指标计算

indice\_ls=[]

for i in range(402):

pur\_i = pur[i]

sup\_i = sup[i]

pur\_count = np.nonzero(pur\_i)[0].shape[0] #下单次数

sup\_count = np.nonzero(sup\_i)[0].shape[0] #供货次数

feedback\_ratio = sup\_count / pur\_count #订单响应比率

avg\_feedback\_time, avg\_weighted\_feedback\_time, default\_nums, feedback\_time\_ls = cal\_feedback\_time(

pur\_i, sup\_i) #平均响应时间，平均加权响应时间,未响应的订单数，订单响应时间序列

'''#计算平均到货率，10个生产周期的到货率序列，

&以及每个有实际供货的生产周期的\*最大周供应量的平均值\*和\*最大周供应量序列\*'''

avg\_arrival\_ratio, arrival\_ratio\_ls, avg\_max\_period\_weeksup, max\_period\_weeksup\_ls, SYI, CV = cal\_period(

pur[i], sup[i])#6

max\_week\_sup = sup[i].max() #五年最大周供应量

median\_week\_sup = np.median(sup[i][np.nonzero(sup[i])]) #五年供应量的中位数

indicators=(pur\_count, sup\_count, #2

feedback\_ratio, avg\_feedback\_time, avg\_weighted\_feedback\_time, default\_nums, avg\_arrival\_ratio, #5

max\_week\_sup, median\_week\_sup,avg\_max\_period\_weeksup, #3

SYI, CV,#2

feedback\_time\_ls, arrival\_ratio\_ls, max\_period\_weeksup\_ls)#3

indice\_ls.append(indicators )

def cal\_period(pur\_i,sup\_i):

pur\_period=[]

sup\_period=[]

max\_period\_weeksup\_ls=[]

for i in range(10):

pur\_year\_i = pur\_i[24\*i:24\*(i+1)]

sup\_year\_i = sup\_i[ 24\*i:24\*(i+1)]

pur\_period.append(pur\_year\_i.sum())

sup\_period.append(sup\_year\_i.sum())

max\_period\_weeksup\_ls.append(sup\_year\_i.max())

pur\_period=np.array(pur\_period)

sup\_period=np.array(sup\_period)

max\_period\_weeksup\_ls=np.array(max\_period\_weeksup\_ls)

avg\_max\_period\_weeksup=np.array(max\_period\_weeksup\_ls)[np.nonzero(sup\_period)].mean()#去掉无供货的生产周期

arrival\_ratio\_ls=sup\_period/pur\_period

avg\_arrival\_ratio=arrival\_ratio\_ls[np.nonzero(pur\_period)].mean()#计算平均到货率时，去掉无订单的生产周期

SYI=(sup\_period[np.nonzero(pur\_period)].mean()-sup\_period[np.nonzero(pur\_period)].std())/sup\_period[np.nonzero(pur\_period)].max()

CV=sup\_period[np.nonzero(pur\_period)].std()/sup\_period[np.nonzero(pur\_period)].mean()

return avg\_arrival\_ratio, arrival\_ratio\_ls, avg\_max\_period\_weeksup, max\_period\_weeksup\_ls,SYI,CV

def cal\_feedback\_time(pur\_i,sup\_i):

pur\_non=np.nonzero(pur\_i)[0]#np.nonzero()返回的是tuple套array，需要手动变为array

sup\_non=np.nonzero(sup\_i)[0]#np.nonzero()返回的是tuple套arra，需要手动变为array

feedback\_time\_ls=[]

# print(pur\_non)

for i in range(len(pur\_non)):

pur\_non\_i=pur\_non[i]

'''求订单i对应的最近的供给i的时间

存在多个订单对应一个供给的情况'''

# print(pur\_non\_i)

a=sup\_non -pur\_non\_i

b=np.where(a>=0,True,False)

a=a[b]

if max(sup\_non)<pur\_non\_i:#如果订单日期后无供货响应

feedback\_time\_i=-1#如果订单迄今没有响应，则响应时间记为-1

#np.inf#如果订单迄今没有响应，则响应时间记为正无穷

else:

feedback\_time\_i=min(a)

feedback\_time\_ls.append(feedback\_time\_i)

feedback\_time\_ls=np.array(feedback\_time\_ls)

weighted\_feedback\_time\_ls=feedback\_time\_ls\*pur\_i[pur\_non]

default\_nums=np.where(feedback\_time\_ls<0,1,0).sum()#记录没有响应的订单次数

avg\_feedback\_time=feedback\_time\_ls[np.where(feedback\_time\_ls>=0,True,False)].mean()

avg\_weighted\_feedback\_time=weighted\_feedback\_time\_ls[np.where(feedback\_time\_ls>=0,True,False)].mean()

# print(feedback\_time\_ls)

return avg\_feedback\_time,avg\_weighted\_feedback\_time,default\_nums ,feedback\_time\_ls

def cal\_fb\_sup\_time(s=sup[0],i=42):

a=np.array(np.nonzero(s)) -i

b=np.where(a>=0,True,False)

if b==np.array([]):

sup\_non\_i=np.inf#如果订单迄今没有响应，则响应时间记为正无穷

a=a[b]

sup\_non\_i=min(a)

return sup\_non\_i

def tral():

pur\_non=np.nonzero(pur[0])[0]#np.nonzero()返回的是tuple套array，需要手动变为array

sup\_non=np.nonzero(sup[0])[0]#np.nonzero()返回的是tuple套arra，需要手动变为array

for i in range(len(pur\_non)):

pur\_non\_i=pur\_non[i]

print(cal\_fb\_sup\_time(sup[0],pur\_non\_i))

tral()

df\_indice = pd.DataFrame(data=np.c\_[names,types,indice\_array],

columns=['供货商ID','材料分类',

'下单周次数',

'供货次数',

'订单响应比率',

'平均响应时间',

'加权平均响应时间',

'沉没订单数',

'平均到货率',

'五年最大周供应量', '五年周供应量的中位数','五年生产周期内最大周供应量的平均值', 'SYI', 'CV',

'五年响应时间序列', '五年到货率序列', '五年生产周期内的最大周供应量序列'

])

# df\_indice.to\_excel('附件1指标.xlsx',index=False)

df\_indice

指标筛选

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

# 当VIF<10,说明不存在多重共线性；当10<=VIF<100,存在较强的多重共线性，当VIF>=100,存在严重多重共线性

x=df\_only\_ind

x1=df\_only\_ind.iloc[:,2:-1]

x2=df\_only\_ind.iloc[:,3:-1]

vif = [variance\_inflation\_factor(x.values, x.columns.get\_loc(i)) for i in x.columns]

Vif

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

tol = [1./variance\_inflation\_factor(x.values, x.columns.get\_loc(i)) for i in x.columns]

Tol

plt.figure(figsize=(12,8))

sns.heatmap(df\_only\_ind.corr(),annot=True)

Decompose数据分解

# def get\_decom\_data():

decom\_data\_ls=[]

for i in [0,1,5,6,7]:

decomposition=seasonal\_decompose(tran[i],period=24)

trend = decomposition.trend #趋势效应

seasonal = decomposition.seasonal #季节效应

residual = decomposition.resid #随机效应

data\_i\_array=np.c\_[trend,seasonal,residual].T

decom\_data\_ls.append(data\_i\_array)

# for i in range(len(decom\_data\_ls)):

decom\_data\_array=np.r\_[decom\_data\_ls[0],decom\_data\_ls[1],decom\_data\_ls[2],decom\_data\_ls[3],decom\_data\_ls[4] ]

np.isnan(decom\_data\_array).sum(axis=1)

from statsmodels.tsa.seasonal import seasonal\_decompose

# plt.figure(figsize=(12,20))

result1 = seasonal\_decompose(tran[0],period=24)

# result1.trend.shape,result1.seasonal.shape,

np.c\_[result1.trend,result1.seasonal].shape

def draw\_decom(tran\_i):

decomposition=seasonal\_decompose(tran\_i,period=24)

trend = decomposition.trend #趋势效应

seasonal = decomposition.seasonal #季节效应

residual = decomposition.resid #随机效应

plt.figure(figsize=(16,8))

plt.subplot(411)

plt.plot(tran\_i, label=u'原始数据')

plt.legend(loc='best')

plt.xlabel('周')

# plt.ylabel('损耗率%')

plt.subplot(412)

plt.plot(trend, label=u'趋势')

plt.legend(loc='best')

plt.xlabel('周')

# plt.ylabel('损耗率%')

plt.subplot(413)

plt.plot(seasonal,label=u'季节性')

plt.legend(loc='best')

plt.xlabel('周')

# plt.ylabel('损耗率%')

plt.subplot(414)

plt.plot(residual, label=u'残差')

plt.legend(loc='best')

plt.xlabel('周')

# plt.ylabel('损耗率%')

plt.tight\_layout()

#return np.c\_[trend, seasonal, residual].T

draw\_decom(tran[0])

def cal\_tran3\_period(tran\_3):

tran\_3=tran\_3.copy()

tran\_3[tran\_3==max(tran\_3)]=0

period=[]

tran\_3\_i=tran\_3[:24]

for i in range(1,10):

tran\_3\_i = np.c\_[tran\_3\_i,tran\_3[24\*i:24\*(i+1)]]

# plt.plot(tran\_3\_i)

tran\_3\_i=tran\_3\_i.T

avg\_week=tran\_3\_i.sum(axis=0)/np.count\_nonzero(tran\_3\_i,axis=0)

plt.plot(avg\_week)

return tran\_3\_i ,avg\_week

cal\_tran3\_period(tran\_3)[1]

def draw2(tran\_3):

plt.figure(figsize=(16,6))

for i in range(10):

tran\_3\_i = tran\_3[24\*i:24\*(i+1)]

# tran\_3\_i=tran\_3\_i[np.nonzero(tran\_3\_i)]

plt.subplot(121)

plt.scatter(range(1,25),tran\_3\_i,label='生产周期{}'.format(i+1))

# plt.legend()

plt.subplot(122)

plt.plot(range(1,25),tran\_3\_i,label='生产周期{}'.format(i+1))

# plt.legend()

plt.show()

draw2(tran\_3)

数据探索与可视化

A=df\_pur[df\_pur['材料分类']=='A'].iloc[:,2:].values

A\_year=A[:, :24]

# Period=[]

# Period.append(A\_year.sum())

# for i in range(1,10):

# A\_year +=A[:, 24\*i:24\*(i+1)]

# Period.append(A[:,24\*i:24\*(i+1)].sum())

# A\_year,Period

def cal\_year(A):

A\_year=A[:, :24]

Period=[]

Period.append(A\_year.sum())

for i in range(1,10):

A\_year +=A[:, 24\*i:24\*(i+1)]

Period.append(A[:,24\*i:24\*(i+1)].sum())

return A\_year,np.array(Period)

A\_year,A\_period =cal\_year(A)

def draw(A\_year):

A\_year\_std=A\_year.std(axis=0)

A\_year\_mean=A\_year.mean(axis=0)

x=range(1,25)

plt.figure(figsize=(15,6))

plt.subplot(131)

plt.plot(x,A\_year\_std,label='std')

plt.title('std')

plt.xlim((1,24))

plt.subplot(132)

plt.plot(x,A\_year\_mean,label='mean')

plt.title('mean')

plt.xlim((1,24))

plt.subplot(133)

plt.plot(x,A\_year.sum(axis=0),label='mean')

plt.title('sum')

plt.xlim((1,24))

plt.show()

draw(A\_year)

B=df\_pur[df\_pur['材料分类']=='B'].iloc[:,2:].values

def cal\_year(A):

A\_year=A[:, :24]

Period=[]

Period.append(A\_year.sum())

for i in range(1,10):

A\_year +=A[:, 24\*i:24\*(i+1)]

Period.append(A[:,24\*i:24\*(i+1)].sum())

return A\_year,np.array(Period)

B\_year,B\_period =cal\_year(B)

draw(B\_year)

C=df\_pur[df\_pur['材料分类']=='C'].iloc[:,2:].values

def cal\_year(A):

A\_year=A[:, :24]

Period=[]

Period.append(A\_year.sum())

for i in range(1,10):

A\_year +=A[:, 24\*i:24\*(i+1)]

Period.append(A[:,24\*i:24\*(i+1)].sum())

return A\_year,np.array(Period)

C\_year,C\_period =cal\_year(C)

draw(C\_year)

plt.plot(range(1,11),Period)

plt.title('十个生产周期总订单的的逐期变化')

plt.show()

x=range(1,11)

for i in range(3):

plt.plot(x,[j[i] for j in S\_period],label="供应商S00{}".format(i+1))

plt.legend()

plt.show()

plt.figure(figsize=(16,6))

plt.plot(range(1,241),pur.sum(axis=0),label='订货量')

plt.plot(range(1,241),sup.sum(axis=0),label='供货量')

plt.legend()

plt.xlabel('周')

plt.ylabel('原材料数量$m^3$')

plt.figure(figsize=(16,6))

plt.plot(range(1,25),pur\_year.mean(axis=0),label='订货量')

plt.plot(range(1,25),sup\_year.mean(axis=0),label='供货量')

plt.legend()

plt.xticks(range(25),range(25))

plt.grid(axis='x')

plt.xlabel('周')

plt.ylabel('原材料数量$m^3$')

stock\_A=A\_sup.sum(axis=0)

stock\_B=B\_sup.sum(axis=0)

stock\_C=C\_sup.sum(axis=0)

stock\_ABC=stock\_A+stock\_B+stock\_C

#订货量

plt.figure(figsize=(16,6))

plt.plot(cal\_year(A)[0].mean(axis=0),label='A')

plt.plot(cal\_year(B)[0].mean(axis=0),label='B')

plt.plot(cal\_year(C)[0].mean(axis=0),label='C')

plt.legend()

#供货量

plt.figure(figsize=(16,6))

plt.plot(cal\_year(A\_sup)[0].mean(axis=0),label='A')

plt.plot(cal\_year(B\_sup)[0].mean(axis=0),label='B')

plt.plot(cal\_year(C\_sup)[0].mean(axis=0),label='C')

plt.legend()

plt.figure(figsize=(16,6))

plt.plot(stock\_A/stock\_ABC,label='A')

plt.plot(stock\_B/stock\_ABC,label='B')

plt.plot(stock\_C/stock\_ABC,label='C')

plt.legend()

# plt.axhline(y=2\*28200,c='r')

# plt.axhline(y=stock\_ABC.mean(),c='y')#平均库存量

# plt.ylabel('库存量')

def cal\_stock(worn\_rate=0.025, prod=prod):

stock\_in=28200\*2#假设已有库存为两周产能，即2\*2822

# stock\_ls=[]

# worn\_rate=0.025

stock\_worn=28200\*2#假设已有库存为两周产能，即2\*2822

stock\_worn\_ls=[]

gap\_ls=[]

for i in prod:

stock\_in+=i

stock\_worn+=i\*(1-worn\_rate)

gap=0

if stock\_worn-28200<0:

gap=stock\_worn-28200

stock\_worn=0

# elif stock\_in-28200<0:

# stock\_in=0

else:

# stock\_in=stock\_in-28200

stock\_worn=stock\_worn-28200

gap=0

# stock\_ls.append(stock\_in)#库存量

stock\_worn\_ls.append(stock\_worn)#考虑转运损耗后的库存量

gap\_ls.append(gap)

# stock\_array=np.array(stock\_ls)

stock\_worn\_array=np.array(stock\_worn\_ls)

gap\_array=np.array(gap\_ls)

return stock\_worn\_array,gap\_array

np.count\_nonzero(cal\_stock()[0])

def cal\_stock\_1(prod=prod):

# stock\_in=28200\*2#假设已有库存为两周产能，即2\*2822

# stock\_ls=[]

# worn\_rate=0.025

stock\_worn=28200\*2#假设已有库存为两周产能，即2\*2822

stock\_worn\_ls=[]

gap\_ls=[]

for i in range(240):

# stock\_in+=

worn\_rate=avg\_week\_worn[i]/100

stock\_worn+=prod[i]\*(1-worn\_rate)

gap=0

if stock\_worn-28200<0:

gap=stock\_worn-28200

stock\_worn=0

# elif stock\_in-28200<0:

# stock\_in=0

else:

# stock\_in=stock\_in-28200

stock\_worn=stock\_worn-28200

gap=0

# stock\_ls.append(stock\_in)#库存量

stock\_worn\_ls.append(stock\_worn)#考虑转运损耗后的库存量

gap\_ls.append(gap)

# stock\_array=np.array(stock\_ls)

stock\_worn\_array=np.array(stock\_worn\_ls)

gap\_array=np.array(gap\_ls)

return stock\_worn\_array,gap\_array

np.count\_nonzero(cal\_stock()[0])

plt.figure(figsize=(16,6))

plt.plot(range(1,241),(cal\_stock\_1()[1]+28200)/28200,c='#F5B14C')

plt.ylabel('产能利用率')

plt.xticks([1,50,100,150,200,240],[1,50,100,150,200,240])

plt.xlim((1,240))

plt.ylim((0.6,1.01))

plt.fill\_between(range(1,241), 0.6, (cal\_stock\_1()[1]+28200)/28200, facecolor='#F5B14C', alpha=0.3)

plt.axhline(y=0.9804187153020208,c='#661D98')

plt.annotate(

'总产能利用率=98.04%',# 注释的文字

xy=(110,0.97),# 箭头箭尖的位置

xytext=(122,0.92),# 注释文字的位置

color = "black",# 文字的颜色

fontsize = 15,# 文字的尺寸

arrowprops=dict(facecolor='black', shrink=0.001))

plt.show()

plt.figure(figsize=(16,6))

# plt.plot((cal\_stock(0)[1]+28200)/28200,label='损耗率=0')

plt.plot((cal\_stock(0.01)[1]+28200)/28200, label='损耗率=1%')

plt.plot((cal\_stock(0.02)[1]+28200)/28200, label='损耗率=2%')

plt.plot((cal\_stock(0.03)[1]+28200)/28200, label='损耗率=3%')

plt.legend(loc='best')

# plt.axhline(y=2\*28200,c='r')

# plt.axhline(y=stock\_array.mean(),c='y')#平均库存量

plt.ylabel('产能利用率')

plt.figure(figsize=(12,6))

plt.plot(cal\_stock(0)[0],label='损耗率=0')

plt.plot(cal\_stock\_1()[0], label='损耗率=每周平均损耗率')

# plt.plot(cal\_stock(0.02), label='损耗率=2%')

# plt.plot(cal\_stock(0.03), label='损耗率=3%')

plt.legend()

plt.axhline(y=2\*28200,c='r')

plt.annotate(text='两周生产量',xy=(160,28200\*2),xytext=(165,40000),color='g',arrowprops=dict(facecolor='black', shrink=0.5))

# plt.axhline(y=stock\_array.mean(),c='y')#平均库存量

plt.ylabel('库存可生产的产品量')

plt.figure(figsize=(16,6))

plt.subplot(211)

plt.plot(np.cumsum(prod))#累计到货率——不计消耗

plt.plot(np.cumsum(28200\*np.ones(240)),c='r')

plt.subplot(212)

plt.plot(np.cumsum(prod)-np.cumsum(28200\*np.ones(240)),)

数据录入

def clean\_trans(plan\_trans\_3):

# week=1

all\_ls=[]

for sup\_name in range(1,plan\_trans\_3.iloc[:,0].nunique()+1):

# plan\_trans\_3[plan\_trans\_3['周数']==week]

# sup\_name=1

i\_ls=[sup\_name]

for week in range(1,25):

i\_week\_trans=plan\_trans\_3[plan\_trans\_3['周数']==week][plan\_trans\_3['供应商']==sup\_name].iloc[:,-1].values

i\_ls.extend(i\_week\_trans)

print(len(i\_week\_trans))

all\_ls.append(i\_ls)

all\_array=np.array(all\_ls)

return all\_array

clean\_trans(plan\_trans\_3).shape

ranked\_names=pd.read\_excel('./剔除指标后/新供应商排名.xlsx').iloc[:,0]#.values

ranked\_names.nunique()

rank2names=[(i,ranked\_names[i]) for i in range(402)]

rank2names=dict(rank2names)

rank2names

pd.DataFrame(np.c\_[np.arange(1,403),np.zeros((402,192))]).to\_excel('转运方案0值底稿.xlsx',index=False)

cleand\_tran3[0]=cleand\_tran3[0].map(rank2names)

cleand\_tran3.to\_excel('cleaned问题3转运方案(有供应商名称).xlsx',index=False,)

cleand\_tran3.index=cleand\_tran3[0]

cleand\_tran3

tmpl=pd.read\_excel('./转运方案None值底稿.xlsx')

tmpl.iloc[cleand\_tran3.index-1,:]=cleand\_tran3.values

tmpl.to\_excel('问题3转运方案（正式版）.xlsx',index=False)

tmpl.iloc[cleand\_tran3.index-1,:]

plan\_trans\_4=pd.read\_excel('./plan/转运方案.xlsx',sheet\_name='问题4')

plan\_trans\_4

cleand\_tran4=pd.DataFrame(clean\_trans(plan\_trans\_4))#.to\_excel('cleaned问题3转运方案.xlsx',index=False)

cleand\_tran4

cleand\_tran4[0]=cleand\_tran4[0].map(rank2names)

cleand\_tran4#.to\_excel('cleaned问题4转运方案(有供应商名称).xlsx',index=False,)

cleand\_tran4.index=cleand\_tran4[0]

cleand\_tran4

tmpl=pd.read\_excel('./转运方案None值底稿.xlsx')

tmpl.iloc[cleand\_tran4.index-1,:]=cleand\_tran4.values

tmpl.to\_excel('问题4转运方案（正式版）.xlsx',index=False)

tmpl.iloc[cleand\_tran4.index-1,:]

plan\_sup\_2=pd.read\_excel('./plan/副本pro1buyfinal.xlsx').reset\_index()

plan\_sup\_2.iloc[:,0]=plan\_sup\_2.iloc[:,0].map(rank2names)

plan\_sup\_2.index=plan\_sup\_2.iloc[:,0]

plan\_sup\_2

temp\_sup=pd.read\_excel('./供应方案None值底稿 .xlsx').iloc[:,:25]

# print(temp\_sup.head(20))

temp\_sup.iloc[plan\_sup\_2.index-1,:]=plan\_sup\_2#.values

temp\_sup.to\_excel('问题2供应方案（正式版）.xlsx',index=False)

plan\_sup\_3=pd.read\_excel('./plan/pro3buyf.xlsx').reset\_index()

plan\_sup\_3.iloc[:,0]=plan\_sup\_3.iloc[:,0].map(rank2names)

plan\_sup\_3.index=plan\_sup\_3.iloc[:,0]

plan\_sup\_3

temp\_sup=pd.read\_excel('./供应方案None值底稿 .xlsx').iloc[:,:25]

# print(temp\_sup.head(20))

temp\_sup.iloc[plan\_sup\_3.index-1,:]=plan\_sup\_3#.values

temp\_sup.to\_excel('问题3供应方案（正式版）.xlsx',index=False)

plan\_sup\_4=pd.read\_excel('./plan/pro4buyfinal.xlsx').reset\_index()

plan\_sup\_4.iloc[:,0]=plan\_sup\_4.iloc[:,0].map(rank2names)

plan\_sup\_4.index=plan\_sup\_4.iloc[:,0]

plan\_sup\_4

temp\_sup=pd.read\_excel('./供应方案None值底稿 .xlsx').iloc[:,:25]

# print(temp\_sup.head(20))

temp\_sup.iloc[plan\_sup\_4.index-1,:]=plan\_sup\_4#.values

temp\_sup.to\_excel('问题4供应方案（正式版）.xlsx',index=False)

实施方案评价

name2type=dict(zip(np.arange(1,403),types))

name2type

def cal\_stock\_1(prod=prod,avg\_week\_worn=avg\_week\_worn,init=28200\*2):

# stock\_in=28200\*2#假设已有库存为两周产能，即2\*2822

# stock\_ls=[]

# worn\_rate=0.025

stock\_worn=init#假设已有库存为两周产能，即2\*2822

stock\_worn\_ls=[]

gap\_ls=[]

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

# stock\_in+=

worn\_rate=avg\_week\_worn[i]/100

stock\_worn+=prod[i]\*(1-worn\_rate)

gap=0

if stock\_worn-28200<0:

gap=stock\_worn-28200

stock\_worn=0

# elif stock\_in-28200<0:

# stock\_in=0

else:

# stock\_in=stock\_in-28200

stock\_worn=stock\_worn-28200

gap=0

# stock\_ls.append(stock\_in)#库存量

stock\_worn\_ls.append(stock\_worn)#考虑转运损耗后的库存量

gap\_ls.append(gap)

# stock\_array=np.array(stock\_ls)

stock\_worn\_array=np.array(stock\_worn\_ls)

gap\_array=np.array(gap\_ls)

return stock\_worn\_array,gap\_array

np.count\_nonzero(cal\_stock()[0])

worn=pd.read\_excel('损耗率.xlsx').iloc[:,1:]

worn.shape

remark\_p\_2=pd.read\_excel('./plan/问题2供应方案（正式版）.xlsx')

remark\_p\_2.iloc[:,0]=remark\_p\_2.iloc[:,0].map(name2type)

p2A=remark\_p\_2[remark\_p\_2.iloc[:,0]=='A'].fillna(0).values[:,1:]

p2B=remark\_p\_2[remark\_p\_2.iloc[:,0]=='B'].fillna(0).values[:,1:]

p2C=remark\_p\_2[remark\_p\_2.iloc[:,0]=='C'].fillna(0).values[:,1:]

p2=remark\_p\_2.fillna(0).values[:,1:]

p2

plt.figure(figsize=(12,6))

# plt.plot(cal\_stock(0)[0],label='损耗率=0')

plt.plot(range(1,25),y,label='实时库存量')

#cal\_stock\_1(prod2,worn\_2)[0]) #label='损耗率=每周平均损耗率')

# plt.plot(cal\_stock(0.02), label='损耗率=2%')

# plt.plot(cal\_stock(0.03), label='损耗率=3%')

# plt.legend()

plt.axhline(y=2\*28200,c='r')

plt.annotate(

'满足两周生产的库存量',# 注释的文字

xy=(11,2\*28200),# 箭头箭尖的位置

xytext=(12,2\*26000),# 注释文字的位置

color = "black",# 文字的颜色

fontsize = 12,# 文字的尺寸

arrowprops=dict(facecolor='black', shrink=0.001))

# plt.axhline(y=stock\_array.mean(),c='y')#平均库存量

plt.xticks(range(25),np.arange(25))

plt.ylabel('库存可生产的产品量')

plt.grid(axis='x')

plt.legend()

#采购花费，1.2, 1.1, 1

p2A.sum()\*1.2+p2B.sum()\*1.1+p2C.sum()\*1.1

t21 = []

for i in range(24):

t21.append(

(

# (t2.iloc[:, 1 + i\*8:8 \* i + 9].values)\*(worn.iloc[:, i + 1].values/100).sum()

(t2.iloc[:,i\*8:8\*(i+1)].sum(axis=0).values\*worn.iloc[:,i]).sum()/100

)

)

t21=np.array(t21)

print(t21.shape)

#总损耗量5502.493

#总量为457525.63622437755

#整体损耗率0.012026635220626317

t21.sum(),t2.sum().sum(),t21.sum()/457525.63622437755

#每周损耗量

t21.sum(axis=1),t21.sum()#,t2.iloc[:,1:].sum()

#总损耗量为550249.3930955562立方米

#每周损耗率

worn\_2=t21/p2.sum(axis=0)

worn\_2

p3.iloc[:,0]=p3.iloc[:,0].map(name2type)

p3A=p3[p3.iloc[:,0]=='A'].fillna(0).values[:,1:]

p3B=p3[p3.iloc[:,0]=='B'].fillna(0).values[:,1:]

p3C=p3[p3.iloc[:,0]=='C'].fillna(0).values[:,1:]

p3=p3.fillna(0).values[:,1:]

p3

plt.figure(figsize=(12,6))

# plt.plot(cal\_stock(0)[0],label='损耗率=0')

plt.plot(range(1,25),cal\_stock\_1(prod3,worn\_2,init=0)[0],label='实时库存量')

#cal\_stock\_1(prod2,worn\_2)[0]) #label='损耗率=每周平均损耗率')

# plt.plot(cal\_stock(0.02), label='损耗率=2%')

# plt.plot(cal\_stock(0.03), label='损耗率=3%')

# plt.legend()

plt.axhline(y=2\*28200,c='r')

plt.annotate(

'满足两周生产的库存量',# 注释的文字

xy=(11,2\*28200),# 箭头箭尖的位置

xytext=(12,2\*26000),# 注释文字的位置

color = "black",# 文字的颜色

fontsize = 12,# 文字的尺寸

arrowprops=dict(facecolor='black', shrink=0.001))

# plt.axhline(y=stock\_array.mean(),c='y')#平均库存量

plt.xticks(range(25),np.arange(25))

plt.ylabel('库存可生产的产品量')

plt.grid(axis='x')

plt.legend()