:

return False

import configparser

config = configparser.ConfigParser()

config.read('parameter\_ini.ini')

# from a file get the json data.

itemname = config.get("statics","itemname")

file\_prefix = config.get("statics","file\_prefix")

file\_path\_prefix = config.get("statics","file\_path\_prefix")

csv\_time\_name = config.get("statics","csv\_time\_name")

print(csv\_time\_name)

user\_list = ['金牌会员', '银牌会员', '钻石会员', 'PLUS会员', '铜牌会员', 'PLUS会员[试用]', '企业会员', '注册会员']

user\_list\_reverse = []

#反转user\_list 按月份作图从少的用户开始、按小时作图也从少的用户开始

for i in range(len(user\_list)):

user\_list\_reverse.append(user\_list[len(user\_list)-i-1])

# print(user\_list\_reverse)

'''按月份'''

months\_list = ['1月', '2月', '3月','4月', '5月','6月', '7月', '8月', '9月', '10月', '11月', '12月']

#用于比较

months\_list\_2 = [1,2,3,4,5,6,7,8,9,10,11,12]

result\_months = months\_list

# 初始化格式

result\_user\_count = [([0]\*len(months\_list)) for i in range(len(user\_list))]

# print(result\_user\_count)

'''按日期'''

result\_date\_count\_map={}

result\_date=[]

result\_date\_count=[]

'''按小时'''

hours\_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]

result\_hour = []

result\_hour\_count = [([0]\*len(hours\_list)) for i in range(len(user\_list))]

flag=0#标志第一行

i=0

import json

#写入文件

outfile\_path\_prefix = config.get("statics","outfile\_path\_prefix")

outfile\_name = 'statics\_'+ itemname + '\_11.json'

outfile\_name\_withprefix = outfile\_path\_prefix + outfile\_name

print(outfile\_name\_withprefix)

outfile\_dic ={}

with open(csv\_time\_name,'r') as csvfile:

reader = csv.reader(csvfile)

for line in reader:

#跳过第一行

if(flag==0):

flag = 1

print(line[6])

print(line)

continue

'''按月份统计'''

if(line[6].isdigit()):

temp\_month\_index = months\_list\_2.index(int(line[6]))

temp\_user\_index = user\_list\_reverse.index(line[2])

# print(temp\_user\_index)

result\_user\_count[temp\_user\_index][temp\_month\_index] += 1

'''按日期'''

if (is\_date(line[3])):

if(line[3] in result\_date\_count\_map.keys()):

result\_date\_count\_map[line[3]] += 1

else:

result\_date\_count\_map[line[3]] = 1

'''按小时统计'''

if (line[7].isdigit()):

temp\_hour\_index = hours\_list.index(int(line[7]))

temp\_user\_index = user\_list\_reverse.index(line[2])

# print(temp\_user\_index)

result\_hour\_count[temp\_user\_index][temp\_hour\_index] += 1

'''按月份统计输出'''

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

print('按月份统计输出')

for i in range(len(user\_list\_reverse)):

print(user\_list\_reverse[i])

print(result\_user\_count[i])

outfile\_dic['count\_month\_' + str(i)] = result\_user\_count[i]

'''按日期统计并输出'''

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

print('按日期统计输出')

result\_date\_count\_map\_sorted\_list = sorted(result\_date\_count\_map.items(), key=lambda x: x[0], reverse=False)

for item in result\_date\_count\_map\_sorted\_list:

result\_date.append(item[0])

result\_date\_count.append(item[1])

print(result\_date)

print(result\_date\_count)

outfile\_dic['result\_date'] = result\_date

outfile\_dic['result\_date\_count'] = result\_date\_count

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

print('按小时统计输出')

for i in range(len(user\_list\_reverse)):

print(user\_list\_reverse[i])

print(result\_hour\_count[i])

outfile\_dic['result\_hour\_count'] = result\_hour\_count

#统计24小时的差评总和

outfile\_dic['result\_hour\_count\_total'] = []

for i in range(24):

temp\_hour\_sum = 0

for j in range(len(result\_hour\_count)):

temp\_hour\_sum += result\_hour\_count[j][i]

outfile\_dic['result\_hour\_count\_total'].append(temp\_hour\_sum)

print(user\_list\_reverse)

print(outfile\_dic['result\_hour\_count\_total'])

outfile\_dic['user\_list'] = user\_list\_reverse

#

with open(outfile\_name\_withprefix, 'w+', encoding='utf-8') as f:

f.write(json.dumps(outfile\_dic, indent=4, ensure\_ascii=False))

{#base.html#}

{% extends "bootstrap/base.html" %}

{% block title %}“鹰眼”电商问题发现平台V1.0{% endblock %}

{% block head %}

{{ super() }}

<link rel="shortcut icon" href="{{ url\_for('static', filename='favicon.ico') }}" type="image/x-icon">

<link rel="icon" href="{{ url\_for('static', filename='eagle.ico') }}" type="image/x-icon">

{% endblock %}

{% block navbar %}

<div class="navbar navbar-inverse" role="navigation">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="{{ url\_for('hello\_world') }}" style="color: #f6f6f6;font-size: 25px;padding:0px 15px">

<img src={{ url\_for('static',filename = 'eagle.png')}}>

</a>

</div>

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li><a href="{{ url\_for('hello\_world') }}" style="color: #f6f6f6">系统主页</a></li>

</ul>

<ul class="nav navbar-nav">

<li><a href="{{ url\_for('test') }}" style="color: #f6f6f6">评论标注</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

{% if current\_user.is\_authenticated %}

<li><a href="{{ url\_for('logout') }}" style="color: #f6f6f6">

<img style="padding-right: 3px;padding-bottom: 4px" src={{ url\_for('static',filename = 'logout.png')}}>注销

</a>

</li>

{% else %}

<li><a href="{{ url\_for('signup') }}" style="color: #f6f6f6">

<img style="padding-right: 3px;padding-bottom: 4px" src={{ url\_for('static',filename = 'signup.png')}}>注册

</a>

</li>

<li><a href="{{ url\_for('login') }}" style="color: #f6f6f6">

<img style="padding-right: 3px;padding-bottom: 4px" src={{ url\_for('static',filename = 'login.png')}}>登录

</a>

</li>

{% endif %}

</ul>

<ul class="nav navbar-nav">

<li><a href="{{ url\_for('statics') }}" style="color: #f6f6f6">质量分析</a></li>

</ul>

</div>

</div>

</div>

{% endblock %}

{% block content %}

<div class="container">

{% for message in get\_flashed\_messages() %}

<div class="alert alert-warning">

<button type="button" class="close" data-dismiss="alert">&times;</button>

{{ message }}

</div>

{% endfor %}

{% block page\_content %}{% endblock %}

</div>

{% endblock %}

{% block scripts %}

{{ super() }}

{% endblock %}

{#statics.html#}

{% extends "base.html" %}

{% block head %}

{{ super() }}

<link rel="stylesheet" type="text/css" href="/static/style2.css">

<script src="/static/jquery-1.12.4.min.js"></script>

<script src="/static/jquery-labelauty.js"></script>

<script type="text/javascript" src="/static/echarts.js"></script>

<link rel="stylesheet" href="/static/jquery-labelauty.css">

<script type="text/javascript">

$(function(){

var \_this1=null;

$('.nav1>li').hover(function(){

\_this1=$(this);

\_this1.find('.second-nav').show();

var \_this2=null;

\_this1.find('.second-nav').find('li').hover(function(){

\_this2=$(this);

\_this2.find('.third-nav').show();

\_this2.find('.third-nav').hover(function(){

$(this).show();

},function(){

$(this).hide();

});

},function(){

\_this2.find('.third-nav').hide();

});

},function(){

\_this1.find('.second-nav').hide();

});

});

</script>

{% endblock %}

{% block page\_content %}

<div class="header1">

<ul class="nav1">

<li><a href="{{ url\_for('statics\_graph',filename = 'statics\_phone\_1') }}">手机行业</a>

</li>

<li><a href="{{ url\_for('statics\_graph',filename = 'statics\_laptop\_1') }}">笔记本电脑行业</a>

</li>

<li><a href="{{ url\_for('statics\_graph',filename = 'statics\_air\_conditioner\_1') }}">空调行业</a>

<li><a href="#">微波炉行业</a>

</li>

<li><a href="#">电视行业</a>

</li>

<li><a href="#">果汁机行业</a>

</li>

<li><a href="#">其他行业</a>

</li>

</ul>

</div>

{% endblock %}

{#statics\_1.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8" xmlns="http://www.w3.org/1999/html" xmlns="http://www.w3.org/1999/html"

xmlns="http://www.w3.org/1999/html">

<script type="text/javascript" src="/static/echarts.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_2') }}">各品牌差评占比</a>

</div>

<center>

<div id="main" style="width: 600px;height:600px;"></div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('main'));

var data = genData();

option = {

title : {

text: '{{ title\_itemname }}' + ' 总体差评因素分布图',

subtext:'样本容量：{{ json\_alldata['sum'] }}',

x:'center'

},

tooltip : {

trigger: 'item',

formatter: "{a} <br/>{b} : {c} ({d}%)"

},

legend: {

type: 'scroll',

orient: 'vertical',

right: 10,

top: 100,

bottom: 50,

data: data.legendData

},

series : [

{

name: '差评因素',

type: 'pie',

radius : '55%',

center: ['40%', '50%'],

data: data.seriesData,

itemStyle: {

emphasis: {

shadowBlur: 10,

shadowOffsetX: 0,

shadowColor: 'rgba(0, 0, 0, 0.5)'

}

}

}

]

};

function genData() {

{#字符 空格 ' "" 等被转义成其他字符 防止js注入 利用js的tojson #}

var legendData = {{ json\_alldata['factor']|tojson }};

var seriesData\_count = {{ json\_alldata['result'] }};

var seriesData = []

for (var i = 0; i < 4; i++) {

seriesData.push({

name: legendData[i],

value: seriesData\_count[i]

});

}

return {

legendData: legendData,

seriesData: seriesData

};

}

myChart.setOption(option);

</script>

</center>

</div>

{% endblock %}

{#statics\_11.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

<script>

var myChart;

function changeData(self){

if ($("#mySelect").val()==2){

var myChart = echarts.init(document.getElementById('main'));

myChart.setOption(option\_date,true);

}

else if ($("#mySelect").val()==1){

var myChart = echarts.init(document.getElementById('main'));

myChart.setOption(option\_month,true);

}

else if ($("#mySelect").val()==3) {

var myChart = echarts.init(document.getElementById('main'));

myChart.setOption(option\_hour,true);

}

}

</script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_2') }}">各品牌差评占比</a>

</div>

<center>

<select id = 'mySelect' onchange = 'changeData(this)'>

<option value='1'>按月份</option>

<option value='2'>按日期</option>

<option value='3'>按小时</option>

</select>

<div id="main" style="width: 1000px;height:650px;"></div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('main'));

var user\_list = {{json\_alldata['user\_list']|tojson}}

for (var i=0;i<user\_list.length;i++)

{

user\_list[i] = user\_list[i].replace("\"","").replace("\"","")

}

option\_month = {

title: {

text: '{{ title\_itemname }}' + ' 每月用户差评分布图',

x:'center'

},

tooltip : {

trigger: 'axis',

axisPointer: {

type: 'cross',

label: {

backgroundColor: '#6a7985'

}

}

},

legend: {

data:['','','',{name: user\_list[7],textStyle:{color:'#ca0035'}},

{name: user\_list[6],textStyle:{color:'#660099'}},

{name: user\_list[5],textStyle:{color:'#ff9933'}},

{name: user\_list[4],textStyle:{color:'#ffde33'}},

{name: user\_list[3],textStyle:{color:'#2e629d'}},

{name: user\_list[2],textStyle:{color:'#009c2f'}},

{name: user\_list[1],textStyle:{color:'#008f9c'}},

{name: user\_list[0],textStyle:{color:'#97929c'}}

]

},

grid: {

left: '3%',

right: '4%',

bottom: '3%',

containLabel: true

},

xAxis : [

{

type : 'category',

boundaryGap : false,

data : ['1月', '2月', '3月','4月', '5月','6月', '7月', '8月', '9月', '10月', '11月', '12月'],

name:'月份'

}

],

yAxis : [

{

type : 'value'

}

],

series : [

{

name: user\_list[0],

type:'line',

itemStyle: {

normal: {

color: '#97929c'

}

},

data: {{ json\_alldata['count\_month\_0']|tojson }}

},

{

name: user\_list[1],

type:'line',

itemStyle: {

normal: {

color: '#008f9c'

}

},

data:{{ json\_alldata['count\_month\_1']|tojson }}

},

{

name: user\_list[2],

type:'line',

itemStyle: {

normal: {

color: '#009c2f'

}

},

data:{{ json\_alldata['count\_month\_2']|tojson }}

},

{

name: user\_list[3],

type:'line',

itemStyle: {

normal: {

color: '#2e629d'

}

},

data:{{ json\_alldata['count\_month\_3']|tojson }}

},

{

name: user\_list[4],

type:'line',

itemStyle: {

normal: {

color: '#ffde33'

}

},

data:{{ json\_alldata['count\_month\_4']|tojson }}

},

{

name: user\_list[5],

type:'line',

itemStyle: {

normal: {

color: '#ff9933'

}

},

data:{{ json\_alldata['count\_month\_5']|tojson }}

},

{

name: user\_list[6],

type:'line',

itemStyle: {

normal: {

color: '#660099'

}

},

data:{{ json\_alldata['count\_month\_6']|tojson }}

},

{

name: user\_list[7],

type:'line',

itemStyle: {

normal: {

color: '#ca0035'

}

},

data:{{ json\_alldata['count\_month\_7']|tojson }}

}

]

};

option\_date = {

title: {

text: '{{ itemname }}' + '日差评折线图',

x:'center'

},

tooltip: {

trigger: 'axis'

},

xAxis: {

data: {{ json\_alldata['result\_date']|tojson }}

},

yAxis: {

splitLine: {

show: true

}

},

dataZoom: [{

startValue: {{ json\_alldata['result\_date'][0]|tojson }}

}, {

type: 'inside'

}],

series: [

{

name: '日差评数',

type: 'line',

data: {{ json\_alldata['result\_date\_count']|tojson }},

itemStyle: {

normal: {

color: '#418ada'

}

}

}

]

};

option\_hour = {

title: {

text: '{{ itemname }}' + ' 行业差评小时分布图',

x:'center'

},

tooltip: {

trigger: 'axis'

},

xAxis: {

data: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]

},

yAxis: {

splitLine: {

show: true

}

},

legend: {

data:['','','',{name: user\_list[7],textStyle:{color:'#ca0035'}},

{name: user\_list[6],textStyle:{color:'#660099'}},

{name: user\_list[5],textStyle:{color:'#ff9933'}},

{name: user\_list[4],textStyle:{color:'#ffde33'}},

{name: user\_list[3],textStyle:{color:'#2e629d'}},

{name: user\_list[2],textStyle:{color:'#009c2f'}},

{name: user\_list[1],textStyle:{color:'#008f9c'}},

{name: user\_list[0],textStyle:{color:'#97929c'}}

]

},

series: [

{

name: '总和',

type: 'bar',

barGap: '-100%',

label: {

normal: {

textStyle: {

color: '#682d19'

},

position: 'left',

show: false,

formatter: '{b}'

}

},

itemStyle: {

normal: {

color: '#E5F9FB',

borderWidth: 2,

borderColor: '#1FBCD2'

}

},

data:{{ json\_alldata['result\_hour\_count\_total']|tojson }}

},

{

name: user\_list[0],

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

itemStyle: {

normal: {

color: '#97929c'

}

},

data:{{ json\_alldata['result\_hour\_count'][0]|tojson }}

},

{

name: user\_list[1],

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

itemStyle: {

normal: {

color: '#008f9c'

}

},

data: {{ json\_alldata['result\_hour\_count'][1]|tojson }}

},

{

name: user\_list[2],

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

itemStyle: {

normal: {

color: '#009c2f'

}

},

data: {{ json\_alldata['result\_hour\_count'][2]|tojson }}

},

{

name: user\_list[3],

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

itemStyle: {

normal: {

color: '#2e629d'

}

},

data: {{ json\_alldata['result\_hour\_count'][3]|tojson }}

},

{

name: user\_list[4],

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

itemStyle: {

normal: {

color: '#ffde33'

}

},

data: {{ json\_alldata['result\_hour\_count'][4]|tojson }}

},

{

name: user\_list[5],

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

itemStyle: {

normal: {

color: '#ff9933'

}

},

data: {{ json\_alldata['result\_hour\_count'][5]|tojson }}

},

{

name: user\_list[6],

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

itemStyle: {

normal: {

color: '#660099'

}

},

data: {{ json\_alldata['result\_hour\_count'][6]|tojson }}

},

{

name: user\_list[7],

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

itemStyle: {

normal: {

color: '#ca0035'

}

},

data: {{ json\_alldata['result\_hour\_count'][7]|tojson }}

}

]

};

myChart.setOption(option\_month,true)

</script>

</center>

</div>

{% endblock %}

{#statics\_12.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

<script>

var myChart;

function changeData(self){

if ($("#mySelect").val()==1){

myChart.setOption(option);

}

else if ($("#mySelect").val()==1993092){

myChart.setOption(option\_1993092,true)

}

else if ($("#mySelect").val()==1993087){

myChart.setOption(option\_1993087,true)

}

else if ($("#mySelect").val()==2942414){

myChart.setOption(option\_2942414,true)

}

else if ($("#mySelect").val()==1253197){

myChart.setOption(option\_1253197,true)

}

}

</script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index0']) }}">总体情况</a>

</br>

{#<a href="{{ url\_for('statics\_graph',filename = 'statics\_ac\_22') }}">主观倾向</a>#}

{#</br>#}

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index3']) }}">各型号占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index']) }}">差评的时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index3']) }}">差评的用户分布</a>

</div>

<center>

<script>

var product\_list = {{json\_alldata['product\_list']|tojson}}

for (var i=0;i<product\_list.length;i++)

{

product\_list[i] = product\_list[i].replace("\"","").replace("\"","")

}

var brand = {{ json\_alldata['brand']|tojson }}.replace("\"","").replace("\"","")

</script>

<select id = 'mySelect' onchange = 'changeData(this)'>

<option value='1'>全部产品</option>

<option value='1993092'>产品:{{ json\_alldata['product\_list'][0]|tojson }}</option>

<option value='1993087'>产品:{{ json\_alldata['product\_list'][1]|tojson }}</option>

<option value='2942414'>产品:{{ json\_alldata['product\_list'][2]|tojson }}</option>

<option value='1253197'>产品:{{ json\_alldata['product\_list'][3]|tojson }}</option>

</select>

<div id="main" style="width: 1100px;height:600px;"></div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('main'));

option = {

title: {

text: brand + '差评的时间分布',

x:'center'

},

tooltip: {

trigger: 'axis'

},

xAxis: {

data: {{ json\_alldata['all\_date\_list']|tojson }}

},

yAxis: {

splitLine: {

show: true

}

},

dataZoom: [{

startValue: {{ json\_alldata['all\_count\_list'][0]|tojson }}

}, {

type: 'inside'

}],

series: [

{

name: '日差评数',

type: 'line',

data: {{ json\_alldata['all\_count\_list']|tojson }},

itemStyle: {

normal: {

color: '#418ada'

}

}

}

]

};

option\_1993092 = {

title: {

text: '产品:'+product\_list[0]+'差评数量的时间分布'

},

tooltip: {

trigger: 'axis'

},

xAxis: {

data: {{ json\_alldata['date\_list\_0']|tojson }}

},

yAxis: {

minInterval:3,

splitLine: {

show: true

}

},

dataZoom: [{

startValue: {{ json\_alldata['date\_list\_0'][0]|tojson }}

}, {

type: 'inside'

}],

series: [

{

name: '日差评数',

type: 'line',

data: {{ json\_alldata['count\_list\_0']|tojson }},

itemStyle: {

normal: {

color: '#418ada'

}

}

}

]

};

option\_1993087 = {

title: {

text: '产品:' + product\_list[1] + '差评数量的时间分布'

},

tooltip: {

trigger: 'axis'

},

xAxis: {

data: {{ json\_alldata['date\_list\_1']|tojson }}

},

yAxis: {

minInterval:3,

splitLine: {

show: true

}

},

dataZoom: [{

startValue: {{ json\_alldata['date\_list\_1'][1]|tojson }}

}, {

type: 'inside'

}],

series: [

{

name: '日差评数',

type: 'line',

data: {{ json\_alldata['count\_list\_0']|tojson }},

itemStyle: {

normal: {

color: '#418ada'

}

}

}

]

};

option\_2942414 = {

title: {

text: '产品:' + product\_list[2]+ '差评数量的时间分布'

},

tooltip: {

trigger: 'axis'

},

xAxis: {

data: {{ json\_alldata['date\_list\_2']|tojson }}

},

yAxis: {

minInterval:4,

splitLine: {

show: true

}

},

dataZoom: [{

startValue: {{ json\_alldata['date\_list\_2'][0]|tojson }}

}, {

type: 'inside'

}],

series: [

{

name: '日差评数',

type: 'line',

data: {{ json\_alldata['count\_list\_2']|tojson }},

itemStyle: {

normal: {

color: '#418ada'

}

}

}

]

};

option\_1253197 = {

title: {

text: '产品:' + product\_list[3] +'差评数量的时间分布'

},

tooltip: {

trigger: 'axis'

},

xAxis: {

data: {{ json\_alldata['date\_list\_3']|tojson }},

itemStyle: {

normal: {

color: '#418ada'

}

},

},

yAxis: {

minInterval:4,

splitLine: {

show: true

}

},

dataZoom: [{

startValue: {{ json\_alldata['date\_list\_3'][0]|tojson }}

}, {

type: 'inside'

}],

series: [

{

name: '日差评数',

type: 'line',

data: {{ json\_alldata['count\_list\_3']|tojson }},

itemStyle: {

normal: {

color: '#418ada'

}

}

}

]

};

myChart.setOption(option)

</script>

</center>

</div>

{% endblock %}

{#statics\_16.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8" xmlns="http://www.w3.org/1999/html">

<script type="text/javascript" src="/static/echarts.js"></script>

{#echarts统计拓展模块#}

<script type="text/javascript" src="/static/ecStat.min.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div id=url\_info\_l>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_16') }}">各原因用户占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_25') }}">各用户差评占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_24') }}">用户差评热力图</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_23') }}">用户主观倾向</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_22') }}">用户主观倾向箱线图</a>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_2') }}">各品牌差评占比</a>

</div>

<center>

<h4>&nbsp&nbsp&nbsp&nbsp&nbsp单原因用户分布图</h4>

</br>

</center>

<div class="parent">

<div class="row">

<div id="value\_4" class="item" style="width: 550px;height: 360px;"></div>

<div id="value\_3" class="item" style="width: 550px;height: 360px;"></div>

</div>

<div class="row">

<div id="value\_2" class="item" style="width: 550px;height: 360px;"></div>

<div id="value\_1" class="item" style="width: 550px;height: 360px;"></div>

</div>

</div>

<script type="text/javascript">

var myChart\_value\_4 = echarts.init(document.getElementById('value\_4'));

var myChart\_value\_3 = echarts.init(document.getElementById('value\_3'));

var myChart\_value\_2 = echarts.init(document.getElementById('value\_2'));

var myChart\_value\_1 = echarts.init(document.getElementById('value\_1'));

var factor\_list = {{json\_alldata['factor']|tojson}}

for (var i=0;i<factor\_list.length;i++)

{

factor\_list[i] = factor\_list[i].replace("\"","").replace("\"","")

}

var option\_value\_4 = {

title : {

text: factor\_list[0],

x:'center'

},

tooltip : {

trigger: 'item',

formatter: "{a} <br/>{b} : {c} ({d}%)"

},

legend: {

type: 'scroll',

orient: 'vertical',

right: 10,

top: 100,

bottom: 50,

data: {{ json\_alldata['userlevel\_list']|tojson }}

},

series : [

{

name: '用户类型',

type: 'pie',

radius : '55%',

center: ['40%', '50%'],

data:{{ json\_alldata['dic\_list\_list'][0]|tojson }},

itemStyle: {

emphasis: {

shadowBlur: 10,

shadowOffsetX: 0,

shadowColor: 'rgba(0, 0, 0, 0.5)'

}

}

}

]

};

var option\_value\_3 = {

title : {

text: factor\_list[1],

x:'center'

},

tooltip : {

trigger: 'item',

formatter: "{a} <br/>{b} : {c} ({d}%)"

},

legend: {

type: 'scroll',

orient: 'vertical',

right: 10,

top: 100,

bottom: 50,

data: {{ json\_alldata['userlevel\_list']|tojson }}

},

series : [

{

name: '用户类型',

type: 'pie',

radius : '55%',

center: ['40%', '50%'],

data:{{ json\_alldata['dic\_list\_list'][1]|tojson }},

itemStyle: {

emphasis: {

shadowBlur: 10,

shadowOffsetX: 0,

shadowColor: 'rgba(0, 0, 0, 0.5)'

}

}

}

]

};

var option\_value\_2 = {

title : {

text: factor\_list[2],

x:'center'

},

tooltip : {

trigger: 'item',

formatter: "{a} <br/>{b} : {c} ({d}%)"

},

legend: {

type: 'scroll',

orient: 'vertical',

right: 10,

top: 100,

bottom: 50,

data: {{ json\_alldata['userlevel\_list']|tojson }}

},

series : [

{

name: '用户类型',

type: 'pie',

radius : '55%',

center: ['40%', '50%'],

data:{{ json\_alldata['dic\_list\_list'][2]|tojson }},

itemStyle: {

emphasis: {

shadowBlur: 10,

shadowOffsetX: 0,

shadowColor: 'rgba(0, 0, 0, 0.5)'

}

}

}

]

};

var option\_value\_1 = {

title : {

text: factor\_list[3],

x:'center'

},

tooltip : {

trigger: 'item',

formatter: "{a} <br/>{b} : {c} ({d}%)"

},

legend: {

type: 'scroll',

orient: 'vertical',

right: 10,

top: 100,

bottom: 50,

data: {{ json\_alldata['userlevel\_list']|tojson }}

},

series : [

{

name: '用户类型',

type: 'pie',

radius : '55%',

center: ['40%', '50%'],

data:{{ json\_alldata['dic\_list\_list'][3]|tojson }},

itemStyle: {

emphasis: {

shadowBlur: 10,

shadowOffsetX: 0,

shadowColor: 'rgba(0, 0, 0, 0.5)'

}

}

}

]

};

myChart\_value\_4.setOption(option\_value\_4);

myChart\_value\_3.setOption(option\_value\_3);

myChart\_value\_2.setOption(option\_value\_2);

myChart\_value\_1.setOption(option\_value\_1);

</script>

</div>

{% endblock %}

{#statics\_17.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div id=url\_info\_l>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_16') }}">各原因用户占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_25') }}">各用户差评占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_24') }}">用户差评热力图</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_23') }}">用户主观倾向</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_22') }}">用户主观倾向箱线图</a>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_2') }}">各品牌差评占比</a>

</div>

<center>

<div id="main" style="width: 700px;height:600px;"></div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('main'));

var data = genData();

option = {

title : {

text: '给予差评的用户分布图',

subtext:'样本容量：{{ json\_alldata['sum'] }}',

x:'center'

},

tooltip : {

trigger: 'item',

formatter: "{a} <br/>{b} : {c} ({d}%)"

},

legend: {

type: 'scroll',

orient: 'vertical',

right: 10,

top: 100,

bottom: 50,

data: data.legendData

},

series : [

{

name: '差评因素',

type: 'pie',

radius : '55%',

center: ['40%', '50%'],

data: data.seriesData,

itemStyle: {

emphasis: {

shadowBlur: 10,

shadowOffsetX: 0,

shadowColor: 'rgba(0, 0, 0, 0.5)'

}

}

}

]

};

function genData() {

var legendData = {{ json\_alldata['user\_list']|tojson }};

var seriesData\_count = {{ json\_alldata['user\_count']|tojson }};

var seriesData = []

for (var i = 0; i < 8; i++) {

seriesData.push({

name: legendData[i],

value: seriesData\_count[i]

});

}

return {

legendData: legendData,

seriesData: seriesData

};

}

myChart.setOption(option);

</script>

</center>

</div>

{% endblock %}

{#statics\_18.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index0']) }}">总体情况</a>

</br>

{#<a href="{{ url\_for('statics\_graph',filename = 'statics\_ac\_22') }}">主观倾向</a>#}

{#</br>#}

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index2']) }}">各型号占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index3']) }}">差评的时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index']) }}">差评的用户分布</a>

</div>

<center>

<div id="main" style="width: 700px;height:600px;"></div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('main'));

var data = genData();

var brand = {{ json\_alldata['brand']|tojson }};

brand = brand.replace("\"","").replace("\"","")

var user\_list = {{json\_alldata['user\_list']|tojson}}

for (var i=0;i<user\_list.length;i++)

{

user\_list[i] = user\_list[i].replace("\"","").replace("\"","")

}

option = {

title : {

text: '给予' + brand + '差评的用户分布图',

subtext:'样本容量：' + '{{ json\_alldata['totalcount'] }}',

x:'center'

},

tooltip : {

trigger: 'item',

formatter: "{a} <br/>{b} : {c} ({d}%)"

},

legend: {

type: 'scroll',

orient: 'vertical',

right: 10,

top: 100,

bottom: 50,

data: data.legendData

},

series : [

{

name: '差评因素',

type: 'pie',

radius : '55%',

center: ['40%', '50%'],

data: data.seriesData,

itemStyle: {

emphasis: {

shadowBlur: 10,

shadowOffsetX: 0,

shadowColor: 'rgba(0, 0, 0, 0.5)'

}

}

}

]

};

function genData() {

var legendData = {{ json\_alldata['user\_list']|tojson }};

var seriesData\_count = {{ json\_alldata['count\_list'] }};

var seriesData = []

for (var i = 0; i < legendData.length; i++) {

seriesData.push({

name: legendData[i],

value: seriesData\_count[i]

});

}

return {

legendData: legendData,

seriesData: seriesData

};

}

myChart.setOption(option);

</script>

</center>

</div>

{% endblock %}

{#statics\_2.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_2') }}">各品牌差评占比</a>

</div>

<center>

<div id="main" style="width: 700px;height:800px;"></div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('main'));

var data = genData();

option = {

title : {

text: '{{ title\_itemname }}' + ' 总体各品牌差评占比图',

subtext:'样本容量：' + {{ json\_alldata['sum'] }},

x:'center'

},

tooltip : {

trigger: 'item',

formatter: "{a} <br/>{b} : {c} ({d}%)"

},

legend: {

type: 'scroll',

orient: 'vertical',

right: 20,

top: 100,

bottom: 1245.5,

data: data.legendData

},

series : [

{

name: '品牌',

type: 'pie',

radius : '55%',

center: ['40%', '50%'],

data: data.seriesData,

itemStyle : {

normal : {

label : {

show : false //隐藏标示文字

},

labelLine : {

show : false //隐藏标示线

}

}

}

}

]

};

function genData() {

var legendData = {{ json\_alldata['brand']|tojson }};

var seriesData\_count = {{ json\_alldata['count']|tojson }}

var seriesData = [];

for (var i = 0; i < legendData.length; i++) {

seriesData.push({

name: legendData[i],

value: seriesData\_count[i]

});

}

return {

legendData: legendData,

seriesData: seriesData

};

}

myChart.setOption(option);

myChart.on('click', function (param){

var name=param.name;

if(name=={{ json\_alldata['brand'][0]|tojson }}){

window.location.href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_3') }}";

}else if(name=={{ json\_alldata['brand'][1]|tojson }}){

window.location.href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_4') }}";

}else if(name=={{ json\_alldata['brand'][2]|tojson }}){

window.location.href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_5') }}";

}else if(name=={{ json\_alldata['brand'][3]|tojson }}){

window.location.href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_6') }}";

}

});

myChart.on('click',myChart);

</script>

</center>

</div>

{% endblock %}

{#statics\_22.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

{#echarts统计拓展模块#}

<script type="text/javascript" src="/static/ecStat.min.js"></script>

<script type="text/javascript" src="/static/dataTool.min.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div>

<div id=url\_info\_l>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_16') }}">各原因用户占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_25') }}">各用户差评占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_24') }}">用户差评热力图</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_23') }}">用户主观倾向</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_22') }}">用户主观倾向箱线图</a>

</div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_11\_v2') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_2') }}">各品牌差评占比</a>

</div>

<center>

<h4>&nbsp&nbsp&nbsp&nbsp&nbsp用户主观倾向</h4>

</center>

<center>

<div id="value" style="width: 800px;height:600px;"></div>

</center>

</div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('value'));

var list\_data = [];

var itemname = "{{ itemname }}";

var data = echarts.dataTool.prepareBoxplotData(

list\_data,

{

layout: 'vertical'

});

option = {

title: [

{

text: '用户感情倾向指数箱形图',

x: 'center'

}

],

tooltip: {

trigger: 'item',

axisPointer: {

type: 'shadow'

}

},

grid: {

left: '10%',

right: '10%',

bottom: '15%'

},

yAxis: {

type: 'category',

data: {{ json\_alldata['axisData']|tojson }},

boundaryGap: true,

nameGap: 30,

splitArea: {

show: false

},

axisLabel: {

formatter: function(value){

var i = Number(value)+Number(1);

return 'score='+i

}

},

splitLine: {

show: false

}

},

xAxis: {

type: 'value',

name: '情感倾向',

splitArea: {

show: true

}

},

series: [

{

name: 'boxplot',

type: 'boxplot',

data: {{ json\_alldata['boxData']|tojson }},

tooltip: {

formatter: function (param) {

var j = Number(param.name)+Number(1);

return [

'score=' + j + ': ',

'upper: ' + param.data[5],

'Q3: ' + param.data[4],

'median: ' + param.data[3],

'Q1: ' + param.data[2],

'lower: ' + param.data[1]

].join('<br/>')

}

}

},

{

name: 'outlier',

type: 'scatter',

data:{{ json\_alldata['outliers']|tojson }}

}

]

};

myChart.setOption(option);

</script>

</div>

{% endblock %}

{#statics\_23.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8" xmlns="http://www.w3.org/1999/html">

<script type="text/javascript" src="/static/echarts.js"></script>

{#echarts统计拓展模块#}

<script type="text/javascript" src="/static/ecStat.min.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div>

<div id=url\_info\_l>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_16') }}">各原因用户占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_25') }}">各用户差评占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_24') }}">用户差评热力图</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_23') }}">用户主观倾向</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_22') }}">用户主观倾向箱线图</a>

</div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_2') }}">各品牌差评占比</a>

</div>

<center>

<h4>&nbsp&nbsp&nbsp&nbsp&nbsp用户主观倾向</h4>

</center>

<div class="parent">

<div class="row">

<div id="value\_5" class="item" style="width: 315px;height:292px;"></div>

<div id="value\_4" class="item" style="width: 315px;height:292px;"></div>

<div id="value\_3" class="item" style="width: 315px;height:292px;"></div>

</div>

</div>

<div id="value\_2" class="left" style="width: 315px;height:292px;"></div>

<div id="value\_1" class="right" style="width: 315px;height:292px;"></div>

</div>

<script type="text/javascript">

var myChart\_value\_5 = echarts.init(document.getElementById('value\_5'));

var myChart\_value\_4 = echarts.init(document.getElementById('value\_4'));

var myChart\_value\_3 = echarts.init(document.getElementById('value\_3'));

var myChart\_value\_2 = echarts.init(document.getElementById('value\_2'));

var myChart\_value\_1 = echarts.init(document.getElementById('value\_1'));

var option\_value\_5= {

// title:'5',

title : {

text: ' score=5',

x:'center'

},

color: ['#346fb1'],

tooltip : {

trigger: 'axis',

axisPointer : { // 坐标轴指示器，坐标轴触发有效

type : 'shadow' // 默认为直线，可选为：'line' | 'shadow'

}

},

grid: {

left: '3%',

right: '4%',

bottom: '3%',

containLabel: true

},

xAxis : [

{

type : 'category',

data :{{ json\_alldata['map\_key'] }},

axisLabel :{

interval: 49,

align:'center',

fontSize:20,

max:1

},

boundaryGap:false,

axisTick:{

alignWithLabel:true,

show:false

},

splitLine:{

show:true,

lineStyle:{

width:4

}

}

}

],

yAxis : [

{

type : 'value',

// axisLine : {onZero: true},

offset:2,

splitLine:{

show:true,

lineStyle:{

width:3

}

}

}

],

series : [

{

name:'倾向区间人数',

type:'bar',

barWidth: '100%',

data:{{ json\_alldata['score\_5'] }}

}

]

};

var option\_value\_4= {

title : {

text: ' score=4',

x:'center'

},

color: ['#346fb1'],

tooltip : {

trigger: 'axis',

axisPointer : { // 坐标轴指示器，坐标轴触发有效

type : 'shadow' // 默认为直线，可选为：'line' | 'shadow'

}

},

grid: {

left: '3%',

right: '4%',

bottom: '3%',

containLabel: true

},

xAxis : [

{

type : 'category',

data :{{ json\_alldata['map\_key'] }},

axisLabel :{

interval: 49,

align:'center',

fontSize:20,

max:1

},

boundaryGap:false,

axisTick:{

alignWithLabel:true,

show:false

},

splitLine:{

show:true,

lineStyle:{

width:4

}

}

}

],

yAxis : [

{

type : 'value',

// axisLine : {onZero: true},

offset:2,

splitLine:{

show:true,

lineStyle:{

width:3

}

}

}

],

series : [

{

name:'倾向区间人数',

type:'bar',

barWidth: '100%',

data:{{ json\_alldata['score\_4'] }}

}

]

};

var option\_value\_3= {

title : {

text: ' score=3',

x:'center'

},

color: ['#346fb1'],

tooltip : {

trigger: 'axis',

axisPointer : { // 坐标轴指示器，坐标轴触发有效

type : 'shadow' // 默认为直线，可选为：'line' | 'shadow'

}

},

grid: {

left: '3%',

right: '4%',

bottom: '3%',

containLabel: true

},

xAxis : [

{

type : 'category',

data :{{ json\_alldata['map\_key'] }},

axisLabel :{

interval: 49,

align:'center',

fontSize:20,

max:1

},

boundaryGap:false,

axisTick:{

alignWithLabel:true,

show:false

},

splitLine:{

show:true,

lineStyle:{

width:4

}

}

}

],

yAxis : [

{

type : 'value',

// axisLine : {onZero: true},

offset:2,

splitLine:{

show:true,

lineStyle:{

width:3

}

}

}

],

series : [

{

name:'倾向区间人数',

type:'bar',

barWidth: '100%',

data:{{ json\_alldata['score\_3'] }}

}

]

};

var option\_value\_2= {

title : {

text: ' score=2',

x:'center'

},

color: ['#346fb1'],

tooltip : {

trigger: 'axis',

axisPointer : { // 坐标轴指示器，坐标轴触发有效

type : 'shadow' // 默认为直线，可选为：'line' | 'shadow'

}

},

grid: {

left: '3%',

right: '4%',

bottom: '3%',

containLabel: true

},

xAxis : [

{

type : 'category',

data :{{ json\_alldata['map\_key'] }},

axisLabel :{

interval: 49,

align:'center',

fontSize:20,

max:1

},

boundaryGap:false,

axisTick:{

alignWithLabel:true,

show:false

},

splitLine:{

show:true,

lineStyle:{

width:4

}

}

}

],

yAxis : [

{

type : 'value',

// axisLine : {onZero: true},

offset:2,

splitLine:{

show:true,

lineStyle:{

width:3

}

}

}

],

series : [

{

name:'倾向区间人数',

type:'bar',

barWidth: '100%',

data:{{ json\_alldata['score\_2'] }}

}

]

};

var option\_value\_1= {

title : {

text: ' score=1',

x:'center'

},

color: ['#346fb1'],

tooltip : {

trigger: 'axis',

axisPointer : { // 坐标轴指示器，坐标轴触发有效

type : 'shadow' // 默认为直线，可选为：'line' | 'shadow'

}

},

grid: {

left: '3%',

right: '4%',

bottom: '3%',

containLabel: true

},

xAxis : [

{

type : 'category',

data :{{ json\_alldata['map\_key'] }},

axisLabel :{

interval: 49,

align:'center',

fontSize:20,

max:1

},

boundaryGap:false,

axisTick:{

alignWithLabel:true,

show:false

},

splitLine:{

show:true,

lineStyle:{

width:4

}

}

}

],

yAxis : [

{

type : 'value',

// axisLine : {onZero: true},

offset:2,

splitLine:{

show:true,

lineStyle:{

width:3

}

}

}

],

series : [

{

name:'倾向区间人数',

type:'bar',

barWidth: '100%',

data:{{ json\_alldata['score\_1'] }}

}

]

};

myChart\_value\_5.setOption(option\_value\_5);

myChart\_value\_4.setOption(option\_value\_4);

myChart\_value\_3.setOption(option\_value\_3);

myChart\_value\_2.setOption(option\_value\_2);

myChart\_value\_1.setOption(option\_value\_1);

</script>

</div>

{% endblock %}

{#statics\_24.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

{#echarts统计拓展模块#}

<script type="text/javascript" src="/static/ecStat.min.js"></script>

<script type="text/javascript" src="/static/dataTool.min.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div id=url\_info\_l>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_16') }}">各原因用户占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_25') }}">各用户差评占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_24') }}">用户差评热力图</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_23') }}">用户主观倾向</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_22') }}">用户主观倾向箱线图</a>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_2') }}">各品牌差评占比</a>

</div>

<center>

<h4>&nbsp&nbsp&nbsp用户类型与差评原因热力图</h4>

</center>

<center>

<div id="value" style="width: 800px;height:600px;"></div>

</center>

</div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('value'));

var hours = {{ json\_alldata['factor']|tojson }};

var days = {{ json\_alldata['userlevel\_list']|tojson }};

var data = {{ json\_alldata['data'] }};

var max = {{ json\_alldata['max'] }};

data = data.map(function (item) {

return [item[1], item[0], item[2] || '-'];

});

option = {

tooltip: {

position: 'top'

},

animation: false,

grid: {

height: '50%',

y: '10%'

},

xAxis: {

type: 'category',

data: hours,

splitArea: {

show: true

}

},

yAxis: {

type: 'category',

data: days,

splitArea: {

show: true

}

},

visualMap: {

min: 0,

max: max,

calculable: true,

orient: 'horizontal',

left: 'center',

bottom: '15%'

},

series: [{

name: '差评数量',

type: 'heatmap',

data: data,

label: {

normal: {

show: true,

}

},

itemStyle: {

emphasis: {

shadowBlur: 10,

shadowColor: 'rgba(0, 0, 0, 0.5)'

}

}

}]

};

myChart.setOption(option);

</script>

</div>

{% endblock %}

{#statics\_25.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

{#echarts统计拓展模块#}

<script type="text/javascript" src="/static/ecStat.min.js"></script>

<script type="text/javascript" src="/static/dataTool.min.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div id=url\_info\_l>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_16') }}">各原因用户占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_25') }}">各用户差评占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_24') }}">用户差评热力图</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_23') }}">用户主观倾向</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_22') }}">用户主观倾向箱线图</a>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_2') }}">各品牌差评占比</a>

</div>

<center>

<h4>&nbsp&nbsp&nbsp&nbsp&nbsp各类型用户差评的占比情况</h4>

</center>

<center>

<div id="value" style="width: 800px;height:600px;"></div>

</center>

</div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('value'));

option = {

tooltip : {

trigger: 'axis',

axisPointer : { // 坐标轴指示器，坐标轴触发有效

type : 'shadow' // 默认为直线，可选为：'line' | 'shadow'

}

},

legend: {

orient: 'vertical',

right: 0,

top: 100,

bottom: 50,

data: {{ json\_alldata['factor']|tojson }}

},

grid: {

left: '3%',

right: '15%',

bottom: '20%',

containLabel: true

},

xAxis: {

type: 'value',

splitNumber:5,

max:1,

},

yAxis: {

type: 'category',

data:{{ json\_alldata['userlevel\_list']|tojson }}

},

series: [

{

name: '售后原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data: {{ json\_alldata['result'][2]|tojson }}

},

{

name: '营销原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight',

}

},

data: {{ json\_alldata['result'][3]|tojson }}

},

{

name: '产品原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data: {{ json\_alldata['result'][1]|tojson }}

},

{

name: '物流原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data:{{ json\_alldata['result'][0]|tojson }}

}

]

};

myChart.setOption(option);

</script>

</div>

{% endblock %}

{#statics\_26.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

{#echarts统计拓展模块#}

<script type="text/javascript" src="/static/ecStat.min.js"></script>

<script type="text/javascript" src="/static/dataTool.min.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div id=url\_info\_l>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_27') }}">产品原因树状图 名词</a>

</br>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_2') }}">各品牌差评占比</a>

</div>

<center>

<h4>&nbsp&nbsp&nbsp&nbsp&nbsp产品原因树状图 形容词</h4>

</center>

<center>

<div id="value" style="width: 1200px;height:700px;"></div>

</center>

</div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('value'));

myChart.showLoading();

$.get('{{ url\_for('getData',filename = 'statics\_' + itemname +'\_tree\_a') }}', function (data) {

myChart.hideLoading();

echarts.util.each(data.children, function (datum, index) {

index % 1 === 0 && (datum.collapsed = true);

});

var feature\_comments = new Array();

feature\_comments = {{ json\_alldata|tojson }}

console.log(feature\_comments);

myChart.setOption(option = {

tooltip: {

trigger: 'item',

triggerOn: 'mousemove',

formatter: function(params){

console.log(!!feature\_comments[params.name]);

if(params.value === undefined)

{

return (params.name)

}else{

if(!!feature\_comments[params.name])

{

return (params.name + ': '+params.value + '</br>' + "<p style='text-align:left'>" + feature\_comments[params.name] + "</p>")

}else{

return (params.name + ': '+params.value)

}

}

}

},

series: [

{

type: 'tree',

data: [data],

top: '2%',

left: '15%',

bottom: '10%',

right: '20%',

symbolSize: 7,

label: {

normal: {

position: 'left',

verticalAlign: 'middle',

align: 'right',

fontSize: 15

}

},

leaves: {

label: {

normal: {

position: 'right',

verticalAlign: 'middle',

align: 'left'

}

}

},

expandAndCollapse: true,

animationDuration: 550,

animationDurationUpdate: 750

}

]

});

});

</script>

</div>

{% endblock %}

{#statics\_27.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

{#echarts统计拓展模块#}

<script type="text/javascript" src="/static/ecStat.min.js"></script>

<script type="text/javascript" src="/static/dataTool.min.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div id=url\_info\_l>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">产品原因树状图 形容词</a>

</br>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_1') }}">总体情况</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_11') }}">总体差评时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_17') }}">差评用户分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_26') }}">问题分析</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname +'\_2') }}">各品牌差评占比</a>

</div>

<center>

<h4>&nbsp&nbsp&nbsp&nbsp&nbsp产品原因树状图 名词</h4>

</center>

<center>

<div id="value" style="width: 1200px;height:700px;"></div>

</center>

</div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('value'));

myChart.showLoading();

$.get('{{ url\_for('getData',filename = 'statics\_' + itemname +'\_tree\_n') }}', function (data) {

myChart.hideLoading();

echarts.util.each(data.children, function (datum, index) {

index % 1 === 0 && (datum.collapsed = true);

});

var feature\_comments = new Array();

feature\_comments = {{ json\_alldata|tojson }}

console.log(feature\_comments);

myChart.setOption(option = {

tooltip: {

trigger: 'item',

triggerOn: 'mousemove',

formatter: function(params){

console.log(!!feature\_comments[params.name]);

if(params.value === undefined)

{

return (params.name)

}else{

if(!!feature\_comments[params.name])

{

return (params.name + ': '+params.value + '</br>' + "<p style='text-align:left'>" + feature\_comments[params.name] + "</p>")

}else{

return (params.name + ': '+params.value)

}

}

}

},

series: [

{

type: 'tree',

data: [data],

top: '2%',

left: '15%',

bottom: '10%',

right: '20%',

symbolSize: 7,

label: {

normal: {

position: 'left',

verticalAlign: 'middle',

align: 'right',

fontSize: 15

}

},

leaves: {

label: {

normal: {

position: 'right',

verticalAlign: 'middle',

align: 'left'

}

}

},

expandAndCollapse: true,

animationDuration: 550,

animationDurationUpdate: 750

}

]

});

});

</script>

</div>

{% endblock %}

{#statics\_3.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index']) }}">总体情况</a>

</br>

{#<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_22') }}">主观倾向</a>#}

{#</br>#}

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index2'])}}">各型号占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index3']) }}">差评的时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index4']) }}">差评的用户分布</a>

</br>

</div>

<center>

<div id="main" style="width: 600px;height:600px;"></div>

<script type="text/javascript">

var myChart = echarts.init(document.getElementById('main'));

var data = genData();

var title\_brand = {{ json\_alldata['brand']|tojson }}

title\_brand = title\_brand.replace("\"","").replace("\"","");

option = {

title : {

text: title\_brand +' 差评因素占比图',

subtext:'样本容量：' + ' {{ json\_alldata['sum']|tojson }} ',

x:'center'

},

tooltip : {

trigger: 'item',

formatter: "{a} <br/>{b} : {c} ({d}%)"

},

legend: {

type: 'scroll',

orient: 'vertical',

right: 10,

top: 100,

bottom: 50,

data: data.legendData

},

series : [

{

name: '差评因素',

type: 'pie',

radius : '55%',

center: ['40%', '50%'],

data: data.seriesData,

itemStyle: {

emphasis: {

shadowBlur: 10,

shadowOffsetX: 0,

shadowColor: 'rgba(0, 0, 0, 0.5)'

}

}

}

]

};

function genData() {

var legendData = {{ json\_alldata['factor']|tojson }};

var seriesData\_count = {{ json\_alldata['count']|tojson }};

var seriesData = []

for (var i = 0; i < legendData.length; i++) {

seriesData.push({

name: legendData[i],

value: seriesData\_count[i]

});

}

return {

legendData: legendData,

seriesData: seriesData

};

}

myChart.setOption(option);

</script>

</center>

</div>

{% endblock %}

{#statics\_7.html#}

{% extends "statics.html" %}

{% block head %}

{{ super() }}

<meta charset="utf-8">

<script type="text/javascript" src="/static/echarts.js"></script>

<script>

var myChart;

function changeData(self){

if ($("#mySelect").val()==1){

myChart.setOption(option);

window.location.reload(true)

}

else{

myChart.setOption(option\_2)

}

}

</script>

{% endblock %}

{% block page\_content %}

{{ super() }}

<div>

</br>

</div>

<div>

<div id=url\_info>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index0']) }}">总体情况</a>

</br>

{#<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_22') }}">主观倾向</a>#}

{#</br>#}

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index']) }}">各型号占比</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index2']) }}">差评的时间分布</a>

</br>

<a href="{{ url\_for('statics\_graph',filename = 'statics\_' + itemname + '\_' + json\_alldata['index3']) }}">差评的用户分布</a>

</div>

<center>

<select id = 'mySelect' onchange = 'changeData(this)'>

<option value='1'>百分比统计</option>

<option value='2'>数值统计</option>

</select>

<div id="main"></div>

<script type="text/javascript">

var myChartContainer = document.getElementById('main');

var resizemyChartContainer = function () {

myChartContainer.style.width = '1100px';

myChartContainer.style.height = window.innerHeight+'px';

};

resizemyChartContainer();

var myChart = echarts.init(myChartContainer);

var title\_brand = {{ json\_alldata['brand']|tojson }}

title\_brand = title\_brand.replace("\"","").replace("\"","");

option = {

title : {

text: title\_brand + '各型号差评因素累积占比图',

x: 'center'

},

tooltip : {

trigger: 'axis',

axisPointer : { // 坐标轴指示器，坐标轴触发有效

type : 'shadow' // 默认为直线，可选为：'line' | 'shadow'

},

formatter: function(params){

res = params[0].name+'</br>';

for(var j=0;j<params.length;j++){

res += params[j].seriesName+': '+params[j].value+'</br>'

}

return res

}

},

legend: {

orient: 'vertical',

right: 10,

top: 100,

bottom: 50,

data: {{ json\_alldata['factor']|tojson }}

},

grid: {

left: '3%',

right: '15%',

bottom: '20%',

containLabel: true

},

xAxis: {

type: 'value',

splitNumber:5,

max:1

},

yAxis: {

type: 'category',

showMinLabel:false,

showMaxLabel:true,

axisLabel:{

interval:0,

verticalAlign:'middle'

},

data: {{ json\_alldata['product\_list']|tojson }}

},

series: [

{

name: '售后原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data: {{ json\_alldata['count\_p\_label2']|tojson }}

},

{

name: '营销原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data: {{ json\_alldata['count\_p\_label3']|tojson }}

},

{

name: '产品原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data: {{ json\_alldata['count\_p\_label1']|tojson }}

},

{

name: '物流原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data: {{ json\_alldata['count\_p\_label0']|tojson }}

}

]

};

option\_2 = {

title : {

text: title\_brand +'各型号差评数量统计图',

x: 'center'

},

tooltip : {

trigger: 'axis',

axisPointer : { // 坐标轴指示器，坐标轴触发有效

type : 'shadow' // 默认为直线，可选为：'line' | 'shadow'

}

},

legend: {

orient: 'vertical',

right: 0,

top: 100,

bottom: 50,

data: {{ json\_alldata['factor']|tojson }}

},

grid: {

left: '3%',

right: '15%',

bottom: '20%',

containLabel: true

},

xAxis: {

type: 'value',

max:{{ json\_alldata['max'] }}

},

yAxis: {

type: 'category',

data: {{ json\_alldata['product\_list']|tojson }}

},

series: [

{

name: '售后原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data: {{ json\_alldata['count\_label2']|tojson }}

},

{

name: '营销原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight',

}

},

data: {{ json\_alldata['count\_label3']|tojson }}

},

{

name: '产品原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data: {{ json\_alldata['count\_label1']|tojson }}

},

{

name: '物流原因',

type: 'bar',

stack: '总量',

label: {

normal: {

show: false,

position: 'insideRight'

}

},

data: {{ json\_alldata['count\_label0']|tojson }}

}

]

};

myChart.setOption(option);

</script>

</center>

</div>

{% endblock %}