## 附录：

### 附录1：收货量+发货量汇总程序.py

import xlrd

import numpy as np

from openpyxl import Workbook

wb=xlrd.open\_workbook("C:\\Users\\Administrator\\Desktop\\vs工作界面\\附件1.xlsx")

ws=wb.sheet\_by\_index(0)

start,end,num=[],[],[]

##去掉头部的标签

start=ws.col\_values(1)

end=ws.col\_values(2)

num=ws.col\_values(3)

start.pop(0);end.pop(0);num.pop(0)

##deli为发货量，rece为收货量

deli=dict()

rece=dict()

for i in range(0,len(start)):

##以下两行代码的意思是如果城市号不在对应的字典的键中，那么就说明是第一次读到，进行赋值，否则加上即可

if(start[i] not in deli.keys()):

deli[start[i]]=num[i]

else :

deli[start[i]]+=num[i]

if(end[i] not in rece.keys()):

rece[end[i]]=num[i]

else :

rece[end[i]]+=num[i]

##以下存储发货量

res1=Workbook()

ws1=res1.active

##城市标签行赋值

ws1.cell(1,1).value="城市"

ws1.cell(1,2).value="发货量"

##以下逐行进行赋值

j=2

for name,tot in zip(list(deli.keys()),list(deli.values())):

ws1.cell(j,1).value=name

ws1.cell(j,2).value=tot

j+=1

res1.save("发货量总结表.xlsx")

##以下为存储收货量

res2=Workbook()

ws2=res2.active

ws2.cell(1,1).value="城市"

ws2.cell(1,2).value="收货量"

j=2

for name,tot in zip(list(rece.keys()),list(rece.values())):

ws2.cell(j,1).value=name

ws2.cell(j,2).value=tot

j+=1

res2.save("收货量总结表.xlsx")

### 附录2：趋势汇总程序.py

import pandas as pd

import numpy as np

from openpyxl import Workbook

data=pd.read\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\附件1.xlsx",parse\_dates=["日期(年/月/日) (Date Y/M/D)"])

month=data['日期(年/月/日) (Date Y/M/D)'].dt.strftime('%Y-%m')

month\_ori=month

##下列代码用来计算不重复的时间列表

month=month.drop\_duplicates()

month.index=range(0,10)

nowline,nextline=0,0

needdata=Workbook()

des=needdata.active

##代表对其进行从头到尾遍历

for i in range(0,9):

##nowcity表示现在城市的快递收发货情况，nextcity表示下一个时间段城市的收发货情况，change表示相比于上个月的差异平均差异

nowcity,nextcity,change=dict(),dict(),dict()

##下列两个变量分别用来判断该城市有没有进入过上述字典中

jud1,jud2=dict(),dict()

##nowline表示现在遍历到的年份，只要符合现在的就可以继续往下遍历，一直遍历到下一年终止

while(month\_ori.iloc[nowline]==month[i]):

bgi,end,num=data.iloc[nowline,1],data.iloc[nowline,2],data.iloc[nowline,3]

if(bgi not in jud1.keys()):

nowcity[bgi]=num

jud1[bgi]=1

else :

nowcity[bgi]+=num

if(end not in jud1.keys()):

nowcity[end]=num

jud1[end]=1

else :

nowcity[end]+=num

nowline+=1

##然后遍历下一年的，同上，nextline表示其遍历到的位置

nextline=nowline

while(month\_ori.iloc[nextline]==month[i+1]):

bgi,end,num=data.iloc[nextline,1],data.iloc[nextline,2],data.iloc[nextline,3]

if(bgi not in jud2.keys()):

nextcity[bgi]=num

jud2[bgi]=1

else :

nextcity[bgi]+=num

if(end not in jud2.keys()):

nextcity[end]=num

jud2[end]=1

else :

nextcity[end]+=num

nextline+=1

if(nextline==16962): ##表明这是数据的最后一行

break

same=list(set(list(nextcity.keys())+list(nowcity.keys())))##表明当前月份出现的城市列

temp=2

for j in sorted(same):

if((j in nowcity.keys())&(j in nextcity.keys())): ##如果都出现了，就计算其增长率

change[j]=(nextcity[j]-nowcity[j])/nowcity[j] ##计算某月的增加或下降趋势，后面两个是为了避免0

elif(j not in nowcity.keys()): ##如果上一年没有出现过，就赋值为1，否则为0

change[j]=1

else :

change[j]=-1

des.cell(1+i\*2,temp).value=j

temp+=1

des.cell(2+i\*2,1).value=month.iloc[i+1]

temp=1

for j in sorted(same):

des.cell(2+i\*2,temp+1).value=change[j]

temp+=1

needdata.save("趋势数据.xlsx")

### 附录3：各个节点的中心度计算程序.py

import xlrd

from openpyxl import Workbook

import numpy as np

import networkx as nx

import matplotlib.pyplot as plt

wb=xlrd.open\_workbook("C:\\Users\\Administrator\\Desktop\\vs工作界面\\附件1.xlsx")

ws=wb.sheet\_by\_index(0)

left,right,num=[],[],[]

left=ws.col\_values(1)

right=ws.col\_values(2)

num=ws.col\_values(3)

left.pop(0);right.pop(0);num.pop(0)

##上述代码为读入数据，然后去掉标签

jud=dict()

g=nx.DiGraph()##创建一个图

for i in range(0,len(num)):

##以下则是判断相应的点或边是否在图中，没有则加入

if(left[i] not in jud.keys()):

jud[left[i]]=1

g.add\_node(left[i])

if(right[i] not in jud.keys()):

jud[left[i]]=1

g.add\_node(right[i])

if(g.has\_edge(left[i],right[i])==0):

g.add\_edge(left[i],right[i],weight=num[i])

else :

g.edges[left[i],right[i]].update({"weight":num[i]+g.get\_edge\_data(left[i],right[i])["weight"]})

nx.draw\_networkx(g,pos=nx.shell\_layout(g),node\_size=200,node\_shape='o',width=1,style='solid',font\_size=8)##进行可视化

plt.show()

des=Workbook()

ws=des.active

ws.cell(1,1).value="节点"

ws.cell(1,2).value="度中心值排序"

j=2

np=nx.pagerank(g,alpha=0.9)

top\_k = sorted(np.items(), key=lambda x: x[1], reverse=True)

for node, value in top\_k:

ws.cell(j,1).value=node

ws.cell(j,2).value=value

j+=1

des.save("中心值的值.xlsx")

### 附录4：topisis评价.py

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from openpyxl import Workbook

plt.rcParams['font.family'] = ['sans-serif']

plt.rcParams['font.sans-serif'] = ['SimHei']

df=pd.read\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\各个指标整理.xlsx")

df=df.dropna(axis=0,how="all")

data=df.values

##提取城市标签

X=data[:,:1]

##提取城市评价指标

data=data[:,1:]

##进行标准化

data=data/np.sum(data\*data,axis=0)\*\*0.5

max\_score=np.max(data,axis=0)

min\_score=np.min(data,axis=0)

##求出其对应与最大值与最小值的距离

max\_dist=np.sum((max\_score-data)\*(max\_score-data),axis=1)\*\*0.5

min\_dist=np.sum((min\_score-data)\*(min\_score-data),axis=1)\*\*0.5

##进行得分计算

final\_score=(min\_dist/(max\_dist+min\_dist))

final\_score/=np.sum(final\_score)

final\_score=final\_score.tolist()

X=X.tolist()

x=dict()

j=0

##将其变成一个字典，方便之后从大到小排序

for i in X:

x[i[0]]=final\_score[j]

j+=1

res=sorted(x.items(),key=lambda x:x[1],reverse=True)

x,final\_score=[],[]

##以下皆为输出+可视化

wb=Workbook()

ws=wb.active

ws.cell(1,1).value="城市"

ws.cell(1,2).value="得分"

for num,sco in res:

x.append(num)

final\_score.append(sco)

for i in range(len(x)):

ws.cell(i+2,1).value=x[i]

ws.cell(i+2,2).value=final\_score[i]

wb.save("topsis评分表.xlsx")

ax=plt.bar(x=x,height=final\_score)##绘制条形图

plt.title("各个城市topsis评分排序")

plt.xlabel("各个城市")

plt.ylabel("各个城市得分情况")

plt.show()

### 附录5: 互动提取各天各城快递运输数量程序.py

import pandas as pd

import numpy as np

from openpyxl import Workbook

data=pd.read\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\附件1.xlsx",parse\_dates=["日期(年/月/日) (Date Y/M/D)"])

tim=data['日期(年/月/日) (Date Y/M/D)'].dt.strftime("%Y/%m/%d")

tim\_ori=tim

tim=tim.drop\_duplicates()

tim=tim.values

b=input("请输入你要查找数据的路径起点：")

e=input("请输入你要查找数据的路径终点：")

nowline=0

needdata=Workbook()

des=needdata.active

daydata=dict()

for i in range(0,len(tim)):

while(tim\_ori.iloc[nowline]==tim[i]):

bgi,end,num=data.iloc[nowline,1],data.iloc[nowline,2],data.iloc[nowline,3]

if(bgi==b) & (end==e):

if(tim[i] not in daydata.keys()):

daydata[tim[i]]=num

else :

daydata[tim[i]]+=num

nowline+=1

if(nowline==16962):

break

wb=Workbook()

ws=wb.active

ws.cell(1,1).value="时间"

ws.cell(1,2).value="总快递运输数量"

i=2

for temp1,temp2 in zip(list(daydata.keys()),list(daydata.values())):

ws.cell(i,1).value=temp1

ws.cell(i,2).value=temp2

i+=1

wb.save("各天{}城到{}城快递运输数量提取收据.xlsx".format(b,e))

### 附录6：求可达及其路径.py

import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM

import datetime

data = pd.read\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\附件2.xlsx").values

##求出不重复的城市对

lines = np.unique(np.array([i[0]+i[1]for i in data[:,[1,2]]]))

lines = np.array([[i[0],i[1]] for i in lines])

date = np.unique(data[:,0])

d = np.zeros(shape=(lines.shape[0],date.shape[0]))

d1 = np.zeros(shape=(lines.shape[0],date.shape[0]))

for i in range(d.shape[0]):

data1 = data[np.logical\_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])]

for j in range(d.shape[1]):

data2 = data1[data1[:,0]==date[j]]

if data2.shape[0]==0:

d[i,j]=0

d1[i,j]=0

else:

d[i,j]=1

d1[i,j]=data2[0,-1]

all\_pre1 = []

def create\_dataset(data, look\_back=1):

X, Y = [], []

for i in range(len(data) - look\_back):

X.append(data[i:(i + look\_back), 0])

Y.append(data[i + look\_back, 0])

return np.array(X), np.array(Y)

for i in range(lines.shape[0]):

y = d[i]

# 将数据划分为训练集和测试集

train\_size = int(len(y) \* 0.8)

train, test = y[:train\_size], y[train\_size:]

# 数据归一化

scaler = MinMaxScaler()

train = scaler.fit\_transform(train.reshape(-1, 1))

test = scaler.transform(test.reshape(-1, 1))

# 创建数据生成器

look\_back = 7

X\_train, y\_train = create\_dataset(train, look\_back)

X\_test, y\_test = create\_dataset(test, look\_back)

# 重塑数据以适应LSTM模型的输入格式

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))

# 创建LSTM模型

model = Sequential()

model.add(LSTM(50, input\_shape=(look\_back, 1)))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# 训练模型

model.fit(X\_train, y\_train, epochs=20, batch\_size=1, verbose=1)

# 预测

train\_predict = model.predict(X\_train)

test\_predict = model.predict(X\_test)

# 将预测结果转换回原始尺度

train\_predict = scaler.inverse\_transform(train\_predict)

y\_train = scaler.inverse\_transform(y\_train.reshape(-1, 1))

test\_predict = scaler.inverse\_transform(test\_predict)

y\_test = scaler.inverse\_transform(y\_test.reshape(-1, 1))

# 计算预测准确性，例如使用均方误差（MSE）作为评估指标

train\_mse = np.mean((train\_predict - y\_train) \*\* 2)

test\_mse = np.mean((test\_predict - y\_test) \*\* 2)

# 计算预测日期与最后一个训练日期之间的天数

last\_train\_date = datetime.date(2019, 4, 17)

start\_pred\_date = datetime.date(2019, 4, 18)

end\_pred\_date = datetime.date(2019, 4, 20)

days\_to\_predict = (end\_pred\_date-start\_pred\_date).days

# 使用训练数据的最后一部分来开始预测

input\_data = train[-look\_back:]

predictions = []

# 预测每一天的货量

for i in range(days\_to\_predict):

input\_data\_reshaped = input\_data.reshape(1, look\_back, 1)

pred = model.predict(input\_data\_reshaped)

predictions.append(pred[0, 0])

# 更新输入数据，用预测值替换最早的值

input\_data = np.roll(input\_data, -1)

input\_data[-1] = pred

# 将预测值转换回原始尺度

predictions = scaler.inverse\_transform(np.array(predictions).reshape(-1, 1))

tdays = []

# 打印预测结果

for i, pred in enumerate(predictions, start=1):

pred\_date = start\_pred\_date + datetime.timedelta(days=i - 1)

tdays += [pred[0]]

all\_pre1+=[tdays]

all\_pre1 = np.array(all\_pre1)

all\_pre = []

def create\_dataset(data, look\_back=1):

X, Y = [], []

for i in range(len(data) - look\_back):

X.append(data[i:(i + look\_back), 0])

Y.append(data[i + look\_back, 0])

return np.array(X), np.array(Y)

for i in range(lines.shape[0]):

y = d1[i]

y[y==0]=y.mean()

# 将数据划分为训练集和测试集

train\_size = int(len(y) \* 0.8)

train, test = y[:train\_size], y[train\_size:]

# 数据归一化

scaler = MinMaxScaler()

train = scaler.fit\_transform(train.reshape(-1, 1))

test = scaler.transform(test.reshape(-1, 1))

# 创建数据生成器

look\_back = 7

X\_train, y\_train = create\_dataset(train, look\_back)

X\_test, y\_test = create\_dataset(test, look\_back)

# 重塑数据以适应LSTM模型的输入格式

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))

# 创建LSTM模型

model = Sequential()

model.add(LSTM(50, input\_shape=(look\_back, 1)))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# 训练模型

model.fit(X\_train, y\_train, epochs=20, batch\_size=1, verbose=1)

# 预测

train\_predict = model.predict(X\_train)

test\_predict = model.predict(X\_test)

# 将预测结果转换回原始尺度

train\_predict = scaler.inverse\_transform(train\_predict)

y\_train = scaler.inverse\_transform(y\_train.reshape(-1, 1))

test\_predict = scaler.inverse\_transform(test\_predict)

y\_test = scaler.inverse\_transform(y\_test.reshape(-1, 1))

# 计算预测准确性，例如使用均方误差（MSE）作为评估指标

train\_mse = np.mean((train\_predict - y\_train) \*\* 2)

test\_mse = np.mean((test\_predict - y\_test) \*\* 2)

# 计算预测日期与最后一个训练日期之间的天数

last\_train\_date = datetime.date(2019, 4, 17)

start\_pred\_date = datetime.date(2019, 4, 18)

end\_pred\_date = datetime.date(2019, 4, 20)

days\_to\_predict = (end\_pred\_date-start\_pred\_date).days

# 使用训练数据的最后一部分来开始预测

input\_data = train[-look\_back:]

predictions = []

# 预测每一天的货量

for i in range(days\_to\_predict):

input\_data\_reshaped = input\_data.reshape(1, look\_back, 1)

pred = model.predict(input\_data\_reshaped)

predictions.append(pred[0, 0])

# 更新输入数据，用预测值替换最早的值

input\_data = np.roll(input\_data, -1)

input\_data[-1] = pred

# 将预测值转换回原始尺度

predictions = scaler.inverse\_transform(np.array(predictions).reshape(-1, 1))

tdays = []

# 打印预测结果

for i, pred in enumerate(predictions, start=1):

pred\_date = start\_pred\_date + datetime.timedelta(days=i - 1)

tdays += [pred[0]]

all\_pre+=[tdays]

all\_pre = np.array(all\_pre)

all\_pre1[all\_pre1>0.5]=1

all\_pre1[all\_pre1<=0.5]=0

p2 = all\_pre\*all\_pre1

temp=pd.DataFrame(np.c\_["1",lines,all\_pre1,p2],columns=["起点","终点","28号是否开通","29号是否开通","28号预测值","29号预测值"])

temp.to\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面[\\第三问结果.xlsx](file:///\\第三问结果.xlsx)")

### 附录7：最短路计算代码.py

import networkx as nx

import matplotlib.pyplot as plt

import pandas as pd

from copy import deepcopy

plt.rcParams['font.sans-serif']=['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

fixeddata=pd.read\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\附件3.xlsx")

##以下为提取相对应的数据

sta=fixeddata.loc[:,"起点 (Start)"]

end=fixeddata.loc[:,"终点 (End)"]

fixcos=fixeddata.loc[:,"固定成本 (Fixed cost)"]

dot\_inf=dict()

ori=nx.DiGraph()

##以下表示给相对应的路径编码，并插入到图中

wegnum=0

weg=dict()

for i in range(len(sta)):

weg[wegnum]=(sta[i],end[i])

wegnum+=1

dot\_inf[(sta[i],end[i])]=fixcos[i]

ori.add\_edge(sta[i],end[i],weight=fixcos[i])

pos=nx.shell\_layout(ori)##表示画图的格式，下面是画图相关代码

nx.draw\_networkx(ori,pos=pos,node\_size=200,node\_shape='o',width=1,style='solid',font\_size=8)

node\_labels=nx.get\_node\_attributes(ori,"des")

nx.draw\_networkx\_labels(ori,pos=pos,labels=node\_labels)

plt.title("铁路路线图",fontsize=10)

plt.show()

nodes=[]

for i in ori.nodes.data():

nodes.append(i[0])

node=dict()

letters=dict()

for i in range(0,len(nodes)):##表示对于点进行编码

node[nodes[i]]=i

letters[i]=nodes[i]

dist=dict()

pre=dict()

INF=9999999

def bellman\_ford(bgi,end,weight):##最短路算法

backup=dict()

for i in range(0,len(node)):

dist[i]=INF

dist[bgi]=0

for i in range(0,5):##限制次数为5

backup=deepcopy(dist)

for j in range(0,wegnum): ##代表走wegnum条路

wega=node[weg[j][0]]

wegb=node[weg[j][1]]

w=dot\_inf[weg[j]]\*(1+(weight/200)\*\*3)##计算对应的成本

if(dist[wegb]>backup[wega]+w):

dist[wegb]=backup[wega]+w

return dist[end]

for j in range(23,28):##代表逐天读取，并且进行逐天预测

g=nx.DiGraph()

data=pd.read\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\{}日数据.xlsx".format(str(j)))

dci=data["发货城市 (Delivering city)"]

rci=data["收货城市 (Receiving city)"]

num=data["快递运输数量(件) (Express delivery quantity (PCS))"]

today=0

for i in range(0,len(dci)):

start=node[dci[i]]

end=node[rci[i]]

today+=bellman\_ford(start,end,num[i])

print("4月{}日的最低运输成本是{}".format(j,today))

### 附录8：第五问22年代码.py

import numpy as np

import pandas as pd

from sklearn.neighbors import KernelDensity

import matplotlib.pyplot as plt

import matplotlib as mpl

mpl.rcParams["font.sans-serif"]=["SimHei"]

mpl.rcParams["axes.unicode\_minus"]=False

data = pd.read\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\22年季度数据.xlsx").values

date = np.unique(data[:,0]) ##表示全部数据

lines = np.unique([i[0]+i[1]for i in data[:,[1,2]]])

lines = np.array([[i[0],i[1]] for i in lines]) ##表示总共的城市对

line\_mean = []

line\_min = []

xu = []

f\_mean = []

f\_std = []

mt = np.zeros(shape=(lines.shape[0],date.shape[0]))

print(mt)

for i in range(lines.shape[0]):

d1 = data[np.logical\_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,-1] ##d1为相对应的月份的数据

rq = data[np.logical\_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,0] ##rq为相对应的时间序列的数据

line\_mean+=[d1.mean()] ##表示每个城市对的均值

line\_min +=[d1.min()] ##表示每个城市对的最小值

fixed =d1.mean()-2\*d1.std() ##表示3sigma定则

if(fixed<0):

fixed=0

xu +=[fixed]

notfixed = d1-(fixed) ##代表各个城市对的非固定需求

##以下则为计算均值与标准差

f\_mean += [notfixed.mean()]

f\_std += [notfixed.std()]

for j in range(rq.shape[0]):

mt[i,date==rq[j]] = notfixed[j]

xu = np.array(xu)

print("22年固定需求",np.c\_["1",lines,np.round(xu)])

print("22年非固定需求均值标准差\n",np.c\_["1",lines,np.round(f\_mean,4),np.round(f\_std,4)])

temp=pd.DataFrame(np.c\_["1",lines,np.round(xu)],columns=["发货城市","收货城市","固定需求"])

temp.to\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\固定需求(22年).xlsx")

temp=pd.DataFrame(np.c\_["1",lines,np.round(f\_mean,2),np.round(f\_std,2)],columns=["发货城市","收货城市","均值","标准差"])

temp.to\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\非固定需求(均值与标准差)(22年).xlsx")

temp=pd.DataFrame(np.c\_["1",lines,mt],columns=np.r\_["0",["发货城市","收货城市"],date])

temp.to\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\非固定需求(22年).xlsx")

sc = np.array([["V","N"],["V","Q"]])

i=0

for i in range(2):

d1 = data[np.logical\_and(data[:,1]==sc[i,0],data[:,2]==sc[i,1])][:,-1]

fixed =d1.mean()-2\*d1.std() ##求出固定需求常数，即3sigma准则

if(fixed<0):

fixed=0

d2 = d1-fixed

sample\_data = d2.reshape(-1, 1)

# KDE模型实例化,高斯核函数是一个常用的核函数，可以对连续变量进行估计，带宽参数影响核函数的宽度，决定了估计出的概率密度函数的平滑度。

kde = KernelDensity(kernel='gaussian', bandwidth=6).fit(sample\_data)

# 指定评估点（根据实际需求调整范围和间隔）

ep = np.linspace(d2.min(), d2.max(), num=300).reshape(-1, 1)

# 评估KDE模型

ld = kde.score\_samples(ep)

ds = np.exp(ld)

# 绘制KDE结果和直方图

fig, ax = plt.subplots()

ax.plot(ep, ds, label='KDE')

ax.hist(sample\_data, bins=5, density=True, alpha=0.5, color='blue', label='直方图')

ax.set\_xlabel('{}到{}(非固定需求)'.format(sc[i,0],sc[i,1]),fontsize=15)

ax.set\_ylabel('概率密度',fontsize=14)

ax.legend(fontsize=14)

plt.savefig('{}到{}(非固定需求)分布图.png'.format(sc[i,0],sc[i,1]))

plt.show()

temp=pd.DataFrame(np.c\_["1",ep,ds[:,None]],columns=["非固定需求量","概率密度"])

temp.to\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\{}\_{}(非固定需求).xlsx".format(sc[i,0],sc[i,1]))

### 附录9：第五问23年代码.py

import numpy as np

import pandas as pd

from sklearn.neighbors import KernelDensity

import matplotlib.pyplot as plt

import matplotlib as mpl

mpl.rcParams["font.sans-serif"]=["SimHei"]

mpl.rcParams["axes.unicode\_minus"]=False

data = pd.read\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\23年季度数据.xlsx").values

date = np.unique(data[:,0])

lines = np.unique([i[0]+i[1]for i in data[:,[1,2]]])

lines = np.array([[i[0],i[1]] for i in lines])

line\_mean = []

line\_min = []

xu = []

f\_mean = []

f\_std = []

mt = np.zeros(shape=(lines.shape[0],date.shape[0]))

for i in range(lines.shape[0]):

d1 = data[np.logical\_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,-1]

rq = data[np.logical\_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,0]

line\_mean+=[d1.mean()]

line\_min +=[d1.min()]

fixed =d1.mean()-2\*d1.std()

if fixed<0 :

fixed=0

xu +=[fixed]

notfixed = d1-(fixed) ##这一部分是非固定需求

f\_mean += [notfixed.mean()]

f\_std += [notfixed.std()]

for j in range(rq.shape[0]):

mt[i,date==rq[j]] = notfixed[j]

xu = np.array(xu)

print("23年固定需求",np.c\_["1",lines,np.round(xu)])

print("23年非固定需求均值标准差\n",np.c\_["1",lines,np.round(f\_mean,4),np.round(f\_std,4)])

temp=pd.DataFrame(np.c\_["1",lines,np.round(xu)],columns=["发货城市","收货城市","固定需求"])

temp.to\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\固定需求(23年).xlsx")

temp=pd.DataFrame(np.c\_["1",lines,np.round(f\_mean,2),np.round(f\_std,2)],columns=["发货城市","收货城市","均值","标准差"])

temp.to\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\非固定需求(均值与标准差)(23年).xlsx")

temp=pd.DataFrame(np.c\_["1",lines,mt],columns=np.r\_["0",["发货城市","收货城市"],date])

temp.to\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\非固定需求(23年).xlsx")

sc = np.array([["J","I"],["O","G"]])

i=0

for i in range(2):

d1 = data[np.logical\_and(data[:,1]==sc[i,0],data[:,2]==sc[i,1])][:,-1]

fixed =d1.mean()-2\*d1.std() ##求出固定需求常数，即3sigma准则

if(fixed<0):

fixed=0

d2 = d1-fixed

sample\_data = d2.reshape(-1, 1)

# KDE模型实例化,高斯核函数是一个常用的核函数，可以对连续变量进行估计，带宽参数影响核函数的宽度，决定了估计出的概率密度函数的平滑度。

kde = KernelDensity(kernel='gaussian', bandwidth=6).fit(sample\_data)

# 指定评估点（根据实际需求调整范围和间隔）

ep = np.linspace(d2.min(), d2.max(), num=300).reshape(-1, 1)

# 评估KDE模型

ld = kde.score\_samples(ep)

ds = np.exp(ld)

# 绘制KDE结果和直方图

fig, ax = plt.subplots()

ax.plot(ep, ds, label='KDE')

ax.hist(sample\_data, bins=5, density=True, alpha=0.5, color='green', label='直方图')

ax.set\_xlabel('{}到{}(非固定需求)'.format(sc[i,0],sc[i,1]),fontsize=15)

ax.set\_ylabel('概率密度',fontsize=14)

ax.legend(fontsize=14)

plt.savefig('{}到{}(非固定需求)分布图.png'.format(sc[i,0],sc[i,1]))

plt.show()

temp=pd.DataFrame(np.c\_["1",ep,ds[:,None]],columns=["非固定需求量","概率密度"])

temp.to\_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\{}\_{}(非固定需求).xlsx".format(sc[i,0],sc[i,1]))

### 附录10：22年固定需求（第一幅图）+23年固定需求（第二幅图）

|  |  |  |
| --- | --- | --- |
| 发货城市 | 收货城市 | 固定需求 |
| A | O | 24.0 |
| A | Q | 0.0 |
| C | M | 6.0 |
| C | N | 0.0 |
| C | U | 10.0 |
| C | V | 34.0 |
| D | A | 14.0 |
| D | E | 16.0 |
| D | L | 4.0 |
| D | R | 9.0 |
| E | F | 14.0 |
| E | I | 4.0 |
| G | L | 15.0 |
| G | N | 5.0 |
| G | O | 0.0 |
| G | Q | 16.0 |
| G | R | 6.0 |
| G | V | 0.0 |
| G | X | 40.0 |
| H | J | 0.0 |
| H | K | 14.0 |
| H | L | 36.0 |
| I | E | 4.0 |
| I | F | 31.0 |
| I | J | 3.0 |
| I | S | 28.0 |
| J | H | 22.0 |
| J | I | 5.0 |
| J | K | 0.0 |
| J | L | 14.0 |
| K | H | 0.0 |
| K | J | 14.0 |
| K | L | 0.0 |
| L | D | 12.0 |
| L | G | 42.0 |
| L | H | 19.0 |
| L | J | 0.0 |
| L | K | 0.0 |
| L | O | 24.0 |
| L | P | 27.0 |
| L | R | 0.0 |
| L | W | 0.0 |
| L | X | 0.0 |
| M | C | 0.0 |
| M | G | 17.0 |
| M | N | 9.0 |
| M | U | 0.0 |
| M | V | 12.0 |
| N | G | 22.0 |
| N | M | 0.0 |
| N | V | 3.0 |
| O | G | 0.0 |
| O | Q | 0.0 |
| O | R | 0.0 |
| P | D | 16.0 |
| Q | A | 0.0 |
| Q | M | 14.0 |
| Q | N | 0.0 |
| Q | O | 22.0 |
| Q | V | 0.0 |
| R | D | 6.0 |
| R | G | 0.0 |
| R | L | 0.0 |
| R | O | 0.0 |
| R | S | 0.0 |
| S | D | 0.0 |
| S | I | 0.0 |
| S | L | 36.0 |
| S | Q | 0.0 |
| S | R | 0.0 |
| U | A | 30.0 |
| U | G | 16.0 |
| U | O | 9.0 |
| U | V | 17.0 |
| V | A | 0.0 |
| V | C | 41.0 |
| V | G | 0.0 |
| V | M | 0.0 |
| V | N | 0.0 |
| V | Q | 0.0 |
| W | L | 0.0 |
| W | X | 1.0 |
| W | Y | 0.0 |
| X | G | 28.0 |
| X | L | 0.0 |
| X | W | 0.0 |
| X | Y | 0.0 |
| Y | L | 0.0 |
| Y | W | 0.0 |
| Y | X | 0.0 |

以上为22年固定需求。

|  |  |  |
| --- | --- | --- |
| **发货城市** | **收货城市** | **固定需求** |
| A | O | 21.0 |
| A | Q | 0.0 |
| C | M | 28.0 |
| C | N | 29.0 |
| C | U | 20.0 |
| C | V | 22.0 |
| D | A | 13.0 |
| D | E | 9.0 |
| D | L | 16.0 |
| D | R | 16.0 |
| E | F | 14.0 |
| E | I | 2.0 |
| G | L | 15.0 |
| G | N | 29.0 |
| G | O | 0.0 |
| G | Q | 21.0 |
| G | R | 0.0 |
| G | V | 0.0 |
| G | X | 39.0 |
| H | J | 0.0 |
| H | K | 17.0 |
| H | L | 37.0 |
| I | E | 6.0 |
| I | F | 26.0 |
| I | J | 0.0 |
| I | S | 24.0 |
| J | H | 32.0 |
| J | I | 4.0 |
| J | K | 0.0 |
| J | L | 14.0 |
| K | H | 4.0 |
| K | J | 5.0 |
| K | L | 0.0 |
| L | D | 6.0 |
| L | G | 41.0 |
| L | H | 18.0 |
| L | J | 10.0 |
| L | K | 10.0 |
| L | O | 20.0 |
| L | P | 28.0 |
| L | R | 22.0 |
| L | W | 0.0 |
| L | X | 41.0 |
| M | C | 17.0 |
| M | G | 26.0 |
| M | N | 0.0 |
| M | U | 16.0 |
| M | V | 27.0 |
| N | G | 28.0 |
| N | M | 0.0 |
| N | V | 6.0 |
| O | G | 0.0 |
| O | Q | 0.0 |
| O | R | 0.0 |
| P | D | 15.0 |
| Q | A | 14.0 |
| Q | M | 1.0 |
| Q | N | 0.0 |
| Q | O | 22.0 |
| Q | V | 0.0 |
| R | D | 19.0 |
| R | G | 20.0 |
| R | L | 2.0 |
| R | O | 0.0 |
| R | S | 0.0 |
| S | D | 24.0 |
| S | I | 25.0 |
| S | L | 38.0 |
| S | Q | 0.0 |
| S | R | 0.0 |
| U | A | 8.0 |
| U | G | 22.0 |
| U | O | 10.0 |
| U | V | 15.0 |
| V | A | 13.0 |
| V | C | 38.0 |
| V | G | 0.0 |
| V | M | 11.0 |
| V | N | 10.0 |
| V | Q | 7.0 |
| W | L | 0.0 |
| W | X | 13.0 |
| W | Y | 0.0 |
| X | G | 37.0 |
| X | L | 4.0 |
| X | W | 0.0 |
| X | Y | 0.0 |
| Y | L | 0.0 |
| Y | W | 0.0 |
| Y | X | 0.0 |

以上为23年固定需求