**问题一**

**1.各公司总金额税额统计**

import pandas as pd

file\_path = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/附件2.xlsx'

input = pd.read\_excel(file\_path, sheet\_name='进项发票信息')

output = pd.read\_excel(file\_path, sheet\_name='销项发票信息')

# 删除作废发票（假设“发票状态”列标记为“作废”）

input = input[input['发票状态'] != '作废发票']

output = output[output['发票状态'] != '作废发票']

Ja = input.groupby('企业代号')['金额'].sum().reset\_index()

# 进项发票金额

Jt = input.groupby('企业代号')['税额'].sum().reset\_index()

# 进项发票税额

Xa = output.groupby('企业代号')['金额'].sum().reset\_index()

# 销项发票金额

Xt = output.groupby('企业代号')['税额'].sum().reset\_index()

# 销项发票税额

# 第一次合并

temp1 = pd.merge(Ja, Jt, on='企业代号', how='outer')

# 第二次合并

temp2 = pd.merge(temp1, Xa, on='企业代号', how='outer')

# 第三次合并

result = pd.merge(temp2, Xt, on='企业代号', how='outer')

# 填充缺失值为0

result = result.fillna(0)

# 导出到新的Excel文件

result.to\_excel('302金额税额表.xlsx', index=False)

print("已保存到 '302金额税额表.xlsx'")

**2.利润标准化**

import pandas as pd

from sklearn.preprocessing import StandardScaler

file\_path = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/302金额税额表.xlsx'

df = pd.read\_excel(file\_path)

#标准化

scaler = StandardScaler()

df['标准化利润'] = scaler.fit\_transform(df[['利润']])

# 导出到新的Excel文件

df.to\_excel('302标准化利润表.xlsx', index=False)

print("已保存")

**3.计算变异系数**

import pandas as pd

file\_path = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/附件2.xlsx'

df = pd.read\_excel(file\_path, sheet\_name='进项发票信息')

#df = pd.read\_excel(file\_path, sheet\_name='销项发票信息')

df = df[df['发票状态'] != '作废发票']

df['开票日期'] = pd.to\_datetime(df['开票日期'], format='%Y/%m/%d')

df['月份'] = df['开票日期'].dt.strftime('%Y/%m')

monthly\_sum = df.groupby(['企业代号', '月份'])[['金额']].sum().reset\_index()

# 对每个公司，计算金额的均值、标准差和变异系数

result = monthly\_sum.groupby('企业代号').agg(

进项金额均值=('金额', 'mean'),

进项金额标准差=('金额', 'std'),

).reset\_index()

# 4. 计算变异系数

#result['进项金额变异系数'] = result['进项金额标准差'] / result['进项金额均值']

result['进项金额变异系数'] = result['进项金额标准差'] / result['进项金额均值']

# 5. 保存结果到Excel

result.to\_excel('302进项金额变异系数.xlsx', index=False)

**4.合并各个指标**

import pandas as pd

os = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷'

#file\_path\_1 = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/附件1.xlsx'

#company = pd.read\_excel(file\_path\_1,sheet\_name='企业信息')

file\_path\_2 = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/302进项金额变异系数.xlsx'

jinxiang\_bianyi = pd.read\_excel(file\_path\_2)

file\_path\_3 = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/302销项金额变异系数.xlsx'

xiaoxiang\_bianyi = pd.read\_excel(file\_path\_3)

file\_path\_4 = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/302发票率.xlsx'

df\_fapiaorate = pd.read\_excel(file\_path\_4)

file\_path\_5 = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/302标准化利润表.xlsx'

df\_p = pd.read\_excel(file\_path\_5)

file\_path\_6 = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/302金额税额表.xlsx'

jinershuier = pd.read\_excel(file\_path\_6)

# 利润

p = df\_p[['企业代号', '标准化利润']]

# 进项变异系数

cv\_a = jinxiang\_bianyi[['企业代号', '进项金额变异系数']]

# 销项变异系数

cv\_b = xiaoxiang\_bianyi[['企业代号', '销项金额变异系数']]

# 进项有效率

# er\_a = df\_youxiao[['企业代号', '进项有效概率']]

# 销项有效率

# er\_b = df\_youxiao[['企业代号', '销项有效概率']]

# 是否违约

#y = company[['企业代号', '违约']]

# 有效概率

# er = df\_youxiao[['企业代号', '有效概率']]

# 进项金额

jinxiangjine = jinershuier[['企业代号','进项金额']]

# 销项金额

xiaoxiangjine = jinershuier[['企业代号','销项金额']]

# 进项作废发票率

rate\_jinxiangzuofei = df\_fapiaorate[['企业代号','进项发票作废率']]

# 进项负金额发票率

rate\_jinxiangfu = df\_fapiaorate[['企业代号','进项负金额发票率']]

# 销项作废发票率

rate\_xiaoxiangzuofei = df\_fapiaorate[['企业代号','销项发票作废率']]

# 销项负金额发票率

rate\_xiaoxiangfu = df\_fapiaorate[['企业代号','销项负金额发票率']]

# 按企业代号合并

#merged = y.merge(p, on='企业代号', how='outer').merge(cv\_a, on='企业代号', how='outer').merge(cv\_b, on='企业代号', how='outer').merge(jinxiangjine, on='企业代号', how='outer').merge(xiaoxiangjine, on='企业代号', how='outer').merge(rate\_jinxiangzuofei, on='企业代号', how='outer').merge(rate\_jinxiangfu, on='企业代号', how='outer').merge(rate\_xiaoxiangzuofei, on='企业代号', how='outer').merge(rate\_xiaoxiangfu, on='企业代号', how='outer')

merged = p.merge(cv\_a, on='企业代号', how='outer').merge(cv\_b, on='企业代号', how='outer').merge(jinxiangjine, on='企业代号', how='outer').merge(xiaoxiangjine, on='企业代号', how='outer').merge(rate\_jinxiangzuofei, on='企业代号', how='outer').merge(rate\_jinxiangfu, on='企业代号', how='outer').merge(rate\_xiaoxiangzuofei, on='企业代号', how='outer').merge(rate\_xiaoxiangfu, on='企业代号', how='outer')

#pd.set\_option('future.no\_silent\_downcasting', True)

#merged['违约'] = merged['违约'].replace({'是': 0, '否': 1})

# 保存结果

merged.to\_excel('302指标汇总.xlsx', index=False)

**5.作废发票占比、负金额发票占比**

import pandas as pd

file\_path = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/附件2.xlsx'

input = pd.read\_excel(file\_path, sheet\_name='进项发票信息')

output = pd.read\_excel(file\_path, sheet\_name='销项发票信息')

A\_stats = input.groupby('企业代号').agg(

进项总发票数=('发票状态', 'count'),

进项作废发票数=('发票状态', lambda x: (x == '作废发票').sum()),

进项负金额发票数=('金额', lambda x: (x < 0).sum())

).reset\_index()

B\_stats = output.groupby('企业代号').agg(

销项总发票数=('发票状态', 'count'),

销项作废发票数=('发票状态', lambda x: (x == '作废发票').sum()),

销项负金额发票数=('金额', lambda x: (x < 0).sum())

).reset\_index()

# 合并A表和B表统计结果

result = pd.merge(A\_stats, B\_stats, on='企业代号', how='outer').fillna(0)

# 4. 保存结果

result.to\_excel('302发票率.xlsx', index=False)

**6.比较不同曲线拟合效果**

import pandas as pd  
import numpy as np  
from sklearn.metrics import r2\_score  
import matplotlib.pyplot as plt  
plt.rcParams['font.sans-serif'] = ['SimHei'] *# 黑体，或改为你系统有的中文字体*plt.rcParams['axes.unicode\_minus'] = False *# 正确显示负号  
  
# 1. 读取Excel数据*df = pd.read\_excel('D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/表格/附件3.xlsx', header=[0,1]) *# 修改为你的文件路径  
  
# 读取各列数据（根据你的 MultiIndex 实际列名）*y = df[('贷款年利率', 'Unnamed: 0\_level\_1')].values  
xa = df[('客户流失率', '信誉评级A')].values  
xb = df[('客户流失率', '信誉评级B')].values  
xc = df[('客户流失率', '信誉评级C')].values  
  
x = xc  
  
*# 一次函数拟合*coef1 = np.polyfit(x, y, 1)  
y\_pred1 = np.polyval(coef1, x)  
r2\_1 = r2\_score(y, y\_pred1)  
  
*# 二次函数拟合*coef2 = np.polyfit(x, y, 2)  
y\_pred2 = np.polyval(coef2, x)  
r2\_2 = r2\_score(y, y\_pred2)  
  
*# 三次函数拟合*coef3 = np.polyfit(x, y, 3)  
y\_pred3 = np.polyval(coef3, x)  
r2\_3 = r2\_score(y, y\_pred3)  
  
print(f'一次函数拟合的R²: {r2\_1:.4f}')  
print(f'二次函数拟合的R²: {r2\_2:.4f}')  
print(f'三次函数拟合的R²: {r2\_3:.4f}')  
  
*# 可选：画图显示拟合效果*plt.scatter(x, y, label='原始数据')  
plt.plot(x, y\_pred1, label='一次拟合', linestyle='-')  
plt.plot(x, y\_pred2, label='二次拟合', linestyle='--')  
plt.plot(x, y\_pred3, label='三次拟合', linestyle='-.')  
plt.legend()  
plt.show()

**7.求流失率与利率函数关系**

import pandas as pd

import numpy as np

from sklearn.metrics import r2\_score

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei'] # 黑体，或改为你系统有的中文字体

plt.rcParams['axes.unicode\_minus'] = False # 正确显示负号

# 1. 读取Excel数据

df = pd.read\_excel('D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/表格/附件3.xlsx', header=[0,1]) # 修改为你的文件路径

# 读取各列数据（根据你的 MultiIndex 实际列名）

y = df[('贷款年利率', 'Unnamed: 0\_level\_1')].values

xa = df[('客户流失率', '信誉评级A')].values

xb = df[('客户流失率', '信誉评级B')].values

xc = df[('客户流失率', '信誉评级C')].values

x = xc

# 三次函数拟合

coef3 = np.polyfit(x, y, 3)

y\_pred3 = np.polyval(coef3, x)

r2\_3 = r2\_score(y, y\_pred3)

equation = f"y = {coef3[0]:.4f}x³ + {coef3[1]:.4f}x² + {coef3[2]:.4f}x + {coef3[3]:.4f}"

print("三次拟合方程为：", equation)

**8.遗传算法进行决策贷款方案**

import numpy as np

import pandas as pd

from geneticalgorithm import geneticalgorithm as ga

from openpyxl.utils.dataframe import dataframe\_to\_rows

# 假设企业数据已加载

data = pd.read\_excel(

'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/表格/123信誉及风险.xlsx')

N = len(data)

T = 1e9 # 总额度上限

# B级降级为C级

for i in range(N):

credit = data.loc[i, '信誉评级']

if credit == 'B' and data.loc[i, '违约'] == '是':

credit = 'C'

# 获取所有C级企业的违约概率

c\_credit\_data = data[data['信誉评级'] == 'C'] # 过滤出C级企业

c\_credit\_data\_sorted = c\_credit\_data.sort\_values(by='违约概率', ascending=False) # 按违约概率降序排列

# 计算C级企业中前9%的阈值

threshold = c\_credit\_data\_sorted['违约概率'].quantile(0.91) # 获取前9%的违约概率阈值

print('阈值：',threshold)

# 定义流失率函数，根据信誉等级选择不同的函数

def func\_lr\_A(ri):

# A级企业的流失率函数

return 0.1859 \* ri\*\*3 - 0.1280 \* ri\*\*2 + 0.0814 \* ri + 0.0378

def func\_lr\_B(ri):

# B级企业的流失率函数

return 0.1677 \* ri\*\*3 - 0.0921 \* ri\*\*2 + 0.0756 \* ri + 0.0385

def func\_lr\_C(ri):

# C级企业的流失率函数

return 0.1425 \* ri\*\*3 - 0.0678 \* ri\*\*2 + 0.0731 \* ri + 0.0388

# 修改后的约束处理函数

def constraint\_handling(x): # x: [a1, r1, a2, r2, ...]

total\_loan = 0

fitness\_sum = 0

for i in range(N):

ai, ri = x[2 \* i], x[2 \* i + 1]

credit = data.loc[i, '信誉评级']

di = data.loc[i, '违约概率']

# D级不放贷

if credit == 'D':

ai = 0

# C级前9%不放贷

if credit == 'C' and di > threshold:

ai = 0

# ai范围

ai = np.clip(ai, 1e5, 1e6)

# ri范围

ri = np.clip(ri, 0.04, 0.15)

# 根据信誉等级选择流失率函数

if credit == 'A':

lr = func\_lr\_A(ri)

elif credit == 'B':

lr = func\_lr\_B(ri)

elif credit == 'C':

lr = func\_lr\_C(ri)

elif credit == 'D':

lr = 1

total\_loan += ai

fitness\_sum += (ai \* ri \* (1 - di) - ai \* di) \* (1 - lr)

# 总放贷额度不超过T，若超过则加罚分

if total\_loan > T:

fitness\_sum -= 1e9

return -fitness\_sum # 遗传算法为最大化问题

# 遗传算法参数

varbound = np.array([[1e5, 1e6], [0.04, 0.15]] \* N)

algorithm\_param = {

'max\_num\_iteration': 200,

'population\_size': 100,

'mutation\_probability': 0.1,

'mutation\_type': 'uniform',

'elit\_ratio': 0.01,

'selection\_type': 'roulette',

'parents\_portion': 0.5,

'crossover\_probability': 0.8,

'crossover\_type': 'uniform',

'max\_iteration\_without\_improv': 50, # 设置没有改进的最大迭代次数

}

# 初始化遗传算法模型

model = ga(

function=constraint\_handling,

dimension=2 \* N,

variable\_type='real',

variable\_boundaries=varbound,

algorithm\_parameters=algorithm\_param

)

# 运行遗传算法

model.run()

best\_solution = model.output\_dict['variable']

best\_total\_loan = 0

import pandas as pd

# 收集每家企业的贷款信息到列表中

best\_loan\_distribution = []

for i in range(N):

ai, ri = best\_solution[2 \* i], best\_solution[2 \* i + 1]

credit = data.loc[i, '信誉评级']

di = data.loc[i, '违约概率']

name = data.loc[i, '企业代号']

if credit == 'D':

ai = 0

if credit == 'C' and di > threshold:

ai = 0

best\_loan\_distribution.append({

'企业代号': name,

'信誉评级': credit,

'贷款额度': round(ai, 2),

'利率': round(ri, 4),

'违约概率': di

})

# 将结果转为 DataFrame

df\_result = pd.DataFrame(best\_loan\_distribution)

# 导出为 Excel 文件

df\_result.to\_excel('贷款方案.xlsx', index=False)

print("贷款方案已成功导出为 Excel 文件：贷款方案.xlsx")

**问题二**

**1.BP神经网络对302家企业进行信誉评级**

import pandas as pd

from sklearn.neural\_network import MLPClassifier

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.impute import SimpleImputer

from sklearn.model\_selection import cross\_val\_score, train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.decomposition import PCA

from imblearn.over\_sampling import SMOTE

import matplotlib.pyplot as plt

import seaborn as sns

# 数据路径

io = 'D:/Users/person/AppData/Local/Programs/Python/PythonProject/中小企业信贷/表格'

# 读取A表和B表

A = pd.read\_excel(io + '/123企业违约概率.xlsx')

B = pd.read\_excel(io + '/302企业违约概率.xlsx')

# 1. 数据预处理

X\_train = A.drop(['企业代号', '违约', '不违约概率', '信誉评级'], axis=1)

y\_train = A['信誉评级']

X\_pred = B.drop(['企业代号', '不违约概率'], axis=1)

# 数据缺失值填补

imputer = SimpleImputer(strategy='mean')

X\_train = pd.DataFrame(imputer.fit\_transform(X\_train), columns=X\_train.columns)

X\_pred = pd.DataFrame(imputer.transform(X\_pred), columns=X\_pred.columns)

# 标签编码

le = LabelEncoder()

y\_train\_encoded = le.fit\_transform(y\_train) # A->0, B->1, C->2, D->3

# 特征标准化

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_pred\_scaled = scaler.transform(X\_pred)

# 使用SMOTE进行过采样，以平衡数据

smote = SMOTE(random\_state=42)

X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train\_scaled, y\_train\_encoded)

# PCA降维，减少维度，以减少噪声

pca = PCA(n\_components=0.95) # 保留95%的方差

X\_train\_resampled = pca.fit\_transform(X\_train\_resampled)

X\_pred\_scaled = pca.transform(X\_pred\_scaled)

# 2. 构建神经网络（MLPClassifier）

clf = MLPClassifier(hidden\_layer\_sizes=(128, 64, 32), activation='relu', max\_iter=2000, random\_state=42,

solver='adam', learning\_rate\_init=0.001, early\_stopping=True, validation\_fraction=0.1)

# 划分训练集与验证集

X\_train\_split, X\_val\_split, y\_train\_split, y\_val\_split = train\_test\_split(X\_train\_resampled, y\_train\_resampled, test\_size=0.2, random\_state=42)

# 训练模型

clf.fit(X\_train\_split, y\_train\_split)

# 3. 预测并计算训练集准确率

y\_train\_pred = clf.predict(X\_train\_split)

train\_accuracy = accuracy\_score(y\_train\_split, y\_train\_pred)

# 4. 计算验证集的准确率

y\_val\_pred = clf.predict(X\_val\_split)

val\_accuracy = accuracy\_score(y\_val\_split, y\_val\_pred)

# 输出训练集和验证集准确率

print(f"训练集准确率: {train\_accuracy \* 100:.2f}%")

print(f"验证集准确率: {val\_accuracy \* 100:.2f}%")

# 5. 在B表上进行预测

y\_pred\_encoded = clf.predict(X\_pred\_scaled)

y\_pred = le.inverse\_transform(y\_pred\_encoded)

# 6. 结果输出

B['预测信誉等级'] = y\_pred

B.to\_excel('302信誉等级预测.xlsx', index=False)

print("已完成企业信誉等级预测，结果保存在 302信誉等级预测.xlsx")

# 7. 显示混淆矩阵和分类报告

conf\_matrix = confusion\_matrix(y\_val\_split, y\_val\_pred)

class\_report = classification\_report(y\_val\_split, y\_val\_pred, target\_names=le.classes\_)

print("混淆矩阵:")

print(conf\_matrix)

print("\n分类报告:")

print(class\_report)

# 8. 可视化混淆矩阵

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=le.classes\_, yticklabels=le.classes\_)

plt.title("混淆矩阵")

plt.ylabel('实际值')

plt.xlabel('预测值')

plt.show()