**附录**

**附录1**

数据清理部分代码（全部见附件）

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from pandas import read\_csv

f = open('data 1.csv',encoding='UTF-8')

names = ['starttime','speed','volume','occupancy']

# filename=input("文件名：")

# f=open(filename,encoding='UTF-8')

# names=['作业日期','ηCO','ηH2','TF(℃)','TC(℃)','mass','送风流量']

data=read\_csv(f,names=names,header=1)

print(data)

data.starttime.drop\_duplicates(keep=False,inplace=True)

# 把重复的值全部删除

print(data[data.duplicated(keep=False)])

# 检验。现在没有重复值

# 缺省值处理

#这里有个问题，就是当我们面临一大堆nan的时候，是否应该根据时间

#speed缺省值填补

count = 1

for count in range(data.shape[0]):

# print(tup[0],'-->',tup[1::],type(tup[1:]))

speedtmp = data.loc[count,["speed"]].values

# print(speedtmp)

# print(speedtmp.dtype)

if pd.isnull(speedtmp) :

if count == 1 : #先给第一个缺省值赋值

tmpcount = 0

while pd.isnull(data.loc[count+tmpcount,["speed"]].values) :

tmpcount+=1

data.loc[count,["speed"]] = data.loc[count+tmpcount,["speed"]]

print("speed第一行为空")

elif count >2 and count <= data.shape[0]: #或者使用 len(data)

tmpcount = 0

while pd.isnull(data.loc[count+tmpcount,["speed"]].values) :

tmpcount+=1

data.loc[count,["speed"]] = (data.loc[count-1,["speed"]].values.astype(float)+data.loc[count+tmpcount,["speed"]].values.astype(float))/2

print(count ,"speed修改成功")

elif count == data.shape[0]+1:

data.loc[count,["speed"]] = data.loc[count-1,["speed"]]

print("speed最后一行为空")

else :

pass

#volume缺省值填补

count = 1

for count in range(data.shape[0]):

# print(tup[0],'-->',tup[1::],type(tup[1:]))

volumetmp = data.loc[count,["volume"]].values

# print(speedtmp)

# print(speedtmp.dtype)

if pd.isnull(volumetmp) :

if count == 1 : #先给第一个缺省值赋值

tmpcount = 0

while pd.isnull(data.loc[count+tmpcount,["volume"]].values) :

tmpcount+=1

data.loc[count,["volume"]] = data.loc[count+tmpcount,["volume"]]

print("volume第一行为空")

elif count >2 and count <= data.shape[0]: #或者使用 len(data)

tmpcount = 0

while pd.isnull(data.loc[count+tmpcount,["volume"]].values) :

tmpcount+=1

data.loc[count,["volume"]] = (data.loc[count-1,["volume"]].values.astype(float)+data.loc[count+tmpcount,["volume"]].values.astype(float))/2

print(count ,"volume修改成功")

elif count == data.shape[0]+1:

data.loc[count,["volume"]] = data.loc[count-1,["volume"]]

print("volume最后一行为空")

else :

pass

#occupancy缺省值填补

count = 1

for count in range(data.shape[0]):

# print(tup[0],'-->',tup[1::],type(tup[1:]))

occupancytmp = data.loc[count,["occupancy"]].values

# print(speedtmp)

# print(speedtmp.dtype)

if pd.isnull(occupancytmp) :

if count == 1 : #先给第一个缺省值赋值

tmpcount = 0

while pd.isnull(data.loc[count+tmpcount,["occupancy"]].values) :

tmpcount+=1

data.loc[count,["occupancy"]] = data.loc[count+tmpcount,["occupancy"]]

print("occupancy第一行为空")

elif count >2 and count <= data.shape[0]: #或者使用 len(data)

tmpcount = 0

while pd.isnull(data.loc[count+tmpcount,["occupancy"]].values) :

tmpcount+=1

data.loc[count,["occupancy"]] = (data.loc[count-1,["occupancy"]].values.astype(float)+data.loc[count+tmpcount,["occupancy"]].values.astype(float))/2

print(count ,"occupancy修改成功")

elif count == data.shape[0]+1:

data.loc[count,["occupancy"]] = data.loc[count-1,["occupancy"]]

print("occupancy最后一行为空")

else :

pass

print (data)

#显示data的相关信息

data.dtypes

data.info()

# 异常值处理

#先进行方差检验，若为零则进行阈值检验，若不为零则进行组合检验，然后进行方差检验

# #以最大值的百分之九十五为界限

# data.max() #最值、累加

# data['speed'].max()

# threshold\_speed = data['speed'].max() \* 0.95

# data.loc[data['speed'] >= threshold\_speed]

from pandas import Series

s= pd.Series()

print(s,s.dtype)

for outliercount in range(2 , data.shape[0]):

# print(outliercount)

# tmp = data.loc[outliercount,["speed"]].values.astype(float) - data.loc[outliercount - 1,["speed"]].values.astype(float)

# print(tmp,tmp.dtype)

if ( (data.loc[outliercount,["speed"]].values.astype(float) - data.loc[outliercount-1,["speed"]].values.astype(float))\*\*2 ==0 and

(data.loc[outliercount,["volume"]].values.astype(float) - data.loc[outliercount-1,["volume"]].values.astype(float))\*\*2 ==0 and

(data.loc[outliercount,["occupancy"]].values.astype(float) - data.loc[outliercount-1,["occupancy"]].values.astype(float))\*\*2 ==0) :

#阈值检测设置速度：0-140；流量：0-225；占用率：0-100；

if (0<= data.loc[outliercount,["speed"]].values <= 140) & (0<= data.loc[outliercount,["volume"]].values <= 225) & (0<= data.loc[outliercount,["occupancy"]].values <= 100) :#阈值检验

pass

# print("阈值检测通过") #阈值检测通过

else :

s= s.append(Series([outliercount]))#添加该行标记到series，之后删除该行数据

print("添加行", outliercount)

elif ((data.loc[outliercount,["speed"]].values ==0 and data.loc[outliercount,["volume"]].values ==0 and data.loc[outliercount,["occupancy"]].values !=0) or

(data.loc[outliercount,["speed"]].values ==0 and data.loc[outliercount,["volume"]].values !=0 and data.loc[outliercount,["occupancy"]].values ==0) or

(data.loc[outliercount,["speed"]].values !=0 and data.loc[outliercount,["volume"]].values ==0 and data.loc[outliercount,["occupancy"]].values ==0) or

(data.loc[outliercount,["speed"]].values !=0 and data.loc[outliercount,["volume"]].values !=0 and data.loc[outliercount,["occupancy"]].values ==0) or

(data.loc[outliercount,["speed"]].values ==0 and data.loc[outliercount,["volume"]].values !=0 and data.loc[outliercount,["occupancy"]].values !=0)) :

s= s.append(Series([outliercount])) #加该行标记到series，之后删除该行数据

print("添加行", outliercount)

else :

#组合检验+阈值检验

#组合检验（注意量纲）：（速度\*1000/60）\*（占用率\*5）/（流量\*5）=结果，结果：0-2038

Combination\_testing = (data.loc[outliercount,["speed"]].values\*1000/60)\*(data.loc[outliercount,["occupancy"]].values\*5)/(data.loc[outliercount,["volume"]].values\*5)

if (0<= Combination\_testing <=2038) :

pass

# print("组合检验通过") #组合检验通过

else :

s= s.append(Series([outliercount])) #添加该行标记到series，之后删除该行数据

if (0<= data.loc[outliercount,["speed"]].values <= 140) & (0<= data.loc[outliercount,["volume"]].values <= 225) & (0<= data.loc[outliercount,["occupancy"]].values <= 100) :#阈值检验

pass

# print("阈值检测通过") #阈值检测通过

else :

s= s.append(Series([outliercount])) #添加该行标记到series，之后删除该行数据

print("添加行", outliercount)

s = s.astype('int')

s = s.sort\_values(ascending=False)

print(s)

#删除对应列数据

print(data.shape[0])

data.drop(s,inplace=True) #根据S的编号删除

print(data.shape[0])

data.to\_csv('data 1\_change.csv') #写入csv文件

**附录2**

曲线拟合部分代码（全部见附件）

import pandas as pd

#显示所有列

pd.set\_option('display.max\_columns', None)

#显示所有行

pd.set\_option('display.max\_rows', None)

#设置value的显示长度为100，默认为50

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from pandas import read\_csv

f = open('data2\_2014.csv',encoding='UTF-8')

names = ['Index\_Num','starttime','resolution','speed','volume','occupancy','速度','车流量','占有率','年','月','日','日期','时间','速度均值','车流量均值','占有率均值','ID','filter']

data=read\_csv(f,names=names,header=1)

print(data)

#单变量曲线拟合

#拟合高斯分布

from scipy.optimize import curve\_fit

import math

#自定义函数 e指数形式

def func(x, a,u, sig):

return a\*(np.exp(-(x - u) \*\* 2 /(2\* sig \*\*2))/(math.sqrt(2\*math.pi)\*sig))\*(431+(4750/x))

#定义x、y散点坐标

x = np.array(data['ID'])

x=np.array(x)

# x = np.array(range(20))

print('x is :\n',x)

num = np.array(data['占有率均值'])

y = np.array(num)

print('y is :\n',y)

popt, pcov = curve\_fit(func, x, y,p0=[3.1,4.2,3.3])

#获取popt里面是拟合系数

a = popt[0]

u = popt[1]

sig = popt[2]

yvals = func(x,a,u,sig) #拟合y值

print(u'系数a:', a)

print(u'系数u:', u)

print(u'系数sig:', sig)

#绘图

plot1 = plt.plot(x, y, 's',label='original values')

plot2 = plt.plot(x, yvals, 'r',label='polyfit values')

plt.xlabel('x')

plt.ylabel('y')

plt.legend(loc=4) #指定legend的位置右下角

plt.title('curve\_fit')

plt.show()

# 高斯拟合

#单个高斯模型，如果曲线有多个波峰，可以分段拟合

def func(x, a,u, sig):

return a\*np.exp(-(x - u) \*\* 2 / (2 \* sig \*\* 2)) / (sig \* math.sqrt(2 \* math.pi))

#混合高斯模型，多个高斯函数相加

def func3(x, a1, a2, a3, a4, a5 ,m1, m2, m3, m4, m5, s1, s2, s3, s4, s5):

return a1 \* np.exp(-((x - m1) / s1) \*\* 2) + a2 \* np.exp(-((x - m2) / s2) \*\* 2) + a3 \* np.exp(-((x - m3) / s3) \*\* 2) + a4 \*np.exp(-((x - m4) / s4) \*\* 2) + a5 \*np.exp(-((x - m5) / s5) \*\* 2)

#正弦函数拟合

#def fmax(x,a,b,c):

# return a\*np.sin(x\*np.pi/6+b)+c

#fita,fitb=optimize.curve\_fit(fmax,x,ymax,[1,1,1])

#非线性最小二乘法拟合

#def func(x, a, b,c):

# return a\*np.sqrt(x)\*(b\*np.square(x)+c)

# 用3次多项式拟合，可推广到n次多项式，数学上可以证明，任意函数都可以表示为多项式形式

# f1 = np.polyfit(x, y, 3)

# p1 = np.poly1d(f1)

# yvals = p1(x) #拟合y值

#也可使用yvals=np.polyval(f1, x)

#拟合，并对参数进行限制，bounds里面代表参数上下限，p0是初始范围，默认是[1,1,1]

x=np.arange(1,data.shape[0]+1,1)

num = np.array(data['占有率均值']) #<-自己的y值

numhunt = np.array(data['占有率均值']) #<-自己的y值

y = np.array(num)

yhunt = np.array(numhunt)

# param\_bounds=([-np.inf,0],[np.inf,1])#设定B和n的下界和上界。其中B为负无穷到正无穷，n为0到1

popt, pcov = curve\_fit(func3, x, y,maxfev=500000)

popthunt, pcovhunt = curve\_fit(func, x, yhunt,p0=[2,2,2])

ahunt = popthunt[0]

uhunt = popthunt[1]

sighunt = popthunt[2]

a1 = popt[0]

u1 = popt[1]

sig1 = popt[2]

a2 = popt[3]

u2 = popt[4]

sig2 = popt[5]

a3 = popt[6]

u3 = popt[7]

sig3 = popt[8]

a4 = popt[9]

u4 = popt[10]

sig4 = popt[11]

a5 = popt[12]

u5 = popt[13]

sig5 = popt[14]

yvals = func3(x,a1,u1,sig1,a2,u2,sig2,a3,u3,sig3,a4,u4,sig4,a5,u5,sig5) #拟合y值

yhuntvals = func(x,ahunt,uhunt,sighunt) #拟合y值

print(u'系数ahunt:', ahunt)

print(u'系数uhunt:', uhunt)

print(u'系数sighunt:', sighunt)

#绘图

plot1 = plt.plot(x, y, 's',label='insect original values')

plot2 = plt.plot(x, yvals, 'r',label='insect polyfit values')

plot3 = plt.plot(x, yhunt, 's',label='predator original values')

plot4 = plt.plot(x, yhuntvals, 'g',label='predator polyfit values')

plt.xlabel('date')

plt.ylabel('Nightly catches log10(N+1)')

plt.legend(loc=4) #指定legend的位置右下角

plt.title('insect/predator')

plt.show()

# 多项式拟合

#单个高斯模型，如果曲线有多个波峰，可以分段拟合

def func(x, a,u, sig):

return a\*np.exp(-(x - u) \*\* 2 / (2 \* sig \*\* 2)) / (sig \* math.sqrt(2 \* math.pi))

#混合高斯模型，多个高斯函数相加

def func3(x, a1, a2, a3, a4, a5 ,m1, m2, m3, m4, m5, s1, s2, s3, s4, s5):

return a1 \* np.exp(-((x - m1) / s1) \*\* 2) + a2 \* np.exp(-((x - m2) / s2) \*\* 2) + a3 \* np.exp(-((x - m3) / s3) \*\* 2) + a4 \*np.exp(-((x - m4) / s4) \*\* 2) + a5 \*np.exp(-((x - m5) / s5) \*\* 2)

#正弦函数拟合

#def fmax(x,a,b,c):

# return a\*np.sin(x\*np.pi/6+b)+c

#fita,fitb=optimize.curve\_fit(fmax,x,ymax,[1,1,1])

#非线性最小二乘法拟合

#def func(x, a, b,c):

# return a\*np.sqrt(x)\*(b\*np.square(x)+c)

# 用3次多项式拟合，可推广到n次多项式，数学上可以证明，任意函数都可以表示为多项式形式

# f1 = np.polyfit(x, y, 3)

# p1 = np.poly1d(f1)

# yvals = p1(x) #拟合y值

#也可使用yvals=np.polyval(f1, x)

#拟合，并对参数进行限制，bounds里面代表参数上下限，p0是初始范围，默认是[1,1,1]

x=np.arange(1,data.shape[0]+1,1)

num = np.array(data['占有率均值']) #<-自己的y值

numhunt = np.array(data['占有率均值']) #<-自己的y值

y = np.array(num)

f1 = np.polyfit(x, y, 13)

p1 = np.poly1d(f1)

yvals = p1(x)

yhunt = np.array(numhunt)

# param\_bounds=([-np.inf,0],[np.inf,1])#设定B和n的下界和上界。其中B为负无穷到正无穷，n为0到1

popt, pcov = curve\_fit(func3, x, y,maxfev=500000)

popthunt, pcovhunt = curve\_fit(func, x, yhunt,p0=[2,2,2])

ahunt = popthunt[0]

uhunt = popthunt[1]

sighunt = popthunt[2]

yhuntvals = func(x,ahunt,uhunt,sighunt) #拟合y值

print(u'系数ahunt:', ahunt)

print(u'系数uhunt:', uhunt)

print(u'系数sighunt:', sighunt)

#绘图

plot1 = plt.plot(x, y, 's',label='insect original values')

plot2 = plt.plot(x, yvals, 'r',label='insect polyfit values')

plot3 = plt.plot(x, yhunt, 's',label='predator original values')

plot4 = plt.plot(x, yhuntvals, 'g',label='predator polyfit values')

plt.xlabel('date')

plt.ylabel('Nightly catches log10(N+1)')

plt.legend(loc=4) #指定legend的位置右下角

plt.title('insect/predator')

plt.show()

# 多元多项式拟合

import pandas as pd

import numpy as np

from sklearn.preprocessing import PolynomialFeatures

from sklearn import linear\_model

x = np.array(data[['速度均值','车流量均值']])

y = np.array(data['占有率均值'])

def stdError\_func(y\_test, y):

return np.sqrt(np.mean((y\_test - y) \*\* 2))

def R2\_1\_func(y\_test, y):

return 1 - ((y\_test - y) \*\* 2).sum() / ((y.mean() - y) \*\* 2).sum()

def R2\_2\_func(y\_test, y):

y\_mean = np.array(y)

y\_mean[:] = y.mean()

return 1 - stdError\_func(y\_test, y) / stdError\_func(y\_mean, y)

poly\_reg =PolynomialFeatures(degree=2) #三次多项式

X\_ploy =poly\_reg.fit\_transform(x)

lin\_reg\_2=linear\_model.LinearRegression()

lin\_reg\_2.fit(X\_ploy,y)

predict\_y = lin\_reg\_2.predict(X\_ploy)

strError = stdError\_func(predict\_y, y)

R2\_1 = R2\_1\_func(predict\_y, y)

R2\_2 = R2\_2\_func(predict\_y, y)

score = lin\_reg\_2.score(X\_ploy, y) ##sklearn中自带的模型评估，与R2\_1逻辑相同

print("coefficients", lin\_reg\_2.coef\_)

print("intercept", lin\_reg\_2.intercept\_)

print('degree={}: strError={:.2f}, R2\_1={:.2f}, R2\_2={:.2f}, clf.score={:.2f}'.format(

3, strError,R2\_1,R2\_2,score))

#clf.score它提供了一个缺省的评估法则来解决问题，简要的说，它用你训练好的模型在测试集上进行评分（0~1）1分代表最好

#算一下得到的方程表达式跟实际的差值

intercept = lin\_reg\_2.intercept\_

coefficients =lin\_reg\_2.coef\_

yres =[]

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

variable\_x = [1,data.loc[i,["速度均值"]].values,data.loc[i,["车流量均值"]].values,(data.loc[i,["速度均值"]].values)\*\*2,(data.loc[i,["速度均值"]].values\*data.loc[i,["车流量均值"]].values),(data.loc[i,["车流量均值"]].values)\*\*2]

tmp\_varX = np.array(variable\_x,dtype=np.float32).T

# print(tmp\_varX, tmp\_varX.dtype,tmp\_varX.shape)

yres.append((intercept + np.dot(coefficients , tmp\_varX)).astype(float))

#做一下和源数据之间的差值

yres = np.array(yres,dtype=np.float32)

print(yres, yres.dtype,yres.shape)

dval= np.abs(yres - data['占有率均值'].values)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print(dval)

#画出散点图

x = range(0,dval.shape[0],1)

print(len(x),len(dval))

plt.scatter(x, dval, marker = 'x',color = 'red', s = 40 ,label = 'First')

plt.show()

print(max(dval))

#开启一个窗口，num设置子图数量，figsize设置窗口大小，dpi设置分辨率

fig = plt.figure(num=1, figsize=(15, 8),dpi=80)

#直接用plt.plot画图，第一个参数是表示横轴的序列，第二个参数是表示纵轴的序列

plt.plot(x,yres)

plt.plot(x,data['占有率均值'])

#显示绘图结果

plt.show()

# 三参数各自表达式画图及预测和误差分析

# 速度-时间（流量-时间、占有率-时间类似）

import matplotlib.pyplot as plt

import numpy as np

import math

# x = range(1,dval.shape[0]+1,1)

# x= np.array(x)

Xlabel = range(1,dval.shape[0]+1+200,1)

Xlabel = np.array(Xlabel)

a1 = 13.33

b1 = 64.76

c1 = 18.38

a2 = 59.19

b2 = 286.9

c2 = 84.72

a3 = 37.64

b3 = 139

c3 = 35.25

a4 = 59.91

b4 = 15.44

c4 = 103.8

a5 = 26.85

b5 = 208

c5 = 44.01

a6 = 9.154

b6 = 173.1

c6 = 18.65

f1=[]

for x in range(1,dval.shape[0]+1+200,1):

f1.append(a1\*math.exp(-((x-b1)/c1)\*\*2) + a2\*math.exp(-((x-b2)/c2)\*\*2) + a3\*math.exp(-((x-b3)/c3)\*\*2) + a4\*math.exp(-((x-b4)/c4)\*\*2) + a5\*math.exp(-((x-b5)/c5)\*\*2) + a6\*math.exp(-((x-b6)/c6)\*\*2))

print(f1)

#开启一个窗口，num设置子图数量，figsize设置窗口大小，dpi设置分辨率

fig = plt.figure(num=1, figsize=(15, 8),dpi=80)

#直接用plt.plot画图，第一个参数是表示横轴的序列，第二个参数是表示纵轴的序列

plt.plot(Xlabel,f1)

# plt.plot(Xlabel,data["速度均值"].values)

#显示绘图结果

plt.show()

#画出散点图

plt.scatter(x, dval, marker = 'x',color = 'red', s = 40 ,label = 'First')

plt.show()

print(max(dval))

**附录3**

预处理后补全缺省值后数据集（见附件）