# 数据集成作业三说明文档

# 成员分工

| 姓名 | 学号 | 任务 |
| --- | --- | --- |
| 黄韬 | 201250111 | 数据盘点、数据预处理、特征工程 |
| 李佳骏 | 201250113 | 字段选择 |
| 张笑恺 | 201250118 | 特征工程 |
| 邱兴驰 | 201250112 | 机器学习、模型评估 |
| 刘亚嘉 | 201250115 | 机器学习、模型应用 |

# 重要的中间数据

保存在 作业三\data\ 中，credit、star 的中间数据分开存储。

# 结果

保存在 作业三\data\{credit or star}\ 中，文件名为{credit or star}_test_predict.csv

# 数据收集

## 字段选择

## 一、评估信用等级

1. 根据查询当前银行常用的信用评估等级，我们选取了以下字段作预测，将根据字段对个人信用的正负加成进行分析。

**贷记卡开户明细**

| | |
|---|---|
| uid | 证件号码 |
| cred_limit | 信用额度 |
| over_draft | 普通额度透支 |
| dlay_amt | 逾期金额 |
| Five_class | 五级分类 |
| Bank acct_bal | 还款账号余额 |

通过当前信用额度可以直接反应用户信用等级。

通过五级分类可以评估当前用户的信用等级。

普通额度透着和还款账号余额结合可以评估账号的还款能力。

逾期金额越大反应用户信用低的可能性越大。

**贷记卡分期付款明细**

| | |
|---|---|
| UID | 证件号码 |
| rem_ppl | 剩余未还本金 |
| Rem_fee | 剩余未还费用 |

**合同明细**

| | |
|---|---|
| uid | 证件号码 |
| fdlay_bal | 逾期余额 |
| fdull_bal | 呆滞余额 |
| fowed_int_in | 表内欠息金额 |
| fowed_int_out | 表外欠息余额 |

| | |
|---|---|
| fine_pr_int | 本金罚息 |
| fine_intr_int | 利息罚息 |
| dlay_days | 逾期天数 |
| five_class | 五级分类 |
| class_date | 最新风险分类时间 |
| is_bad | 不良记录标志 |
| due_intr_days | 欠息天数 |

逾期余额、呆滞金额、欠息、罚息、逾期天数、欠息天数等都会对信用进行负加成。

通过五级分类和最新风险分类时间可以评估当前用户的信用等级。

如果存在不良记录标志，那么应该会直接影响到用户信用等级。

**贷记卡开户明细**

| | |
|---|---|
| Uid | 证件号码 |
| cred_limit | 信用额度 |
| over_draft | 普通额度透支 |
| dlay_amt | 逾期金额 |
| five_class | 五级分类 |
| bankacct_bal | 还款账号余额 |

通过当前信用额度可以直接反应用户信用等级。

通过五级分类可以评估当前用户的信用等级。

普通额度透着和还款账号余额结合可以评估账号的还款能力。

逾期金额越大反应用户信用低的可能性越大。

**贷记卡分期付款明细**

| | |
|---|---|
| Uid | 证件号码 |
| rem_ppl | 剩余未还本金 |

| | |
|---|---|
| rem_fee | 剩余未还费用 |

**借据明细**

| | |
|---|---|
| uid | 证件号码 |
| dlay_amt | 逾期金额 |
| dull_amt | 呆滞金额 |
| bad_debt_amt | 呆帐金额 |
| owed_int_in | 表内欠息金额 |
| owed_int_out | 表外欠息金额 |
| fine_pr_int | 本金罚息 |
| fine_intr_int | 利息罚息 |
| dlay_days | 逾期天数 |
| due_intr_days | 欠息天数 |
| pay_freq | 还款频率 |
| fvouch_type | 主要担保方式 |

同合同明细。

**存款账号信息**

| | |
|---|---|
| uid | 证件号码 |
| frz_sts | 冻结状态 |

如果用户账户已经被冻结，那么信用评分低的概率更高。

**基本信息**

| | |
|---|---|
| uid | 证件号码 |
| sex | 性别 |
| birthday | 出生日期 |

| | |
|---|---|
| marrige | 婚姻状况 |
| education | 教育程度 |
| career | 职业 |
| Prof_titl | 职称 |
| Is_employee | 员工标志 |
| is_shareholder | 是否股东 |
| is_black | 是否黑名单 |
| is_contract | 是否关联人 |
| | |

年龄位于25-45之间为最佳，因为这个年龄段的人正是工作上升期有赚钱能力。

已婚并有子女的用户往往因为需要供养家庭而需要保证良好信用，并且在银行中更愿意进行交易。

学历越高，知识水平越高，信用越好的可能性越大

工作越稳定，还款能力越强，信用往往越好。

是股东、或者职位越高，收入往往越高，与银行交易的频率可能越高，信用越有可能良好。

如果用户基本信息中表示该用户在黑名单中，那么信用评分低的概率更高。

**贷款账号信息**

| | |
|---|---|
| uid | 证件号码 |
| five_class | 五级分类 |
| overdue_class | 逾期细分 |
| overdue_flag | 逾期标志 |
| owed_int_flag | 欠息标志 |
| defect_type | 贷款瑕疵类型 |
| owed_int_in | 表内欠息金额 |
| owed_int_out | 表外欠息金额 |
| delay_bal | 逾期金额 |

通过五级分类可以评估当前用户的信用等级。

普通额度透着和还款账号余额结合可以评估账号的还款能力。

逾期金额越多、欠息金额越多反应用户信用低的可能性越大。

**贷款账户汇总**

| uid | 证件号码 |
| --- | --- |
| all_ball | 总余额 |
| Bad_ball | 不良余额 |
| Due_intr | 欠息总额 |
| Delay_bal | 逾期总额 |

总余额中不良余额、欠息余额、逾期余额占比越高信用评级可能越差。

2. 筛选合并信用等级相关字段

```sql
select distinct credit.uid as uid, credit_level,
djk.cred_limit, djk.over_draft, djk.dlay_amt, djk.five_class, djk.bank acct_bal,
djkfq.rem_ppl, djkfq.rem_fee,
contract.dlay_bal, contract.dull_bal, contract.owed_int_in, contract.owed_int_out, contract.fine_pr_int,contract.fine_intr_int, contract.dlay_days, contract.five_class, contract.class_date, contract.is_bad, contract.due_intr_days,
duebill.dlay_amt,duebill.dull_amt, duebill.bad_debt_amt,duebill.owed_int_in, duebill.owed_int_out, duebill.fine_pr_int, duebill.fine_intr_int, duebill.dlay_days, duebill.due_intr_days, duebill.vouch_type,
base.sex,base.birthday,base.marrige , base.education, base.reg_add, base.career, base.prof_titl, base.is_employee, base.is_shareholder, base.is_contact, base.is_black,
liabacct.five_class, liabacct.overdue_class, liabacct.overdue_flag, liabacct.owed_int_flag, liabacct.defect_type, liabacct.owed_int_in, liabacct.owed_int_out, liabacct.delay_bal,
liab.bad_bal, liab.due_intr, liab.delay_bal
from pri_credit_info as credit
left join dm_v_as_djk_info as djk on credit.uid = djk.uid
left join dm_v_as_djkfq_info as djkfq on credit.uid = djkfq.uid
left join dm_v_tr_contract_mx as contract on credit.uid = contract.uid
left join dm_v_tr_duebill_mx as duebill on credit.uid = duebill.uid
left join pri_cust_base_info as base on credit.uid = base.uid
left join pri_cust_liab_acct_info as liabacct on credit.uid = liabacct.uid
left join pri_cust_liab_info as liab on credit.uid = liab.uid
where credit.credit_level <> '-1'
order by credit.uid;
```

## 二、评估客户星级

中国工商银行官方网站回答：为了为您提供更好的服务，我行可以根据您在我行的业务情况进行星级评估。如果达到一定的星点值，则可以评估为相应的等级。星级评估是基于您在我行近半年的综合业务量。也就是说，您在我行办理的存款、贷款、投资理财(包括基金、理财产品、国债、保险、外汇、贵金属等。)、信用卡消费、汇款、异地存取款等。，都是按照一定的转换规则累计的。通常，你处理的业务越多，积累的星值就越多，你的星级就越高。根据相应的星级，我会为您提供不同的优惠服务。星级越高，一些服务费的折扣就越大。当然，三星级及以下的星级没有折扣。如果您处理相同的业务，如果此费用与上次费用不同，可能是由于星级调整带来的折扣程度不同。

1. 根据银行回答、从业人士分析、并结合资料，我们选择了以下字段用来预测客户星级。

**借据明细**

| | |
|---|---|
| uid | 证件号码 |
| bal | 余额 |
| norm_bal | 正常余额 |
| Pay_freq | 还款频率 |
| Vouch_type | 主要担保方式 |

还款频率越高，正常余额越多，担保方式越可靠用户星级可能越高。

**存款账号信息**

| | |
|---|---|
| uid | 证件号码 |
| bal | 余额 |
| avg_mth | 月日均 |
| Avg_qur | 季度日均 |
| Avg_year | 年日均 |
| is_secu_card | 是否社保卡 |
| acct_sts | 账户状态 |
| frz_sts | 冻结状态 |
| stp_sts | 止付状态 |

余额越多，三种日均较高，说明与银行的交易越多，在银行星级评级中更有可能被评为星级用户。账户状态同样可以影响用户星级的评估。

**存款汇总信息**

| | |
|---|---|
| uid | 证件号码 |
| all_bal | 总余额 |
| Avg_mth | 月日均 |
| Avg_qur | 季度日均 |

| | |
|---|---|
| avgyear | 年日均 |
| Sa_bal | 活期余额 |
| Td_bal | 定期余额 |
| Fin_bal | 理财余额 |
| Sa_crd_bal | 卡活期余额 |
| Td_crd_bal | 卡内定期 |
| Sa_td_bal | 定活两便 |
| Etc_bal | 通知存款 |
| Td_3m_bal | 定期3个月 |
| Td_6m_bal | 定期6个月 |
| Td_1y_bal | 定期1年 |
| Td_2y_bal | 定期2年 |
| Td_3y_bal | 定期3年 |
| Td_5y_bal | 定期5年 |
| oth_td_bal | 定期其他余额 |
| cd_bal | 大额存单余额 |

在银行中办理的存款、贷款、投资理财(包括基金、理财产品、国债、保险、外汇、贵金属等。)、信用卡消费、汇款、异地存取款等交易越多，星级越高。

**基本信息**

| | |
|---|---|
| uid | 证件号码 |
| sex | 性别 |
| birthday | 出生日期 |
| marrige | 婚姻状况 |
| education | 教育程度 |
| Reg_add | 户籍地址 |

| | |
|---|---|
| career | 职业 |
| Prof_titl | 职称 |
| Is_employee | 员工标志 |
| is_shareholder | 是否股东 |
| is_black | 是否黑名单 |
| is_contract | 是否关联人 |

年龄位于25-45之间为最佳，因为这个年龄段的人正是工作上升期有赚钱能力。

已婚并有子女的用户往往因为需要供养家庭而需要保证良好信用，并且在银行中更愿意进行交易，越有可能被评为星级用户。

学历越高，知识水平越高，与银行交易的信用越好，星级越高的可能性越大

工作越稳定，还款能力越强，交易频率越高，星级往往越高。

是股东、或者职位越高，收入往往越高，与银行交易的频率可能越高，星级越高。

如果用户基本信息中表示该用户在黑名单中，那么星级低的概率更高。

2. 筛选合并星级相关字段

SQL | 复制代码

```sql
    select distinct star.uid as uid, star_level,
    asset.all_bal, asset.avg_mth, asset.avg_qur, asset.avg_year, asset.sa_bal, asset.td_bal , asset.fin_bal, asset.sa_crd_bal, asset.td_crd_bal, asset.sa_td_bal, asset.ntc_bal, asset.td_3m_bal, asset.td_6m_bal, asset.td_1y_bal, asset.td_2y_bal, asset.td_3y_bal, asset.td_5y_bal, asset.oth_td_bal, asset.cd_bal,
    base.sex,base.birthday,base.marrige , base.education, base.reg_add, base.career, base.prof_titl, base.is_employee, base.is_shareholder, base.is_contact, base.is_black,
    acct.bal,acct.avg_mth, acct.avg_qur, acct.avg_year,acct.is_secu_card,
    acct.acct_sts,acct.frz_sts,acct.stp_sts
    from pri_star_info as star
```

## 三、相关资料

1. 信用评级：A credit level is **the measure of a business's creditworthiness, which is made up**

**from a number of factors to understand the its level of financial risk**. The score ranges from 0 to 100, with 0 representing a high risk and 100 representing a low risk.

2. 信用评级：https://wiki.mbalib.com/wiki/信用评级
3. 信用评级标准：

```
Payment History (35%)
Accounts that have a positive payment history are preferred by banks. The reports show any late or missed payments, as well as bankruptcy or collection actions, according to each line of credit.
Amounts Owed (30%)
Having a lot of debt does not necessarily indicate low credit ratings. Rather, FICO takes into account the proportion of outstanding debt to total credit volume.

For example, someone who owes 10,000 but has all of their lines of credit fully extended and all loans at their maximum would have a lower score than someone who owes the same $10,000 but has only used half of their lines of credit and loan balances.

Length of Credit History (15%)
Lenders will also want to know how long you have been using credit. Longer credit history is generally better than a shorter one because it shows you have a track record of responsible borrowing.

Credit Mix (10%)
This refers to the different types of credit you have, such as revolving credit (e.g., credit cards) and installment loans (e.g., auto loans).

Having a mix of different types of credit can improve your FICO score. It is not required to have all types of credit but having a variety shows that you can responsibly manage different obligations that come with each type.

New Credit (10%)
Opening new lines of credit can be seen as a sign of financial instability. Lenders will want to know if you have been opening new lines of credit frequently.
```

4. 贷款分类制度：贷款五级分类制度是**根据内在风险程度将商业贷款划分为正常、关注、次级、可疑、损失五类**。 这种分类方法是银行主要依据借款人的还款能力，即最终偿还贷款本金和利息的实际能力，确定贷款遭受损失的风险程度，其中后三类称为不良贷款。 此前的贷款四级分类制度是将贷款划分为正常、逾期、呆滞、呆账四类。

## 数据盘点

首先将 `train_set.describe()` 存入文件中，随后调用 `visualize.py` 中的方法对数据盘点进行可视化。

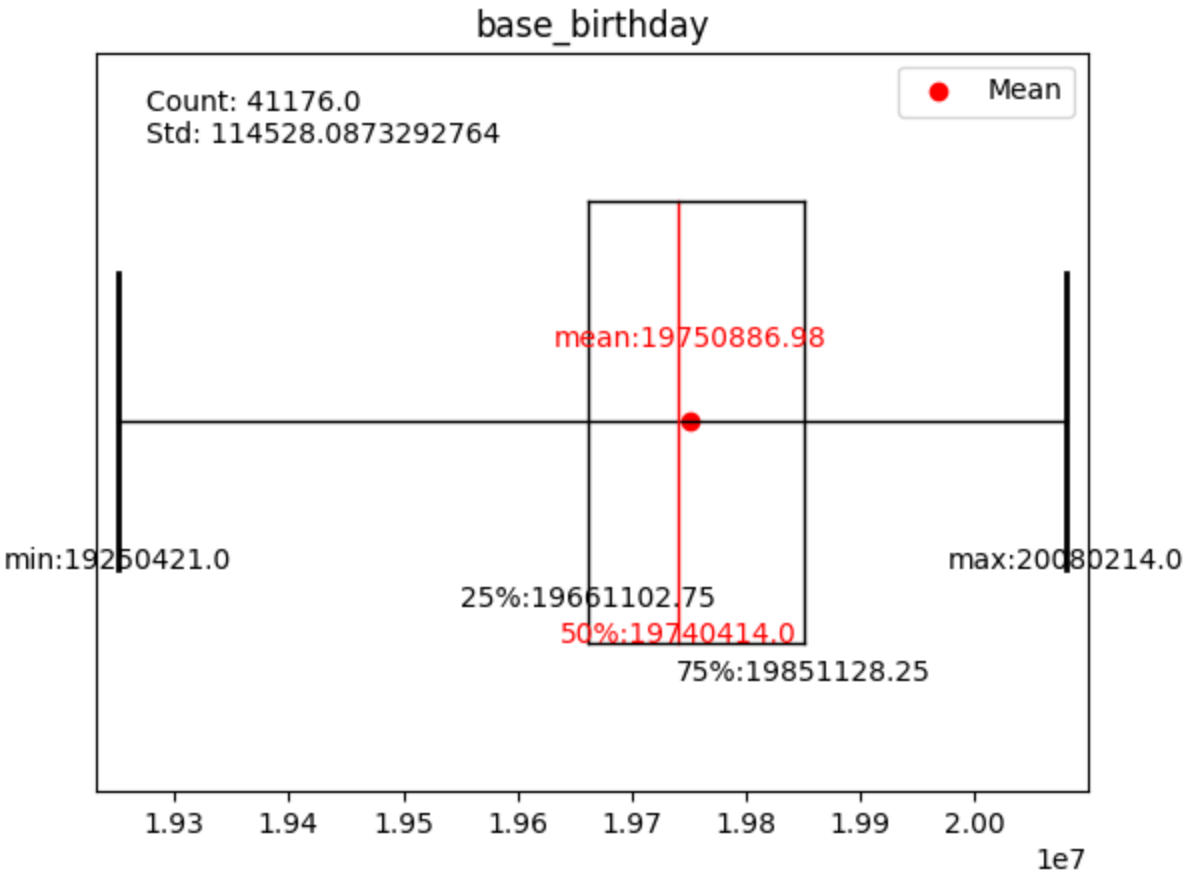
对于数据盘点的可视化，我们选择绘制箱线图的形式，通过对数据盘点的每一个属性绘制对应的箱线图，从而在图上直观地展示数据集中该属性的均值、最小值、下四分位数、中位数、上四分位数、最大值。

对于计数与标准差这类无法体现在箱线图中的，将其标准在图的右上角。

具体的处理，我们使用了pandas库与matplotlib库

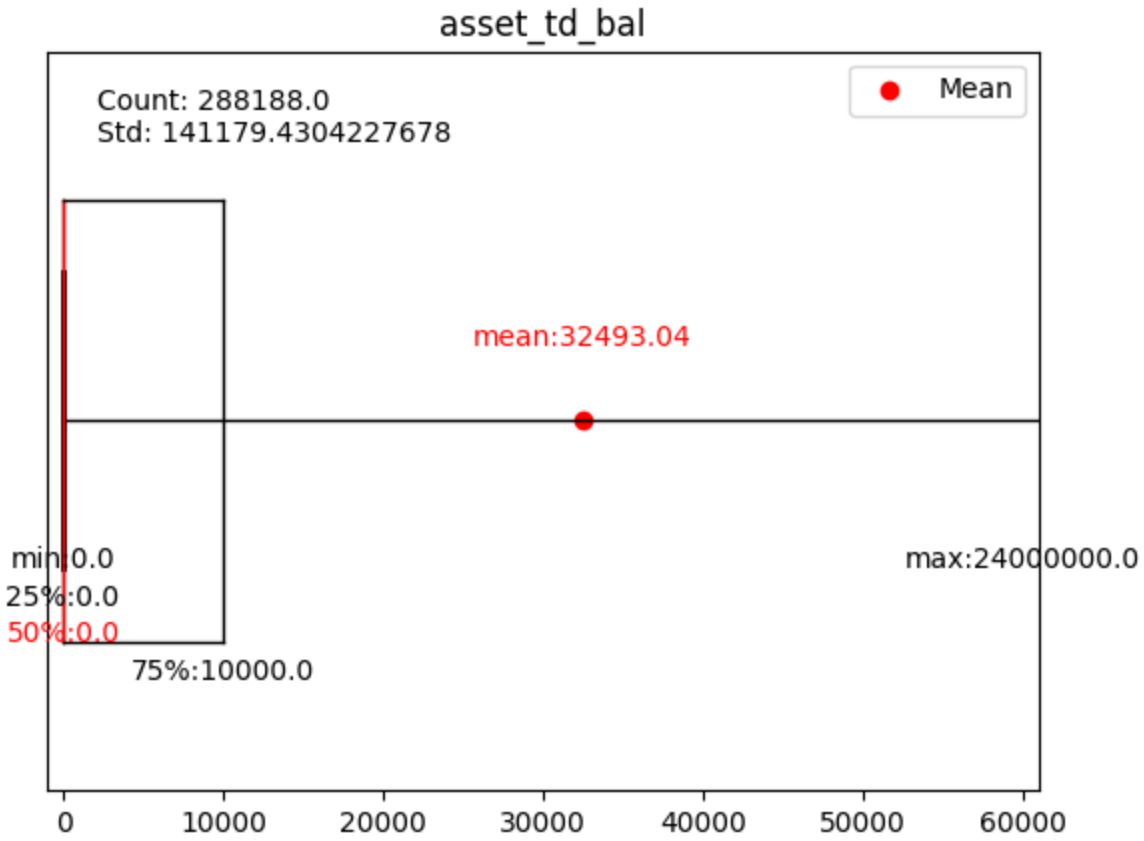
```python
import pandas as pd
import matplotlib.pyplot as plt
```

绘制出的图如下所示：




图片存储位置为 \data\{credit or star}\{credit or star}_describe_visualization\

当出现极端数据分布时，如下图：




限制箱线图x轴的上下限，超过上下限的数据不在图中画出，仅用文字标识其值

visualize Python 复制代码

```python
# iqr为四分位距（上四分位数 - 下四分位数）
x_min = max(min_value, min(mean, q1) - 5 * iqr)
x_max = min(max_value, max(mean, q3) + 5 * iqr)
# 避免iqr为0时出错。
if iqr > 0:
    x_min = max(x_min, lower_bound)
    x_max = min(x_max, upper_bound)
```

设置x轴范围时略微超过x轴上下限，避免最大最小值绘制于图的边缘，难以看清

visualize Python 复制代码

```python
ax.set_xlim(x_min - iqr * 0.1 - 1, x_max + iqr * 0.1 + 1)  # 设置X轴范围
```

# 数据预处理

## 缺失值处理

1. 去除掉缺失值大于0.7的列，每一列的缺失值储存在 {credit or star}_missing.csv 中

```
train_set.isnull().mean().to_csv(form_csv_path('train_missing'))
train_set = train_set.loc[:, train_set.isnull().mean() < 0.7]
```

2. 使用 KNN 插补填充数值型缺失值，用众数填充类别型缺失值

main Python 复制代码

```
train_set = pp.handle_missing_knn(train_set)
```

KNNImputer通过欧几里德距离矩阵寻找最近邻样本，使用最近邻样本的对应位置的非空数值的均值填补缺失的数值。此处K取5。

preprocess Python 复制代码

```
def handle_missing_knn(data):
    # 选择数值型列
    numeric_columns = get_numeric_columns(data)
    # 使用KNN插补填充数值型缺失值
    imputer = KNNImputer(n_neighbors=5)
    data[numeric_columns] = imputer.fit_transform(data[numeric_columns])
    # 类别型缺失值仍用众数填充
    data = handle_missing_object(data)

    print('缺失值处理完毕！')
    return data

def handle_missing_object(data):
    # 选择类别型列
    categorical_columns = get_categorical_columns(data)
    # 使用众数填充类别型列的缺失值
    data.loc[:, categorical_columns] = data.loc[:, categorical_columns].fillna(data[categorical_columns].mode().iloc[0])
    return data
```

以下是经过KNN插补处理的结果，可以看出有一定的可靠性

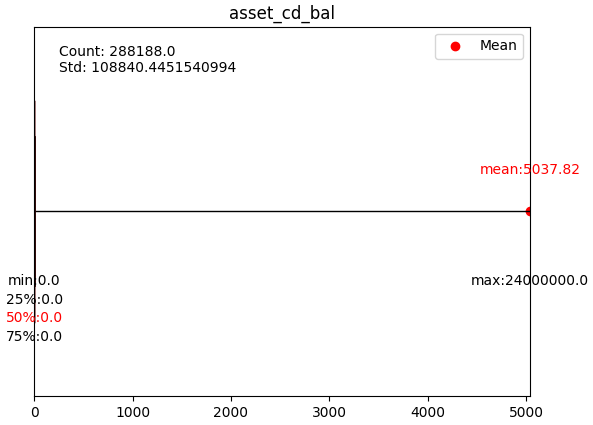
（左为原始数据；右为删除原始数据后经过KNN插补得到的数据）

| asset_all_bal | asset_all_bal |
|---|---|
| 0.01 | 0.01 |
| 8909.38 | 8266.42 |
| 1.43 | 0.21 |
| 13428.43 | 13060.608 |
| 2518.77 | 2724.262 |
| 0 | 0.048 |
| 30643.5 | 30437.626 |
| 0 | 58.886 |
| 2437.29 | 2335.002 |
| 37.46 | 41.806 |

## 异常值处理

通过箱线图寻找离群值并对其进行异常值处理。

1. 由于数据集中存在分布极端的数据，至少75%的数据均为0，如下图：




   对于这样的数据，不能简单地通过离群值来判断异常值，这会导致绝大多数有效的、非0的数据被视为异常值处理。因此异常值处理时忽略此处为0的数据，只针对非0的有效数据进行离群值的判断。

```
train_set.replace(0, np.nan, inplace=True)
```

   将数据集中为0的数据替换为nan，这样能使后续箱线图的处理中忽略掉这些数据，返回数据集时再将nan替换回0即可。

2. 对于数据集中为数值型的每一列，绘制箱线图，计算其上四分位数与下四分位数，相减得到四分位距

```python
# 绘制箱线图
plt.figure()
train_set.boxplot(column=column)
plt.title(column)
# 计算异常值阈值
q1 = train_set[column].quantile(0.25)  # 第一四分位数
q3 = train_set[column].quantile(0.75)  # 第三四分位数
iqr = q3 - q1  # 四分位距（IQR）
```

3. 设置上下边界，将超出边界值的数据置为边界值

```python
lower_bound = q1 - 1.5 * iqr  # 下边界
upper_bound = q3 + 1.5 * iqr  # 上边界
# 处理异常值，将超出边界值的数值设为边界值
train_set[column] = np.where(train_set[column].isnull(), np.nan,
                             train_set[column].clip(lower=lower_bound,
upper=upper_bound))
```

4. 最后将第一步中置为nan的数据替换回0即完成异常值的处理。

```python
train_set.replace(np.nan, 0, inplace=True)
```

## 数据转换

数据转换使用基本的Label Encoding方法，将转换后的结果存入 {credit or star}_train_trans.csv 中

preprocess　Python　复制代码

```python
def encode(data, catelist):
    le = LabelEncoder()
    data[catelist] = data[catelist].apply(le.fit_transform)
    print('数据转换完成！')
    return data
```

## 数据标准化

数据标准化使用基本的StandardScaler方法，将标准化后的结果存入 {credit or star}_train_std.csv 中

preprocess Python 复制代码

```python
def standardize(data, colist):
    scaler = StandardScaler()
    data[colist] = scaler.fit_transform(data[colist])
    print('数据标准化完成！')
    return data
```

# 特征工程

## 属性间相关性

1. 概念：相关系数：又叫简单相关系数或线性相关系数，一般用字母r表示，用来度量两个变量间的线性关系。
2. 计算方法：

$$r = \frac{\sum(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum(x_i - \bar{X})^2 (y_i - \bar{Y})^2}}$$

3. 剔除原理：如果相关系数R>0.9时就可能存在较强相关性
4. 代码实现：
    a. 设A、B、C为三个属性，可能存在：

    corr(A, B) > threshold

    corr(B, C) > threshold

    corr(A, C) < threshold

    的情况，此时应该去除B属性，因此不能进行“遇到相关系数大于阈值的两个属性则去除任一个”的简单处理。选择下述数理方法。

    b. 定义一个方法，从相关系数矩阵中找出与其他属性相关系数超出阈值最多的一个属性。

preprocess Python 复制代码

```python
def my_find(corr_mat, colist, limit):
    n = len(colist)
    target = '-1'
    max_count = 0
    for i in range(n):
        row = colist[i]
        count = 0
        for j in range(i + 1, n):
            column = colist[j]
            if abs(corr_mat.loc[row, column]) > limit:
                count += 1
        if count > max_count:
            target = row
            max_count = count

    return target
```

c. 在数据集中反复调用my_find，删除返回的目标，直到没有要删除的目标为止。

```python
while drop_target != '-1':
    remaining_colist.remove(drop_target)
    drop_target = my_find(corr_mat, remaining_colist, threshold)
```

d. 此时即去除了所有不合规的属性，返回新的属性列表 remaining_colist 即可。

## 多重共线性（使用方差膨胀系数进行检验剔除）

1. 概念：共线性问题指的是输入的自变量之间存在较高的线性相关度。共线性问题会导致回归模型的稳定性和准确性大大降低，另外，过多无关的维度计算也很浪费时间。
2. 计算方法：

$$VIF_i = \frac{1}{1 - R_i^2} \quad i = 1, 2, \cdots, k$$

3. 剔除原理：VIF是容忍度的倒数，值越大则共线性问题越明显，通常以10作为判断边界。当VIF<10,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。
4. 代码实现：

a. 使用了statsmodels库

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

b. 在数据集上添加一列截距项。

```
# 截距项
mat['c'] = 1
```

原因：在使用方差膨胀系数进行剔除时，如果只考虑自变量的系数，没有截距项，那么剔除一个自变量后，剩下的模型可能会出现一个不合理的截距项。这是因为截距项代表了在所有自变量为零时的响应值，而如果移除了一个自变量，模型的整体结构发生了改变，截距项可能需要进行调整。因此，在使用方差膨胀系数剔除矩阵时，为了保持模型的一致性和正确性，通常会将截距项纳入考虑范围。这样，在剔除一个自变量后，通过重新拟合模型，可以得到相应的调整后的截距项，从而保持模型的准确性。

c. 计算VIF_list，将结果与属性名合并形成对应的dataframe，并删除添加的截距项。

```
# 计算vif
name = mat.columns
x = np.matrix(mat)
VIF_list = [variance_inflation_factor(x, i) for i in range(x.shape[1
])]
VIF = pd.DataFrame({'feature': name, "VIF": VIF_list})
VIF.drop(VIF[VIF['feature'] == 'c'].index, inplace=True)
```

d. 在返回的VIFdataframe中取出 VIF<=10 的属性名，返回即可。

```
VIF = cal_vif(train_set)
remaining_colist = VIF[VIF['VIF'] <= 10]['feature'].tolist()
```

## 岭回归（除基本方法之外的剔除方法）

1. 概念：岭回归是专门用于共线性数据分析的有偏估计的回归方法，实际上是一种改良的最小二乘法，但它放弃了最小二乘的无偏性，损失部分信息，放弃部分精确度为代价来寻求效果稍差但更符合实际的回归方程。
2. 计算方法：

$$\min_{w} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2^{\ 2} + \alpha||\boldsymbol{w}||_2^{\ 2}$$

3. 剔除原理：岭回归是通过引入k个单位阵，使回归系数可以估计，得到的回归估计值要比简单线性回归系数更加稳定，也更加接近真实情况。虽然引入单位阵会导致信息丢失，但同时也换来回归模型的合理估计。
4. 代码实现：

filter Python 复制代码

```python
# 通过岭回归法判断多重共线性
def remove_high_collinearity(train_set):
    # Separate input features and target variable
    X = train_set.iloc[:, :-1]  # 所有列除了最后一列
    y = train_set.iloc[:, -1]  # 最后一列

    # 标准化输入特征
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(X)

    # 创建 Ridge 回归模型
    ridge = Ridge(alpha=0.1)

    # 将 Ridge 模型拟合到数据
    ridge.fit(scaled_data, y)

    # 获取特征重要性
    feature_importances = abs(ridge.coef_)

    # 按重要性排序并获取最重要的特征
    selected_features = [f for _, f in sorted(zip(feature_importances, train_set.columns[:-1]), reverse=True)]

    return selected_features
```

进行数据预处理之后创建回归模型（alpha值根据具体情况进行确定，这里选为0.1），之后进行拟合，获得特征重要性，并返回一个根据重要性排列好的特征列表，最后根据列表选择剔除符合需求的属性。

## 与目标的相关性

1. 与目标的相关性与属性间相关性相似，均是基于相关性系数来进行处理。不同的是，这一步计算的是各属性与目标属性，即 credit_level 或 star_level 之间的相关性。
2. 剔除原理：当属性与 credit_level 或 star_level 之间相关性较少时（此处我们取0.1），判断该属性对目标的影响较小，因此将该属性剔除。
3. 代码实现：
    a. 获得各属性与 credit_level 或 star_level 的相关性系数列表。

    ```python
    level_corr_list = corr_mat[level_coname][colist]
    ```

    b. 遍历该列表，若当前属性与 credit_level 或 star_level 的相关性系数小于阈值，则删除该属性。

```
for column in colist:
    if abs(level_corr_list[column]) < threshold:
        remaining_colist.remove(column)
```

d. 返回新的属性列表 remaining_colist 即可。

## 模型选择

选择了四种机器学习模型进行比较和对比，分别为逻辑回归、决策树、随机森林、XGBoost，以下为各自实现代码，其中对于xgboost需要进行一个额外的数据转换，才能确保其正确执行，在得出结果之后进行相应的逆转换即可得到正确的结果。除此之外，各个分类器的参数都可以进行调整，但是在调整之后选择使用默认参数进行训练，因为调整后的参数效果不如默认参数

机器学习基本实现 Python 复制代码

```python
    def logistic_regression(self):
        log_model = LogisticRegression()
        log_model.fit(self.X_train, self.y_train)
        # 预测
        y_predict = log_model.predict(self.X_test)
        return y_predict

    def decision_tree(self):
        de_model = DecisionTreeClassifier(criterion='gini', random_state=0
)
        de_model.fit(self.X_train, self.y_train)
        # 预测
        y_predict = de_model.predict(self.X_test)
        return y_predict

    def random_forest(self):
        random_model = RandomForestClassifier()
        random_model.fit(self.X_train, self.y_train)
        # 预测
        y_predict = random_model.predict(self.X_test)
        return y_predict

    def xgboost(self):
        le = LabelEncoder()
        self.y_train = le.fit_transform(self.y_train)
        xg_model = xgb.XGBClassifier()
        xg_model.fit(self.X_train, self.y_train)
        # 预测
        y_predict = xg_model.predict(self.X_test)
        y_predict = le.inverse_transform(y_predict)
        return y_predict
```

# 模型评估

star：

| 模型 | 准确率 | 精确率 | 召回率 | F1分数 | Cohen's Kappa系数 |
|---|---|---|---|---|---|
| 逻辑回归模型 | 0.81984 | 0.63746 | 0.55034 | 0.57759 | 0.71518 |
| 决策树模型 | 0.87159 | 0.69412 | 0.69462 | 0.69436 | 0.80238 |

| | | | | | |
|---|---|---|---|---|---|
| 随机森林模型 | 0.89036 | 0.73464 | 0.70339 | 0.71530 | 0.83168 |
| XGBoost模型 | 0.87551 | 0.71117 | 0.66128 | 0.67289 | 0.80904 |

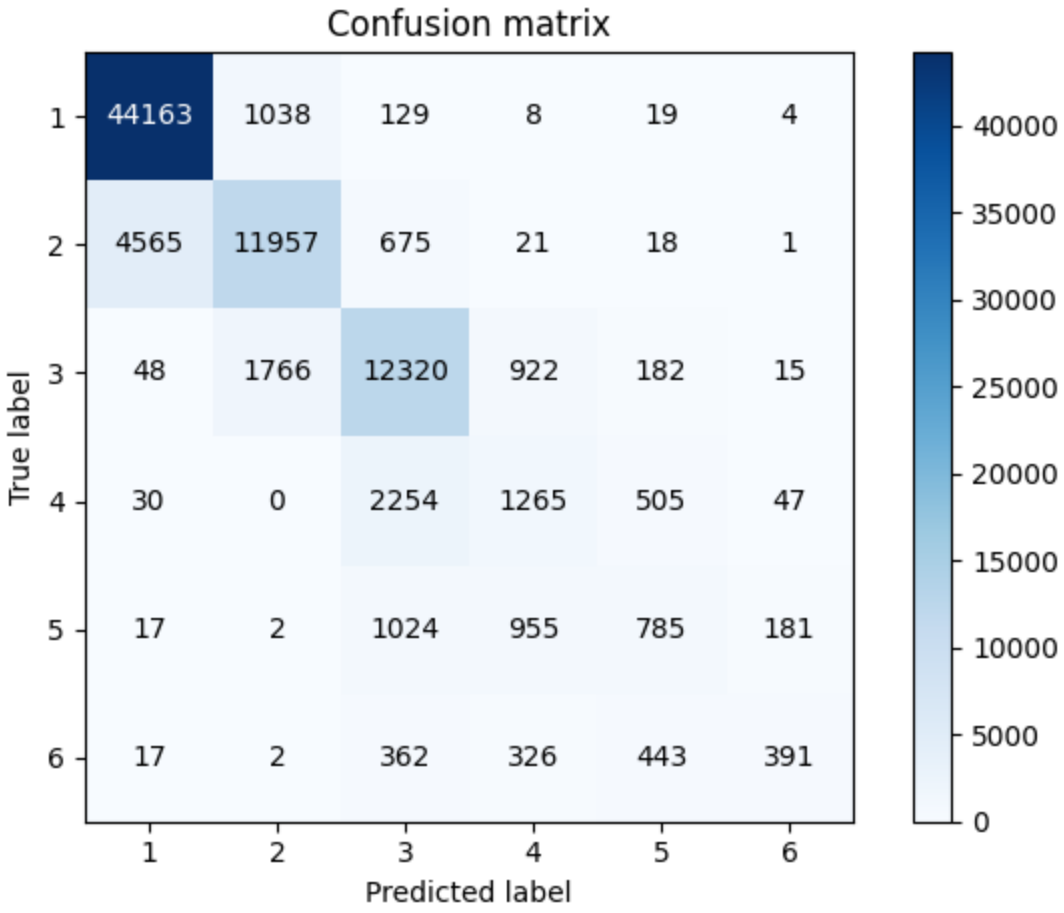
credit：

| 模型 | 准确率 | 精确率 | 召回率 | F1分数 | Cohen's Kappa系数 |
|---|---|---|---|---|---|
| 逻辑回归模型 | 0.63061 | 0.24365 | 0.24213 | 0.22888 | 0.19860 |
| 决策树模型 | 0.71852 | 0.58128 | 0.65701 | 0.61152 | 0.48088 |
| 随机森林模型 | 0.74273 | 0.67229 | 0.66547 | 0.66835 | 0.51630 |
| XGBoost模型 | 0.74184 | 0.79305 | 0.62108 | 0.68435 | 0.49061 |

根据上面四个模型的在star和credit两个数值上的预测结果以及表现我们可以基本确认随机森林模型与XGBoost模型的效果最好并且各项指标基本相近，随机森林模型除了精确率略低于XGBoost，其余均高过，原因可能是由于XGBoost模型的默认参数训练效果略有不佳，但是调参没有炼丹成功达成预期目标，因此选择采用随机森林模型进行相应的应用

混淆矩阵：

star：




credit：

由于其分值过多，不像星级能够清楚地分出等级，因此认为其混淆矩阵的可视性不佳，因此不予展示。

# 模型应用

对测试数据进行数据预处理，标准化之后选取不同的预测模型进行预测，并将数据保存为csv格式文件

Python | 复制代码

```python
test_set = pd.read_csv(form_csv_path('test'))

test_set.isnull().mean().to_csv(form_csv_path('test_missing'))

test_set = test_set.loc[:, test_set.isnull().mean() < 0.7]

test_set = pp.handle_missing_knn(test_set)

test_set = pp.replace_data(test_set)

test_set = pp.encode(test_set, catelist)

test_set = pp.standardize(test_set, colist)

y_pred = model.predict_test(test_set, colist)

test_set[credit_or_star + '_level'] = y_pred

test_set.to_csv(credit_or_star + '_test_predict.csv', index=False)


    def predict_test(self, test_set, col_list):
        pred = self.true_model.predict(test_set[col_list])
        pred = pred.astype(int)
        return pred

```