信用卡高风险客户识别

1 项目背景

某地区的信用卡产业始于 20 世纪 80 年代，1989 年开始向外资银行开放市场后，依托合理风险下的经营理念，特别是开创了信用卡销售外包模式，迅速占领了该地区的信用卡市场。发卡量从 1989 年的 50 多万张一跃到 1995 年的 500 多万张。

信用卡高速发展的背后是坏账风险的不断增大。据统计，2006 年，该地区有 900 多万人拥有信用卡，现金卡，交不起卡债的人已达 70 多万，造成银行呆账超过 1500 亿新台币。100 名有收入的人之中就有 6 人是卡奴，人均欠债 100 多万新台币。给整个地区的银行信用卡业务蒙上了一层阴影。

为了推进信用卡业务良性发展，减少坏账风险，该地区各大银行都进行了信用卡客户风险识别相关工作，建立了相应的客户风险识别模型。某银行因旧的风险识别模型随时间推移，不再适应业务发展需求，需要重新进行风险识别模型构建

2 项目目标

（1） 判断识别出哪些客户为高风险类客户，哪些客户为禁入类客户。

（2） 对不同客户类别进行特征分析，比较不同客户的风险。

（3） 评估该机构的信用卡业务风险，针对目前的情况提出风控建议。

3 项目步骤

3.1 工程前期准备

3.1.1 导入数据

（1） 介绍数据

目前，银行给出的数据的数据如表 3-1 所示。

表 3-1 信用卡信息数据说明表

变量名称	变量取值说明	示例
顾客编号		CDMS0000001
申请书来源	1.Take-One 邮寄件 2.现场办卡 3.电访 4.亲签亲访 5.亲访 6.亲签 7.本行 VIP、PB8.其他	1
瑕疵户	1.是 2.否	2
逾期	1.是 2.否	1
呆账	1.是 2.否	2
借款余额	1.是 2.否	1
退票	1.是 2.否	2
拒往记录	1.是 2.否	1
强制停卡记录	1.是 2.否	2
张数	1.1 张 2.2 张 3.3 张 4.4 张 5.大于 4 张	5
频率	1.天天用 2.经常用 3.偶而用 4.很少用 5.没有用	2
户籍	1.北部 2.中部 3.南部 4.东部	3
都市化程度	1.都会 2.都市 3.城镇	2
性别	1.女	1
年龄	1.此信用卡持有人之 15-19 岁 2.20-24 岁 3.25-29 岁 4.30-34 岁 5.35-39 岁 6.40-44 岁 7.45-49 岁 8.50-54 岁 9.55-59 岁	5
婚姻	1.未婚 2.已婚 3.其他	1
学历	1.小学及以下 2.国初中 3.高中职 4.专科 5.大学及以上	2
职业	1.国中及以下学生 2.高中、高职学生 3.夜间部高中、高职学生 4.专科学生 5.夜间部专科学生 5.夜间部专科学生 6.大学生 7.夜间部大学生 8.管理职 9.专门职 10.技术职 11.事务职 12.销售职 13.劳务职 14.服务职 15.农林渔牧自营 16.商工服务自营(员工 9 人以下)17.自由业自营 18.经营者(员工 10 人以上)19.家庭主妇(没有兼副业)20.家	3

	庭主妇(有兼副业)21.无职 22.其他	
个人月收入	1.无收入 2.10000 元以下 3.10001-20000 元 4.20001-30000 元 5.30001-40000 元 6.40001-50000 元 7.50001-60000 元 8.60001 元以上	4
个人月开销	1.10000 元以下 2.10001-20000 元 3.20001-30000 元 4.30001-40000 元 5.40001 元以上	5
住家	1.租赁 2.宿舍 3.本人所有 4.父母所有 5.配偶所有 6.其他	2
家庭月收入	1.20000 元以下 2.20001-40000 元 3.40001-60000 元 4.60001-80000 元 5.80001-100000 元 6.100001 元以上	3
月刷卡额	1.20000 元以下 2.20001-40000 元 3.40001-60000 元 4.60001-80000 元 5.80001-100000 元 6.100001-150000 元 7.150001-200000 元 8.200000 以上	4
宗教信仰	1.佛教 2.道教 3.基督教 4.天主教 5.一贯道 6.拜拜 7.其他	2
人口数	持卡人共同居住的人口数 1.一人 2.二人 3.三人 4.四人 5.五人 6.六人 7.七人 8.八人 9.九人以上	2
家庭经济	1.上 2.中上 3.中 4.中下 5.下	1
血型	1.A 型 2.B 型 3.AB 型 4.O 型	1
星座	1.牧羊座 2.金牛座 3.双子座 4.巨蟹座 5.狮子座 6.处女座 7.天秤座 8.天蝎座 9.射手座 10.魔羯座 11.双鱼座	1

（2） 上传数据到 Python 数据挖掘建模平台

在新增数据源上，选择本地上传数据，如图 1 所示。

图 1 本地上传数据源

在本地路径上选择文件，填写在平台新建的目标表名，如图 2 所示。

图 2 本地选择文件上传

根据文件的数据，可以修改文件的字段名和类型，如图 3 所示。

图 3 字段设置

上传成功，可以在平台的数据源上查看数据，单击数据源操作的查看按钮如图 4 所示，数据预览如图 5 所示。

图 4 单击预览数据按钮

credict_user_id	application_source	flaw_user	overdue	bad_debt	loan_balance	refund	refuse_record	mandatory_stop
CDMS0000001	5	2	2	2	2	2	2	2
CDMS0000002	3	2	2	2	2	2	2	2
CDMS0000003	2	2	2	2	2	2	2	2
CDMS0000004	3	2	2	2	2	2	2	2
CDMS0000005	6	2	2	2	2	2	2	2
CDMS0000006	5	2	2	2	2	2	2	2
CDMS0000007	7	2	2	2	2	2	2	2
CDMS0000008	6	2	2	2	2	2	2	2

图 5 数据预览

3.1.2 新建空白工程

右击我的工程，新建一个空白的工程，如图 6 所示。

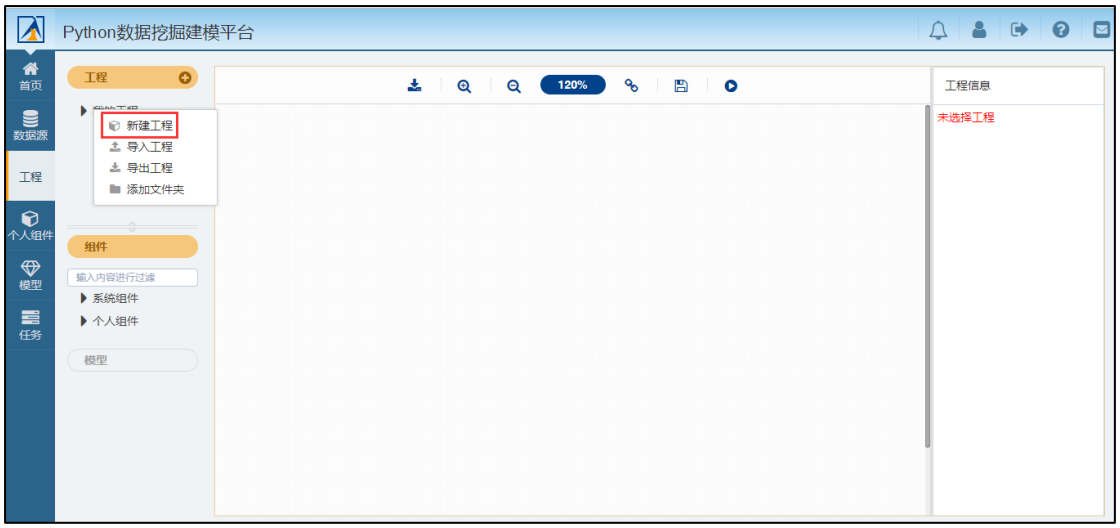

图 6 新建工程

填写工程的信息，包括工程名称和工程描述，如图 7 所示。

图 7 填写工程信息

3.2 数据预处理

读取 credict_card 数据，步骤如图 8 所示。

(1) 选择工程。

(2) 选择输入源组件。

(3) 拖入输入源组件。

(4) 填写数据表名。

(5) 单击更新按钮，更新出数据。

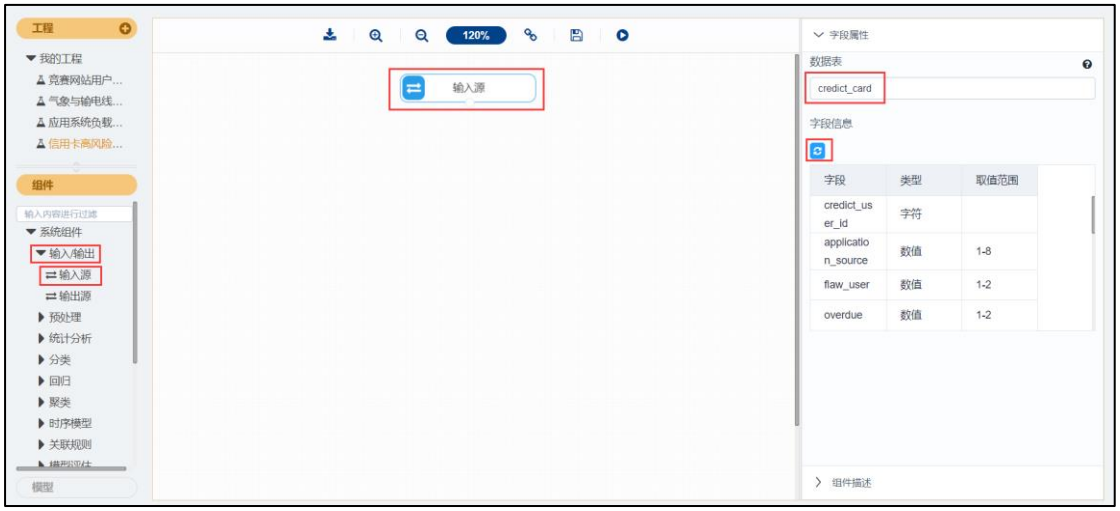

图 8 输入源组件

3.2.1 异常数据筛选

接下来进行异常数据筛选，步骤如图 9 所示。

(1) 找到预处理→Python 脚本组件。

(2) 拖入 Python 脚本组件，并将输入源和 Python 脚本组件连接。

(3) 选择字段属性，在脚本处填入数据变换代码，如表 3-2 所示。

(4) 对 Python 脚本组件右键，选择运行该节点。

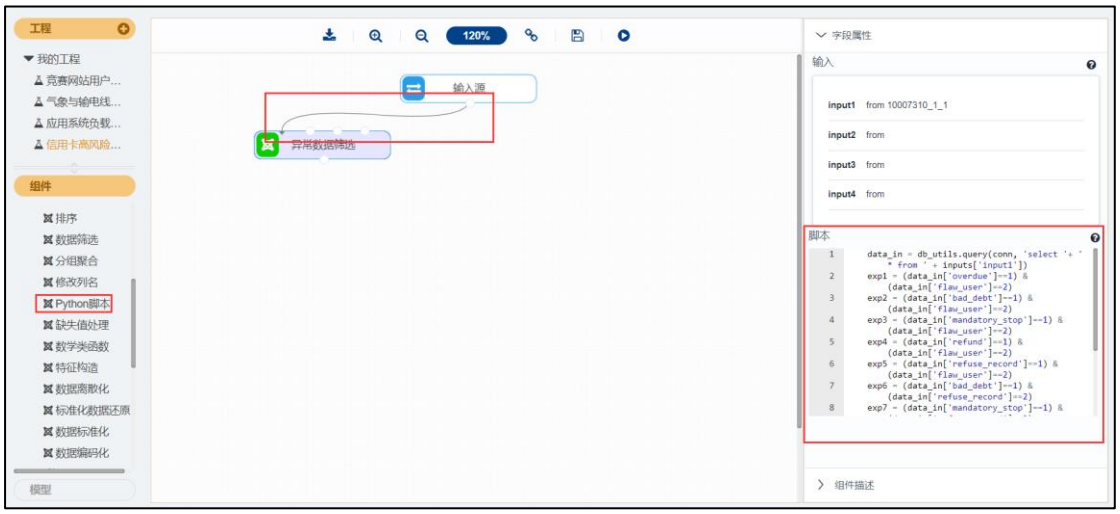

图 9 异常数据筛选组件

表 3-2 异常数据筛选代码

```
data_in = db_utils.query(conn, 'select '+ ' * from ' + inputs['input1'])
exp1 = (data_in['overdue']==1) & (data_in['flaw_user']==2)
exp2 = (data_in['bad_debt']==1) & (data_in['flaw_user']==2)
exp3 = (data_in['mandatory_stop']==1) & (data_in['flaw_user']==2)
exp4 = (data_in['refund']==1) & (data_in['flaw_user']==2)
exp5 = (data_in['refuse_record']==1) & (data_in['flaw_user']==2)
exp6 = (data_in['bad_debt']==1) & (data_in['refuse_record']==2)
exp7 = (data_in['mandatory_stop']==1) & (data_in['refuse_record']==2)
exp8 = (data_in['refund']==1) & (data_in['refuse_record']==2)
exp9 = (data_in['freq']==1) & (data_in['month_amount']>1)
data_out = data_in.loc[(exp1 | exp2 | exp3 | exp4 | exp5 | exp6 | exp7 | exp8 | exp9).apply(lambda x:not(x)),:]
data_out.reset_index(inplace=True)
return(data_out)
```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 10 所示。

预览数据

index	credict_user_id	application_source	flaw_user	overdue	bad_debt	loan_balance
0	CDMS0000001	5	2	2	2	2
1	CDMS0000002	3	2	2	2	2
2	CDMS0000003	2	2	2	2	2
3	CDMS0000004	3	2	2	2	2
4	CDMS0000005	6	2	2	2	2
5	CDMS0000006	5	2	2	2	2
6	CDMS0000007	7	2	2	2	2
7	CDMS0000008	6	2	2	2	2

共 61400 条 25 条/页 1 2 3 4 5 6 ··· 2456 前往 1 页

图 10 数据变换结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为异常数据筛选。

3.2.2 数据校正

接下来进行数据校正，步骤如图 11 所示。

(1) 找到预处理→Python 脚本组件。

(2) 拖入 Python 脚本组件，并将异常数据筛选和 Python 脚本组件连接。

(3) 选择字段属性，在脚本处填入数据变换代码，如表 3-3 所示。

(4) 对 Python 脚本组件右键，选择运行该节点。

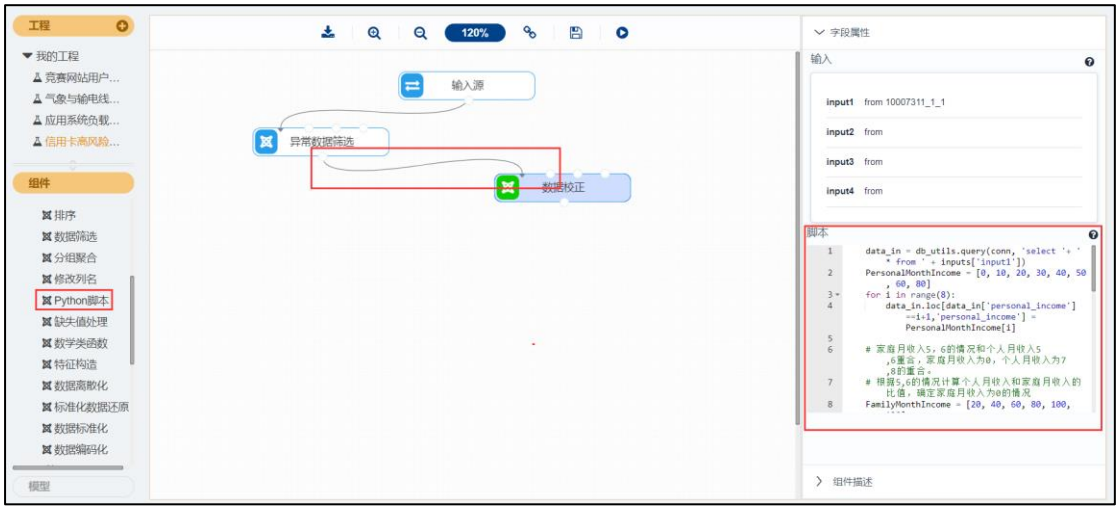

图 11 数据校正组件

表 3-3 数据校正代码

```
data_in = db_utils.query(conn, 'select '+ ' * from ' + inputs['input1'])
PersonalMonthIncome = [0, 10, 20, 30, 40, 50, 60, 80]
for i in range(8):
    data_in.loc[data_in['personal_income']==i+1,'personal_income'] = PersonalMonthIncome[i]
```

```
# 家庭月收入 5，6 的情况和个人月收入 5,6 重合，家庭月收入为 0，个人月收入为 7,8 的重合。
# 根据 5,6 的情况计算个人月收入和家庭月收入的比值，确定家庭月收入为 0 的情况
FamilyMonthIncome = [20, 40, 60, 80, 100, 120]

m = (data_in.loc[:,'family_income']==5)
data_in.loc[m,'family_income'] = FamilyMonthIncome[4]
ratio5 = data_in.loc[m, 'personal_income'] / data_in.loc[m, 'family_income']
m1 = data_in.loc[:,'family_income']==6
data_in.loc[m1,'family_income'] = FamilyMonthIncome[5]
ratio6 = data_in.loc[m1, 'personal_income'] / data_in.loc[m1, 'family_income']

# 家庭收入(千元)
FamilyMonthIncome = [20, 40, 60, 80, 100, 150]
data_in.loc[data_in['family_income'] == 0, 'family_income'] = 6
for i in range(6):
    m2 = data_in.loc[:, 'family_income'] == i+1
    data_in.loc[m2, 'family_income'] = FamilyMonthIncome[i]

# 月刷卡额(千元)
MonthCardPay = [20, 40, 60, 80, 100, 150, 200, 250]
for i in range(8):
    m = data_in.loc[:, 'month_amount'] == i+1
    data_in.loc[m, 'month_amount'] = MonthCardPay[i]

# 个人月开销（千元）
PersonalMonthOutcome = [10, 20, 30, 40, 60]
for i in range(5):
    m = data_in['personal_outcome'] == i+1
    data_in.loc[m, 'personal_outcome'] = PersonalMonthOutcome[i]
return(data_in)
```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 12 所示。

预览数据

index	credict_user_id	application_source	flaw_user	overdue	bad_debt	loan_balance
0	CDMS0000001	5	2	2	2	2
1	CDMS0000002	3	2	2	2	2
2	CDMS0000003	2	2	2	2	2
3	CDMS0000004	3	2	2	2	2
4	CDMS0000005	6	2	2	2	2
5	CDMS0000006	5	2	2	2	2
6	CDMS0000007	7	2	2	2	2
7	CDMS0000008	6	2	2	2	2

共 61400 条 25 条/页 1 2 3 4 5 6 ··· 2456 前往 1 页

图 12 数据校正结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为数据校正。

3.2.3 历史信用风险特征构建

接下来进行历史信用风险特征构建，步骤如图 13 所示。

(1) 找到预处理→Python 脚本组件。

(2) 拖入 Python 脚本组件，并将数据校正和 Python 脚本组件连接。

(3) 选择字段属性，在脚本处填入数据变换代码，如表 3-4 所示。

(4) 对 Python 脚本组件右键，选择运行该节点。

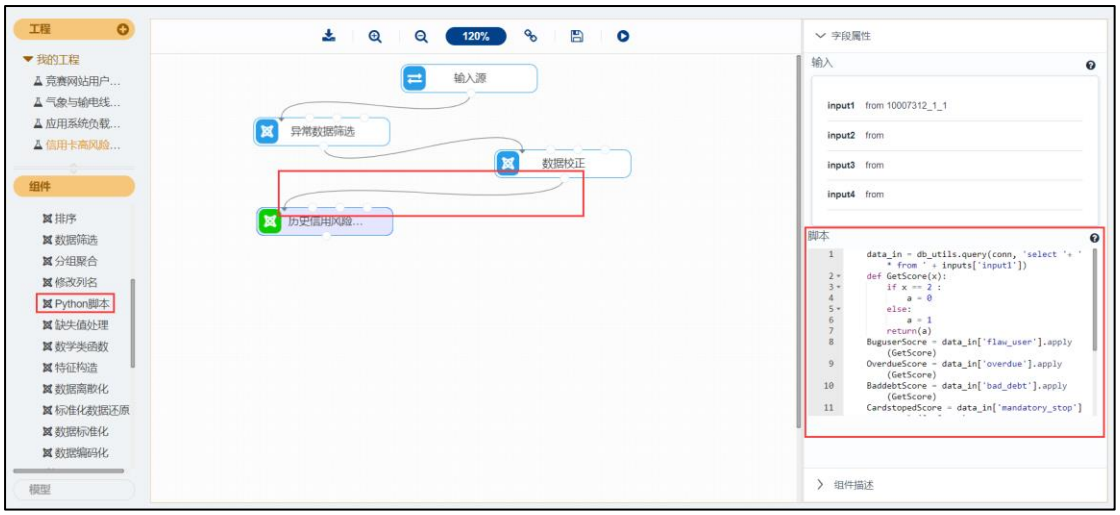

图 13 历史信用风险特征构建组件

表 3-4 历史信用风险特征构建代码

```
data_in = db_utils.query(conn, 'select '+ ' * from ' + inputs['input1'])
def GetScore(x):
    if x == 2 :
        a = 0
    else:
        a = 1
    return(a)
BuguserSocre = data_in['flaw_user'].apply(GetScore)
OverdueScore = data_in['overdue'].apply(GetScore)
BaddebtScore = data_in['bad_debt'].apply(GetScore)
CardstopedScore = data_in['mandatory_stop'].apply(GetScore)
BounceScore = data_in['refund'].apply(GetScore)
RefuseScore = data_in['refuse_record'].apply(GetScore)
data_in.loc[:,'history_credit_risk'] = BuguserSocre  + OverdueScore * 2 +  BaddebtScore * 3 +
```

```
CardstopedScore * 3 + BounceScore * 3 + RefuseScore * 3
    return(data_in)
```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 14 所示。

index	credict_user_id	application_source	flaw_user	overdue	bad_debt	loan_balance
0	CDMS0000001	5	2	2	2	2
1	CDMS0000002	3	2	2	2	2
2	CDMS0000003	2	2	2	2	2
3	CDMS0000004	3	2	2	2	2
4	CDMS0000005	6	2	2	2	2
5	CDMS0000006	5	2	2	2	2
6	CDMS0000007	7	2	2	2	2
7	CDMS0000008	6	2	2	2	2

图 14 历史信用风险特征构建结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为历史信用风险特征构建。

3.2.4 经济风险特征构建

接下来进行经济风险特征构建，步骤如图 15 所示。

(1) 找到预处理→Python 脚本组件。

(2) 拖入 Python 脚本组件，并将历史信用风险特征构建和 Python 脚本组件连接。

(3) 选择字段属性，在脚本处填入数据变换代码，如表 3-5 所示。

(4) 对 Python 脚本组件右键，选择运行该节点。

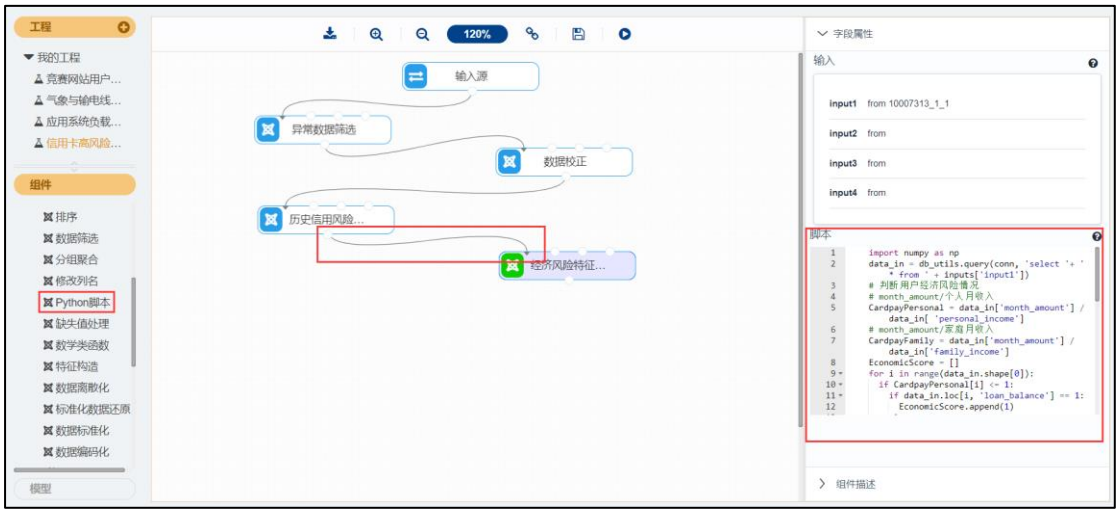

图 15 经济风险特征构建组件

表 3-5 经济风险特征构建代码

```
import numpy as np
data_in = db_utils.query(conn, 'select '+ ' * from ' + inputs['input1'])
#  判断用户经济风险情况
# month_amount/个人月收入
CardpayPersonal = data_in['month_amount'] / data_in[ 'personal_income']
# month_amount/家庭月收入
CardpayFamily = data_in['month_amount'] / data_in['family_income']
EconomicScore = []
for i in range(data_in.shape[0]):
    if CardpayPersonal[i] <= 1:
        if data_in.loc[i, 'loan_balance'] == 1:
            EconomicScore.append(1)
        else:
            EconomicScore.append(0)

    if CardpayPersonal[i] > 1:
        if CardpayFamily[i] <= 1:
            if data_in.loc[i, 'loan_balance'] == 1:
                EconomicScore.append(2)
            else:
                EconomicScore.append(1)

        if CardpayFamily[i] > 1:
            if data_in.loc[i, 'loan_balance'] == 1:
                EconomicScore.append(4)
            else:
                EconomicScore.append(2)

#  个人月开销/month_amount
OutcomeCardpay = data_in['personal_outcome'] / data_in['month_amount']
OutcomeCardpayScore = []
for i in range(data_in.shape[0]):
    if(OutcomeCardpay[i] <= 1):
        OutcomeCardpayScore.append(1)
    else:
        OutcomeCardpayScore.append(0)

data_in['economic_risk'] = np.array(EconomicScore) + np.array(OutcomeCardpayScore)
return(data_in)
```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 16 所示。

预览数据

index	credict_user_id	application_source	flaw_user	overdue	bad_debt	loan_balance
0	CDMS0000001	5	2	2	2	2
1	CDMS0000002	3	2	2	2	2
2	CDMS0000003	2	2	2	2	2
3	CDMS0000004	3	2	2	2	2
4	CDMS0000005	6	2	2	2	2
5	CDMS0000006	5	2	2	2	2
6	CDMS0000007	7	2	2	2	2
7	CDMS0000008	6	2	2	2	2

共 61400 条 25 条/页 1 2 3 4 5 6 ··· 2456 前往 1 页

图 16 经济风险特征构建结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为经济风险特征构建。

3.2.5 收入风险特征构建

接下来进行收入风险特征构建，步骤如图 17 所示。

(1) 找到预处理→Python 脚本组件。

(2) 拖入 Python 脚本组件，并将经济风险特征构建和 Python 组件连接。

(3) 选择字段属性，在脚本处填入数据变换代码，如表 3-6 所示。

(4) 对 Python 脚本组件右键，选择运行该节点。

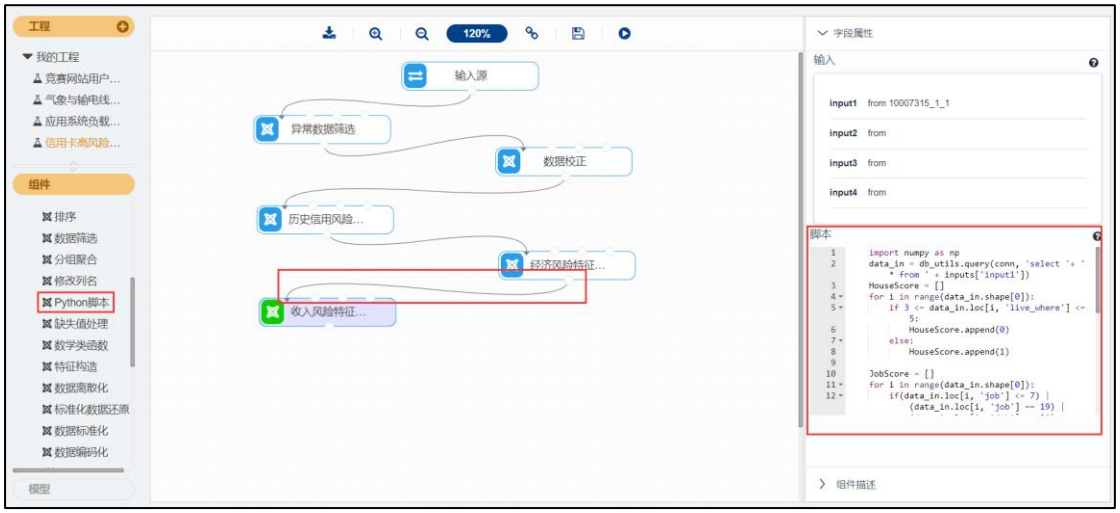

图 17 收入风险特征构建组件

表 3-6 收入风险特征构建代码

```
import numpy as np
data_in = db_utils.query(conn, 'select '+ ' * from ' + inputs['input1'])
HouseScore = []
for i in range(data_in.shape[0]):
```

```
        if 3 <= data_in.loc[i, 'live_where'] <= 5:
            HouseScore.append(0)
        else:
            HouseScore.append(1)

    JobScore = []
    for i in range(data_in.shape[0]):
        if(data_in.loc[i, 'job'] <= 7) | (data_in.loc[i, 'job'] == 19) | (data_in.loc[i, 'job'] == 21):
            JobScore.append(2)
        if(data_in.loc[i, 'job'] >= 8) & (data_in.loc[i, 'job'] <= 11):
            JobScore.append(1)
        if(data_in.loc[i, 'job'] <= 18) & (data_in.loc[i, 'job'] >= 12) | (data_in.loc[i, 'job'] == 20) |
(data_in.loc[i, 'job'] == 22):
            JobScore.append(0)

    AgeScore = []
    for i in range(data_in.shape[0]):
        if data_in.loc[i, 'age'] <= 2:
            AgeScore.append(1)
        else:
            AgeScore.append(0)

    data_in['income_risk'] = np.array(HouseScore) + np.array(JobScore) + np.array(AgeScore)


    return(data_in)
```

(5) 运行完成后，对 Python 脚本组件右键，选择查看数据，如图 18 所示。

index	credict_user_id	application_source	flaw_user	overdue	bad_debt	loan_balance
0	CDMS0000001	5	2	2	2	2
1	CDMS0000002	3	2	2	2	2
2	CDMS0000003	2	2	2	2	2
3	CDMS0000004	3	2	2	2	2
4	CDMS0000005	6	2	2	2	2
5	CDMS0000006	5	2	2	2	2
6	CDMS0000007	7	2	2	2	2
7	CDMS0000008	6	2	2	2	2

图 18 收入风险特征构建结果

(6) 运行完成后，对 Python 脚本组件右键，重命名为收入风险特征构建。

3.2.6 数据标准化

当属性间的量级相差较大时，数据标准化将数据统一映射到特定的区间，消除数据的量纲，步骤如图 19 所示。

(1) 找到预处理→数据标准化组件。

(2) 拖入数据标准化组件，将收入风险特征构建和数据标准化组件连接。

(3) 选择字段属性，单击更新数据，勾选 history_credit_risk、economic_risk、income_risk 字段。

(4) 对数据标准化组件右键，选择运行该节点。

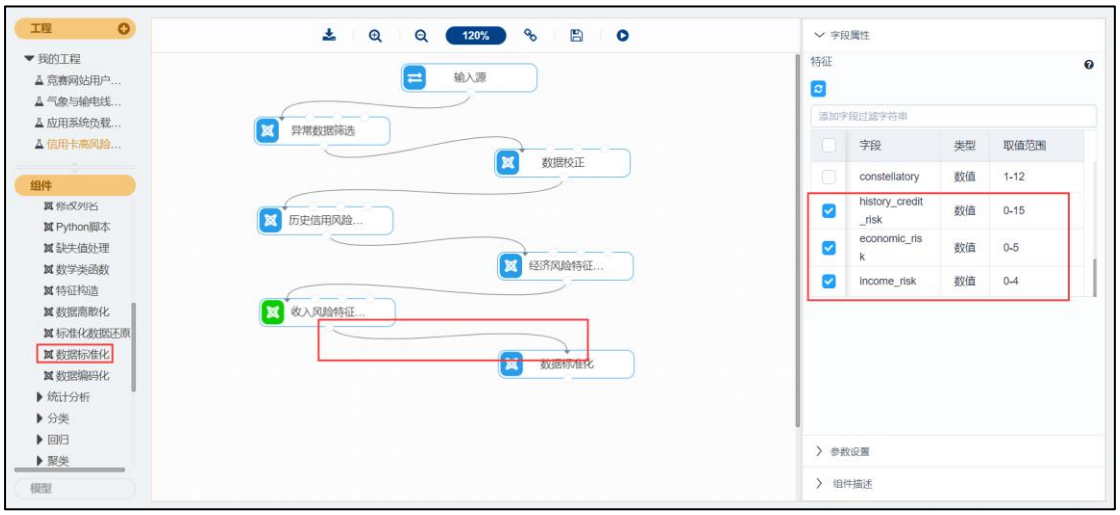

图 19 数据标准化组件

(5) 运行完成后，对数据标准化组件右键，选择查看数据，数据标准化的输出表结果如图 20 所示。

预览数据

history_credit_risk	economic_risk	income_risk
-0.2255617128526497	-0.4332165071939898	-0.2677378801586858
-0.2255617128526497	0.8977563995716324	1.5885297171474695
-0.2255617128526497	0.8977563995716324	1.5885297171474695
-0.2255617128526497	0.8977563995716324	-0.2677378801586858
-0.2255617128526497	0.8977563995716324	0.6603959184943918
-0.2255617128526497	0.8977563995716324	-0.2677378801586858
-0.2255617128526497	0.8977563995716324	-0.2677378801586858
-0.2255617128526497	0.8977563995716324	1.5885297171474695

共 61400 条 25 条/页 1 2 3 4 5 6 ··· 2456 前往 1 页

图 20 数据标准化结果

3.3 模型构建

3.3.1 K-Means 聚类算法

选择 K-Means 聚类算法模型，步骤如图 21、图 22 所示。

(1) 找到聚类→K-Means 组件。

(2) 拖入 K-Means 组件，将数据标准化和 K-Means 组件连接。

(3) 选择字段属性，单击更新数据，勾选全部字段。

(4) 选择参数设置，设置聚类数（n_clusters）的值为 5，其他的参数都设置为默认值。

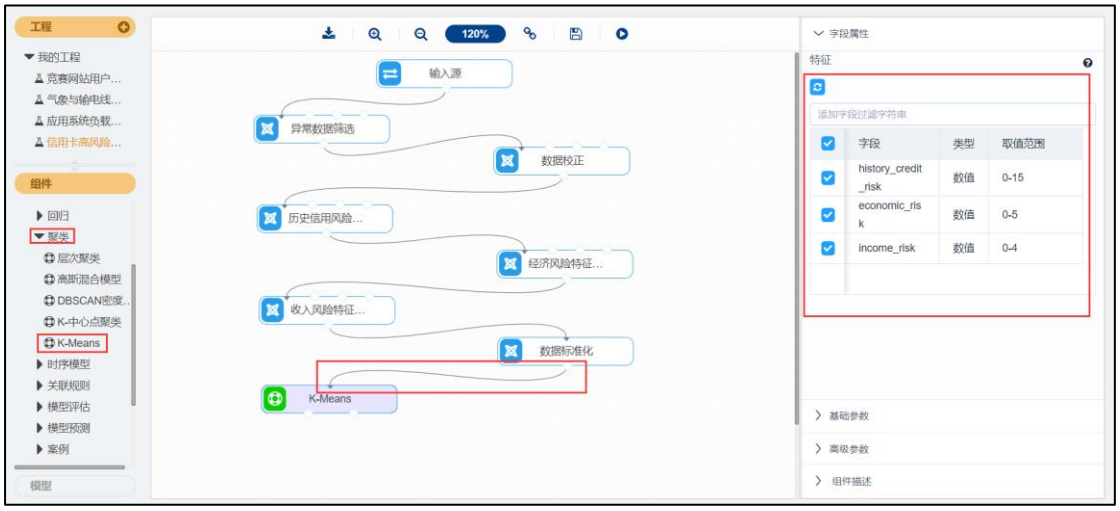

图 21 K-Means 聚类组件_字段属性

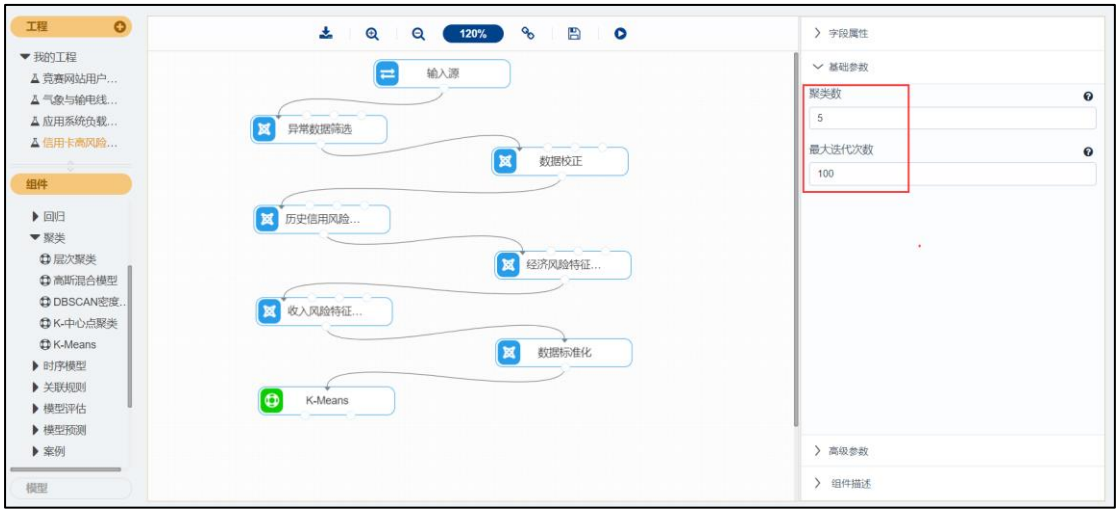

图 22 K-Means 组件_参数设置

(5) 运行完成后，对 K-Means 组件右键，选择查看数据，K-Means 的输出表结果如图 23 所示。选择查看报告，K-Means 的报告如图 24 所示。

预览数据

history_credit_risk	economic_risk	income_risk	cluster_id
-0.22556171285265	-0.43321650719399	-0.267737880158686	3
-0.22556171285265	0.897756399571632	1.58852971714747	2
-0.22556171285265	0.897756399571632	1.58852971714747	2
-0.22556171285265	0.897756399571632	-0.267737880158686	2
-0.22556171285265	0.897756399571632	0.660395918494392	2
-0.22556171285265	0.897756399571632	-0.267737880158686	2
-0.22556171285265	0.897756399571632	-0.267737880158686	2
-0.22556171285265	0.897756399571632	1.58852971714747	2

共 61400 条 25 条/页 1 2 3 4 5 6 ··· 2456 前往 1 页

图 23 K-Means 聚类算法的结果

图 24 K-Means 聚类算法的报告