互联网金融 Lending Club 信贷项目实践

-10215501435 杨茜雅

实验引言：

过去的十年中，互联网金融平台如 Lending Club 通过提供借贷匹配服务，极大地便利了个人和小企业的融资路径，成为现代金融生态系统的重要组成部分。然而，随着这些平台用户数量的激增，信贷违约的问题逐渐显现，对贷款机构的资金安全构成了威胁。传统的信贷评估模型在处理海量的、异质性的网络贷款数据时，往往效率低下且准确度不高，因而急需一种能够精确预测贷款违约风险的新模型。

本研究聚焦于使用 Lending Club 平台在 2015 年度发布的贷款数据，通过深入分析和挖掘数据中蕴含的信息，旨在构建一个创新的信贷违约预测模型，以预测和识别潜在的违约借款人。该数据集包含约 890,000 笔贷款记录，覆盖贷款金额、利率、借款人的信用评分、就业情况、居住状态等多达 75 个属性。本研究首先通过数据预处理和特征工程技术，如缺失值处理、变量转换和特征选择，优化了模型的输入数据结构。进而，我们采用了逻辑回归分类器，并结合过采样技术来平衡样本分布，以应对原始数据中正负样本严重不均衡的问题。在模型构建过程中，特别关注了如何从庞大的特征集中筛选出对信贷违约预测最具影响力的因素。通过综合运用过滤方法、嵌入方法和包装方法等多种特征选择技术，本研究成功地识别出了与贷款违约最相关的关键因素，从而提高了模型的预测准确度和泛化能力。最终，该预测模型不仅能够为信贷平台提供一个准确评估借款人违约风险的工具，进而优化贷款审批流程，降低信贷风险，同时也为金融科技领域中的风险管理研究提供了一种新的方法论框架和实证研究基础。

未来，随着机器学习和人工智能技术的不断进步，信贷违约预测模型将持续优化升级，为实现更高效、更安全的信贷服务提供坚实的技术支撑。

1、实验工具

•采用了 Lending Club 信用贷款违约数据是美国网络贷款平台 LendingClub 在 2007-2015 年间的信用贷款情况数据，主要包括贷款状态和还款信息。附加属性包括：信用评分、地址、邮编、所在州等，累计 75 个属性（列），890000 笔 贷款（行）。
•贷款违约预测模型，使用了 Numpy，Pandas，Sklearn 科学计算包完成数据清洗，构建特征工程，以及完成预约模型的训练，数据可视化采用了 Matplotlib 及 Seaborn 等可视化包。

2、实验过程

本次实验分为三个阶段来完成，分别是：
•数据的初步分析和整理
•数据的探索性分析及可视化
•借贷违约预测（LogisticRegression）

2.1 数据的初步分析和整理

2.1.1 导入相关数据分析及可视化包

```
#导入相关库
import numpy as np
import pandas as pd

%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('ggplot')  #风格设置近似R这种的ggplot库

import seaborn as sns
sns.set_style('whitegrid')
```

导入 LendingClub 贷款数据

```
#导入数据及预览前三行
data=pd.read_csv("./dataset/loan.csv")
data.head(3)
```

Out[2]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	tota
0	1077501	1296599	5000.0000	5000.0000	4975.0000	36 months	10.6500	162.8700	B	B2	...	
1	1077430	1314167	2500.0000	2500.0000	2500.0000	60 months	15.2700	59.8300	C	C4	...	
2	1077175	1313524	2400.0000	2400.0000	2400.0000	36 months	15.9600	84.3300	C	C5	...	

选择 2015 年度的贷款数据

```
#选择2015年度的贷款数据
data_15=data[(data.issue_d=='Jan-2015')\
             |(data.issue_d=='Feb-2015')\
             |(data.issue_d=='Mar-2015')\
             |(data.issue_d=='Apr-2015')\
             |(data.issue_d=='Apr-2015')\
             |(data.issue_d=='Apr-2015')\
             |(data.issue_d=='May-2015')\
             |(data.issue_d=='Jun-2015')\
             |(data.issue_d=='Jul-2015')\
             |(data.issue_d=='Aug-2015')\
             |(data.issue_d=='Sep-2015')\
             |(data.issue_d=='Oct-2015')\
             |(data.issue_d=='Nov-2015')\
             |(data.issue_d=='Dec-2015')\
             ]
```

统计 2015 年度数据每列的缺失值情况

```
#统计每列的缺失值情况
check_null = data_15.isnull().sum(axis=0).sort_values(ascending=False)/float(len(data)) #查看缺失值比例
print(check_null[check_null > 0.2]) # 查看缺失比例大于20%的属性。
```

```
desc                          0.4745
dti_joint                     0.4740
annual_inc_joint              0.4740
verification_status_joint     0.4740
il_util                       0.4536
mths_since_rcnt_il            0.4511
inq_last_12m                  0.4505
open_acc_6m                   0.4505
open_il_6m                    0.4505
open_il_24m                   0.4505
open_il_12m                   0.4505
total_bal_il                  0.4505
open_rv_12m                   0.4505
open_rv_24m                   0.4505
max_bal_bc                    0.4505
all_util                      0.4505
total_cu_tl                   0.4505
inq_fi                        0.4505
mths_since_last_record        0.3907
mths_since_last_major_derog   0.3362
mths_since_last_delinq        0.2298
dtype: float64
```

从上图中可以看出，数据集中有很多列都有缺失值，所以我们要判断此列的数据对预测结果是否有影响，如果没有影响，可以将此列删除，本文中我们将缺失值超过 40%的列删除。

```
#删除缺失值超过40%的列
thresh_count = len(data_15)*0.4 # 设定阀值
data_15 = data_15.dropna(thresh=thresh_count, axis=1 ) #若某一列数据缺失的数量超过阀值就会被删除
```

再次检查缺失值的情况，只有 6 列的数据还有缺失值。

```
#按缺失值比例从大到小排列
data_15.isnull().sum(axis=0).sort_values(ascending=False)/float(len(data_15))
```

```
Out[6]: mths_since_last_delinq       0.4844
        next_pymnt_d                 0.0612
        emp_title                    0.0567
        last_pymnt_d                 0.0410
        revol_util                   0.0004
        title                        0.0003
        last_credit_pull_d           0.0000
        home_ownership               0.0000
        zip_code                     0.0000
        purpose                      0.0000
        url                          0.0000
        pymnt_plan                   0.0000
        loan_status                  0.0000
        issue_d                      0.0000
        verification_status          0.0000
```

查看数据类型的大概分布情况

```
data_15.dtypes.value_counts() # 分类统计数据类型
```

```
Out[7]: float64    31
        object     21
        int64       2
        dtype: int64
```

使用 pandas 的 loc 切片方法，得到每列至少有 2 个分类特征的数组集

```
#loc切片得到每列至少有2个分类特征的数组集
data_15 = data_15.loc[:,data_15.apply(pd.Series.nunique)!=1]
```

查看数据的变化，列数少了 1 列。

```
data_15.dtypes.value_counts()# 分类统计数据类型
```

```
Out[9]: float64    30
        object     21
        int64       2
        dtype: int64
```

上述过程，删除了较多缺失值的特征，以下将对有缺失值的特征进行处理。

2.1.2 缺失值处理

Object”和“float64“类型缺失值的处理方法不一样，所以将两者分开进行处理。

首先处理“Object”分类变量缺失值。

```
#便于理解将变量命设置为loans
loans=data_15
loans.shape
```

```
Out[11]: (421094, 53)
```

初步了解“Object”变量概况。

```
#初步了解“Object”变量概况
pd.set_option('display.max_rows',None)
loans.select_dtypes(include=['object']).describe().T
```

Out[12]:

	count	unique	top	freq
term	421094	2	36 months	283172
grade	421094	7	C	120567
sub_grade	421094	35	C1	26434
emp_title	397220	120812	Teacher	8070
emp_length	421094	12	10+ years	141520
home_ownership	421094	4	MORTGAGE	207682
verification_status	421094	3	Source Verified	179565
issue_d	421094	12	Oct-2015	48631
loan_status	421094	8	Current	377553
pymnt_plan	421094	2	n	421093
url	421094	421094	https://www.lendingclub.com/browse/loanDetail....	1
purpose	421094	14	debt_consolidation	250020
title	420962	27	Debt consolidation	249926
zip_code	421094	914	945xx	4466
addr_state	421094	49	CA	58067
earliest_cr_line	421094	668	Aug-2002	3235
initial_list_status	421094	2	w	267251
last_pymnt_d	403811	13	Jan-2016	290530
next_pymnt_d	395337	3	Feb-2016	345013
last_credit_pull_d	421083	14	Jan-2016	402875
application_type	421094	2	INDIVIDUAL	420583

Object”分类变量缺失值概况。

```
#查看“Object”分类变量缺失值概况。
objectColumns = loans.select_dtypes(include=["object"]).columns
loans[objectColumns].isnull().sum().sort_values(ascending=False)
```

```
Out[13]: next_pymnt_d            25757
         emp_title               23874
         last_pymnt_d            17283
         title                     132
         last_credit_pull_d         11
         application_type            0
         issue_d                     0
         grade                       0
         sub_grade                   0
         emp_length                  0
         home_ownership              0
         verification_status         0
         url                         0
         loan_status                 0
         pymnt_plan                  0
         purpose                     0
         zip_code                    0
         addr_state                  0
         earliest_cr_line            0
         initial_list_status         0
         term                        0
         dtype: int64
```

使用'unknown'来填充缺失值。

```
#使用'unknown'来填充缺失值
objectColumns = loans.select_dtypes(include=["object"]).columns # 筛选数据类型为object的数据
loans[objectColumns] = loans[objectColumns].fillna("Unknown") #以分类"Unknown"填充缺失值
```

确认"Object"分类变量无缺失值。

```
#查看"Object"分类变量缺失值情况
loans[objectColumns].isnull().sum().sort_values(ascending=False)
```

```
Out[20]: application_type       0
         pymnt_plan             0
         grade                  0
         sub_grade              0
         emp_title              0
         emp_length             0
         home_ownership         0
         verification_status    0
         issue_d                0
         loan_status            0
         url                    0
         last_credit_pull_d     0
         purpose                0
         title                  0
         zip_code               0
         addr_state             0
         earliest_cr_line       0
         initial_list_status    0
         last_pymnt_d           0
         next_pymnt_d           0
         term                   0
         dtype: int64
```

处理"float64"数值型变量缺失值。

```
loans.select_dtypes(include=[np.number]).isnull().sum().sort_values(ascending=False)
```

```
Out[21]: mths_since_last_delinq    203961
         revol_util                   162
         tot_cur_bal                    0
         member_id                      0
         loan_amnt                      0
         funded_amnt                    0
         funded_amnt_inv                0
         int_rate                       0
         installment                    0
         annual_inc                     0
         dti                            0
         delinq_2yrs                    0
         inq_last_6mths                 0
         open_acc                       0
         pub_rec                        0
         revol_bal                      0
         total_rev_hi_lim               0
         total_acc                      0
         out_prncp                      0
         out_prncp_inv                  0
         total_pymnt                    0
```

结果发现只有两个变量存在缺失值，使用 mean 值来填充缺失值。

```
#利用sklearn模块中的Imputer模块填充缺失值
numColumns = loans.select_dtypes(include=[np.number]).columns
from sklearn.preprocessing import Imputer
imr = Imputer(missing_values='NaN', strategy='mean', axis=0)  # 针对axis=0 列来处理
imr = imr.fit(loans[numColumns])
loans[numColumns] = imr.transform(loans[numColumns])
```

再次查看数值变量缺失值。

```
loans.select_dtypes(include=[np.number]).isnull().sum().sort_values(ascending=False)
```

```
Out[25]: total_rev_hi_lim            0
         tot_cur_bal                 0
         member_id                   0
         loan_amnt                   0
         funded_amnt                 0
         funded_amnt_inv             0
         int_rate                    0
         installment                 0
         annual_inc                  0
         dti                         0
         delinq_2yrs                 0
         inq_last_6mths              0
         mths_since_last_delinq      0
         open_acc                    0
         pub_rec                     0
         revol_bal                   0
         revol_util                  0
         total_acc                   0
         out_prncp                   0
         out_prncp_inv               0
         total_pymnt                 0
```

从上表中可以看到数值变量中已经没有缺失值。

2.1.3 数据过滤

本文的目的是对平台用户的贷款违约做出预测，所以需要筛选得到一些对用户违约有影响的信息，其他不相关的冗余信息，需要将其删除掉。

首先查看所有的分类标签

```
loans.columns
```

```
Out[26]: Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
                'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_title',
                'emp_length', 'home_ownership', 'annual_inc', 'verification_status',
                'issue_d', 'loan_status', 'pymnt_plan', 'url', 'purpose', 'title',
                'zip_code', 'addr_state', 'dti', 'delinq_2yrs', 'earliest_cr_line',
                'inq_last_6mths', 'mths_since_last_delinq', 'open_acc', 'pub_rec',
                'revol_bal', 'revol_util', 'total_acc', 'initial_list_status',
                'out_prncp', 'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv',
                'total_rec_prncp', 'total_rec_int', 'total_rec_late_fee', 'recoveries',
                'collection_recovery_fee', 'last_pymnt_d', 'last_pymnt_amnt',
                'next_pymnt_d', 'last_credit_pull_d', 'collections_12_mths_ex_med',
                'application_type', 'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal',
                'total_rev_hi_lim'],
               dtype='object')
```

- sub_grade：与 Grade 的信息重复
- emp_title：缺失值较多，同时不能反映借款人收入或资产的真实情况
- zip_code：地址邮编，邮编显示不全，没有意义
- addr_state：申请地址所属州，不能反映借款人的偿债能力
- last_credit_pull_d：LendingClub 平台最近一个提供贷款的时间，没有意义
- policy_code：变量信息全为 1
- pymnt_plan 基本是 n

•title： title 与 purpose 的信息重复，同时 title 的分类信息更加离散
•next_pymnt_d : 下一个付款时间，没有意义
•policy_code : 没有意义
•collection_recovery_fee: 全为 0，没有意义
•earliest_cr_line : 记录的是借款人发生第一笔借款的时间
•issue_d： 贷款发行时间，这里提前向模型泄露了信息
•last_pymnt_d、collection_recovery_fee、last_pymnt_amnt： 预测贷款违约模型是贷款前的风险控制手段，这些贷后信息都会影响我们训练模型的效果，在此将这些信息删除
•url:所有的行都不同，没有分类意义

将以上重复或对构建预测模型没有意义的属性进行删除。

```
#删除对模型没有意义的列
loans2=loans.drop(['sub_grade', 'emp_title',  'title', 'zip_code', 'addr_state','url'], axis=1, inplace = True)
loans3=loans.drop(['issue_d', 'pymnt_plan',  'earliest_cr_line', 'initial_list_status', 'last_pymnt_d','next_pymnt_d','last_credit_pull_d'], axis=1, inplace = True)
```

再次查看'Object'类型变量，只剩下 8 个分类变量。

```
object_columns_df3 =loans.select_dtypes(include=["object"]) #筛选数据类型为object的变量
print(object_columns_df3.iloc[0])
```

```
term                         60 months
grade                                C
emp_length                    10+ years
home_ownership                 MORTGAGE
verification_status     Source Verified
loan_status                      Issued
purpose               home_improvement
application_type               INDIVIDUAL
Name: 466285, dtype: object
```

2.2 数据的探索性分析及数据可视化

数据预处理完后，接下来探索数据的特征工程，为后续的违约预测模型做好建模准备工作
特征工程是机器学习最重要的一部分，希望找到的特征是最贴近实际业务场景的，所以要反复去找特征，只需要最少的特征得到简单的模型，并且有最好的预测效果。
本节将特征工程主要分 3 大部分：特征抽象 、特征缩放 、特征选择

2.2.1 特征抽象

数据集中有很多的"Object"类型的分类变量存在，但是对于这种变量，机器学习算法不能识别，需要将其转化为算法能识别的数据类型。
首先对于"loan_status"数据类型转换

```
#统计"loan_status"数据的分布
loans['loan_status'].value_counts()
```

```
Out[13]: Current               377553
         Fully Paid             22984
         Issued                  8460
         Late (31-120 days)      4691
         In Grace Period         3107
         Charged Off             2773
         Late (16-30 days)       1139
         Default                  387
         Name: loan_status, dtype: int64
```

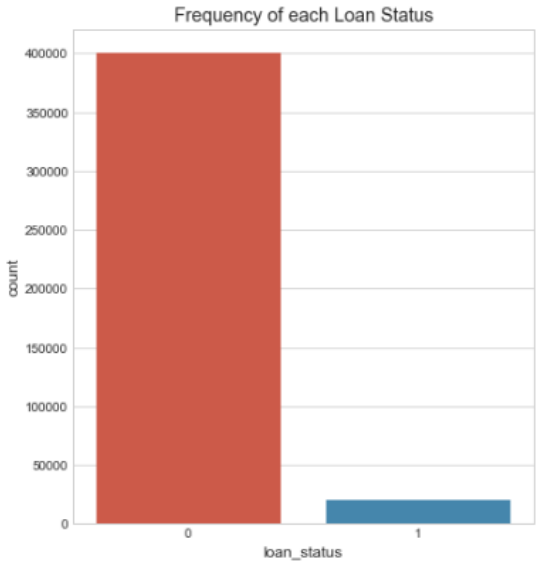
将上表中的违约编码为 1，正常的为 0 进行编码。

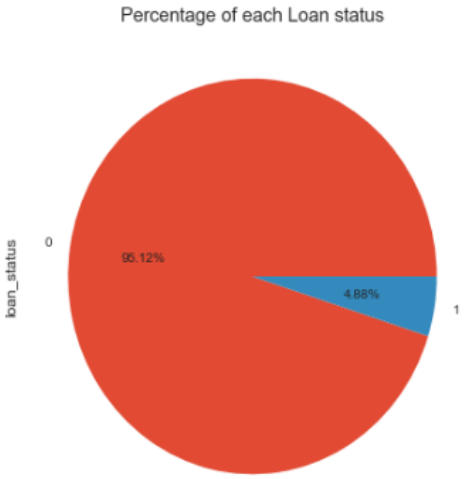
```
#使用Pandas replace函数定义新函数：
def coding(col, codeDict):
    colCoded = pd.Series(col, copy=True)
    for key, value in codeDict.items():
        colCoded.replace(key, value, inplace=True)
    return colCoded

#把贷款状态LoanStatus编码为违约=1，正常=0:
pd.value_counts(loans["loan_status"])
loans["loan_status"] = coding(loans["loan_status"], {'Current':0,'Fully Paid':0\
                                                     ,'In Grace Period':1\
                                                     ,'Late (31-120 days)':1\
                                                     ,'Late (16-30 days)':1\
                                                     ,'Charged Off':1\
                                                     ,"Issued":1\
                                                     ,"Default":1\
                                                     ,"Does not meet the credit policy. Status:Fully Paid":1\
                                              ,"Does not meet the credit policy. Status:Charged Off":1})

print( '\nAfter Coding:')
pd.value_counts(loans["loan_status"])
```

```
After Coding:

Out[33]: 0    400537
         1     20557
         Name: loan_status, dtype: int64
```

可视化查看"loan_status"中不同状态的替换情况。

```
# 贷款状态分布可视化
fig, axs = plt.subplots(1,2,figsize=(14,7))
sns.countplot(x='loan_status',data=loans,ax=axs[0])
axs[0].set_title("Frequency of each Loan Status")
loans['loan_status'].value_counts().plot(x=None,y=None, kind='pie', ax=axs[1],autopct='%1.2f%%')
axs[1].set_title("Percentage of each Loan status")
plt.show()
```





变量“emp_length”、"grade"进行特征抽象化

```
# 构建mapping，对有序变量"emp_length"、“grade”进行转换
mapping_dict = {
    "emp_length": {
        "10+ years": 10,
        "9 years": 9,
        "8 years": 8,
        "7 years": 7,
        "6 years": 6,
        "5 years": 5,
        "4 years": 4,
        "3 years": 3,
        "2 years": 2,
        "1 year": 1,
        "< 1 year": 0,
        "n/a": 0
    },
    "grade":{
        "A": 1,
        "B": 2,
        "C": 3,
        "D": 4,
        "E": 5,
        "F": 6,
        "G": 7
    }
}

loans = loans.replace(mapping_dict) #变量映射
loans[['emp_length','grade']].head() #查看效果
```

Out[40]:

	emp_length	grade
466285	10	3
466286	0	1
466287	5	3
466288	10	3
466289	10	2

变量"home_ownership", "verification_status", "application_type","purpose", "term" 狂热编码

```
#变量狂热编码
n_columns = ["home_ownership", "verification_status", "application_type","purpose", "term"]
dummy_df = pd.get_dummies(loans[n_columns])# 用get_dummies进行one hot编码
loans = pd.concat([loans, dummy_df], axis=1) #当axis = 1的时候，concat就是行对齐，然后将不同列名称的两张表合并
loans = loans.drop(n_columns, axis=1)  #清除原来的分类变量
```

重新查看数据集中的数据类型

```
loans.info() #查看数据信息
```

```
purpose_small_business                 421094 non-null uint8
purpose_vacation                       421094 non-null uint8
purpose_wedding                        421094 non-null uint8
term_ 36 months                        421094 non-null uint8
term_ 60 months                        421094 non-null uint8
dtypes: float64(32), int64(3), uint8(25)
memory usage: 125.7 MB
```

2.2.2 特征缩放

采用标准化的方法进行去量纲操作，加快算法收敛速度，采用 scikit-learn 模块 preprocessing 的子模块 StandardScaler 进行操作。

```
col = loans.select_dtypes(include=['int64','float64']).columns
col = col.drop('loan_status') #剔除目标变量
loans_ml_df = loans # 复制数据至变量loans_ml_df


from sklearn.preprocessing import StandardScaler # 导入模块
sc =StandardScaler() # 初始化缩放器
loans_ml_df[col] =sc.fit_transform(loans_ml_df[col]) #对数据进行标准化
loans_ml_df.head() #查看经标准化后的数据
```

Out[45]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	grade	emp_length	annual_inc
466285	1.5317	1.5531	2.3053	2.3053	2.3070	-0.1417	1.3743	0.1705	1.1171	0.6901
466286	1.5426	1.5633	-0.7689	-0.7689	-0.7685	-1.6864	-0.7407	-1.3637	-1.4719	0.3115
466287	1.4689	1.4913	-1.2851	-1.2851	-1.2849	0.5207	-1.2077	0.1705	-0.1774	-0.5675
466288	1.5383	1.5593	-0.6114	-0.6114	-0.6109	-0.1417	-0.8963	0.1705	1.1171	-0.4661
466289	1.5209	1.5131	0.5553	0.5553	0.5562	-0.4219	-0.0376	-0.5966	1.1171	-0.1889

5 rows × 60 columns

以上过程完成了非数值型特征抽象化处理，使得算法能理解数据集中的数据，这么多的特征，究竟哪些特征对预测结果影响较大，所以以下通过影响大小对特征进行选择。

2.2.3 特征选择

特征的选择优先选取与预测目标相关性较高的特征，不相关特征可能会降低分类的准确率，因此为了增强模型的泛化能力，我们需要从原有特征集合中挑选出最佳的部分特征，并且降低学习的难度，能够简化分类器的计算，同时帮助了解分类问题的因果关系。

一般来说，根据特征选择的思路将特征选择分为 3 种方法：嵌入方法（embedded approach）、过滤方法（filter approach）、包装方法（wrapper approacch）。

●过滤方法（filter approach）：通过自变量之间或自变量与目标变量之间的关联关系选择特征。

●嵌入方法（embedded approach）：通过学习器自身自动选择特征。

●包装方法（wrapper approacch）：通过目标函数（AUC/MSE）来决定是否加入一个变量。

本次项目采用 Filter、Embedded 和 Wrapper 三种方法组合进行特征选择。

首先将数据集中的贷款状态'loan_status'抽离出来

```
#构建X特征变量和Y目标变量
x_feature = list(loans_ml_df.columns)
x_feature.remove('loan_status')
x_val = loans_ml_df[x_feature]
y_val = loans_ml_df['loan_status']
len(x_feature) # 查看初始特征集合的数量
```

Out[46]: 59

重新查看没有贷款状态'loan_status'的数据集。

```
x_val.describe().T # 初览数据
```

Out[47]:

	count	mean	std	min	25%	50%	75%	max
id	421094.0000	0.0000	1.0000	-5.6857	-0.8238	0.1508	0.8370	1.5499
member_id	421094.0000	0.0000	1.0000	-5.6767	-0.8259	0.1340	0.8339	1.5739
loan_amnt	421094.0000	-0.0000	1.0000	-1.6614	-0.7864	-0.1447	0.5553	2.3053
funded_amnt	421094.0000	-0.0000	1.0000	-1.6614	-0.7864	-0.1447	0.5553	2.3053
funded_amnt_inv	421094.0000	-0.0000	1.0000	-1.6730	-0.7860	-0.1440	0.5562	2.3070
int_rate	421094.0000	0.0000	1.0000	-1.6864	-0.7948	-0.0722	0.6921	3.7955
installment	421094.0000	0.0000	1.0000	-1.6816	-0.7267	-0.2305	0.5592	4.0987
grade	421094.0000	0.0000	1.0000	-1.3637	-0.5966	0.1705	0.9376	3.2389
emp_length	421094.0000	-0.0000	1.0000	-1.4719	-0.9541	0.0815	1.1171	1.1171
annual_inc	421094.0000	-0.0000	1.0000	-1.0408	-0.4187	-0.1618	0.1991	127.4245
dti	421094.0000	-0.0000	1.0000	-0.8167	-0.2811	-0.0254	0.2614	424.5997
delinq_2yrs	421094.0000	0.0000	1.0000	-0.3745	-0.3745	-0.3745	-0.3745	41.6652
inq_last_6mths	421094.0000	-0.0000	1.0000	-0.6600	-0.6600	-0.6600	0.4924	6.2543

Wrapper 方法

选出与目标变量相关性较高的特征。通过暴力的递归特征消除 (Recursive Feature Elimination)方法筛选 30 个与目标变量相关性最强的特征，将特征维度从 59 个降到 30 个。

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
# 建立逻辑回归分类器
model = LogisticRegression()
# 建立递归特征消除筛选器
rfe = RFE(model, 30) #通过递归选择特征，选择30个特征
rfe = rfe.fit(x_val, y_val)
# 打印筛选结果
print(rfe.support_)
print(rfe.ranking_) #ranking 为 1代表被选中，其他则未被代表未被选中
```

```
[ True  True  True  True  True  True  True False False False False False
 False False False False False False False  True  True  True  True  True
  True  True  True  True  True False False False False False  True  True
  True  True  True  True  True  True  True False  True False False False
 False False  True False False False False False False  True  True]
[ 1  1  1  1  1  1  1 11 19 26 30 22 18 15 17 24 20 25 16  1  1  1  1  1
  1  1  1  1 27 28 23 29 21  1  1  1  1  1  1  1  1  1  5  1  4 12  3  8 10
  1  6  2 13  7  9 14  1  1]
```

通过布尔值筛选首次降维后的变量。

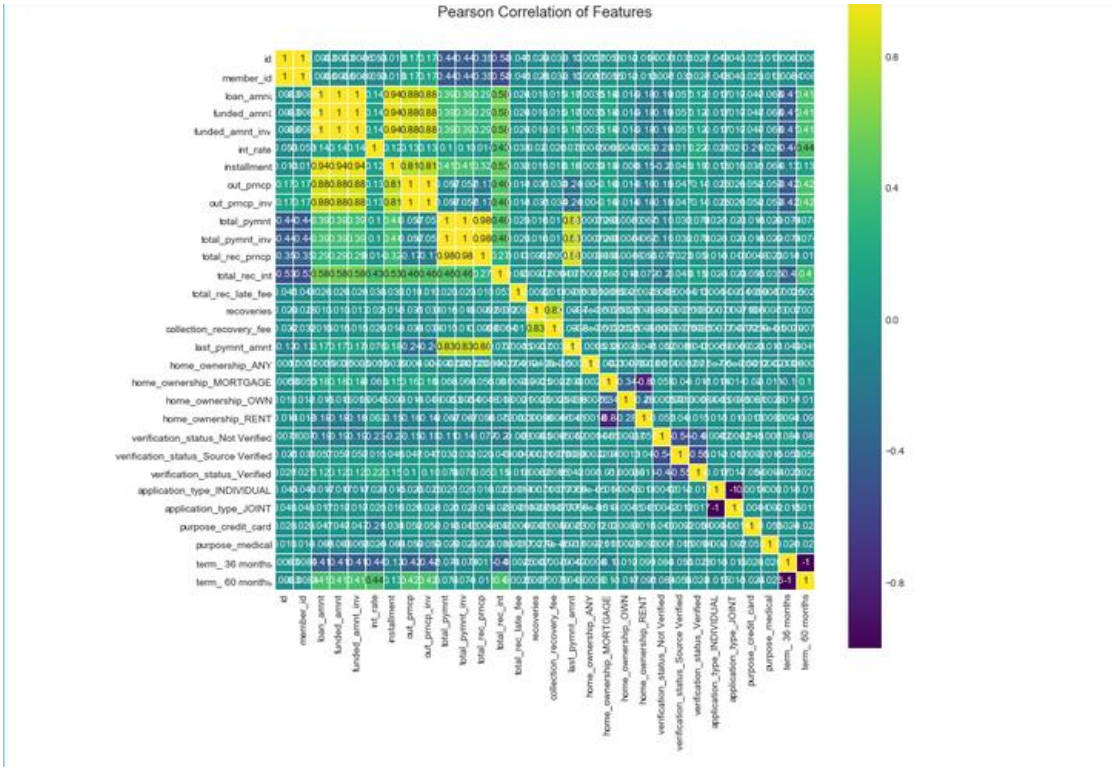
```
col_filter = x_val.columns[rfe.support_] #通过布尔值筛选首次降维后的变量
col_filter # 查看通过递归特征消除法筛选的变量
```

```
Out[49]: Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'int_rate', 'installment', 'out_prncp', 'out_prncp_inv', 'total_pymnt',
       'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_amnt', 'home_ownership_ANY', 'home_ownership_MORTGAGE',
       'home_ownership_OWN', 'home_ownership_RENT',
       'verification_status_Not Verified',
       'verification_status_Source Verified', 'verification_status_Verified',
       'application_type_INDIVIDUAL', 'application_type_JOINT',
       'purpose_credit_card', 'purpose_medical', 'term_ 36 months',
       'term_ 60 months'],
      dtype='object')
```

Filter 方法

正常情况下，影响目标变量的因数是多元性的；但不同因数之间会互相影响（共线性 ），或相重叠，进而影响到统计结果的真实性。下一步，以下通过皮尔森相关性图谱找出冗余特征并将其剔除，且通过相关性图谱进一步引导我们选择特征的方向。

```
colormap = plt.cm.viridis
plt.figure(figsize=(12,12))
plt.title('Pearson Correlation of Features', y=1.05, size=15)
sns.heatmap(loans_ml_df[col_filter].corr(),linewidths=0.1,vmax=1.0, square=True, cmap=colormap, linecolor='white', annot=True)
```



从上图中得到需要删除的冗余特征。

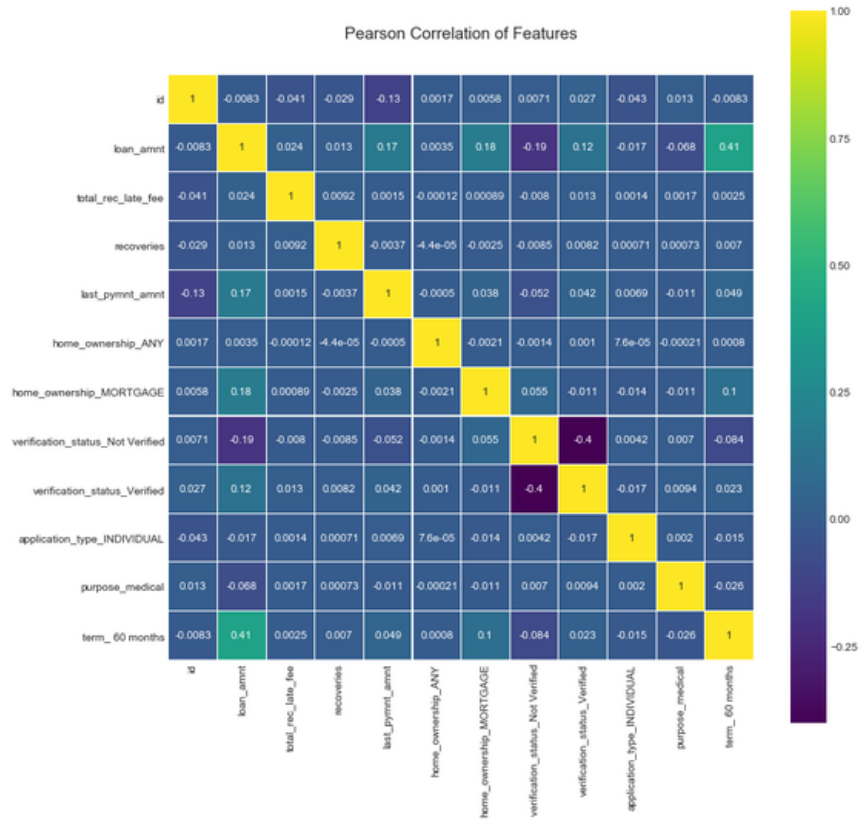
```
drop_col = ['id','member_id','collection_recovery_fee','funded_amnt', 'funded_amnt_inv','installment', 'out_prncp', 'out_prncp_inv',
                    'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int', 'home_ownership_OWN',
                    'application_type_JOINT',  'home_ownership_RENT' ,
                    'term_ 36 months', 'total_pymnt', 'verification_status_Source Verified', 'purpose_credit_card','int_rate']
col_new = col_filter.drop(drop_col) #剔除冗余特征
print(len(col_new))
```

```
Out[53]: 12
```

特征从 30 个降到 12 个，再次确认处理后的数据相关性。

```
col_new # 查看剩余的特征
colormap = plt.cm.viridis
plt.figure(figsize=(12,12))
plt.title('Pearson Correlation of Features', y=1.05, size=15)
sns.heatmap(loans_ml_df[col_new].corr(),linewidths=0.1,vmax=1.0, square=True, cmap=colormap, linecolor='white', annot=True)
```

```
Out[54]: Index(['id', 'loan_amnt', 'total_rec_late_fee', 'recoveries',
                'last_pymnt_amnt', 'home_ownership_ANY', 'home_ownership_MORTGAGE',
                'verification_status_Not Verified', 'verification_status_Verified',
                'application_type_INDIVIDUAL', 'purpose_medical', 'term_ 60 months'],
               dtype='object')
```



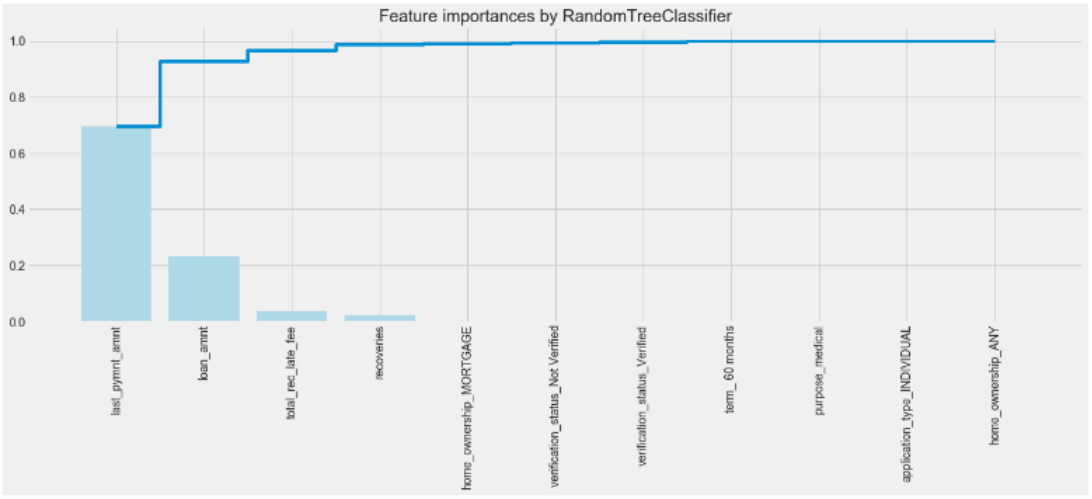
Embedded 方法

为了了解每个特征对贷款违约预测的影响程度，所以在进行模型训练之前，我们需要对特征的权重有一个正确的评判和排序，就可以通过特征重要性排序来挖掘哪些变量是比较重要的，降低学习难度，最终达到优化模型计算的目的

```
#随机森林算法判定特征的重要性
names = loans_ml_df[col_new].columns
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=10,random_state=123)#构建分类随机森林分类器
clf.fit(x_val[col_new], y_val) #对自变量和因变量进行拟合
names, clf.feature_importances_
for feature in zip(names, clf.feature_importances_):
    print(feature)
```

特征重要性从大到小排序及可视化图形，结果发现最具判别效果的特征是收到的最后付款总额'last_pymnt_amnt'

```
('loan_amnt', 0.2320401372170755)
('total_rec_late_fee', 0.038105066238408793)
('recoveries', 0.023185975751540376)
('last_pymnt_amnt', 0.69414506227249018)
('home_ownership_ANY', 2.2497162069184006e-05)
('home_ownership_MORTGAGE', 0.0029571340368987845)
('verification_status_Not Verified', 0.0027043169977816772)
('verification_status_Verified', 0.0025556284177692878)
('application_type_INDIVIDUAL', 0.0006161493143893509)
('purpose_medical', 0.0011238981016409573)
('term_ 60 months', 0.0025441344899357981)
```



2.3 借贷违约预测模型（LogisticRegression）

2.3.1 样本不平衡处理

本项目中，2015 年度贷款平台上违约的借款人比例很低，约为 4.9%，正负样本量非常不平衡，非平衡样本常用的解决方式有 2 种：

•过采样（oversampling），增加正样本使得正、负样本数目接近，然后再进行学习。

•欠采样（undersampling），去除一些负样本使得正、负样本数目接近，然后再进行学习。

```
X = loans_ml_df[col_new]
y = loans_ml_df["loan_status"]
n_sample = y.shape[0]
n_pos_sample = y[y == 0].shape[0]
n_neg_sample = y[y == 1].shape[0]
print('样本个数：{}; 正样本占{:.2%}; 负样本占{:.2%}'.format(n_sample,
n_pos_sample / n_sample,
n_neg_sample / n_sample))
print('特征维数：', X.shape[1])
```

```
样本个数：421094; 正样本占95.12%; 负样本占4.88%
特征维数： 11
```

```
from imblearn.over_sampling import SMOTE # 导入SMOTE算法模块
# 处理不平衡数据
sm = SMOTE(random_state=42)    # 处理过采样的方法
X, y = sm.fit_sample(X, y)
print('通过SMOTE方法平衡正负样本后')
n_sample = y.shape[0]
n_pos_sample = y[y == 0].shape[0]
n_neg_sample = y[y == 1].shape[0]
print('样本个数：{}; 正样本占{:.2%}; 负样本占{:.2%}'.format(n_sample,
                                                  n_pos_sample / n_sample,
                                                  n_neg_sample / n_sample))
```

```
通过SMOTE方法平衡正负样本后
样本个数：801074; 正样本占50.00%; 负样本占50.00%
```

2.3.2 模型训练

采用逻辑回归分类器 分类器进行训练

```
# 构建逻辑回归分类器
from sklearn.linear_model import LogisticRegression
clf1 = LogisticRegression()
clf1.fit(X, y)
```

查看预测结果的准确率

```
predicted1 = clf.predict(X) # 通过分类器产生预测结果
from sklearn.metrics import accuracy_score
print("Test set accuracy score: {:.5f}".format(accuracy_score(predicted1, y,)))
```

```
Test set accuracy score: 0.67768
```

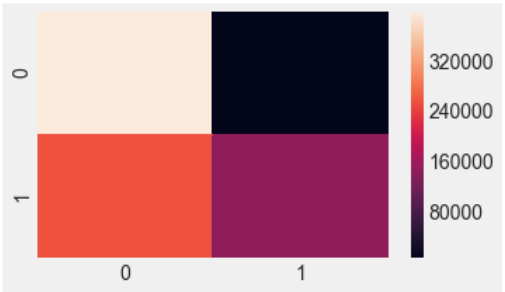
利用混淆矩阵及可视化观察预测结果

```
#生成混淆矩阵
from sklearn.metrics import confusion_matrix
confusion_matrix(y, predicted1)
```

```
Out[73]: array([[396734,   3803],
                [254399, 146138]], dtype=int64)
```

```
# 混淆矩阵可视化
plt.figure(figsize=(5,3))
sns.heatmap(m)
```

```
Out[74]: <matplotlib.axes._subplots.AxesSubplot at 0x49641198>
```



再利用 sklearn.metrics 子模块 classification_report 查看 precision、recall、f1-score 的值

```
#查看precision、recall、f1-score的值
from sklearn.metrics import classification_report
print(classification_report(y, predicted1))
```

	precision	recall	f1-score	support
0	0.61	0.99	0.75	400537
1	0.97	0.36	0.53	400537
avg / total	0.79	0.68	0.64	801074

```
#计算ROC值
from sklearn.metrics import roc_auc_score
roc_auc1 = roc_auc_score(y, predicted1)
print("Area under the ROC curve : %f" % roc_auc1)
```

```
Area under the ROC curve : 0.677680
```

以上完成了全部的模型训练及预测工作。

3、小结

本文基于互联网金融平台 2015 年度贷款数据完成信贷违约预测模型，全文包括了数据清洗，构建特征工程，训练模型，最后得到的模型准确率达到了 0.79，召回率达到了 0.68，具有较好的预测性，本文的模型可以作为信贷平台预测违约借款人的参考。