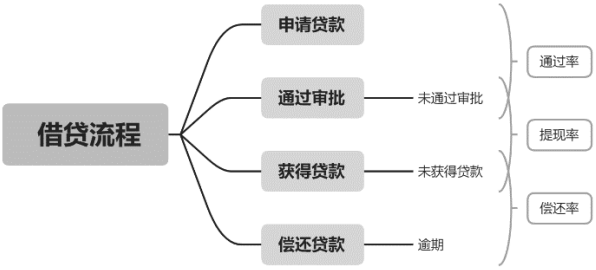
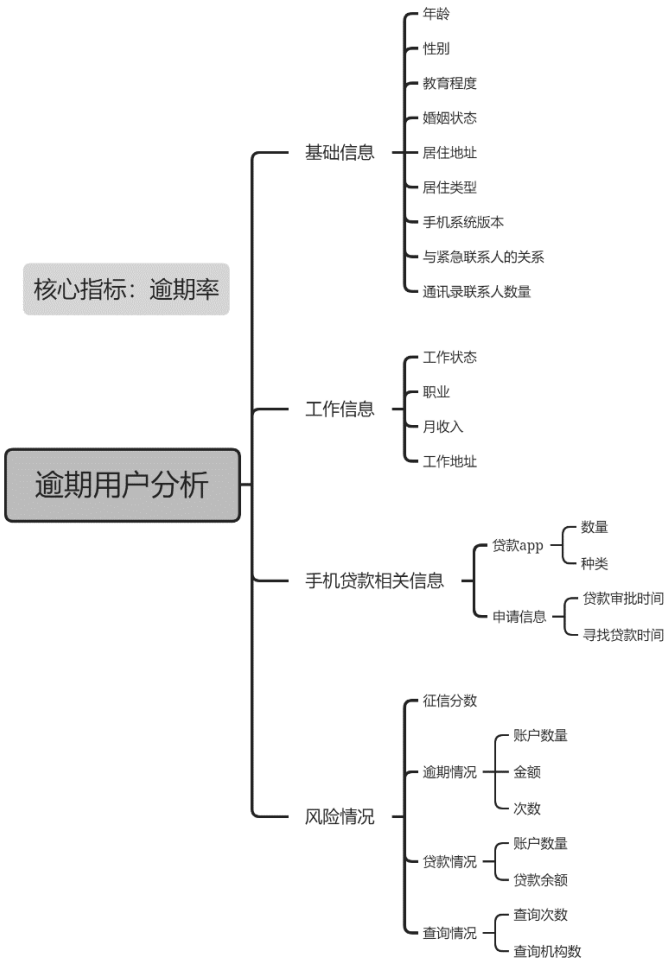
逾期用户分析

一、 研究内容

在互联网消费信贷业务中，个人借贷流程中的各环节都很重要，其中用户偿还贷款环节是关键，逾期率为线上消费借贷业务的公司或者机构重点关注的核心指标，为降低逾期率，公司需要对逾期用户画像并对用户进行筛选。



本文基于提供数据从多维度对逾期用户进行分析，核心指标为逾期率，为相关机构提出参考意见。



二、 数据分析

1. 逾期用户基本信息

表 2-1-1 不同性别逾期率

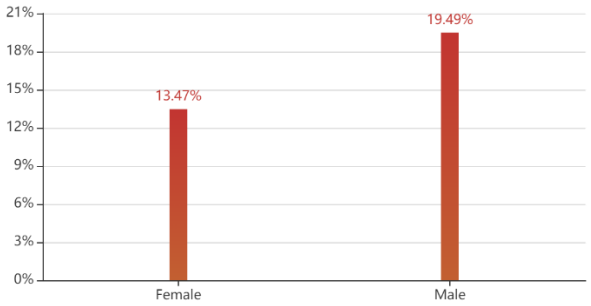


表 2-1-2 不同年龄逾期率

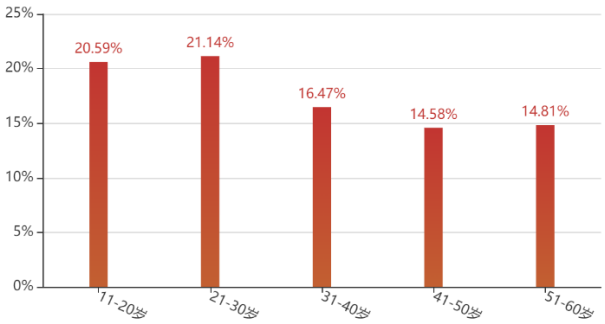


表 2-1-3 不同婚姻状态逾期率

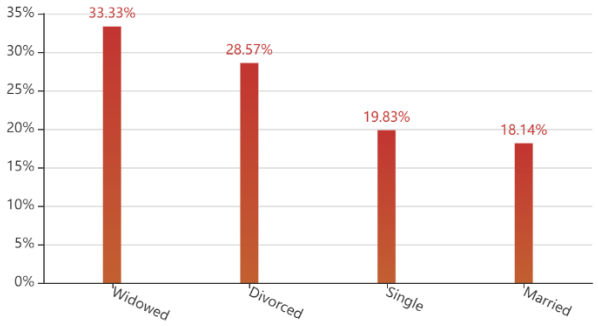


表 2-1-4 不同教育程度逾期率

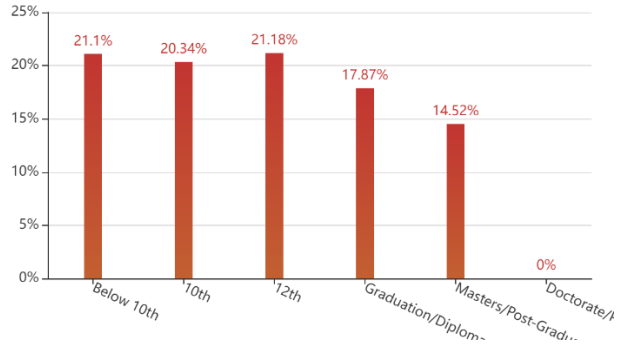


表 2-1-5 不同居住类型逾期率

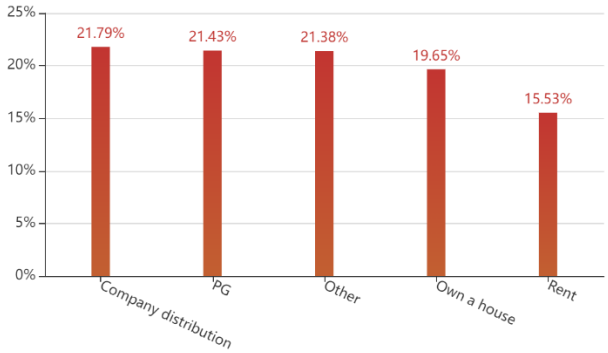
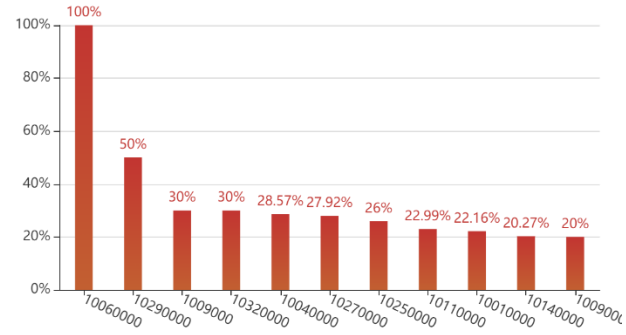


表 2-1-6 逾期率超过 20%的居住州



从上述对逾期用户基本信息的分析可知：

1） 从性别来看，男性用户的逾期率明显高于女性；

2） 从年龄来看，11 至 30 岁的年轻用户更容易逾期，逾期率超过了 20%，年龄越大越不容易逾期；

3） 从婚姻状态来看，离异与丧偶的用户的逾期率高达 30%，结婚的用户比较稳定，不易逾期；

4） 从教育程度来看，逾期率与教育程度呈负相关，教育程度越低，逾期率越高，教育程度越高，逾期率越低，研究生学历的逾期率低至 14.5%，甚至没有博士学历的用户逾期；

5） 从居住类型来看，租赁住房的用户的逾期率明显低于其他居住类型的用户，逾期率为 15.5%；

6） 从居住地址来看，不同州用户的逾期情况出现明显差异，其中有 11 个州的用户的逾期率超过了 20%，1029000 州的用户的逾期率高达 50%，1006000 州的用户的逾期率甚至为 100%。

表 2-1-7 不同手机系统版本逾期率

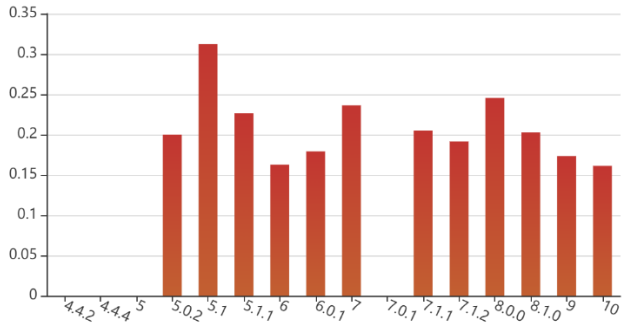


表 2-1-8 不同通讯录联系人数量逾期率

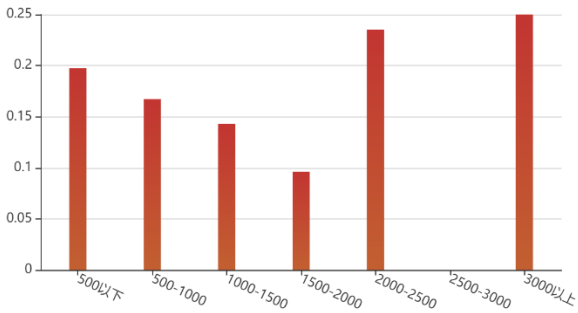


表 2-1-9 不同与紧急联系人 1 关系的逾期率

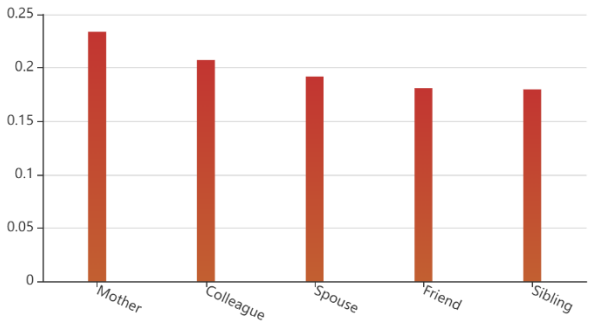
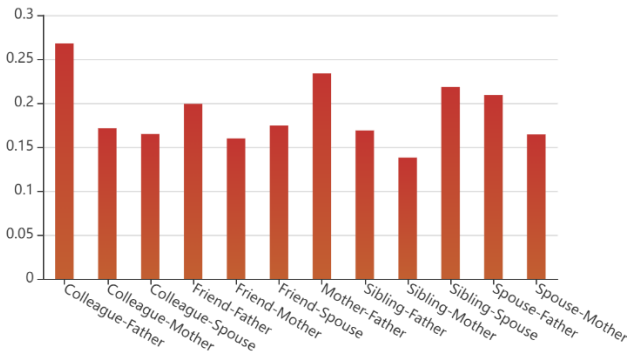


表 2-1-10 不同与紧急联系人关系的逾期率



分析用户手机与人际关系的基本情况可知：

1） 从手机系统版本来看，手机系统版本为 5.1 的用户的逾期率显著高于其他，逾期率达到了 31.3%，其次为 8.0.0 版本的逾期率较高，逾期率达到了 24.6%，手机系统版本为 6 相关、9 与 10 的逾期率较低；

2）从联系人数量来看，数量低于 2000 时，通讯录联系人数量越大，逾期率越低，但当联系人数量超过 2000 时，逾期率就显著提高；

3）从紧急联系人来看，紧急联系人 1 为母亲的用户的逾期率最高，紧急联系人 1 为兄弟姐妹的用户的逾期率最低，紧急联系人组合为同事和父亲的逾期率最高，高达 26.8，紧急联系人组合为兄弟姐妹和目前的逾期率最低，低至 13.8%。

2. 逾期用户工作信息

表 2-2-1 不同月收入的逾期率

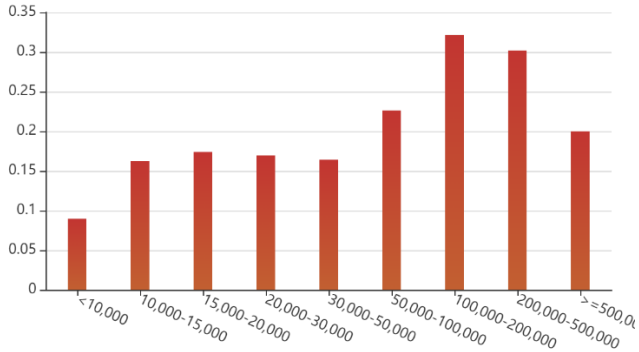


表 2-2-2 不同工作状态的逾期率

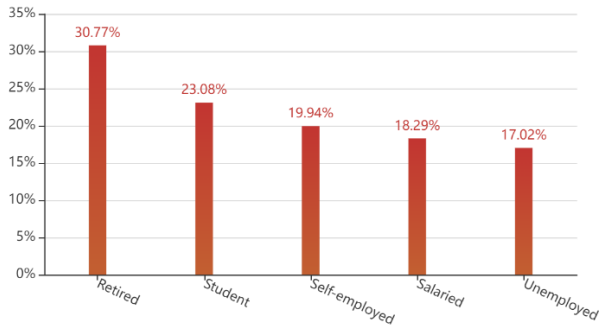
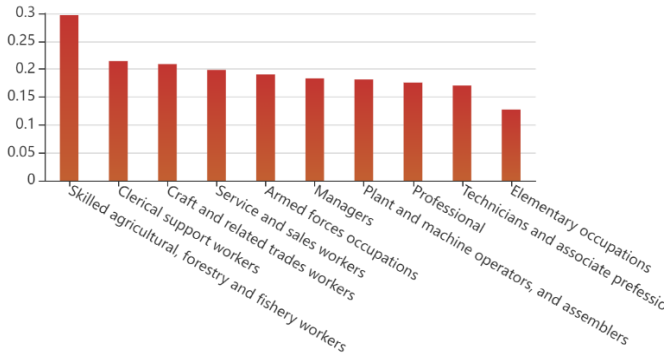
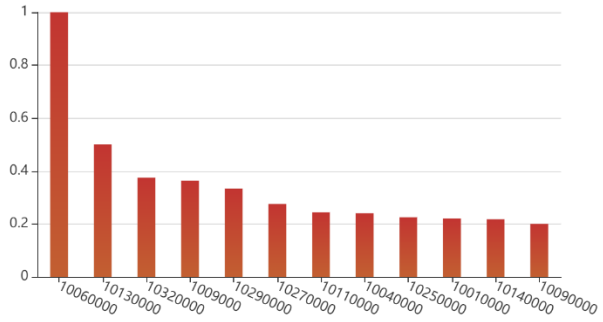


表 2-2-3 不同职业的逾期率

表 2-2-4 逾期率超过 20%的工作州





从上述用户的工作信息分析可知：

1）从工作收入来看，收入低的用户不容易逾期，逾期率在月收入为 10 万至 20 万之间达到峰值，达到了 32.1%，之后随着月收入的提高，逾期率反而下降；

2）从工作状态来看，退休用户的逾期率最高，达到了 30.8%，其次为学生，达到了 23.8%；

3）从职业来看，农渔业的技术工人的逾期率显著高于其他职业，高达 30%，初级职业的逾期率显著低于其他职业，低至 12.7%；

4）从工作地址来看，逾期率超过 20%的州有 20 个，其中 10060000 州的逾期率为 100%，其次为 10130000 州，逾期率达到了 50%。

3. 逾期用户手机贷款相关信息

表 2-3-1 不同短期贷款类 app 数量的逾期情况

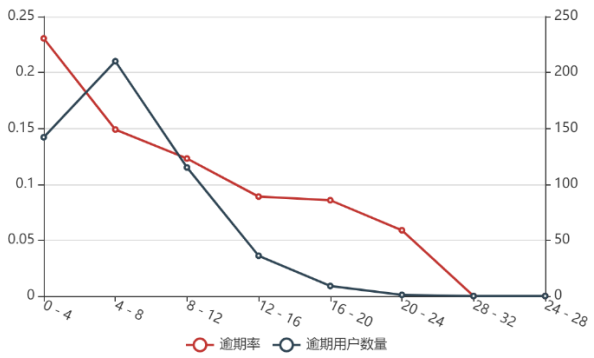
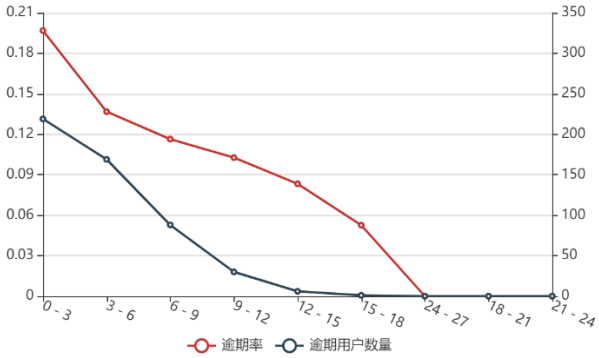


表 2-3-2 不同长期贷款类 app 数量的逾期情况



1） 逾期用户安装 app 总数的平均值为 302.5 个，其中安装短期贷款类 app 的平均数量为 6.1 个，近 30 天内安装短期贷款类 app 平均数为 2.5 个，安装长期贷款类 app 的平均数量为 3.7 个，近 30 天内安装长期贷款类 app 平均数为 2.5 个；

2） 从上图可以看出，不管是短期贷款类 app 还是长期贷款类的 app，逾期率都随着 app 数量的增加而下降。

表 2-3-3 不同申请时间的逾期情况

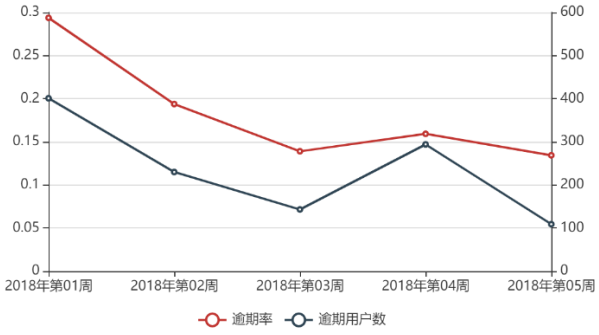


表 2-3-4 不同审批天数的逾期情况

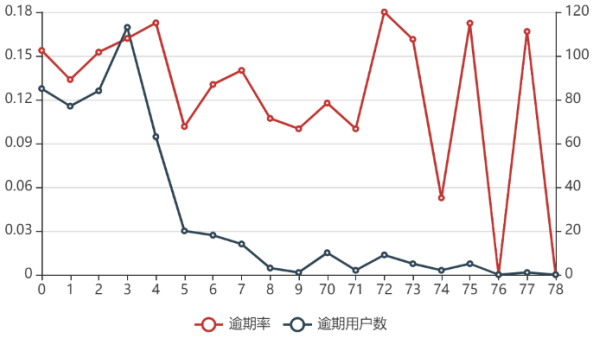
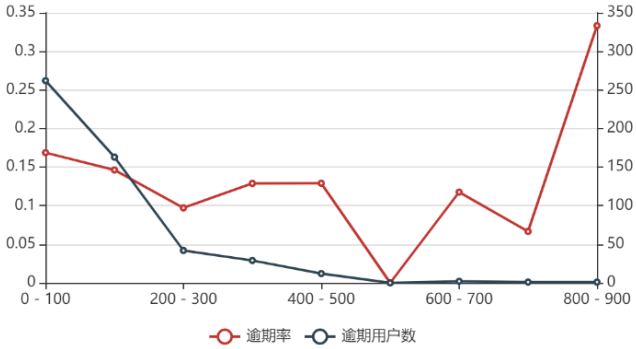


表 2-3-5 不同申请日期距离第 1 个贷款类 app 安装天数的逾期情况



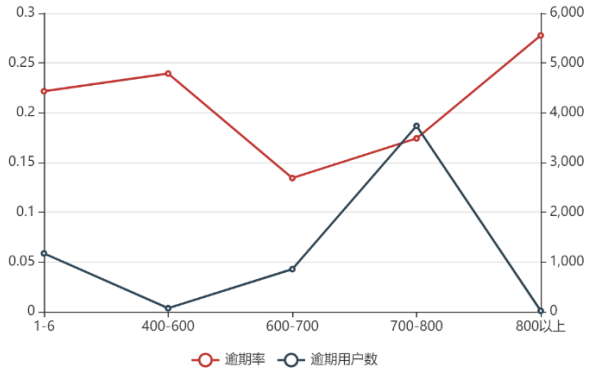
1） 从申请时间来看，逾期率与逾期用户数都随着申请时间呈下降趋势，其中 2018 年第一周的逾期率显著高于其他时间，因此可以得出结论，每年开年申请贷款的用户更容易逾期；

2） 从审批时间来看，机构审批时间集中于 0-3 天，用户逾期率与机构审批时间的关系不大；

3） 从寻找贷款时间来看，用户申请日期距离第 1 个贷款类 app 安装天数集中于 0-200 天，当天数小于 300 天时，逾期率随着相差天数的增大而提高。

4. 逾期用户风险情况

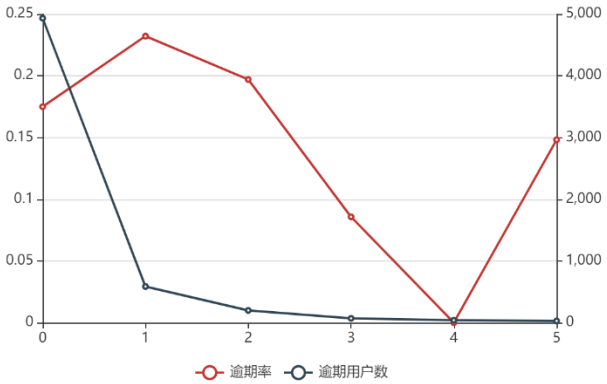
表 2-4-1 不同征信分数的逾期情况



逾期用户的征信分数集中在 700-800 分，其次为 1-6 分，征信分数为 600 分以下的用户的逾期率超过了 20%，征信分数在 600-700 之间的用户的逾期率最低，为 13.4%。

分析用户的历史逾期情况：

表 2-4-2 近 12 个月内不同历史逾期次数的逾期情况



1） 用户的逾期账户数只有 0 和 1 个两种，其中逾期账户数为 1 个的用户的逾期率为 22.4%，显著高于账户数为 0 个的用户的逾期率 17.3%；

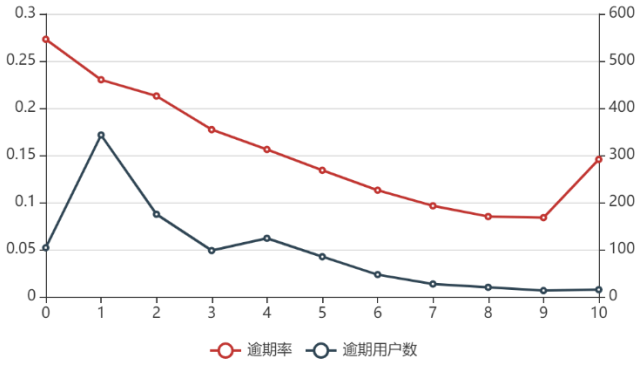
2） 逾期用户历史逾期总金额的平均值为 236，显著高于获得贷款的用户的 170；

3） 大多数逾期用户在近 12 个月内历史逾期次数为 0，从上图可以看出历史逾期次数与逾期率的关系

不大。

分析用户的贷款情况：

表 2-4-3 不同在贷账户数的逾期情况



从上图可得出，逾期率随着用户在贷账户数的增加呈下降趋势；另外，逾期用户的总贷款金额的平均值达到了 78724，远远高于获得贷款的用户的 129633。

从查询情况来看：

表 2-4-4 近 60 天内不同查询次数的逾期情况

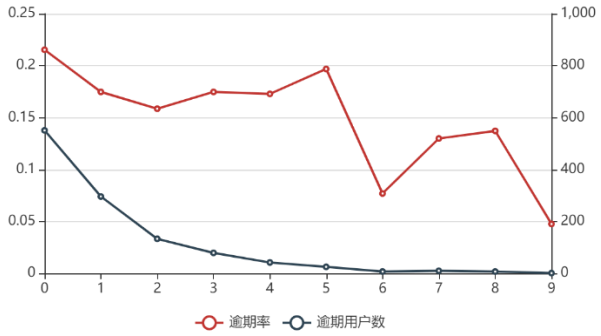
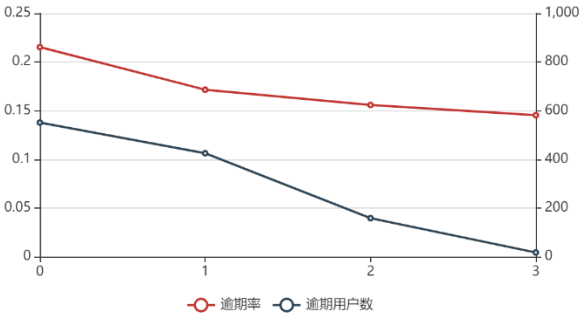


表 2-4-5 近 60 天内不同查询机构数的逾期情况



从上图来看，逾期率和逾期用户数随着查询次数和查询机构数的增加呈下降趋势。

三、 构建预测用户逾期模型

1. 数据处理

候选数据特征为 40 个，数据标签为 is_overdue 列，其中 1 表示最终逾期，0 表示未逾期。

对于特征为分类的数据，根据逾期比例排序赋值，处理过程如下：

```
#gender_v转数字
size_mapping = {'Male':0, 'Female':1}
data_features['gender_v'] = data_features['gender_v'].map(size_mapping)
```

```
#逾期比例的计算与排序在MySQL实现
```

```
#education_v逾期比例从小到大赋值1，2，3。。。
size_mapping = {'Doctorate/PhD':1, 'Masters/Post-Graduation':2, 'Graduation/Diploma':3,
                '10th':4,'Below 10th':5,'12th':6}
data_features['education_v'] = data_features['education_v'].map(size_mapping)
```

```
#marriage_v逾期比例从小到大赋值1，2，3。。。
size_mapping = {'Married':1, 'Single':2, 'Divorced':3,'Widowed':4}
data_features['marriage_v'] = data_features['marriage_v'].map(size_mapping)
```

```
#live_state随机赋值
class_encoder = LabelEncoder()
data_features['live_state'] = class_encoder.fit_transform(data_features['live_state'].values)
```

```
#live_county随机赋值
class_encoder = LabelEncoder()
data_features['live_county'] = class_encoder.fit_transform(data_features['live_county'].values)
```

```
#residence_type_v逾期比例从小到大赋值1，2，3。。。
size_mapping = {'Rent':1, 'Own a house':2, 'Other':3,
                'PG':4,'Company distribution':5}
data_features['residence_type_v'] = data_features['residence_type_v'].map(size_mapping)
```

```
#employ_stat_v逾期比例从小到大赋值1，2，3。。。
size_mapping = {'Unemployed':1, 'Salaried':2, 'Self-employed':3,
                'Student':4,'Retired':5}
data_features['employ_stat_v'] = data_features['employ_stat_v'].map(size_mapping)
```

```
#occupation_v逾期比例从小到大赋值1，2，3。。。
size_mapping = {'Elementary occupations':1, 'Technicians and associate professionals':2,
                'Professional':3,'Plant and machine operators, and assemblers':4,
                'Managers':5, 'Armed forces occupations':6, 'Service and sales workers':7,
               'Craft and related trades workers':8, 'Clerical support workers':9,
               'Skilled agricultural, forestry and fishery workers':10}
data_features['occupation_v'] = data_features['occupation_v'].map(size_mapping)
```

```
#mon_salary_v从小到大赋值1，2，3。。。
size_mapping = {'<10,000':1, '10,000-15,000':2, '15,000-20,000':3,
                '20,000-30,000':4,'30,000-50,000':5, '50,000-100,000':6,
               '100,000-200,000':7,'200,000-500,000':8,'>=500,000':9}
data_features['mon_salary_v'] = data_features['mon_salary_v'].map(size_mapping)
```

```
#work_state随机赋值
class_encoder = LabelEncoder()
data_features['work_state'] = class_encoder.fit_transform(data_features['work_state'].values)
```

```
#work_county随机赋值
class_encoder = LabelEncoder()
data_features['work_county'] = class_encoder.fit_transform(data_features['work_county'].values)
```

```
#system_version逾期比例从小到大赋值0,1，2，3。。。
size_mapping = {'4.4.4':0,'7.0.1':0,'4.4.2':1, '5':1,'10':2, '6':3,
                '9':4,'6.0.1':5, '7.1.2':6,'5.0.2':7,'8.1.0':8,'7.1.1':9,'5.1.1':10,
               '7':11,'8.0.0':12,'5.1':13}
data_features['system_version'] = data_features['system_version'].map(size_mapping)
```

```
#cont1_rel_v逾期比例从小到大赋值1，2，3。。。
size_mapping = {'Sibling':1, 'Friend':2, 'Spouse':3,
                'Colleague':4,'Mother':5}
data_features['cont1_rel_v'] = data_features['cont1_rel_v'].map(size_mapping)
```

```
#cont2_rel_v逾期比例从小到大赋值1，2，3。。。
size_mapping = {'Mother':1, 'Spouse':2, 'Father':3,}
data_features['cont2_rel_v'] = data_features['cont2_rel_v'].map(size_mapping)
```

对于个人基本信息的数据清洗已在之前实现，共得到 33492 条记录，对于手机上安装的 app 统计信

息以及征信报告信息，删去还有空值以及-99 异常值的记录，最终得到 605 条记录。

2. 筛选特征

基于 IV 与随机森林筛选特征，选择标准为 IV 值大于 0.05 和随机森林模型重要性前十。

过程如下：

```
# 计算 IV 函数
def cal_iv(x, y, n_bins=6, null_value=np.nan,):

    # 剔除空值
    x = x[x != null_value]

    # 若 x 只有一个值，返回 0
    if len(x.unique()) == 1 or len(x) != len(y):
        return 0

    if x.dtype == np.number:
        # 数值型变量
        if x.nunique() > n_bins:
            # 若 nunique 大于箱数，进行分箱
            x = pd.qcut(x, q=n_bins, duplicates='drop')

    # 计算IV
    groups = x.groupby([x, list(y)]).size().unstack().fillna(0)
    t0, t1 = y.value_counts().index
    groups = groups / groups.sum()
    not_zero_index = (groups[t0] > 0) & (groups[t1] > 0)
    groups['iv_i'] = (groups[t0] - groups[t1]) * np.log(groups[t0] / groups[t1])
    iv = sum(groups['iv_i'])

    return iv
```

```
# 统计每个特征对应的 iv 值
fea_iv = data_features.apply(lambda x: cal_iv(x, label), axis=0).sort_values(ascending=False)


# 筛选 IV > 0.05 的特征
imp_fea_iv = fea_iv[fea_iv > 0.05].index
imp_fea_iv
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# 设定参数从10到1000
param = {'n_estimators': list(range(10, 1001, 50))}
g = GridSearchCV(estimator = RandomForestClassifier(random_state = 2020), param_grid = param, cv = 5)
g.fit(data_all, label)
g.best_estimator_
```

```
#n_estimators的最佳取值为60
param = {'n_estimators': list(range(10, 110, 10))}
g = GridSearchCV(estimator = RandomForestClassifier(random_state = 2020), param_grid = param, cv = 5)
g.fit(data_all, label)
g.best_estimator_
```

```
rf = g.best_estimator_
rf_impc = pd.Series(rf.feature_importances_, index=data_all.columns).sort_values(ascending=False)
```

```
del rf_impc['target']
```

```
# 筛选重要性前十的特征
imp_fea_rf = rf_impc.index[:10]
```

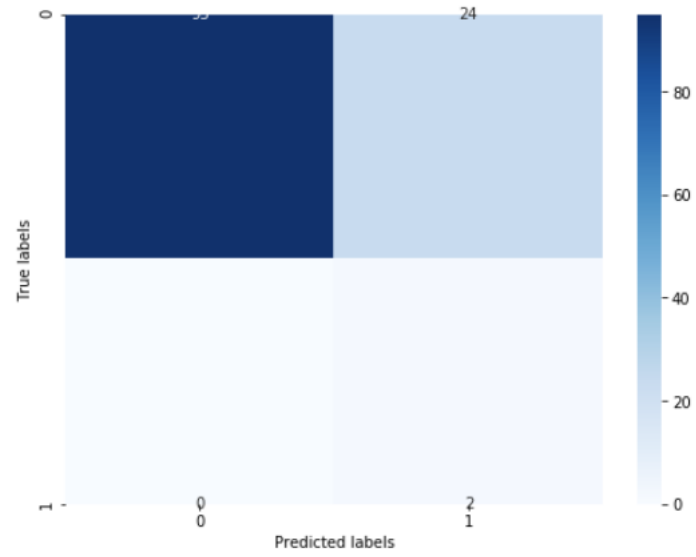
```
# 合并特征并筛选出有用特征
imp_fea = list(set(imp_fea_iv) & set(imp_fea_rf))
data_features_1 = data_features[imp_fea]
```

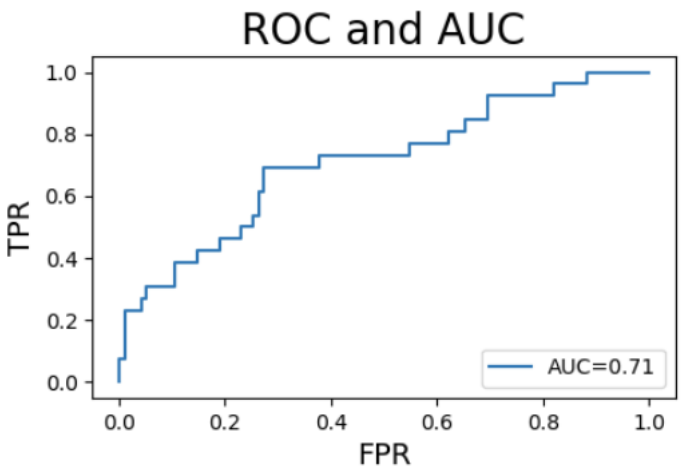
3. 模型训练与评估

将数据按 80%/20%划分为训练集和测试集，并分别对训练集和测试集的特征进行标准化，在训练集上训练模型，在测试集上验证模型性能。

最终结果如下：

```
训练集
准确率： 0.8429752066115702
精确率： 0.6
召回率： 0.03896103896103896
测试集
准确率： 0.8016528925619835
精确率： 1.0
召回率： 0.07692307692307693
The confusion matrix result:
 [[95 24]
 [ 0  2]]
```





模型的 AUC 稳定在 0.7 左右，表面该模型具有预测性。

四、结论与建议

1） 选择 25%与 15%的逾期率作为风险标准，机构应关注的容易逾期的用户的特征包括：丧偶或离异，居住州为 1006000、10290000、10320000、10270000 和 10250000，手机系统版本为 5.1.1，紧急联系人一二为同事和父亲，月收入为 50000-200000，退休，农渔业技术工人，工作州为 1006000、10130000、10320000、1009000、10290000 和 10270000；不容易逾期的用户的特征包括女性，41 岁以上，博士学历，租房，通讯录联系人数量为 1000-2000，紧急联系人一二为兄弟姐妹和母亲，月收入小于 10000，初级职业（未统计不易逾期用户工作与居住地址的特征）。

2） 用户手机里贷款类 app 数量越多，越不容易逾期；每年开年申请贷款的用户更容易逾期。

3） 根据征信报告信息判断用户的逾期可能性并不是很准确，机构在对个人借款者进行信用评估时应注意。

4） 机构应尽量完善个人借款者的信息，尽可能获得更多更准确的信息，特别是在手机上安装的 app 统计信息以及征信报告信息获取上，努力实现流程自动化以及获取信息的准确性与完整性。

5） 机构可以设定评分准入，将各个维度下的特征设定合适的评分标准，在实际贷款工作中可以根据贷款属性和贷款人属性，自动计算相应的分数。

6） 机构在预测个人借款者是否会逾期的方法上，可以通过算法构建模型来预测，本报告基于 IV 以及随机森林对特征进行筛选，并通过逻辑回归进行模型训练，最终得到一个效果良好的预测。